Module 1 - Sensing and Perception:
Sensor Fusion GPS+IMU
Isaac Skog 2016

with modifications by Bo Bernhardsson 2018

Sensor Fusion GPS+IMU

In this assignment you will study an inertial navigation system (INS) constructed using sensor fusion by a Kalman filter. The start code provides you with a working system with an inertial measurement unit (IMU, here accelerometer+gyro) and GNSS (GPS). You will

- evaluate the effects of GPS signal outage on the navigation solution
- implement an improved vehicle motion model (”car does not skid or fly”)
- include measurements from a speedometer to the navigation fusion filter.

and study the improved performance during GPS signal outage.

Download from Canvas the file `GNSSaidedINS.zip` to a folder where matlab can be run. The folder contains Matlab files that implement a GNSS-aided Inertial Navigation System (INS) and a data set with GPS, IMU, and speedometer data. To run the GNSS-aided INS execute the file `main.m`.

The data is from a car driving in local traffic for about 5 minutes, see Figure 1 for a plot of the GPS-signal. Notice the round-about in the middle of the track. We do not have ground-truth from the test, but the GPS signal seems reasonable during the entire test. We will simulate loss of the GPS signal occuring at 200 seconds and onwards, covering a period where the car traverses the roundabout for the second time, the part is marked in red. The data rates of the sensors are: GPS (1Hz), IMU (100Hz) and speedometer (4Hz), though some glitches exist in the dataset.

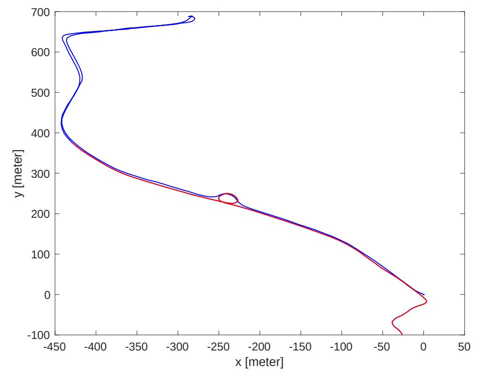

Figure 1: GPS signal from the experiment. The car starts in the origin, travels northwest and turns around back towards the start. We will simulate a GPS outage during the red part of the trajectory, when $t > 200$ sec).

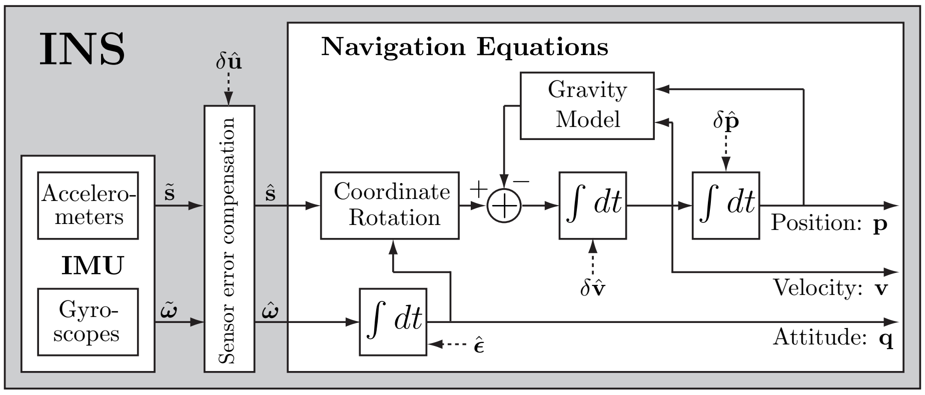

Figure 2: Conceptional sketch of an INS. The dashed arrows indicate possible points for insertion of correction (calibration) data.

Next a short description of the GNSS-aided INS system implemented in the Matlab code, is presented. If interested in a more in-depth description of GNSS-aided INS the book [1] provides an excellent introduction to the topic. If you have any questions about the Matlab code or the project in general contact `isaac.skog@liu.se` or `bob@control.lth.se`.

