Inertial motion estimation for extreme sports modelling

Problem presented by

Simon Fowler

Vert Systems Ltd.

Executive Summary

Problem Sensors record three-axis accelerometer, gyroscope and magnetometer data at high frequency (100 Hz) and GPS data (position in three-dimensional space) at low frequency (≈ 1 Hz).

The goal is to combine these measurements into a trajectory of the point holding the sensors. The trajectory, consisting of the time profiles for position, orientation, velocity and acceleration, has to be suitable for performance analysis and animation (so needs higher time resolution and accuracy than the GPS data). Computations will be performed in a post-processing step, not in real-time.

Proposed solutions Two algorithms to incorporate the GPS data are proposed. The first is a simple generalization of the prior solution provided by the problem presenter. It always produces a result quickly and avoids drift. However, it relies on heuristically tuned "gains". The head orientation is not guaranteed to be correct, which is likely to have knock-on effects on speed and acceleration.

The second approach applies a linear Kalman smoother iteratively. This algorithm exploits the post-processing nature of the computations taking into account past and future measurements in an optimal manner. It avoids heuristic gains and generates covariance matrices that give an estimate of the error in the results.

Further investigations looked into the potential of wavelet smoothing the measurements and potential applications for Android devices.

Version 1.0
May 29, 2013
iii+26 pages

Report author

Jonathan Black (University of Oxford, UK), Jacqueline Christmas (Exeter, UK), Jakub Nowotarski (Wroclaw, Poland), Jan Sieber[1] (Exeter, UK), Jakub Tomczyk (Wroclaw, Poland)

[1] corresponding author: j.sieber@exeter.ac.uk

Contributors

Jonathan Black (University of Oxford, UK),
Jacqueline Christmas (University of Exeter, UK),
Mikhail Krastanov (IMI, Bulgarian Academy of Sciences, Sofia, Bulgaria),
Jakub Nowotarski (Wroclaw University of Technology, Poland),
Jan Sieber (University of Exeter, UK),
Robert Szalai (University of Bristol, UK),
Jakub Tomczyk (Wroclaw University of Technology, Poland),
Ellen Webborn (Univeristy of Warwick, UK)

ESGI91 was jointly organised by
University of Bristol
Knowledge Transfer Network for Industrial Mathematics

and was supported by
Oxford Centre for Collaborative Applied Mathematics
Warwick Complexity Centre

Contents

1 Introduction **1**

2 Problem statement **2**
2.1 Background 2
2.2 Constraints 2
2.3 Potential Solutions 3
2.4 Aims for study group 3
2.5 Test case data provided 4

3 Prior approach and naive solution **6**
3.1 Internal state to be tracked 6
3.2 Prior approach — attitude update and correction 6
3.3 Problems left open from the prior approach— drift 7
3.4 Smoothing and interpolation of the GPS position data 8
3.5 Assimilation of GPS values into sensor-based trajectory 9
3.6 Summary 13

4 Kalman filter based assimilation of all measurement data simultaneously **14**
4.1 Background 14
4.2 The model for the presented problem 15
4.3 Iterative procedure 17
4.4 Initial guesses 19
4.5 Expectation (E) Maximisation (M) iteration 19
4.6 E step 20
4.7 M step 21
4.8 Initial results 22

5 Wavelet smoothing and tests of Android phone **22**
5.1 Wavelet smoothing 22
5.2 Android device as a sensor 24

6 Conclusions and suggestions for next steps **25**

Bibliography **26**

1 Introduction

(1.1) The presented problem is a *data assimilation problem* [1]. A dynamic model describes how its internal state $\mathbf{x}(t_k)$ (in our case position, velocity, orientation) evolves forward in time (from t_{k-1} to t_k for a large number of time steps). As the initial state $\mathbf{x}(t_0)$ is unknown and the model itself has uncertainty and biases one expects systematic drift and inaccuracies. At the same time a time series of measurements $\mathbf{y}(t_k)$ is taken. These measurements are typically only a projection of the full state $\mathbf{x}(t)$. The measurements $\mathbf{y}(t_k)$ are also affected by disturbances and they will be inconsistent with the current belief of what the state has evolved to through the model. Thus, one needs to "assimilate" the measurements into the current belief about the true state. In general the result will be a probability distribution described by a mean and a (co-)variance (this description is complete if the distribution is *normal*, which is typically the case if disturbances originate from many small independent sources).

