COMP0130: ROBOT VISION AND NAVIGATION

Workshop 3: Multisensor Navigation

Paul Groves

Aim

The aim of this workshop is to produce an integrated navigation solution from a combination of different technologies. Using Matlab, you will work with odometry, magnetic compass, inertial navigation and GNSS, applying Kalman filtering to integrating data from different sensors. Step by step instructions, including all equations to implement, are provided. Answers will appear on Moodle at the end of the workshop.

Task 1: Car Dead Reckoning

The file Workshop3_Speed_Heading.csv contains 175s of car odometry and magnetic compass data. Column 1 of this comma-separated variable (CSV) format file contains time in seconds, column 2 contains the forward speed in metres per second and column 3 contains heading in degrees. The compass provides a measurement of the instantaneous heading. However, the odometer counts wheel rotations from which it calculates the distance travelled. The odometer speed output is therefore the average speed since the previous speed measurement, not the instantaneous speed.

The initial geodetic latitude and longitude (at time 0) are 50.4249580° and –3.5957974°, respectively. Use the speed and heading measurements to compute a position solution for the rest of the car's trajectory.

The average velocity between epochs k–1 and k is given by

$$\begin{pmatrix} \bar{v}_{N,k} \\ \bar{v}_{E,k} \end{pmatrix} = \tfrac{1}{2}\begin{pmatrix} \cos\psi_k + \cos\psi_{k-1} \\ \sin\psi_k + \sin\psi_{k-1} \end{pmatrix}\bar{v}_k , \tag{1}$$

where $\bar{v}_k$ is the average speed, measured by the odometer in this case, and ψ is heading, measured by the magnetic compass. Don't forget to convert the heading from degrees to radians before inputting it to the sine and cosine functions. The constant deg_to_rad is included in the Matlab script Define_constants.m

The latitude, L_k, and longitude, λ_k, (both in radians) at epoch k can then be computed from their counterparts at epoch $k-1$ using.

$$L_k = L_{k-1} + \frac{\bar{v}_{N,k}\left(t_k - t_{k-1}\right)}{R_N + h} \qquad \lambda_k = \lambda_{k-1} + \frac{\bar{v}_{E,k}\left(t_k - t_{k-1}\right)}{\left(R_E + h\right)\cos L_k}, \tag{2}$$

where t_k is the time at epoch k, h is the geodetic height, which you may assume remains at 37.4m throughout, R_N is the meridian radius of curvature and R_E is the transverse radius of curvature. The radii of curvature may be computed from the latitude using the Matlab function Radii_of_curvature.

To prepare for task 2, compute the damped instantaneous DR velocity at each epoch using

$$\begin{aligned} v_{N,k} &= 1.7\bar{v}_{N,k} - 0.7v_{N,k-1} \\ v_{E,k} &= 1.7\bar{v}_{E,k} - 0.7v_{E,k-1} \end{aligned}, \tag{3}$$

You may assume that the instantaneous velocity at time zero is given by the speed and heading measurements at that time. Thus $v_{N,0} = v_0\cos\psi_0$ and $v_{E,0} = v_0\sin\psi_0$. Without damping, the instantaneous velocity would be $2\bar{\mathbf{v}}_k - \mathbf{v}_{k-1}$. However, unless the sensors are

perfect, an oscillating error results. Therefore, damping is used to prevent the error build-up at the expense of introducing a time lag.

Task 2 Car DR/GNSS Integration

The file Workshop3_GNSS_Pos_Vel.csv contains the GNSS position and velocity solution for the same car as in Task 1. The format is as follows:

- Column 1 contains time in seconds
- Column 2 contains geodetic latitude in degrees
- Column 3 contains geodetic longitude in degrees
- Column 4 contains geodetic height in metres
- Columns 5 to 7 contain Earth-referenced velocity in metres per second, resolved along north, east and down, respectively.

