Modeling of Energy Consumption in GPS Receivers for Power Aware Localization Systems

Claudio Mandrioli
Department of Automatic Control,
Lund University
claudio.mandrioli@control.lth.se

Alberto Leva
Dipartimento di Elettronica, Informazione e Bioingegneria,
Politecnico di Milano, Italy
alberto.leva@polimi.it

Bo Bernhardsson
Department of Automatic Control,
Lund University
bo.bernhardsson@control.lth.se

Martina Maggio
Department of Automatic Control,
Lund University
martina.maggio@control.lth.se

ABSTRACT

This paper proposes a first-principle model of GPS receivers, that allows us to exploit the trade-off between battery consumption and positioning accuracy. We present the model and propose a GPS sampling strategy that uses both the current positioning confidence, and information about the GPS status. We complement the GPS sensor with internal measurement units and show how the given model exposes the battery-accuracy trade-off in the context of sensor fusion. We demonstrate the usefulness of the proposed sampling strategy using both simulation and real data.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; **Sensors and actuators**; *Embedded software*;

KEYWORDS

GPS Receiver, Power-Aware Computing, Cyber-Physical Modeling, Localization Systems.

ACM Reference Format:
Claudio Mandrioli, Alberto Leva, Bo Bernhardsson, and Martina Maggio. 2019. Modeling of Energy Consumption in GPS Receivers for Power Aware Localization Systems. In *10th ACM/IEEE International Conference on Cyber-Physical Systems (with CPS-IoT Week 2019) (ICCPS '19), April 16–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3302509.3311043

1 INTRODUCTION

Global Positioning System (GPS) receivers are well known to be power-hungry with respect to the power consumed by a small electronic device [1, 2, 9, 10, 12, 14, 16, 18, 19, 21]. This is probably the main reason that motivates research on the optimization of the GPS stack and on multi-sensor data merging, for example via sensor fusion [11]. In fact, combining the information provided by more than one sensor type could allow one to exploit the sensor benefits and to limit their drawbacks.

One alternative is merging data from GPS sensors with data provided by inertial measurements sensors [4]. While the GPS is power-hungry but provides very precise information, inertial measurements sensors are less energy-demanding, but also less precise. In the literature, optimizations of this type are accompanied by experimental data [1, 2, 9, 10, 12, 16, 18–20], that are time-consuming to retrieve and only valid for the specific testing setup.

This paper follows a different approach to tackle the problem of optimizing power consumption for localization in GPS devices. We propose a first-principle model of the GPS receiver. This model captures the physical *phenomena* that determine the sensor behavior. The main advantage of using such a model is its independence from its specific hardware and software. We capture the behavior of a *generic* GPS receiver and its dynamics. This allows us to test strategies for power optimization and to obtain insights on its effectiveness before implementing them on a real device. We envision our work to provide contributions in the design of GPS-assisted navigation strategies for devices with battery constraints, from activity trackers to tracking systems mounted on drones.

Specifically, this paper makes the following contributions:

- **Modeling:** It provides a *first-principle* model of the GPS behavior, identifying the dynamics that regulate it. A first-principle model is a model that captures the technological design choices that are behind the GPS system. These choices greatly influence what can be achieved with any GPS sensor, as they introduce basic limitations and characteristics of the technology. In this specific context, we highlight how a dynamical model is necessary to capture the involved *phenomena*. In fact, GPS sensors that receive the same *stimula* can behave differently, depending on their internal state.
- **Design:** It identifies opportunities for battery savings. Specifically, modeling the GPS-related *phenomena* allows us to devise a sampling strategy that exploits the technology characteristics.
- **Integration:** It integrates the GPS with an ecosystem of inertial measurement sensors. While this is not a new idea, thanks to our model we are able to capture the trade-offs (of the different merging algorithms) programmatically and to expose the characteristics of each solution.

ICCPS '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6285-6/19/04…$15.00
https://doi.org/10.1145/3302509.3311043

Even though the notions used to build the presented model are known in the litterature, their systematic study and integration has not been seen elsewhere by the authors. The development of this first-principle and simulable model enables the possiblity of (i) performing at design phase a quantitative evaluaion of the energy cost of different sampling strategies, (ii) comparing different sampling strategies without depending on the specific testing conditions.

This paper is organized as follows. As much research has been done on the topic of GPS optimization, Section 2 describes related work. Section 3 describes the physical principles behind the GPS receiver and shows how we capture these principles with our model. Section 4 discusses a sensor fusion algorithm that merges the information obtained by Inertial Measurement Units (IMUs) with the GPS data. The sampling strategy that is derived using the given models is described in Section 5. We evaluate our proposal with data from real GPS and IMU devices and simulated traces in Section 6. Finally, Section 7 concludes the paper.

2 RELATED WORK

Battery drain is a serious problem when GPS sensors are used in small devices. This is well testified by the number and variety of works that try to mitigate it. Previous work mainly apply to smartphones and can be categorized in two classes, depending on the type of approach that is used for battery optimization: (i) work on the GPS stack – i.e., work optimizing the behavior of the sensor, (ii) work that reduce the usage of the GPS sensor – i.e., work that tries to sample less frequently or only when needed.

The first class includes results like [5, 13–16]. The authors of [13] aim at outsourcing the device computation (once the data has been retrieved) to some server, using a network connection. [15] improves the GPS receiver power-efficiency selecting only a subset of visible satellites to be tracked. Other works aim at improving the speed of the signal acquisition, either using information from previous acquisitions [16], or using different algorithms for the decoding of the signal [5, 14].

The second class includes several attempts to build an *adaptation* layer that controls the usage of positioning sensors, keeping the receiver off for as much time as possible [2, 12, 19]. Usually, this is achieved implementing a trade-off controller, that trades accuracy for energy consumption. In the same class we can include works that exploit other sensors. When the adaptation layer detects that the user state does not need high accuracy, it minimizes the GPS receiver usage by turning it off and enabling it again only on demand or eventually switching to other positioning techniques [1, 9, 10, 18, 20]. Among the works on this additional adaptation layer, [21] proposes a set of design principles for smartphone applications, to improve the smartphone battery efficiency.

This work has a complementary role with respect to the ones mentioned above. We propose here a modeling approach based only on how the GPS receiver is designed. This is therefore transversal with respect to the implementation details of the specific sensor. We argue that we can study the trade-off between accuracy and battery consumption in a first principled way, using this model. We can also determine how different factors (including satellite visibility, and timing behaviors) affect the receiver.

Modeling GPS sensors is not a new research area, see for example [9, 15]. These prior efforts are mainly data-driven, i.e., they collect data for a specific receiver with a given hardware and software platform and try to infer the behavior of the sensor from this data (and not having any notion of state). This paper represents an attempt of deriving a generic *dynamic* model. We argue that this is needed, since the GPS receiver can behave differently in response to the same input, depending on its internal state.