If the problem is linear, that is, the model has the form

$$\mathbf{x}(t_k) = \mathbf{A}(t_k)\mathbf{x}(t_{k-1}) \tag{1}$$

and the measurements are known to be depending on the state via

$$\mathbf{y}(t_k) = \mathbf{C}(t_k)\mathbf{x}(t_k),$$

($\mathbf{A}$ and $\mathbf{C}$ are matrices and both rules are assumed to be affected by disturbances) then the solution known to be optimal is the *Kalman smoother* [1, 2, 3, 5].

(1.2) The presented problem does not fit into this class of linear problems because the change of orientation creates rotations, which are nonlinear operations (regardless of representation by quaternions, rotation matrix or Euler angles). In our case this implies that the outputs depend nonlinearly on the state: $\mathbf{y}(t_k) = \mathbf{N}(\mathbf{x}(t_k))$. Since the sought trajectory is a small deviation from the trajectory obtained from the GPS data, one can hope to apply the Kalman smoother to the deviations from the GPS trajectory. This results in the iteration process described in Section 4.

(1.3) Section 3 describes a naive way to incorporate measurements, which does not rely on linearity of the model (thus, $\mathbf{Y}(t_k) = \mathbf{N}(\mathbf{X}(t_k))$ is permitted). After each update step following the model (1) one computes the direction $\mathbf{c}(t_k)$ normal to the surface given by $\{\mathbf{x} : \mathbf{N}(\mathbf{x}) = \mathbf{y}(t_k)\}$ and then applies a correction along this normal to the state: $\mathbf{x}_{\text{cor}}(t_k) = \mathbf{x}(t_k) + g\mathbf{c}(t_k)$. The advantage of this approach is simplicity and speed. The disadvantage is that it relies on a heuristic factor (or *gain*) g. This factor is determined by how trustworthy each component of the measurement is. One does not have an objective estimate for this trustworthiness (in contrast to the Kalman

smoother, which estimates variances along with means). The approach discussed in Section 3 is a generalisation of the approach to attitude correction provided by the problem presenter. It comes with example scripts and demo code.

(1.4) The solution based on the Kalman filter is most likely the "best possible" approach (optimal in some mathematical sense). At the moment the implementation does not follow through with the iteration described in Section 4. Its author, Jacqueline Christmas, an expert on data assimilation, is interested in further collaboration to finish and improve the implementation.

(1.5) The report consists of four pieces, independently written within a very short time such that notation is not entirely consistent across all sections (especially, Section 3 and 4). On the other hand, the sections aim to be self-contained, making it possible to read, for example, Section 4 without Section 3. The report is accompanied by some demo code (`Matlab`/`octave` compatible) and example data files to aid experimentation.

2 Problem statement

(as provided by Simon Fowler)

2.1 Background

(2.1) Performance analysis for extreme sports athletes goes beyond lap times and heart rates to include details of their motion across challenging terrain. Vert Systems is currently developing technology to allow the visualisation of this motion, where the core of the system is an inertial navigation algorithm which needs to generate not just a position estimate but also an estimate of the velocity and acceleration of the user. The key requirement is to track the local features of the motion, with absolute position accuracy potentially less of an issue. The specific problem for the study group is based on a mountain biking example, which provides a representative example of both the highly dynamic movement and also the levels of noise inherent in the motion over rough terrain.

2.2 Constraints

(2.2) The sensor data that is available is 3-axis acceleration and rate (gyroscope) inertial data, 3-axis magnetic field data, and GPS data. Although GPS data is available as input to the algorithms, the system requires higher fidelity detail of the motion than is available purely from GPS.

(2.3) The motion of the user is likely to include relatively long duration external forces, such as from cornering and braking, meaning that standard techniques such as high pass filtering to remove the effect of gravity are potentially inappropriate in this environment. The user motion will also include

relatively high levels of noise, particularly in the vertical axis, due to the features of the terrain. The attitude of the sensor relative to the direction of motion is not fixed. The attitude will remain in a generally constant position relative to the motion, but with short term variations likely.

(2.4) The inertial sensors used will be relatively inexpensive MEMS devices, which are relatively noisy and have limited accuracy. Basic calibration of the sensors can be assumed but some level of estimation of sensor errors is likely to be necessary based on the redundant information within the sensor data; a goal is to minimise the required pre-calibration activities.

2.3 Potential Solutions

(2.5) The 'standard' solution would be an estimation technique based on Kalman filtering, but even within this general field there is significant room for different solutions. The current solution uses basic estimators for both attitude and position, although one of the current key sources of error is the propagation of attitude errors into the position estimate.