Your task is to compute an integrated horizontal-only DR/GNSS navigation solution using Kalman filtering. You will implement a 4-state Kalman filter estimating north and east DR velocity error, DR latitude error and DR longitude error. The state vector is thus

$$\mathbf{x} = \begin{pmatrix} \delta v_N \\ \delta v_E \\ \delta L \\ \delta \lambda \end{pmatrix} \tag{4}$$

Initially, you don't know what the DR errors are, so initialise all 4 states at zero. Assume that your initial velocity uncertainty is, σ_v = 0.1 m/s in each direction and that your initial position uncertainty is σ_r = 10m per direction. The state estimation error covariance matrix is therefore initialised at

$$\mathbf{P}_0^+ = \begin{pmatrix} \sigma_v^2 & 0 & 0 & 0 \\ 0 & \sigma_v^2 & 0 & 0 \\ 0 & 0 & \dfrac{\sigma_r^2}{(R_N + h_0)^2} & 0 \\ 0 & 0 & 0 & \dfrac{\sigma_r^2}{(R_E + h_0)^2 \cos^2 L_0} \end{pmatrix}. \tag{5}$$

You can use the GNSS latitude and height from Workshop3_GNSS_Pos_Vel.csv at time 0.

Then follow the ten steps of the Kalman filter as follows:

1. Compute the transition matrix using

$$\mathbf{\Phi}_{k-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \dfrac{\tau_s}{R_N + h_{k-1}} & 0 & 1 & 0 \\ 0 & \dfrac{\tau_s}{(R_E + h_{k-1})\cos L_{k-1}} & 0 & 1 \end{pmatrix}, \tag{6}$$

   where the propagation interval, τ_s, is 0.5s in this case and you can use GNSS-indicated height.

2. Compute the system noise covariance matrix using

$$\mathbf{Q}_{k-1} = \begin{pmatrix} S_{DR}\tau_s & 0 & \frac{1}{2}\frac{S_{DR}\tau_s^2}{R_N + h_{k-1}} & 0 \\ 0 & S_{DR}\tau_s & 0 & \frac{1}{2}\frac{S_{DR}\tau_s^2}{(R_E + h_{k-1})\cos L_{k-1}} \\ \frac{1}{2}\frac{S_{DR}\tau_s^2}{R_N + h_{k-1}} & 0 & \frac{1}{3}\frac{S_{DR}\tau_s^3}{(R_N + h_{k-1})^2} & 0 \\ 0 & \frac{1}{2}\frac{S_{DR}\tau_s^2}{(R_E + h_{k-1})\cos L_{k-1}} & 0 & \frac{1}{3}\frac{S_{DR}\tau_s^3}{(R_E + h_{k-1})^2 \cos^2 L_{k-1}} \end{pmatrix}, \quad (7)$$

where the DR velocity error power spectral density (PSD), S_{DR}, is 0.2 m^2s^{-3}.

3. Propagate the state estimates:

$$\hat{\mathbf{x}}_k^- = \mathbf{\Phi}_{k-1}\hat{\mathbf{x}}_{k-1}^+ . \quad (8)$$

4. Propagate the error covariance matrix:

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+\mathbf{\Phi}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1} \quad (9)$$

5. Compute the measurement matrix (assuming the position measurements proceed the velocity measurements:

$$\mathbf{H}_k = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} \quad (10)$$

6. Compute the measurement noise covariance matrix assuming the GNSS position measurements have an error standard deviation of σ_{Gr} = 5 m per axis and the velocity measurements have an error standard deviation of σ_{Gv} = 0.02 m/s per axis. Thus,

$$\mathbf{R}_k = \begin{pmatrix} \frac{\sigma_{Gr}^2}{(R_N + h_k)^2} & 0 & 0 & 0 \\ 0 & \frac{\sigma_{Gr}^2}{(R_E + h_k)^2 \cos^2 L_k} & 0 & 0 \\ 0 & 0 & \sigma_{Gv}^2 & 0 \\ 0 & 0 & 0 & \sigma_{Gv}^2 \end{pmatrix}. \quad (11)$$