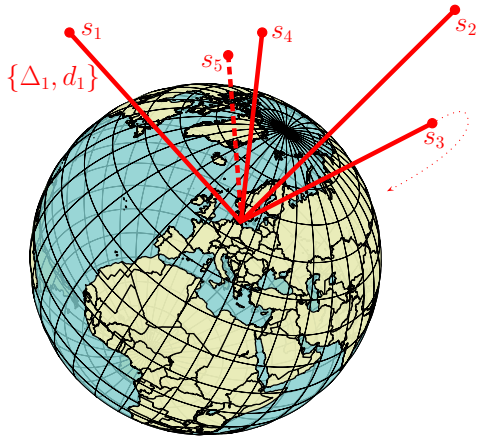

Figure 1: Trilateration: GPS receiver and satellites.

To end this brief review, we would like to notice that this work combines, in a purpose-specific manner, problems and ideas from different domains. To quote just a couple, there is a vast literature on correcting drift in accelerometer-based odometry, see e.g. [3], and the idea of taking action – in our case, accessing the GPS – only when performance is deemed unsatisfactory, is shared by the more general domain of event-triggered sampling [6]. However, the addressed problem has specific characteristics, detailed in the following, that make our solution effective despite – and in some sense thanks to – its simplicity.

3 GPS RECEIVER MODEL

This section introduces the receiver model used in the rest of the paper. Specifically, Section 3.1 describes the physics behind the model and Section 3.2 discusses our modelling choices.

3.1 GPS physics

GPS sensors locate themselves through a process called *trilateration* [8]. This process consists of measuring the distance from 4 or more points in space (satellites), whose positions are known. Given the distance measurements, the GPS sensor then performs a least square estimation to determine its current position. Figure 1 shows an example with five satellites. To correctly estimate the current position, the GPS receiver must measure the distance from s_1, s_2, s_3 and s_4. Additionally, measuring the distance from s_5 is not necessary, but improves the position accuracy.

The GPS framework includes (circa) 30 satellites. These satellites orbit around the Earth following known trajectories. While orbiting, they broadcast periodic signals that encode a set of parameters, called *ephemeris data*. The ephemeris data describe the satellites' orbits (see for example the trajectory of satellite s_3 in Figure 1), and therefore allow the GPS receiver to accurately determine their position in time. The satellite trajectories change over time, due

Table 1: Combinations of Ephemeris Data, Ranging Data and Antenna Status.

State Name	Ephemeris Data	Ranging Data	Antenna	State Number
Position Available	Available	Available	On	④
—	~~Available~~	~~Available~~	~~Off~~	
Warm Start	Available	Not Available	On	⑤
Warm Start Available	Available	Not Available	Off	⑥
Read Ephemeris Data	Not Available	Available	On	③
—	~~Not Available~~	~~Available~~	~~Off~~	
Cold Start	Not Available	Not Available	On	②
Not Available Info	Not Available	Not Available	Off	①

to uncertainties and disturbances, like corrections for collision avoidance.

The ephemeris data are considered valid for a time span of 30 minutes. To correctly estimate the current position, the receiver should ensure that the ephemeris data are up to date. The transmission of the ephemeris data has a duration of 30 seconds, and the satellites continuously broadcast new data. In order to ensure the correct acquisition of one data point, the receiver then has to fetch and decode the signal for a time that is in the interval $[30, 60)$ seconds (in the worst case, the receiver starts reading the message right after the start of a new message transmission).

All the satellites transmit on the same frequency and the different signals are multiplexed using the Code Division Multiple Access (CDMA) technique. Using CDMA, the signal has three components: (i) the carrier wave, (ii) the data waveform, and (iii) a spreading waveform. The spreading waveform is a deterministic signal, different for each satellite[1], transmitted at much higher frequency with respect the data waveform. The spreading waveform is then used to recognize the satellite the received message belongs to.

The spreading waveform is also used for determining the distance between the satellite and the receiver. Assuming the satellite and the receiver share a time reference, the receiver can measure the delay it takes to receive the signal from the satellite. In Figure 1 the delay to receive the message from satellite s_1 is denoted by Δ_1. Multiplying this quantity by the speed of light C, the receiver can determine how far the signal has been travelling, i.e., the distance from the satellite, indicated in Figure 1 with d_1. For a generic satellite x, this can be written as $d_x = \Delta_x \cdot C$. The set of the distances the receiver measures from the visible satellites is called *ranging data*.

The hypothesis that the clocks of the receiver and the satellites are synchronized is not valid, so one extra satellite must be tracked and used for the trilateration procedure. The fourth satellite allows the receiver to compensate its time reference offset.

Due to the satellites' and the receiver's movements, the doppler effect will distort the signal reception. The effect is a shift in the frequency spectrum of the signal. To fetch the signal, the receiver must then perform a two-dimensional search in phase and frequency shift. As a consequence, the process of fetching the signal directly includes the measurement of the ranging data[2]. This process takes some milliseconds (usually in the range from $2ms$ to $10ms$), depending mainly on signal strength.

3.2 GPS modeling

We plan to use a model of the GPS receiver behavior to optimize battery consumption subject to position data availability. Our receiver model needs to capture two relevant *phenomena*: GPS position availability and power consumption.

The GPS receiver provides a position estimate when it has collected both the ephemeris and the ranging data for at least 4 satellites. Detecting data from more than 4 satellites can improve the positioning accuracy. This depents on many factors which are hard to model, like the relative position of the stellites in sapce and the geography of the environment around the sensor. Therefore, for the sake of keeping the model at a reasonable complexity, it is only required to set a minimum number of satellites to be tracked, depending on the specific application. This number has to be equal to or greater than 4 and it being higher represents the the constraint of higher accuracy in the given application. As for power consumption, the receiver always consumes a (negligible) idle power. On top of that, the sensor consumes additional power when its radio is turned on, which is precisely the cause of battery draining. This power has been experimentally shown to be constant in time [2, 16] and usually between 20mW and 400mW. We use the latter for our model, but this is just a constant that can be changed depending on the device.

The important states that our model needs to capture are:

(1) *ephemeris data* are available or not;
(2) *ranging data* are available or not;
(3) the radio *antenna* is turned on or off.

The discrete nature of these phenomena can be well captured using a finite state machine. For our purposes, a *timed automaton* is sufficient. In the following, we describe the states and transitions of the resulting timed automaton. We use the variable `visible_satellites` to represent the number of (different) satellites' signals that reach the receiver.

States: The model *states* are the combination of the three above configurations. Table 1 reports all of them. However, the availability of the ranging data is incompatible with the

[1]Each satellite's spreading waveform is unique and has been chosen to be a *golden code*, i.e., it looks like white noise and it does not correlate with any other satellite sequence.

[2]Additionally, the receiver can estimate its own speed from the measured doppler effect and compensate for the satellite speed retrieved with the ephemeris data.

Table 2: Transitions, summary table with name, conditions and eventual updates.