(2.6) Solutions could take advantage of the dynamics of the user, where the mass of the actual body in motion will limit the rate of change of velocity within certain bounds, with short term variations based on movement of the sensor relative to the centre of mass. Similarly, the mechanisms for change in overall velocity (steering, pedalling, and braking) could be modelled to some extent so that they can be isolated from gravity and noise.

(2.7) The results do not need to be generated in real-time, and so techniques do not need to be based solely on processing the data in a line-by-line manner. Similarly, processing and memory requirements are not a primary concern, as the processing can be performed off-line.

2.4 Aims for study group

(2.8) The challenge for the study group is to bring a new perspective to this problem, and investigate:

- What is the most appropriate and accurate technique for estimation the motion based on the quality of data available in the target environment?
- Techniques for automatic calibration or otherwise minimisation of sensor errors such as bias, drift, and misalignment.

Sample data will be available for actual investigation/experimentation, and further data can be collected for specific scenarios as necessary to support the study group.

output	column in `Data`	variable	dimension	frequency
acceleration	2–4	$\hat{\mathbf{a}}$	$\mathbb{R}^3$	100 Hz
angular velocity	5–7	$\hat{\boldsymbol{\omega}}$	$\mathbb{R}^3$	100 Hz
magnetometer	8–10	$\hat{\mathbf{m}}$	$\mathbb{R}^3$	100 Hz
GPS position	11–13*	$\hat{\mathbf{p}}$	$\mathbb{R}^3$	≈ 1 Hz

*if non-zero

Table 1: Variable names for measurement data

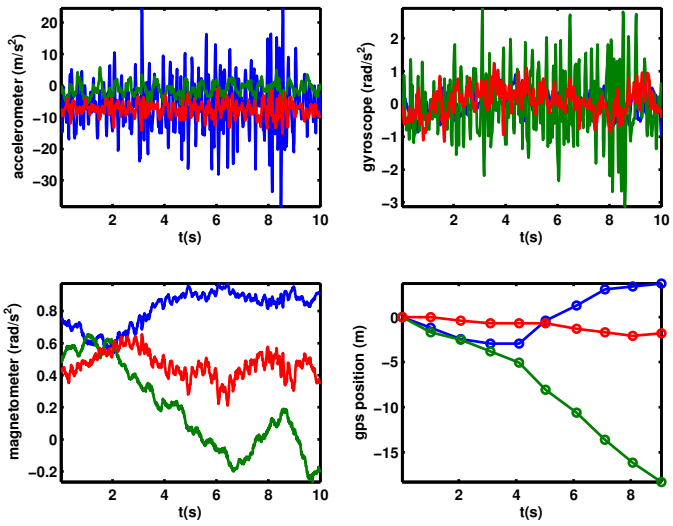


Figure 1: Example time profiles of measurement data for file 2, starting from time point $k_0 = 40,000$. (a) $\hat{\mathbf{a}}$, (b) $\hat{\boldsymbol{\omega}}$, (c) $\hat{\mathbf{m}}$ including correction (4), (d) $\hat{\mathbf{p}}$, relative to the first point in $\hat{\mathbf{p}}$.

2.5 Test case data provided

(2.9) The data files (comma separated tables)

1. `Cal_Data_1_no_calibration.csv`,
2. `INS_23_processed_test.csv`,
3. `Test_Ride_1_no_calibration.csv`,
4. `Test_Ride_2_no_calibration.csv`

contained time profiles (as measured by sensors) in the format listed below and in Table 1. We refer to the csv table as `Data(:,:)` using `octave`/`Matlab` array notation, calling the number of rows N (equalling the number of measurements).

(2.10) `Data(:,1)` time t (unit: s), to be ignored, replace by

$$t_k = k/100 \quad (k = 1, \ldots, N).$$

(2.11) `Data(:,2:4)` accelerometer measurements of linear acceleration $\hat{\mathbf{a}}(t_k) \in \mathbb{R}^3$ $(k = 1, \ldots, N)$ for times t_k (unit: m/s^2).