GNSS aided INS – A short introduction

Inertial navigation is a self-contained navigation technique in which measurements provided by accelerometers and gyroscopes are used to track the position, velocity, and attitude (orientation) of an object relative to a known start position, velocity, and attitude. Since an INS is self-contained, i.e., it does not rely on any external information sources that can be disturbed or jammed, it is an attractive means of navigation for many applications where 100% coverage and a high continuity-of-service is needed. Further, it also provides a full 6 degree-of-freedom navigation solution and generally has a high update rate ($\geq$ 100 Hz); properties of high importance in the design of various autonomous systems. In Fig. 2 a block diagram of an INS for navigation in a geographically limited area using low-cost inertial sensors (accelerometers and gyroscopes), is shown. Mathematically, the navigation process of an INS can be described as follows. Define the navigation state vector and inertial measurement input vector as

$$\mathbf{x}_k \triangleq \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \\ \mathbf{q}_k \end{bmatrix} \in R^{10}, \quad \text{and} \quad \mathbf{u}_k \triangleq \begin{bmatrix} \mathbf{s}_k \\ \omega_k \end{bmatrix} \in R^6, \tag{1}$$

respectively. Here $\mathbf{p}_k \in R^3$ $[m]$, $\mathbf{v_k} \in R^3$ $[m/s]$, and $\mathbf{q}_k \in R^4$ $[-]$ denote the position, velocity, and attitude (quaternion representation) of the navigation system at time instant k. Further, $\mathbf{s}_k \in R^3$ $[m/s^2]$ and $\boldsymbol{\omega}_k \in R^3$ $[rad/s]$ denote the specific force[1] and angular velocity, respectively. A discrete time version of

[1]The output of an accelerometer is the sum of the inertial and gravitational acceleration, which is referred to as specific force.

the inertial navigation equations in Fig. 2 is given by the nonlinear differential equation

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k), \tag{2}$$

where

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_{k-1} + T_s \mathbf{v}_{k-1} + \frac{T_s^2}{2}\Big(\mathbf{R}_b^n(\mathbf{q}_{k-1})\mathbf{s}_k - \mathbf{g}\Big) && (3) \\ \mathbf{v}_k &= \mathbf{v}_{k-1} + T_s\Big(\mathbf{R}_b^n(\mathbf{q}_{k-1})\mathbf{s}_k - \mathbf{g}\Big) && (4) \\ \mathbf{q}_k &= \Big(\cos(0.5\,T_s\,\|\boldsymbol{\omega}_k\|)\mathbf{I}_4 + \frac{1}{\|\boldsymbol{\omega}_k\|}\sin(0.5\,T_s\,\|\boldsymbol{\omega}_k\|)\boldsymbol{\Omega}_k\Big)\mathbf{q}_{k-1} && (5) \end{aligned}$$

and

$$\boldsymbol{\Omega}_k = \begin{bmatrix} 0 & [\boldsymbol{\omega}_k]_z & -[\boldsymbol{\omega}_k]_y & [\boldsymbol{\omega}_k]_x \\ -[\boldsymbol{\omega}_k]_z & 0 & [\boldsymbol{\omega}_k]_x & [\boldsymbol{\omega}_k]_y \\ [\boldsymbol{\omega}_k]_y & -[\boldsymbol{\omega}_k]_x & 0 & [\boldsymbol{\omega}_k]_z \\ -[\boldsymbol{\omega}_k]_x & -[\boldsymbol{\omega}_k]_y & -[\boldsymbol{\omega}_k]_z & 0 \end{bmatrix}. \tag{6}$$

Here, T_s denotes the sampling period of the data and $\mathbf{R}_b^n(\mathbf{q})$ denotes the directional cosine matrix (rotation matrix) that rotates a vector from the body coordinate frame (b-frame) to the navigation coordinate frame (n-frame). Further, $\mathbf{g}$ denotes the gravity vector expressed in the navigation coordinate system. Noteworthy is that the attitude in the mechanized INS equations is described and propagated using the quaternion vector $\mathbf{q}$. It is possible to use other attitude representations such as Euler angles or directional cosine matrices, but the quaternion representation is generally more numerically stable.