Transition	Triggering Condition	Effects (Updates)
`turn_on`	User Controlled	—
`turn_off`	User Controlled	—
`fetch_freq&phase`	Time Delay, usually in the interval [2ms, 10ms]	`sat_tracked_freq&phase := visible_satellites`
`get_ephemeris`	Time Delay, in the interval [30s, 60s)	`sat_tracked_ephemeris := visible_satellites` `expiry_time_ephemeris := time + 30minutes`
`lose_visibility`	`sat_tracked_freq&phase < required_satellites`	—
`ephemeris_expire`	`time > expiry_time_ephemeris or` `sat_tracked_ephemeris < required_satellites`	— —

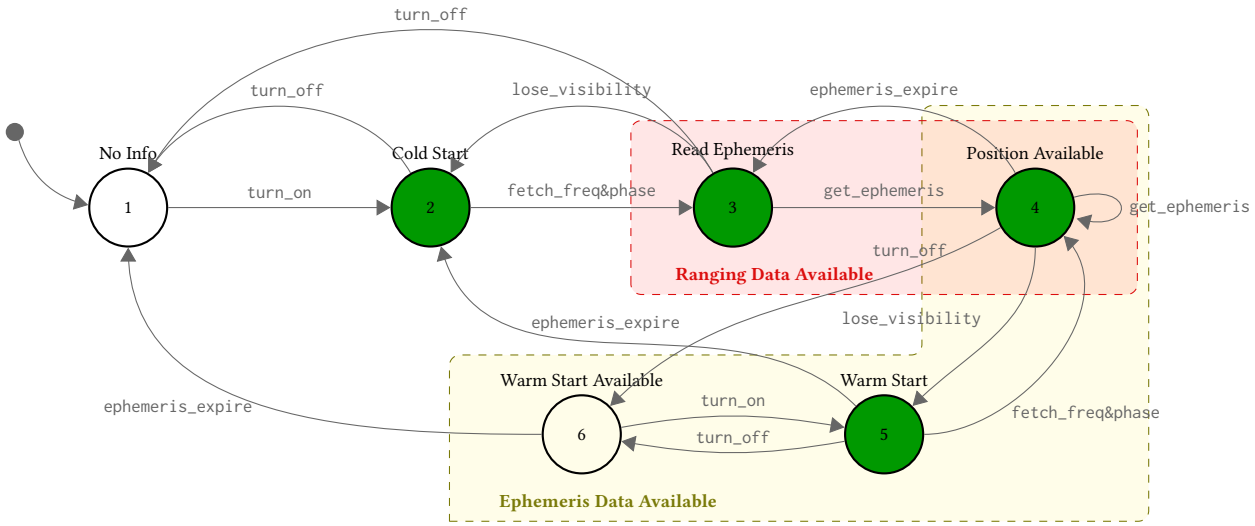

Figure 2: State machine representing the *cyber* dynamics.

radio being turned off, so we remove the two corresponding states from the list. The discrete states are also augmented with variables representing the number of satellites being tracked. These are: (i) the number of satellites for which the ephemeris data are available (`sat_tracked_ephemeris`), (ii) the number of satellites for which the ranging data are available (`sat_tracked_freq&phase`), and (iii) the expiration time of the ephemeris data (`expiry_time_ephemeris`).

Transitions: Analogously, the model *transitions* represent the events that alter the information availability or the antenna state changes. As described in Section 3.1, the ephemeris data become available when the receiver listens consecutively to the satellites' signal for long enough (transition `get_ephemeris`). Their loss of availability happens either at the expiration of the ephemeris data, or when the tracked satellites disappear from the visible sky. In theory, the second event does not necessarily force an update of the ephemeris data. For instance, a satellite may disappear for a limited time interval and appear again before its ephemeris data expiration. For simplicity (and without loss of information with respect to our model usage) we do not include the specific tracking of different satellites in the model and, consequently, we do not distinguish between these two cases. The transition `ephemeris_expire` implements both. The ranging data instead become available as soon as the satellites' signals are fetched. We refer to this transition as `fetch_freq&phase`. The loss of ranging data can have two causes: (i) the antenna is turned off (transition `turn_off`), (ii) the number of visible satellites is less than the required one[3] (transition `lose_visibility`). Lastly the antenna's state is controlled by the user with the events `turn_on` and `turn_off`.

The transitions can be divided in three categories: *inputs*, *delays*, and *disturbances*. The two transitions `turn_on` and `turn_off` are arbitrarily controlled by the user, and can be seen as inputs of the system. The second category, delays, includes transitions that represent the end of data aquisition procedures and can be seen as delays in the system's dynamics: `fetch_freq&phase` and `get_ephemeris`. When fired, these transitions cause the update of the state variables `sat_tracked_ephemeris` and `sat_tracked_freq&phase`, that are set to the number of `visible_satellites`. In addition, when the `get_ephemeris` transition is fired, the expiration time of the ephemeris data is also updated, according to `expiry_time_ephemeris = time + 30minutes`. The *disturbance* class of transitions includes `ephemeris_expire` and `lose_visibility`. Both the transitions are triggered when the

[3]Generally 4. Different applications may require a higher number of satellites to improve positioning accuracy (https://www.developerfusion.com/article/4652/writing-your-own-gps-applications-part-2/2/).

number of visible satellites is less than the required number. Moreover, the `ephemeris_expire` transition is triggered also when the current time exceeds `expiry_time_ephemeris`.

The different transitions, their triggering conditions, and their effects (updates) are summarized in Table 2.

Dynamics: After defining states and transitions, we need to specify which transitions can fire in each state and the resulting system state. The resulting *automaton* is shown in Figure 2. In each of the states, the three properties shown in Table 1 can change, e.g., the ephemeris data can become unavailable if they were available. The states drawn in green are states in which the antenna is on. The pink rectangle encloses the states in which the ranging data are available. The yellow polygon shows the states in which the ephemeris data are available.

In general, each state accepts only the transitions that change the properties that define it. For instance, if the antenna is off (states No Info ① and Warm Start Available ⑥), the automaton can accept the transition `turn_on` and move to a different state, but cannot accept the transition `turn_off`. There is one exception, corresponding to the self-loop in Figure 2. In state Position Available ④ the transition `get_ephemeris` can be fired to indicate an update of the ephemeris data before their expiration.

Finally, two (discrete) equations complement the model dynamics. They saturate the number of tracked satellites (both in terms of ephemeris and ranging data) to the number of visible satellites, implementing the fact that the receiver can only track visible satellites. These are, respectively, `sat_tracked_ephemeris := min{sat_tracked_ephemeris, visible_satellites}` and `sat_tracked_freq&phase := min{sat_tracked_freq&phase, visible_satellites}`.

4 SENSOR FUSION

The proposed model is general and can be used for many different purposes. The abstraction offered by the model devised in this paper allows for analysis and synthesis of control strategies. This section introduces its use in the context of sensor fusion. Sensor fusion [11] refers to a set of techniques for merging measurements that come from different sensors with the aim of enhancing the advantages of the specific sensors while compensating for the disadvantages. For our specific case, we would like to limit battery consumption without penalizing accuracy, using Inertial Measurement Units (IMUs) in conjunction with the GPS receiver.

IMUs are sensors that usually include an accelerometer and a gyroscope. The accelerometer measures the acceleration along the three axes while the gyroscope measures the angular velocity of the orientation changes. IMUs can be considered complementary to the GPS. In fact, they provide a relative measurement (while the GPS position is absolute), and are characterized by low power consumption, since they do not consume the extra power needed to use an antenna. Their measurements are also continuously available, since they do not rely on other devices. A navigation system only based on IMUs is referred to as an Inertial Navigation System (INS).