(2.12) `Data(:,5:7)` gyroscope measurements of angular velocity $\hat{\boldsymbol{\omega}}(t_k) \in \mathbb{R}^3$ $(k = 1, \ldots, N)$ for times t_k (unit: degree/s). We also store the vector $\hat{\boldsymbol{\omega}}(t_k) = (\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3)(t_k)$ of angular velocities in the anti-symmtric matrix

$$\hat{\boldsymbol{\Omega}}(t_k) = \begin{bmatrix} 0 & \hat{\omega}_3 & -\hat{\omega}_2 \\ -\hat{\omega}_3 & 0 & \hat{\omega}_1 \\ \hat{\omega}_2 & -\hat{\omega}_1 & 0 \end{bmatrix}. \tag{2}$$

(2.13) `Data(:,8:10)` magnetometer measurements $\hat{\mathbf{m}}(t_k) \in \mathbb{R}^3$ $(k = 1, \ldots, N)$ for times t_k (unit: arbitrary). The magnetometer points toward the magnetic north $\mathbf{n}$, given by the problem presenter;

$$\mathbf{n} = \begin{bmatrix} \sin n_d \cos n_i \\ \cos n_d \cos n_i \\ \sin n_i \end{bmatrix} \quad \text{where } n_i = 66.727^\circ,\ n_d = -2.072^\circ. \tag{3}$$

Prior experimentation by the problem presenter had shown that the center of rotation of the magnetometer measurements was displaced from the origin by a consistent offset $m_c \in \mathbb{R}^3$. Thus, for the investigations of the study group we preprocessed the measurements by fitting `Data(:,8:10)` to a sphere

```
% finds best fit mc for center
mc=sphereFit(Data(:,8:10));
```

using

$$\hat{\mathbf{m}}_k = \frac{\hat{\mathbf{m}}_{0,k} - \mathbf{m}_c}{|\hat{\mathbf{m}}_{0,k} - \mathbf{m}_c|} \tag{4}$$

where $\hat{\mathbf{m}}_{0,k}$ = `Data(k,8:10)` and $\mathbf{m}_c$ is the center `mc` obtained by `sphereFit`[1].

(2.14) `Data(:,11:13)` (whenever non-zero) GPS position data $\hat{\mathbf{p}} \in \mathbb{R}^3$ (geo-location [east-west, north-south, elevation] relative to $(0^\circ, 0^\circ)$ and zero elevation, unit: m). Select the indices k_{gps} at which `Data(:,11)` is non-zero:

$$k_{\text{gps}} = \texttt{find(Data(:,11)~=0)}$$

and define N_{gps} as the length of k_{gps}. Then $\hat{\mathbf{p}}(t_{k_{\text{gps}},j})$ is the GPS location at time $t_{k_{\text{gps}},j}$.

(2.15) Columns `Data(:,14:end)` were not used in the algorithms.

(2.16) Figure 1 shows a 10-second part of the time profiles of $\hat{\mathbf{a}}$, $\hat{\boldsymbol{\omega}}$, $\hat{\mathbf{m}}$ and $\hat{\mathbf{p}}$ to give a visual impression of the relative temporal and spatial resolution of the sensor signals and their signal-to-noise ratios.

[1] `http://www.mathworks.co.uk/matlabcentral/fileexchange/34129-sphere-fit-least-squared`

3 Prior approach and naive solution

3.1 Internal state to be tracked

(3.1) Table 2 lists the internal states to be tracked and their dimensions. The

state	variable name	dimension
position of sensor	$\mathbf{p}(t_k)$	$\mathbb{R}^3$
velocity of sensor	$\mathbf{v}(t_k)$	$\mathbb{R}^3$
attitude (orientation) of sensor	$\mathbf{R}(t_k)$	$\mathbb{R}^{3\times 3}$

Table 2: Variable names for internal states to be tracked for all times t_k $(k = 1, \ldots, N)$.

attitude (or orientation) of the sensor can be stored in many different ways. Table 2 and the subsequent equations use the convention of storing it as tripod of three orthogonal unit vectors, that is, $\mathbf{R}(t_k)$ is a 3×3 rotation matrix satisfying

$$\mathbf{R}(t_k)^T\mathbf{R}(t_k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

Since $\mathbf{R}(t_k)^T\mathbf{R}(t_k)$ is symmetric, condition (5) imposes 6 conditions (say, the identities for the entries $(1,1)$, $(2,1)$, $(3,1)$, $(2,2)$, $(2,3)$ and $(3,3)$ in condition (5)) on the 9 entries of $R(t_k)$. Storing an orientaton as a rotation matrix is equivalent to storing it as a quaternion (a vector of length 4 and unit length). Code for conversion between rotation matrix $\mathbf{R}$ and quaternion was provided by the problem presenter.