The function `Nav_eq.m` in the folder with the Matlab files implements these navigation equations. The folder also includes several functions for converting between different attitude representations such as Euler angles and directional cosine matrices.

Neglecting the discretization and quantization errors introduced in the mechanization of the navigation equations, the inertial navigation system will perfectly track the position, velocity, and attitude of the navigation platform as long as the specific force and angular velocity is measured without errors. In reality there exist no error free measurements and due to the integrative nature of the inertial navigation equations, the measurement errors will be accumulated and cause the position and velocity errors in the navigation solution to grow without bound. As a rule of thumb, for an INS using low-cost sensors, the position error grows cubically with time and is proportional to the magnitude of the bias in the gyroscope measurements.

TASK #1: Use the functions `errorgrowth.m` and `Nav_eq.m` to evaluate how the position error grows with time. Assume that the navigation system is stationary; the initial position and velocity is zero; the initial roll, pitch, and yaw are zero; the accelerometer measurements are error free; and the gyroscope measurements are error free except from a bias in the x-axis gyroscope with a magnitude of 0.01 $^\circ/s$. Explain the observed behavior: In what directions do you get a position error? Find an approximative formula for the error behavior. Where is error largest, Stockholm or Lund?

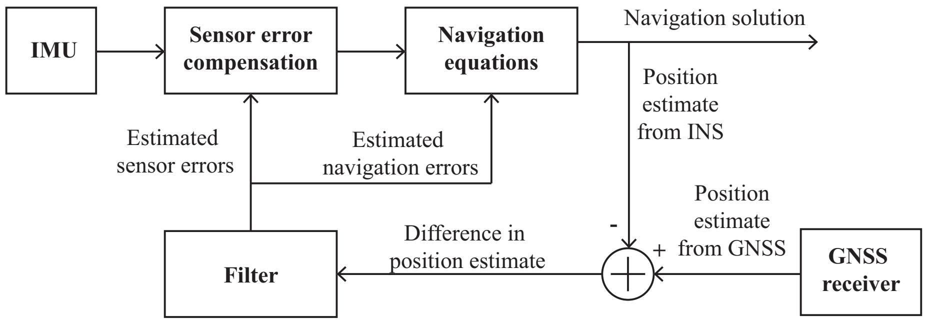
IMU
Sensor error compensation
Navigation equations
Navigation solution
Position estimate from INS
Estimated sensor errors
Estimated navigation errors
Position estimate from GNSS
-
+
Difference in position estimate
Filter
GNSS receiver

Figure 3: Conceptional sketch of a loosely-coupled closed-loop GNSS-aided INS.

From your simulation it should be clear that an INS that uses low-cost inertial sensors only can be used for stand alone navigation for a very limited time period. To overcome this problem, the navigation information provided by a low-cost INS is frequently fused with the navigation information provided by a GNSS-receiver. If done properly the result is a navigation system that: (1) has an position accuracy as a GNSS-receiver; (2) provides a full 6 degrees-of-freedom navigation solution; (3) has a high update rate; (4) for shorter time periods can provide a navigation solution even during GNSS-receiver outages; and (5) has a higher system integrity as faulty GNSS-measurements can be detected and rejected.

In Fig. 3 one way of fusing the information between an INS and a GNSS-receiver is shown. The displayed information fusion strategy is referred to as a loosely-coupled[2] closed-loop GNSS-aided INS configuration. The main idea is that the INS is used as the backbone in the navigation system and constantly provides a 6 degree-of-freedom navigation solution. Whenever the GNSS-receiver produces a position estimate the difference between the position estimates of the two systems is calculated and used as the input for a filter that tries to estimates the errors of the INS navigation solution, as well as the errors in the IMU sensors. The estimated errors are then used to correct the navigation solution and to calibrate the sensors. A short discussion about various GNSS-aided INS information fusion strategies and their pros and cons can be found in [2].