The main weakness of an INS is that, given the differential nature of the sensors, their measurements must be integrated. This exposes the estimation to significant low frequency errors introduced by the biases. As a rule of thumb, those errors are usually considered grow cubically in time. IMUs are therefore considered reliable only for short time intervals. The idea of *GPS-IMU* sensor fusion algorithm is to exploit the IMUs to provide continuous low-power positioning. At the same time, the GPS can be sampled at lower rate to compensate for the low frequency errors.

The GPS model devised in this paper captures the delays introduced by the GPS receiver and its power consumption. This allows us to rigorously study an optimal sampling method for the GPS sensor, with respect to a given accuracy and power consumption target. On the contrary, the model used for the sensor fusion algorithm is well-known, and we base our results on [17].

4.1 The Sensor Fusion Algorithm

This section describes the proposed sensor fusion algorithm. The algorithm integrates the IMU measurements to obtain a trajectory profile. GPS positioning is triggered irregularly, and when the GPS position is received an *Extended Kalman Filter* [7] estimates and compensates the IMU biases.

We want to estimate the following quantities: the tridimensional position, the tridimensional velocity and the orientation (attitude) using its quaternion representation. We define the state $x(k) = [p(k), v(k), q(k)]'$, where we denote with $p(k)$ the position at time k and measure it in meters with respect to a reference point, i.e., $p(k) \in \mathbb{R}^3[m]$. We use $v(k)$ to indicate the velocity at time k measuring it in meters per seconds, i.e., $v(k) \in \mathbb{R}^3[m/s]$. Finally, we denote with $q(k)$ the attitude at time k, using its (dimensionless) quaternion representation, i.e., $q(k) \in \mathbb{R}^4[-]$.

The IMU provides the following measurements at time k: the acceleration and the angular rates. Both the acceleration $s(k)$ and the angular rates are provided along the three axis, i.e., $s(k) \in \mathbb{R}^3[m/s^2]$, and $\omega(k) \in \mathbb{R}^3[rad/s]$. Equation (1) shows the integration of the IMU measurements s and ω to obtain p, v, and q. In the equation, T_s represents the IMU sampling time, R_b^n denotes the directional cosine matrix that rotates a vector from the body coordinate frame b to the body coordinate frame n and g is the gravitational force, I_4 is the identity matrix of order 4,

$$\begin{aligned} p(k) &= p(k-1) + T_s \cdot v(k-1) + \\ &\quad (R_b^n(q(k-1)) \cdot s(k) - g) \cdot {}^{T_s^2}/_2 \\ v(k) &= v(k-1) + (R_b^n(q(k-1)) \cdot s(k) - g) \cdot T_s \\ q(k) &= (\cos(0.5 \cdot T_s \cdot \|\omega(k)\|) \cdot I_4 + \\ &\quad \sin(0.5 \cdot T_s \cdot \|\omega(k)\|) \cdot {}^{\Omega(k)}/_{\|\omega(k)\|}) \cdot q(k-1) \end{aligned} \quad . \quad (1)$$

Finally, Ω is defined as

$$\Omega(k) = \begin{bmatrix} 0 & [\omega(k)]_z & -[\omega(k)]_y & [\omega(k)]_x \\ -[\omega(k)]_z & 0 & [\omega(k)]_x & [\omega(k)]_y \\ [\omega(k)]_y & -[\omega(k)]_x & 0 & [\omega(k)]_z \\ -[\omega(k)]_x & -[\omega(k)]_z & -[\omega(k)]_z & 0 \end{bmatrix}. \quad (2)$$

Equation (1) can also be seen in the form of a discrete-time dynamical system $x(k) = f(x(k-1), u(k))$. This representation highlights that $x(k) = [p(k), v(k), q(k)]'$ is the system state and output, i.e., the position estimation, and $u(k) = [s(k), \omega(k)]'$ is the system input, i.e., the data read by the IMU. The corresponding block diagram is shown in Figure 3.

Our aim is to correct the estimation made by the INS using the more reliable measurements obtained using the GPS receiver. We denote with $u(k)$ the actual acceleration and with $\tilde{u}(k)$ the output

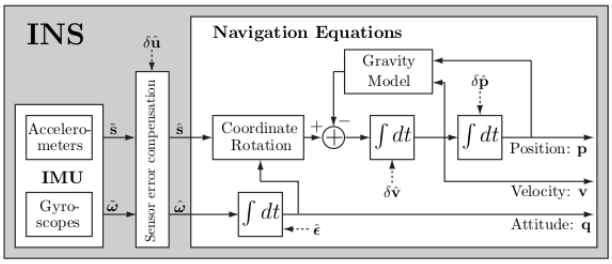

Figure 3: Block Diagram for the Inertial Navigation System.

of the IMU that we will feed in the integration equation (after compensating it for the estimated bias).

We assume the following about the disturbances that affect sensor measurements. Two different disturbances affect the IMU measurements: a slowly varying bias $\delta u(k)$, and an additive white noise $w_1(k)$, with covariance matrix Q_1, as shown in Equation (3). The dynamics of the slowly varying bias $\delta u(k)$ is a *random walk*, shown in equation 4, where $w_2(k)$ is a white noise with covariance matrix Q_2.

$$\tilde{u}(k) = u(k) - \delta u(k) + w_1(k) \tag{3}$$

$$\delta u(k) = \delta u(k-1) + w_2(k) \tag{4}$$

The GPS sensor provides a measurement of the position that only affected by an additive white noise $e(k)$ with covariance R.

$$\tilde{p}_{\text{GPS}}(k) = p(k) + e(k) \tag{5}$$

The Kalman filter measures the discrepancy between the output of the INS and the GPS and uses it to correct the position and biases estimation. The correction is based on the *Kalman gain*, computed online. This technique was proved to be optimal in case of Gaussian disturbances and noises. In fact, the Kalman gain is computed according to the uncertainty of the quantities to be estimated, formally defined as their covariance matrix, usually denoted with $\mathbf{P}$. The covariance matrix is also updated online[4], its evolution is based on the linearized model of the system dynamics (i.e., how the *past* estimation uncertainty propagates to the following one), and on the disturbances and noise models (i.e., their variance).