7. Compute the Kalman gain matrix using

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1} \quad (12)$$

8. Formulate the measurement innovation vector

$$\delta\mathbf{z}_k^- = \begin{pmatrix} L_k^G - L_k^D \\ \lambda_k^G - \lambda_k^D \\ v_{N,k}^G - v_{N,k}^D \\ v_{E,k}^G - v_{E,k}^D \end{pmatrix} - \mathbf{H}_k\hat{\mathbf{x}}_k^- = \begin{pmatrix} L_k^G - L_k^D + \delta L_k^- \\ \lambda_k^G - \lambda_k^D + \delta\lambda_k^- \\ v_{N,k}^G - v_{N,k}^D + \delta v_{N,k}^- \\ v_{E,k}^G - v_{E,k}^D + \delta v_{E,k}^- \end{pmatrix} \quad (13)$$

where superscript *G* denotes the GNSS-indicated solution obtained from Workshop3_GNSS_Pos_Vel.csv and superscript *D* denotes the DR-indicated

solution obtained from task 1. Note that units of radians are assumed for latitude and longitude.

9. Update the state estimates using

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \delta \mathbf{z}_k^- \tag{14}$$

10. Update the error covariance matrix using

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k^- . \tag{15}$$

Use the Kalman filter estimates to correct the DR solution at each epoch, *k*, using

$$\begin{aligned} L_k^C &= L_k^D - \delta L_k^+ \\ \lambda_k^C &= \lambda_k^D - \delta \lambda_k^+ \\ v_{N,k}^C &= v_{N,k}^D - \delta v_{N,k}^+ \\ v_{E,k}^C &= v_{E,k}^D - \delta v_{E,k}^+ \end{aligned} , \tag{16}$$

where the superscript *C* denotes the corrected DR solution.

Task 3: UAV INS/GNSS Integration (OPTIONAL)

Workshop 2 concerned an unmanned air vehicle (UAV) is flying over the Amazon jungle in Brazil, conducting an aerial survey to detect unauthorised logging. As well as GNSS data, the UAV's inertial navigation solution is also logged every second. Your task is to compute an integrated INS/GNSS navigation solution using Kalman filtering.

Use your GNSS position and velocity solution from Workshop 2 Task 2 to correct and calibrate the inertial navigation solution. Make sure that you saved your GNSS position and velocity solution with sufficient precision. Your latitude and longitude should have a precision of at least 10^{-7} degrees, your height 1 mm and your velocity 1 mm/s.

The inertial navigation solution is supplied in the file Aircraft_inertial.csv, which is in comma-separated variable (CSV) format. This contains latitude, L_b (degrees), longitude, λ_b (degrees), height, h_b (m), north, east, down velocity, $\mathbf{v}_{eb}^n$ (m/s), Euler attitude, $\boldsymbol{\psi}_{nb}$ (degrees), and $\mathbf{f}_{ib}^b$ specific force, (m/s^2).

The file Aircraft_inertial.csv is organised as follows:

- Column 1 contains time in seconds
- Column 2 contains geodetic latitude in degrees
- Column 3 contains geodetic longitude in degrees
- Column 4 contains geodetic height in metres
- Columns 5 to 7 contain Earth-referenced velocity in metres per second, resolved along north, east and down, respectively.
- Columns 8 to 10 contain, respectively, roll, pitch and heading, all in degrees.
- Columns 11 to 13 contain the measured specific force, resolved along the body-frame axes in metres per second squared.