Many different filter algorithms can be used to estimate the errors in the navigation solution. In the GNSS-aided INS implemented in the Matlab code, a standard Kalman filter algorithm is used. The Kalman filter uses the following linear state space model to describe the error dynamics of the INS as well as dynamics of the inertial sensor errors. Let the IMU measurements be described by the signal model

$$\tilde{\mathbf{u}}_k = \mathbf{u}_k - \delta\mathbf{u}_k + \mathbf{w}_k^{(1)} \tag{7}$$

where $\mathbf{w}_k^{(1)}$ denotes the measurement noise which is assumed to be additive white noise with covariance matrix $\mathbf{Q}^{(1)}$, and $\delta\mathbf{u}_k$ denotes the slowly varying

[2]The term *loosely-coupled* refers to that the position solutions provided by the GNSS-receiver, and not the pseudo range measurements, are used as measurements in the information fusion.

measurement bias which is modeled as the random walk process

$$\delta\mathbf{u}_k = \delta\mathbf{u}_{k-1} + \mathbf{w}_k^{(2)}. \tag{8}$$

Here, $\mathbf{w}_k^{(2)}$ denotes the random walk process noise which is assumed to be additive white noise with covariance matrix $\mathbf{Q}^{(2)}$; the random walk process noise and the measurement noise are assumed uncorrelated.

Next, represent the perturbation between true navigation state $\mathbf{x}_k$ and the INS estimated $\hat{\mathbf{x}}_k$ as

$$\delta\mathbf{x}_k \triangleq \begin{bmatrix} \delta\mathbf{p}_k \\ \delta\mathbf{v}_k \\ \boldsymbol{\epsilon}_k \end{bmatrix} \in R^9, \tag{9}$$

where the position perturbation (error) $\delta\mathbf{p}_k$ and velocity perturbation (error) $\delta\mathbf{v}_k$ are defined as $\delta\mathbf{p}_k = \mathbf{p}_k - \hat{\mathbf{p}}_k$ and $\delta\mathbf{v}_k = \mathbf{v}_k - \hat{\mathbf{v}}_k$, respectively. The attitude perturbation vector $\boldsymbol{\epsilon}_k$ is defined as the small (Euler) angle sequence that rotates the attitude vector $\hat{\mathbf{q}}_k$ into $\mathbf{q}_k$. That is, $\mathbf{q}_k = \Gamma(\hat{\mathbf{q}}_k, \boldsymbol{\epsilon}_k)$, where the implicit function Γ is defined as

$$\Gamma(\hat{\mathbf{q}}, \boldsymbol{\epsilon}) \triangleq \{\mathbf{q} \in \mathbb{S}^3 \mid \mathbf{R}_b^n(\mathbf{q}) = (\mathbf{I}_3 - [\boldsymbol{\epsilon}]_\times)\mathbf{R}_b^n(\hat{\mathbf{q}})\}. \tag{10}$$

Here, $[\mathbf{a}]_\times$ denotes the skew-symmetric matrix representation of $\mathbf{a}$, for which $[\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b}$.

With the perturbation vector defined as in (9), the error propagation in the navigation equations (2) when fed with the measurement vector $\tilde{\mathbf{u}}_k$ can, for small error perturbations, be described by the linear state space model

$$\mathbf{z}_k = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k)\mathbf{z}_{k-1} + \mathbf{G}_k(\mathbf{x}_k)\mathbf{w}_k, \tag{11}$$

where

$$\mathbf{z}_k \triangleq \begin{bmatrix} \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix} \in R^{15}, \quad \mathbf{w}_k \triangleq \begin{bmatrix} \mathbf{w}_k^{(1)} \\ \mathbf{w}_k^{(2)} \end{bmatrix} \in R^{12} \tag{12}$$

and the state transition matrix and noise gain matrix are defined as

$$\mathbf{F}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \mathbf{I}_3 & T_s\mathbf{I}_3 & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{I}_3 & T_s[\mathbf{R}_b^n(\mathbf{q}_k)\mathbf{s}_k]_\times & T_s\mathbf{R}_b^n(\mathbf{q}_k) & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{I}_3 & \mathbf{0}_{3,3} & -T_s\mathbf{R}_b^n(\mathbf{q}_k) \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{I}_3 & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{I}_3 \end{bmatrix} \tag{13}$$