We now provide details about how the Kalman filter is implemented, and how it corrects the estimates. We would like to obtain an estimate $\hat{x}(k)$ of $x(k)$[5]. We therefore need to estimate the following quantities: the position error $\delta x(k)$ (used to correct the position estimation) and the biases of the sensors $\delta u(k)$ (to compensate the estimate obtained using Equation (1) for future measurements),

$$\delta x(k) = x(k) - \hat{x}(k) = \begin{bmatrix} \delta p(k) \\ \delta v(k) \\ \epsilon(k) \end{bmatrix} = \begin{bmatrix} p(k) - \hat{p}(k) \\ v(k) - \hat{v}(k) \\ \epsilon(k) \end{bmatrix}. \tag{6}$$

Here, the attitude error vector $\epsilon(k) \in \mathbb{R}^3$ is defined as the the small (Euler) angle sequence that rotates the attitude vector $\hat{q}(k)$ into $q(k)$[6]. We can then define the vector collecting all the quantities to be estimated and the vector that collects the disturbances as,

$$\delta z(k) = \begin{bmatrix} \delta x(k) \\ \delta u(k) \end{bmatrix} \in \mathbb{R}^{15}, \quad w(k) = \begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix} \in \mathbb{R}^{12}. \tag{7}$$

The linearized state space model, describing the evolution of $z(k)$, needed to define how the covariance matrix of the estimation changes in time, is

$$z(k) = \mathbf{F}(x(k), u(k)) \cdot z(k-1) + \mathbf{G}(x(k)) \cdot w(k), \tag{8}$$

where the state transition matrix $\mathbf{F}(x(k), u(k))$ and the noise gain matrix $\mathbf{G}(x(k))$ are defined as follows. $\mathbf{F}(x(k), u(k)) =$

$$\begin{bmatrix} I_3 & T_s I_3 & 0_{3,3} & 0_{3,3} & 0_{3,3} \\ 0_{3,3} & I_3 & \left[T_s R_b^n(q(k))\right]_\times & -T_s R_b^n(q(k)) & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & I_3 & 0_{3,3} & -T_s R_b^n(q(k)) \\ 0_{3,3} & 0_{3,3} & 0_{3,3} & I_3 & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & 0_{3,3} & 0_{3,3} & I_3 \end{bmatrix},$$

$$\mathbf{G}(x(k)) = \begin{bmatrix} 0_{3,3} & 0_{3,3} & 0_{3,3} & 0_{3,3} \\ T_s R_b^n(q(k)) & 0_{3,3} & 0_{3,3} & 0_{3,3} \\ 0_{3,3} & T_s R_b^n(q(k)) & 0_{3,3} & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & I_3 & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & 0_{3,3} & I_3 \end{bmatrix}.$$

At every sampling step k, the inertial measurements $\tilde{u}(k)$ are compensated with respect to the estimated biases $\delta u(k)$, according to

$$\tilde{u}(k) \leftarrow \tilde{u}(k) + \delta u(k). \tag{9}$$

The new value of $\tilde{u}$ is then integrated according to Equation (1), obtaining a first estimation $\hat{x}(k)$. The covariance matrix $\mathbf{P}$ as specified in Equation (10), including both the propagation of the uncertainty coming from the previous estimations (first part of the sum) and the contribution of the new disturbances affecting the process (second part of the sum).

$$\mathbf{P} \leftarrow \mathbf{F}(\hat{x}, \hat{u})\mathbf{P}\mathbf{F}^\top(\hat{x}, \hat{u}) + \mathbf{G}(\hat{x}) \begin{bmatrix} Q_1 & 0_{3,3} \\ 0_{3,3} & Q_2 \end{bmatrix} \mathbf{G}^\top(\hat{x}) \tag{10}$$

In absence of GPS measurements, the obtained estimate $\hat{x}$ is used as the position estimate. On the contrary, when GPS measurements are available, the Kalman gain $\mathbf{K}$ is computed according to the following equation,

$$\mathbf{K} \leftarrow \mathbf{P}\left[I_3, 0_{3,3}\right]^\top \left(\left[I_3, 0_{3,3}\right]\mathbf{P}\left[I_3, 0_{3,3}\right]^\top + R\right)^{-1}, \tag{11}$$

which can be interpreted as a weighted sum of: (i) our confidence in the current estimations, defined by $\mathbf{P}$; (ii) how noisy are the new measurement, encoded in the covariance of the additive noise R. This is used to estimate the perturbation state $\hat{z}(k)$:

$$\delta\hat{z}(k) = \begin{bmatrix} 0_{9,1} \\ \delta\hat{u}(k) \end{bmatrix} + \mathbf{K}(\tilde{p}_{GPS}(k) - \hat{p}). \tag{12}$$

Given the availability of the GPS measurements, the quantity $\delta\hat{x}(k)$ is considered zero (i.e., $0_{9,1}$). On the contrary, the sensor bias $\delta\hat{u}(k)$ is propagated and will then be used also when the GPS measurements are not available, in Equation (9).

[4] If the system was linear, the covariance matrix would be updated at every step with the same system dynamics and covariances of disturbances and noises. This would apparently make the evolution of the covariance matrix deterministic. On the contrary, this is not the case here and at every time step the system needs to be newly linearized. As a conclusion, the evolution of the covariance matrix is not deterministic.

[5] In general, we use $\hat{\cdot}$ to indicate the estimate of the variable represented by $\cdot$.

[6] $q(k) = \Gamma(\hat{q}(k), \epsilon(k))$, where $\Gamma(\hat{q}(k), \epsilon(k)) \triangleq \{q \in \mathbb{S}^3 | R_b^n(q(k)) = (I_3 - [\epsilon(k)]_\times) R_b^n(\hat{q}(k))\}$ and $[a]_\times$ defines the antisymmetric matrix representation of a for which $[a]_\times \cdot b = a \cdot b$.

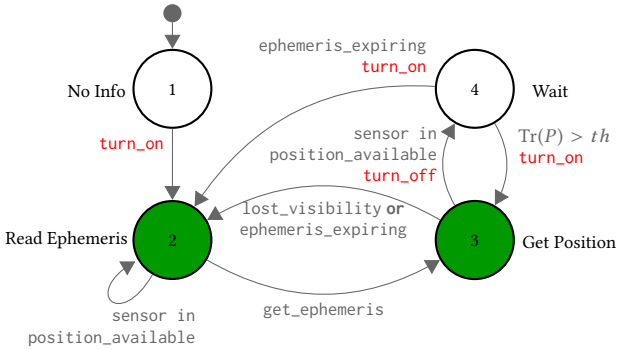

Figure 4: State Machine of the Sampling Strategy Controller.

When the GPS is turned on, the computation of the new error vector allows us to finally update the state estimation as

$$\hat{p} \leftarrow \hat{p} + \delta\hat{p}, \qquad \hat{v} \leftarrow \hat{v} + \delta\hat{v}, \qquad \hat{q} \leftarrow \Gamma(\hat{q}, \hat{\epsilon}). \tag{13}$$

With the sensor fusion algorithm just described, and the sensor model specified in Section 3, we can now describe the dynamical limitations that the GPS sensor imposes and the opportunities available for optimizing its sampling.

5 ANALYSIS AND SAMPLING STRATEGY

This section discusses the general features that characterize an effective sampling strategy and describes the one we advocate using.

5.1 The dynamics

The model presented in Section 3 highlights two dynamics that characterize the sensor. The first one is the availability of the ephemeris data and the second one is the availability of ranging data. The two occur at very different time scales, both in terms of acquisition time and in terms of data validity.

Ephemeris data: The ephemeris data live in the time scale of *minutes*, requiring between 30 and 59 seconds to be acquired and having a validity of 30 minutes from the acquisition. There are two implications of these facts. First, this induces a start-up delay equivalent to the acquisition time of the ephemeris data. This is referred to as *Time To First Fix* (TTFF). Second, an effective sampling strategy refreshes the ephemeris data at least every 30 minutes. This requires the antenna to be turned on for enough time to capture the data and affects the sensor battery consumption.