You can use the Matlab function Euler_to_CTM.m to convert a Euler attitude solution to a coordinate transformation matrix. Note that this function will transform $\boldsymbol{\psi}_{nb}$ (radians) to $\mathbf{C}_n^b$. To obtain $\mathbf{C}_b^n$, you will need to transpose $\mathbf{C}_n^b$. The Matlab function CTM_to_Euler.m can be used to transform $\mathbf{C}_n^b$ to $\boldsymbol{\psi}_{nb}$ (radians). The Matlab function NED_to_ECEF.m will convert

L_b (radians), λ_b (radians), h_b, $\mathbf{v}_{eb}^n$ and $\mathbf{C}_b^n$ to $\mathbf{r}_{eb}^e$, $\mathbf{v}_{eb}^e$ and $\mathbf{C}_b^e$, while the Matlab function ECEF_to_NED.m performs the reverse process.

You will find some useful parameters in the Matlab script Define_Constants.m (available on moodle).

Implement a 15-state Kalman filter estimating attitude error (radians), velocity error (m/s), position error (m), accelerometer bias (m/s^2) and gyro bias (rad/s). The state vector is thus

$$\mathbf{x} = \begin{pmatrix} \delta\boldsymbol{\psi}_{eb}^e \\ \delta\mathbf{v}_{eb}^e \\ \delta\mathbf{r}_{eb}^e \\ \mathbf{b}_a \\ \mathbf{b}_g \end{pmatrix} \tag{17}$$

where $\delta\boldsymbol{\psi}_{eb}^e$ is the INS attitude error expressed as a small angle resolved about ECEF frame axes, $\delta\mathbf{v}_{eb}^e$ is the Cartesian ECEF INS velocity error, $\delta\mathbf{r}_{eb}^e$ is the Cartesian ECEF INS position error, $\mathbf{b}_a$ is the accelerometer bias vector (resolved along the INS body frame) and $\mathbf{b}_g$ is the gyro bias vector (resolved about the INS body frame).

The Kalman filter state vector estimate and error covariance matrix may be initialised using the Matlab function Initialise_Integration_KF.m (supplied on Moodle). The state vector estimates are all initialised at zero while the state uncertainties are 1° per component for the attitude error, 0.1 m/s for the velocity error, 10m for the position error 1000 μg for the accelerometer biases and 10 °/hour for the gyro biases.

The transition matrix is

$$\boldsymbol{\Phi}_{k-1} \approx \begin{bmatrix} \mathbf{I}_3 - \boldsymbol{\Omega}_{ie}^e \tau_s & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \widetilde{\mathbf{C}}_b^e \tau_s \\ \mathbf{F}_{21}\tau_s & \mathbf{I}_3 - 2\boldsymbol{\Omega}_{ie}^e \tau_s & \mathbf{F}_{23}\tau_s & \widetilde{\mathbf{C}}_b^e \tau_s & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \tau_s & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{18}$$

where $\boldsymbol{\Omega}_{ie}^e$ is the skew symmetric matrix of the Earth rotation rate (included in Define_Constants.m), τ_s is the propagation interval (equal to 1s), $\widetilde{\mathbf{C}}_b^e$ is the inertial attitude solution, and the matrices $\mathbf{F}_{21}$ and $\mathbf{F}_{23}$ are given by

$$\mathbf{F}_{21} = \left[-\left(\widetilde{\mathbf{C}}_b^e \widetilde{\mathbf{f}}_{ib}^b \right) \wedge \right], \qquad \mathbf{F}_{23} = -\frac{2\hat{\boldsymbol{\gamma}}_{ib}^e}{r_{eS}^e(\widetilde{L}_b)} \frac{\widetilde{\mathbf{r}}_{eb}^{e\,\mathrm{T}}}{\left| \widetilde{\mathbf{r}}_{eb}^e \right|} \tag{19}$$

where $\widetilde{\mathbf{f}}_{ib}^b$ is the specific force measured by the INS, $\hat{\boldsymbol{\gamma}}_{ib}^e$ is the gravitational acceleration, r_{eS}^e is the geocentric Earth radius, $\widetilde{L}_b$ is the inertial latitude solution and $\widetilde{\mathbf{r}}_{eb}^e$ is the inertial ECEF position solution. The Matlab functions Calculate_F21.m, Calculate_F23.m and Gravity_ECEF.m (called by Calculate_F23.m) may be used to compute these terms.