and

$$\mathbf{G}_k(\mathbf{x}_k) = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ T_s\mathbf{R}_b^n(\mathbf{q}_k) & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & T_s\mathbf{R}_b^n(\mathbf{q}_k) & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{I}_3 & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{I}_3 \end{bmatrix}, \tag{14}$$

respectively. Next, the GNSS-receiver position measurements are modelled as

$$\tilde{\mathbf{p}}_k^{\text{GPS}} = \mathbf{p}_k + \mathbf{e}_k^{(1)} \tag{15}$$

where $\mathbf{e}_k^{(1)}$ is additive white noise with covariance matrix $\mathbf{R}^{(1)}$. The observation equation for the INS error propagation state space model can be written as

$$\mathbf{y}_k^{(1)} \triangleq \tilde{\mathbf{p}}_k^{\text{GPS}} - \hat{\mathbf{p}}_k = \mathbf{H}^{(1)}\mathbf{z}_k + \mathbf{e}_k^{(1)}, \qquad \mathbf{H}^{(1)} \triangleq \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3,12}\end{bmatrix} \tag{16}$$

With the state space model as defined by (11) and (16) the Kalman filter based algorithm for realizing the loosely-coupled closed-loop GNSS-aided INS is given in Algorithm 1 (See last page.).

Noteworthy is that the Kalman filter time update does not include any propagation of an estimate of the full state vector $\hat{\mathbf{z}}_k$ (instead $\hat{\mathbf{x}}_k$ and $\delta\hat{\mathbf{u}}_k$ are propagated). The reason for this is that after the estimated navigation state perturbations $\delta\hat{\mathbf{x}}$ and sensor biases $\delta\hat{\mathbf{u}}$ have been fed back into the INS and used to correct the navigation solution, the errors $\delta\hat{\mathbf{x}}$ are considered to equal zero.

TASK #2: Familiarize yourself with the Matlab code that implements the GNSS-aided INS and execute the script `main.m`. Modify the code to simulate a GNSS-receiver outage from 200 seconds and onward. Plot the estimated car position $\hat{x}_k$ (`output_data.x_h`). Also plot the difference between estimated car position and the GPS position without outage. Experiment with varying settings of filter parameters in `get_settings.m` and try to improve the filter performance during GPS outage.

Motion model:
Even though the GNSS-aided INS estimates the sensor biases the position error growth during the GNSS-signal outages is substantial and the navigation solution soon becomes useless. One way to reduce the error growth rate is to include a model of the vehicle's motion dynamics into the information fusion process. There are many different ways to include motion dynamics models into the information process. One way is to define a set of motion constraints and enforce these constraints on the navigation solution using so called constrained filtering. See [4] for an introduction to different constraint filtering techniques.

The motion of a wheeled vehicle on a surface is governed by two non-holonomic constraints. During ideal driving conditions, a vehicle experiences no side slip and no motion in the direction normal to the road surface, and the velocity in the y-axis and z-axis direction of the vehicle coordinate frame (p-frame) should therefore be equal to zero, i.e.

$$\mathbf{0}_{2,1} = \mathbf{A}\mathbf{R}_n^p\mathbf{v}, \qquad \mathbf{A} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \tag{17}$$

where $\mathbf{R}_n^p$ denotes the transformation from the navigation frame (n-frame) to the vehicle coordinate frame (p-frame). In reality the velocity in the y-axis and z-axis direction of the vehicle coordinate frame will not be perfectly zero and these constraints have to be relaxed. Modeling the error as a white noise term $\mathbf{e}^{(3)}$ the pseudo-observation can be written[3]

$$\mathbf{y}_k^{(3)} \triangleq \mathbf{0}_{2,1} - \mathbf{A}\mathbf{R}_n^p\hat{\mathbf{v}}_k = \mathbf{H}^{(3)}\mathbf{z}_k + \mathbf{e}_k^{(3)}, \quad \mathbf{H}^{(3)} \triangleq \begin{bmatrix}\mathbf{0}_{2,3} & \mathbf{A}\mathbf{R}_n^p & \mathbf{0}_{2,9}\end{bmatrix}. \tag{18}$$