Ranging data: The ranging data are characterized by a time scale of the order of milliseconds. They require from 2 to 10 milliseconds to be acquired. This could be critical for real-time applications. The data validity is instantaneous, since they are used as soon as they are received to compute the current position (and moving will invalidate them). The time scale allows us to derive a bound in the sensor sampling period. Sampling as frequently as this (lower) bound is equivalent to keeping the sensor always on.

Another important consequence of the sampling policy is the observability of the event `lost_visibility`. The occurrence event is in fact detectable only when the antenna is turned on and the sensor is listening to the visible satellites. When a satellite disappears, if the antenna is turned off, the device cannot detect it. At the next sampling, the receiver needs then to acquire new ephemeris data before being capable to provide positioning information.

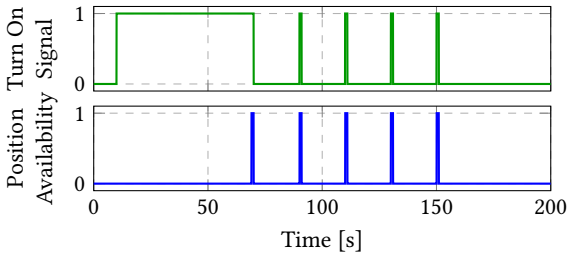

Figure 5: TTFF simulation.

5.2 Sampling Strategy Controller

With these considerations in mind, we designed a sampling strategy that aims at keeping the ephemeris data updated, and sampling the GPS sensor according to the uncertainty of the position estimation (in the Kalman filter). We use the trace of the covariance matrix P, which represents the estimation variance of the position. When the trace overcomes a predefined threshold th, we consider the position too uncertain and request the GPS intervention. This is formally encoded in the state machine shown in Figure 4.

The logical controller sends a `turn_on` signal when the system is starting, to collect the ephemeris data (State ② in Figure 4). Then, once the ephemeris data are available (which is defined by the very same transition of the sensor model), it starts cycling between states ③ and ④, alternatively triggering the `turn_off` and `turn_on` signals. For readability, and consistently with the sensor model shown in Figure 2, the states in which the antenna is turned on are filled in green.

When the ephemeris data are about to expire (intuitively defined as $time > expiry\_time\_ephemeris - 60$), or the sensor loses visibility of the tracked satellites, the controller goes back to State ② and keeps the antenna on, to refresh the ephemeris data. If the former ephemeris data are valid the sensor can actually still be sampled, represented by taking the self-loop transition `sensor in position_avaialable`.

6 EVALUATION

This section describes the results obtained the model proposed in Section 3.2 and two different implementations[7]. The first one is written in Modelica[8], while the second one is written in Matlab[9][10].

The purpose of the Modelica code is to obtain a powerful simulation tool. The nature of Modelica – in terms of composability and object-orientation – makes the implementation easy to embed. We use the Modelica implementation to show how the model captures the relevant dynamics of the GPS receivers, comparing the results obtained with data obtained on real hardware. The implementation in Matlab is used for data analysis of real GPS traces and in combination with sensor fusion algorithm as described in Section 4, to experiment with the accuracy-battery trade-off.

6.1 GPS sensor dynamics

In this section we want to show how our model captures the dynamics of the receiver. We start with the Time To First Fix (TTFF),

[7]The code for both the implementation will be released in case the paper is accepted.
[8]http://www.modelica.org
[9]http://www.mathworks.com/products/matlab.html
[10]The code and the data used for the simulations are available at: https://gitlab.control.lth.se/mmaggio/gps-modeling/

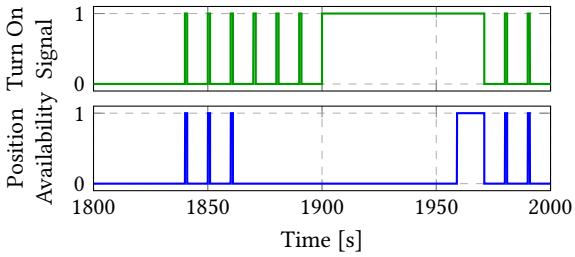

Figure 6: Ephemeris expiration simulation.

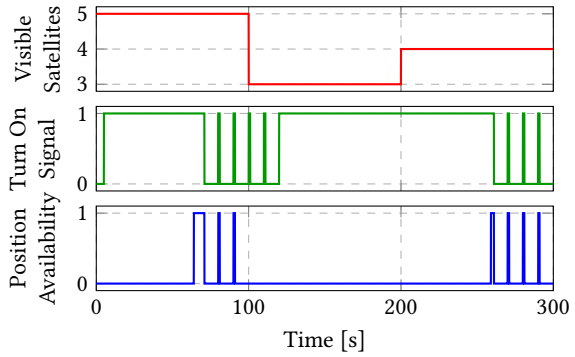

Figure 7: Loss Visibility Simulation.

the loss of ephemeris data, and the loss of visibility. We use our Modelica model to obtain simulation data.

Figure 5 shows the results of a simulation aimed at determining the TTFF. At time $t = 10s$ the GPS receiver is turned on, and we simulate how it fetches the position. The top row shows the command signal. The GPS antenna has to be on for a minute before the position becomes available. The position availability is shown in the second row of the figure. As soon as the ranging data are received, a position reading is taken. Then the sensor can be sampled at regular intervals.

As anticipated in Section 2, previous work discuss the TTFF, e.g., [9, 10, 12]. However, no contribution combines the TTFF with a model of the GPS technology. In previous literature, the performances of GPS receivers in smartphones are evaluated by looking at how much time it takes for a generic application to get a position measure after the request is sent to the GPS module. GPS sensors in smartphones implement an *Assisted-GPS* function, that allows the retrieval of the ephemeris data from the internet instead of listening to the satellites[11].

Up to the (limited) extent to which they are comparable – due to the fact that are used Assisted-GPS sensors and the extra software layers included in the experiments – the results presented in these works are coherent with the model presented here. Our model on the other side achieves more generality, not being dependent on the specific implementation and on the given device. Moreover, our model allows one to look directly at what are the theoretical performances one can expect from a GPS sensor, without the overhead that is introduced by the operative system of a smartphone.

Figure 6 shows how our model captures the expiration of the ephemeris data. The figure is organized as the previous one, showing the turn on signal in the top row and the position availability in the bottom row. It displays a duty cycling of the sensor for quickly acquiring position – i.e. a series of warm starts. At time $t = 1861s$ the ephemeris data expire, making the duty cycling ineffective. The sensor detects the loss of ephemeris data after a few failed attempts and turns on the antenna. When the device receives an update of the ephemeris data, the position becomes available again as well as the possibility of performing warm starts.