The system noise covariance matrix is

$$\mathbf{Q}_{k-1} \approx \begin{pmatrix} S_{rg}\tau_s\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & S_{ra}\tau_s\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & S_{bad}\tau_s\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & S_{bgd}\tau_s\mathbf{I}_3 \end{pmatrix} \tag{20}$$

where the gyro noise PSD, S_{rg}, is (0.02 deg/√hour)^2 = 3.385×10^{-11} rad^{2}s^{-1}, the accelerometer noise PSD, S_{ra}, is (200μg/√Hz)^2 = 3.847×10^{-6} m^{2}s^{-3}, the accelerometer bias random walk PSD, S_{bad}, is 10^{-7} m^{2}s^{-5} and the gyro bias random walk PSD, S_{bgd}, is 2×10^{-12} rad^{2}s^{-3}. These values are included in Define_Constants.m.

Measurements comprise the difference between the GNSS and inertial position and velocity solutions. The measurement matrix is therefore

$$\mathbf{H}_k \approx \begin{pmatrix} \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{pmatrix}_k \tag{21}$$

The measurement noise covariance matrix is

$$\mathbf{R}_k = \begin{pmatrix} \sigma_r^2\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \sigma_v^2\mathbf{I}_3 \end{pmatrix}_k \tag{22}$$

where the position measurement noise SD, σ_r, is 2.5m per axis and the velocity measurement noise SD, σ_v, is 0.05 m/s per axis. These values are included in Define_Constants.m.

The measurement innovation vector is

$$\delta\mathbf{z}^- = \begin{pmatrix} \hat{\mathbf{r}}_{ea}^e - \widetilde{\mathbf{r}}_{eb}^e + \delta\hat{\mathbf{r}}_{eb}^{e-} \\ \hat{\mathbf{v}}_{ea}^e - \widetilde{\mathbf{v}}_{eb}^e + \delta\hat{\mathbf{v}}_{eb}^{e-} \end{pmatrix} \tag{23}$$

where $\hat{\mathbf{r}}_{ea}^e$ and $\hat{\mathbf{v}}_{ea}^e$ is the GNSS position and velocity solution, $\widetilde{\mathbf{r}}_{eb}^e$ and $\widetilde{\mathbf{v}}_{eb}^e$ is the inertial position and velocity solution, and $\delta\hat{\mathbf{r}}_{eb}^{e-}$ and $\delta\hat{\mathbf{v}}_{eb}^{e-}$ are the propagated position error and velocity error estimates at the current epoch.

Following each epoch, the Kalman filter estimates may be used to correct the inertial attitude, velocity and position solution using

$$\begin{aligned} \hat{\mathbf{C}}_b^e &= \left(\mathbf{I}_3 - \boldsymbol{\Omega}\left(\delta\hat{\boldsymbol{\psi}}_{eb}^{e+}\right)\right)\widetilde{\mathbf{C}}_b^e \\ \hat{\mathbf{v}}_{eb}^e &= \widetilde{\mathbf{v}}_{eb}^e - \delta\hat{\mathbf{v}}_{eb}^{e+} \\ \hat{\mathbf{r}}_{eb}^e &= \widetilde{\mathbf{r}}_{eb}^e - \delta\hat{\mathbf{r}}_{eb}^{e+} \end{aligned}, \tag{24}$$

where $\boldsymbol{\Omega}\left(\delta\hat{\boldsymbol{\psi}}_{eb}^{e+}\right)$ is the skew-symmetric matrix of the estimated attitude error. It may be computed using the Matlab function Skew_symmetric.m (available on Moodle).

Express the corrected inertial attitude solution in the form latitude (degrees), longitude (degrees), height (m), north, east, down velocity (m/s) and Euler attitude (degrees).