[3]this numbering is chosen to give a cleaner result later

Speedometer sensor:
Another way to improve the navigation performance during GNSS signal outages is to include additional sensors. A commonly used sensor in vehicle applications is a speedometer. If a speedometer is available, the measurements $\tilde{v}_k^{\text{SPEED}}$ from this sensor can be included in the Kalman filter framework by introducing the observation equation

$$y_k^{(2)} \triangleq \tilde{v}_k^{\text{SPEED}} - \mathbf{B}\mathbf{R}_n^p \hat{\mathbf{v}}_k^n = \mathbf{H}^{(2)}\mathbf{z}_k + e_k^{(2)}, \tag{19}$$

where $e_k^{(2)}$ denotes the measurement noise and

$$\mathbf{H}^{(2)} \triangleq \begin{bmatrix} \mathbf{0}_{1,3} & \mathbf{B}\mathbf{R}_n^p & \mathbf{0}_{1,9} \end{bmatrix}, \quad \mathbf{B} \triangleq \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \tag{20}$$

The relaxed speedometer measurement and non-holonomic constraint can be written as the joint measurement equation

$$\begin{bmatrix} \mathbf{y}_k^{(2)} \\ \mathbf{y}_k^{(3)} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{R}_n^p & \mathbf{0}_{3,9} \end{bmatrix} z_k + \begin{bmatrix} e_k^{(2)} \\ e_k^{(3)} \end{bmatrix}. \tag{21}$$

The noise $e_k^{(2)}$ and $e_k^{(3)}$ are modeled as Gaussian, with covariance $R^{(2)}$ and $R^{(3)}$ respectively (=(`settings.speed`)^2 and (`settings.sigma_non_holonomic`)^2).

Hint for implementation: If not all observations in

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{y}_k^{(1)} \\ \mathbf{y}_k^{(2)} \\ \mathbf{y}_k^{(3)} \end{bmatrix} = \mathbf{H}\mathbf{z}_k + e_k, \qquad \mathbf{H} \triangleq \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3,3} & \mathbf{0}_{3,9} \\ \mathbf{0}_{3,3} & \mathbf{R}_n^p & \mathbf{0}_{3,9} \end{bmatrix} \tag{22}$$

are available at a certain time-instance, the corresponding rows in $\mathbf{y}_k$ and $\mathbf{H}$ can simply be deleted. Note also that $\mathbf{y}_k$ can be calculated as

$$\mathbf{y}_k = \begin{bmatrix} \tilde{\mathbf{p}}_k^{\text{GPS}} \\ \tilde{v}_k^{\text{SPEED}} \\ \mathbf{0}_{2,1} \end{bmatrix} - \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{R}_n^p \end{bmatrix} \begin{bmatrix} \hat{\mathbf{p}}_k \\ \hat{\mathbf{v}}_k \end{bmatrix} \tag{23}$$

TASK #3: Implement support for non-holonomic motion constraints and speedometer measurements into the GNSS-aided INS framework. Some hints where changes are needed are given in the code. Turn off speedometer measurements and run with only the motion constraints (`settings.speed_aiding='off';settings.non_holonomic='on'`). Tune the filter by changing the magnitude of filter parameters, for example $R^{(3)}$, until you get a decent performance and plot the position estimates and position error, with GNSS outage. You should see a substantial improvement.

TASK #4: The data included in the `GNSSaidedINS.zip` folder also include measurements from a speedometer. Let the fusion filter also use these measurements, trim the filter, and then plot the position estimates and position error, with GNSS outage. What is the resulting rms error of the position trajectory?

Project report

Describe your results in a report and submit in Canvas before the deadline. You do not need to submit any Matlab code, unless you for some reason want to.