Figure 7 shows how the model captures the loss of visibility of the satellites. The top row shows the number of visible satellites, while the second and third row respectively depict the turn on signal and the position availability. As an input to the model, we provide 5 visible satellites in the interval between $t = 0s$ and $t = 100s$, 3 visible satellites in the interval between $t = 100s$ and $t = 200s$ and 4 satellites from $t = 200s$ to $t = 300s$. During the first 100 seconds, the GPS receiver is able to acquire the ephemeris data. When 2 satellites disappear from the visible sky, the 3 visible ones are not sufficient for the device to position itself. The position then stops being available, despite the duty cycling. Then the position becomes available again when a new satellite appears in the visible sky, and after the device acquires its ephemeris data.

This last simulation points out both the realism and one limitation of the proposed model. Theoretically, if the satellite that appears again after the loss of visibility was one of the two that previously disappeared, the device would not need to re-acquire the ephemeris data (provided that it kept in memory the ones acquired at the start up and that those were still valid). To capture this *phenomenon*, we would need to describe separately the acquisition of the signal and ephemeris data of the different satellites, together with their visibility. Apparently, this would increase the complexity of the model and decrease it usability. An extension of the model to include also this phenomenon would not be very difficult to obtain. It is enough to have parallel state machines similar to the one shown in Figure 2, that independently capture the tracking of individual satellites but are synchronized in the antenna's state.

6.2 Positioning Accuracy

Here we use real traces, recorded from a GPS receiver and an IMU sensor that includes accelerometer and gyroscope. We recorded data in two different conditions: a car and a bicycle ride. We recorded traces with continuous GPS sampling and simulated different GPS sensor dynamics on top of that, to compare different sampling policies. We show what the tracking would have been when the sensor fusion algorithm was live, compared to the continuous sampling of the GPS. We then use simulations to further analyze the trade-off between power (and therefore battery) consumption and performance (positioning accuracy). The sampling rate of the used traces is $1Hz$ but the localization information is of course made available only when the model is in the correct state.

Figures 8 and 9 respectively show traces for the tracking of the bike and the car. In each figure, the GPS trace is represented using solid blue lines, while two different executions of the sensor fusion

[11] The model presented in this paper can be adapted to represent this allowing for an external input that possibly triggers the transition `get_ephemeris` before the delay that instead represents the action of listening to the satellites. The modeling of how much time the device requires for fetching this information from the internet and how much power this procedure can take is non trivial because of the many different (and often difficult to predict) involved components.

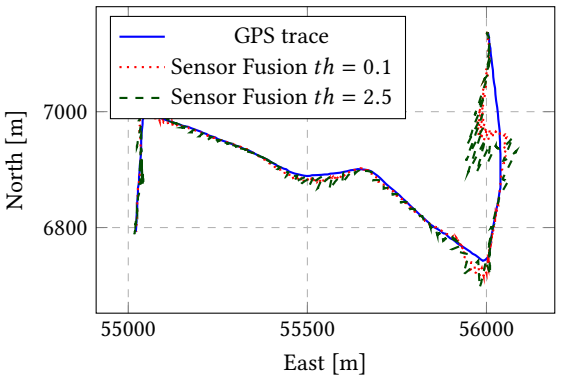

Figure 8: Trace tracked when cycling. GPS measurements and sensor fusion algorithm with different thresholds.

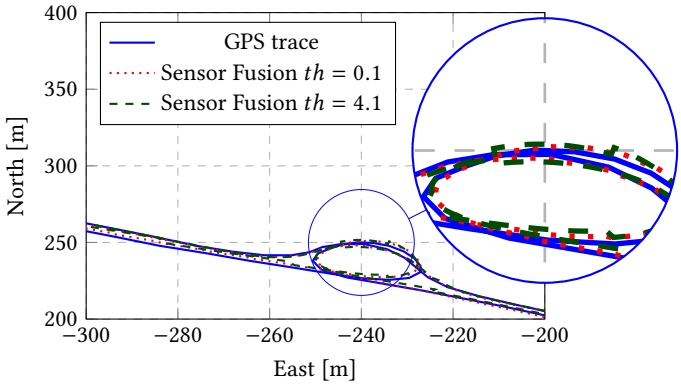

Figure 9: Trace tracked when driving. GPS measurements and sensor fusion algorithm with different thresholds.

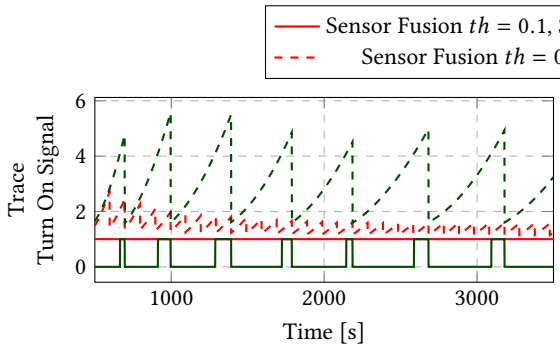

Figure 10: Control Signal and Sampling when cycling.

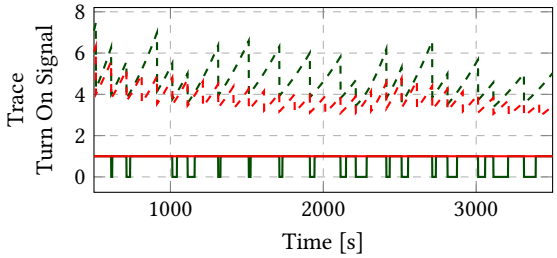

Figure 11: Control Signal and Sampling when driving.

algorithm (with different values of the threshold th) are shown in red dotted lines and green dashed lines. The red dotted lines correspond to the use of a threshold $th = 0.1$, which in turn means leaving the GPS receiver always on, but using the sensor fusion algorithm to incorporate also the IMU data. In general, the cycling trace exposes more complex dynamics, that are harder to track for the sensor fusion algorithm. Low frequency GPS sampling can still guarantee some form of tracking, depending on the precision needed for the given application. In the car trace, the movements of the GPS receiver are reduced and the IMU sensors are able to much better allow for low frequency sampling of the GPS signal. Notice that our implementation of the sensor fusion algorithm is very basic and more advanced versions could improve the tracking performance also in the biking case. For the same simulations, we also show in Figures 10 and 11 the signal used for the GPS triggering (i.e., the trace of P) and the state of the GPS antenna. The figures show how the sampling strategy is able to keep the variance of the estimation bounded, while reducing the on time.

6.3 Tracking the Trade-Off between Performance and Power Consumption

Finally, we would like to expose the trade-off between performance (accuracy) and power (battery) consumption. For both the driving and the biking scenario, we run 30 simulations with different triggering thresholds for the sensor fusion algorithm (specifically, we use $th \in \{4, 5, 6, 8, 10, 12\}$). In principle, higher values of the threshold guarantee a lower power consumption, at the price of larger position estimate errors (and *viceversa*).

The acquisition of satellites signals is modeled as a random variable, uniformly distributed in the intervals specified for the model. In the biking simulation, the number of visible satellites is a constant. In the driving simulation (more realistically), the value of the visible satellites is also a random variable. More in detail, for each time step in the simulation, there is a probability of increasing or decreasing the number of visible satellites (in a realistic bound between 3 and 6). The overall error of a trace is defined as the root-mean-square of the distance between the trace and the pure GPS signal.