3.2 Prior approach — attitude update and correction

(3.2) The problem presenter provided a prior reference solution for one core issue the trajectory recovery has to overcome, a correction and updating rule for the attitude $\mathbf{R}$ (file `attitude_est.sci` attached). If one assumes that the attitude $\mathbf{R}(t_k)$ at the previous time t_k is known and the gyroscope data for the angular velocity $\hat{\mathbf{\Omega}}(t_k)$ is perfect, then the attitude at time t_{k+1} is approximately

$$\mathbf{R}(t_{k+1}) = \exp\left([t_{k+1} - t_k]\mathbf{\Omega}_{\text{cor}}(t_k)\right)\mathbf{R}(t_k), \tag{6}$$

where (except for the corrections below) $\mathbf{\Omega}_{\text{cor}}(t_k) = \hat{\mathbf{\Omega}}(t_k)$. This is the *explicit Euler* formula for the update. The $\exp(\ldots)$ term in (6) refers to the matrix exponential of the 3×3 matrix $(t_{k+1}-t_k)\hat{\mathbf{\Omega}}(t_k)$ (see `Matlab`'s/`octave`'s built-in command `expm`).

(3.3) However, two additional pieces of information are incorporated to modify the rotation, making $\mathbf{\Omega}_{\text{cor}}(t_k)$ different from $\hat{\mathbf{\Omega}}(t_k)$. The magnetometer measurement should equal the magnetic north, $\hat{\mathbf{m}}(t_{k+1}) = \mathbf{R}(t_{k+1})^T\mathbf{n}$, and, most

of the time, the acceleration $\hat{\mathbf{a}}(t_{k+1})$ points upwards. The error is expressed as

$$\mathbf{e}_g = f_g(t_k)\,\frac{\hat{\mathbf{a}}(t_{k+1})}{|\hat{\mathbf{a}}(t_k)|} \times \left[\mathbf{R}(t_k)^T \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}\right] \tag{7}$$

$$\mathbf{e}_m = f_m(t_k)\,\hat{\mathbf{m}}(t_{k+1}) \times \left[\mathbf{R}(t_k)^T \mathbf{n}\right]. \tag{8}$$

In (7) and (8), f_g and f_m are heuristic prefactors (between 0 and 1) depending on how trustworthy $\hat{\mathbf{m}}(t_{k+1})$ is (lower if $\hat{\boldsymbol{\omega}}(t_k)$ is large) and how likely $\hat{\mathbf{a}}(t_k)$ is pointing upward (lower if $|\hat{\mathbf{a}}(k)|$ differs strongly from $9.81\,\mathrm{m/s^2}$). The operation $\times$ is the vector cross product and $\mathbf{n}$ is the magnetic north vector defined in Equation (3). The magnetometer measurement $\hat{\mathbf{m}}$ is already scaled to unit length according to Equation (4). The second term in the cross products in (7) and (8) applies the inverse $\mathbf{R}(t_k)^T$ of the previous attitude to express the upward direction $(0,0,1)^T$ and the magnetic north $\mathbf{n}$ in the coordinates aligned with the previous orientation $\mathbf{R}(t_k)$.

(3.4) The overall error

$$\mathbf{e} = \mathbf{e}_g + \mathbf{e}_m \tag{9}$$

modifies the rotation $\boldsymbol{\Omega}_{\mathrm{cor}}$ in (6) via a PI control term:

$$\mathbf{e}_{\mathrm{int}}(t_{k+1}) = \mathbf{e}_{\mathrm{int}}(t_k) + (t_{k+1} - t_k)\mathbf{e}(t_k) \tag{10}$$

$$\boldsymbol{\omega}_{\mathrm{cor}}(t_k) = \hat{\boldsymbol{\omega}}(t_k) + g_p\mathbf{e}(t_k) + g_{\mathrm{int}}\mathbf{e}_{\mathrm{int}}(t_k). \tag{11}$$

In (11) the prefactors g_p and g_{int} are control gains chosen (heuristically) to avoid introducing an instability but controlling the error $\mathbf{e}$ to zero in the absence of disturbances.

Finally, the matrix $\boldsymbol{\Omega}_{\mathrm{cor}}(t_k)$ is constructed from $\boldsymbol{\omega}_{\mathrm{cor}}$ in the same way as $\hat{\boldsymbol{\Omega}}(t_k)$ from $\hat{\boldsymbol{\omega}}(t_k)$ (see Equation 2).