References

[1] J. A. Farrell and M. Barth, ''The Global Positioning System and Inertial Navigation: Theory and Practice", New York: McGraw-Hill, 370 pp, 1999.

[2] I. Skog and P. Handel, ''In-car positioning and navigation technologies – A survey, *IEEE Transactions on Intelligient Transportation Systems* , vol.10, no.1, pp.4-21, 2009

[3] T. Kailath, A. Sayed, and B. Hassibi, ''Linear Estimation", Prentice Hall, 2000.

[4] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms", *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303-1318, Aug. 2010.

[5] G. Dissanayake, S. Sukkarieh, E. Nebot and H. Durrant-Whyte, "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*, vol. 17, no. 5, pp. 731-747, Oct 2001.

Algorithm 1 GNSS-aided INS algorithm.

```
1:  $k = 1$
2:  $\hat{\mathbf{x}} \leftarrow$ **Process**{ Initial navigation state }
3:  $\delta\hat{\mathbf{u}} \leftarrow$ **Process**{ Initial sensor bias state estimate }
4:  $\mathbf{P} \leftarrow$ **Process**{ Initial Kalman filter state covariance matrix }
5:  loop
6:    % Calibrate the sensor measurements using current sensor
      bias estimate.
7:    $\hat{\mathbf{u}} \leftarrow \tilde{\mathbf{u}}_k + \delta\hat{\mathbf{u}}$
8:
9:    % Update the INS navigation state.
10:   $\hat{\mathbf{x}} \leftarrow f(\hat{\mathbf{x}}, \hat{\mathbf{u}})$
11:
12:   % Time update of the Kalman filter state covariance.
13:   $\mathbf{P} \leftarrow \mathbf{F}(\hat{\mathbf{x}}, \hat{\mathbf{u}})\mathbf{P}\,\mathbf{F}^{\top}(\hat{\mathbf{x}}, \hat{\mathbf{u}}) + \mathbf{G}(\hat{\mathbf{x}}) \begin{bmatrix} \mathbf{Q}^{(1)} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{Q}^{(2)} \end{bmatrix} \mathbf{G}^{\top}(\hat{\mathbf{x}})$
14:
15:   if GNSS measurement available then
16:
17:     % Calculate the Kalman filter gain.
18:     $\mathbf{K} \leftarrow \mathbf{P}(\mathbf{H}^{(1)})^{\top}\Big(\mathbf{H}^{(1)}\mathbf{P}(\mathbf{H}^{(1)})^{\top} + \mathbf{R}^{(1)}\Big)^{-1}$
19:
20:     % Calculate the measurement vector.
21:     $\mathbf{y}^{(1)} \leftarrow \tilde{\mathbf{p}}_k^{\text{GPS}} - \hat{\mathbf{p}}$
22:
23:     % Update the perturbation state estimate.
24:     $\begin{bmatrix} \delta\hat{\mathbf{x}} \\ \delta\hat{\mathbf{u}} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{0}_{9,1} \\ \delta\hat{\mathbf{u}} \end{bmatrix} + \mathbf{K}\left(\mathbf{y}^{(1)} - \mathbf{0}_{3,1}\right)$
25:
26:     % Update the Kalman filter state covariance.
27:     $\mathbf{P} \leftarrow \Big(\mathbf{I}_{15} - \mathbf{K}\mathbf{H}^{(1)}\Big)\mathbf{P}$
28:
29:     % Correct the navigation states using current perturbation
        estimates.
30:     $\hat{\mathbf{p}} \leftarrow \hat{\mathbf{p}} + \delta\hat{\mathbf{p}}$
31:     $\hat{\mathbf{v}} \leftarrow \hat{\mathbf{v}} + \delta\hat{\mathbf{v}}$
32:     $\hat{\mathbf{q}} \leftarrow \Gamma(\hat{\mathbf{q}}, \hat{\boldsymbol{\epsilon}})$
33:   end if
34:   $k \leftarrow k + 1$
35: end loop
```