Figure 12 shows the simulation results for the cycling tracking. Different colors for the marks correspond to different threshold values. The figure highlights the trade-off and show that it is controllable using the choice of the threshold. Furthermore two other interesting phenomena are pointed out by this simulation. First, for low triggering values (blue, red, and green points) the variance of the error is lower, since the simulation is converging to always-on scenario, saturating the achievable tracking precision. At the same time, the energy consumption variance increases due to frequent changes in sensor state (with the receiver being affected by the time it takes to fetch satellites' signals). Second, higher triggering values instead (purple, yellow, and orange points) the opposite behavior is experienced. The variance in terms of energy consumption is smaller, since the antenna is turned on less frequently (implying

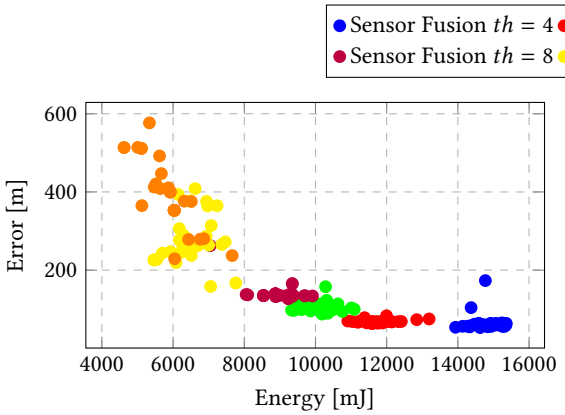

Figure 12: Trade-Off between Energy and Error when cycling.

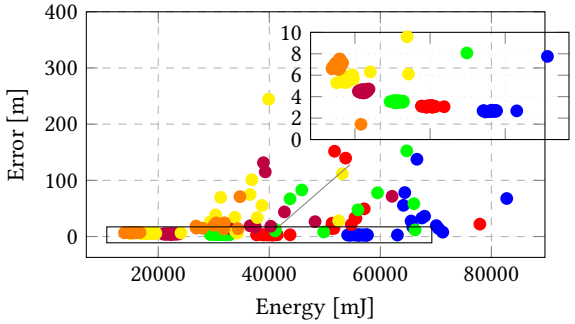

Figure 13: Trade-Off between Energy and Error when driving (with signal loss).

that there will be less uncertainty on the time spent fetching signals). The error here becomes lager and with higher variance, due to the need to exploit more often (less reliable) IMU data.

Finally, Figure 13 shows the simulations performed using the car data. Given the introduction of a varying number of visible satellites and therefore the possibility of losing GPS availability, the behavior of the system becomes much more diverse. Specifically, points spread radially away from the origin. This is reasonable, since the loss of visibility will negatively affect both the accuracy (as the GPS data wont be available until a sufficient number of satellites become visible again) and the energy consumption (as the sensor will have to be turned on for relatively long time to reacquire the ephemeris data). Still, the same behavior is detectable looking only at the simulations where no satellite visibility loss event happens.

7 CONCLUSION

This paper presented a first-principle model of a GPS receiver, able to capture the dynamics of the data acquisition. We used the model to enhance a sensor fusion algorithm that merges data from inertial measurement sensors with the GPS trace, to improve the GPS battery consumption. The paper presented the model, simulation results, and experiments obtained with real GPS traces, and exposes the trade-off between Energy consumption and Positioning Accuracy.

Acknowledgements:
We would like to thank Isaac Skog (Linköping University) that allowed us to reuse and build on his code for the Matlab model. This work was partially supported by the ELLIIT Strategic Research Area, by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, and by the Nordforsk Nordic Hub on Industrial IoT (HI2OT).

REFERENCES

[1] Fehmi Ben Abdesslem, Andrew Phillips, and Tristan Henderson. 2009. Less is More: Energy-efficient Mobile Sensing with Senseless. In *Proceedings of the 1st ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds.*
[2] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox. 2009. EnLoc: Energy-Efficient Localization for Mobile Phones. In *IEEE INFOCOM 2009.*
[3] Gregory Dudek and Michael Jenkin. 2016. Inertial Sensing, GPS and Odometry. In *Springer Handbook of Robotics.*
[4] Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. 2007. *Global Positioning Systems, Inertial Navigation, and Integration.*
[5] Haitham Hassanieh, Fadel Adib, Dina Katabi, and Piotr Indyk. 2012. Faster GPS via the Sparse Fourier Transform. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking.*
[6] WPMH Heemels, Karl Henrik Johansson, and Paulo Tabuada. 2012. An introduction to event-triggered and self-triggered control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on.* IEEE.
[7] Rudolph E. Kalman. 1960. A New Approach to Linear Filtering And Prediction Problems. *ASME Journal of Basic Engineering* (1960).
[8] E.D. Kaplan. 1996. *Understanding GPS: Principles and Applications.*
[9] Mikkel Baun Kjærgaard, Jakob Langdal, Torben Godsk, and Thomas Toftkjær. 2009. EnTracked: Energy-efficient Robust Position Tracking for Mobile Devices. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services.*
[10] Xiaohan Li, Fengpeng Yuan, and J. Lindqvist. 2016. Feasibility of software-based duty cycling of GPS for trajectory-based services. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC).*
[11] Martin E. Liggins, James Llinas, and David L. Hall. 2008. *Multisensor Data Fusion.*
[12] Kaisen Lin, Aman Kansal, Dimitrios Lymberopoulos, and Feng Zhao. 2010. Energy-accuracy Trade-off for Continuous Mobile Device Location. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services.*
[13] P. Misra, W. Hu, Y. Jin, J. Liu, A. S. de Paula, N. WirstrÃűm, and T. Voigt. 2014. Energy efficient GPS acquisition with Sparse-GPS. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks.*
[14] D. Orn, M. Szilassy, B. Dil, and F. Gustafsson. 2016. A novel multi-step algorithm for low-energy positioning using GPS. In *2016 19th International Conference on Information Fusion (FUSION).*
[15] Thomas Olutoyin Oshin, Stefan Poslad, and Athen Ma. 2012. Improving the Energy-Efficiency of GPS Based Location Sensing Smartphone Applications. *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications* (2012).
[16] Heitor S. Ramos, Tao Zhang, Jie Liu, Nissanka B. Priyantha, and Aman Kansal. 2011. LEAP: A Low Energy Assisted GPS for Trajectory-based Services. In *Proceedings of the 13th International Conference on Ubiquitous Computing.*
[17] I. Skog and P. Handel. 2009. In-Car Positioning and Navigation Technologies?A Survey. *IEEE Tran. on Intelligent Transportation Systems* 10, 1 (March 2009).
[18] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. 2009. VTrack: Accurate, Energy-aware Road Traffic Delay Estimation Using Mobile Phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems.*
[19] Sravan Kumar Thokala, Pranav Koundinyaa, Shivakant Mishra, and Larry Shi. 2014. Virtual GPS: A Middleware for Power Efficient Localization of Smartphones Using Cross Layer Approach. In *Proceedings of the Middleware Industry Track.* Article 2.
[20] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A. Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh. 2009. A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services.*
[21] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. 2010. Improving Energy Efficiency of Location Sensing on Smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services.*