3.3 Problems left open from the prior approach— drift

(3.5) The attitude correction (6), (11) alone cannot correct a disturbance-driven drift in the position and velocity. That is, if one updates the other states in the natural way

$$\mathbf{p}(t_{k+1}) = p(t_k) + (t_{k+1} - t_k)\mathbf{v}(t_k) \tag{12}$$

$$\mathbf{v}(t_{k+1}) = \mathbf{v}(t_k) + (t_{k+1} - t_k)\mathbf{R}(t_k)\hat{\mathbf{a}}(t_k) \tag{13}$$

then the disturbances in the measurements $\hat{\mathbf{a}}$ and the uncertainty in the initial position and velocity $\mathbf{p}(t_0)$ and $\mathbf{v}(t_0)$ will introduce rapidly increasing errors in $\mathbf{p}(t_k)$ and $\mathbf{v}(t_k)$.

(3.6) The PI control approach to attitude adjustment relies on a number of heuristics (gains, error factors, etc). It is not clear if the attitude is maintained

correctly, and if/how errors in the attitude $\mathbf{R}$ translate into errors in position $\mathbf{p}$, linear velocity $\mathbf{v}$ and linear acceleration

$$\mathbf{a}(t_{k+1}) \approx \frac{\mathbf{v}(t_{k+1}) - \mathbf{v}(t_k)}{t_{k+1} - t_k}.$$

3.4 Smoothing and interpolation of the GPS position data

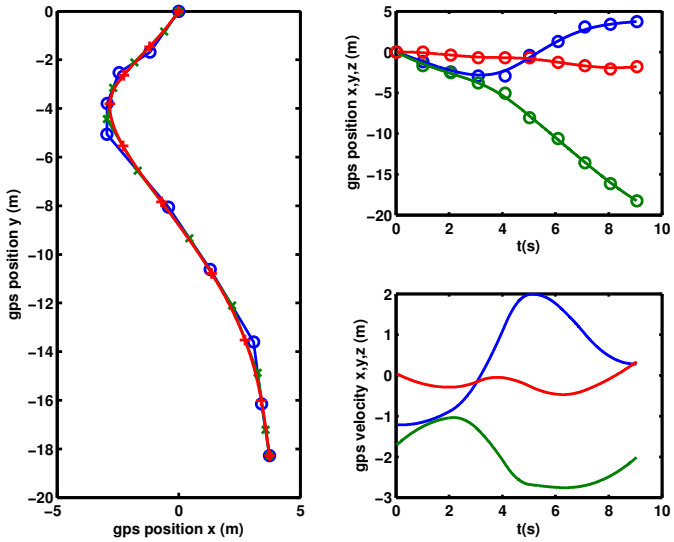


Figure 2: Illustration of simple smoothing and interpolation algorithm based on linear midpoint interpolation. Example time and x-y profiles of measurement data are for file 2, starting from time point $k_0 = 40,000$. (a) Illustration of algorithm, see paragraph (3.8), (b) curves are the resulting time profiles of GPS positions ("o" symbols original data, blue= x, green= y, red= z), (c) resulting time profiles of velocities obtained from time differentiation of spline interpolated smoothed GPS positions (blue= x, green= y, red= z).

(3.7) Several of the simple demonstration algorithms developed during the study week are assuming GPS data at the same frequency as the sensor data. The true GPS signal comes with a low time resolution and aligns along a grid of several meters resolution in space. Thus, a simple approach to connect the GPS data with the assumptions of the algorithms shown in 3.5 is smoothing and interpolation.

(3.8) Figure 2 illustrates a simple smoothing algorithm. Suppose we are given a sequence of GPS points $\hat{\mathbf{p}}_j \in \mathbb{R}^3$ at times $t_{\mathrm{gps},j}$ $(j = 1, \ldots, n_{\mathrm{gps}}$, see circles

in Figure 2(a) and (b)). We define

$$\begin{aligned} s_0 &= t_{\mathrm{gps},1}, & s_j &= \frac{1}{2}\left[t_{\mathrm{gps},j+1} + t_{\mathrm{gps},j}\right], \quad (j = 1, \ldots, n_{\mathrm{gps}} - 1), \\ s_{n_{\mathrm{gps}}} &= t_{\mathrm{gps},n_{\mathrm{gps}}}, & & \\ \mathbf{q}_0 &= \hat{\mathbf{p}}_1, & \mathbf{q}_j &= \frac{1}{2}\left[\hat{\mathbf{p}}_{j+1} + \hat{\mathbf{p}}_j\right], \quad (j = 1, \ldots, n_{\mathrm{gps}} - 1), \\ \mathbf{q}_{n_{\mathrm{gps}}} &= \hat{\mathbf{p}}_{n_{\mathrm{gps}}}, & & \\ \mathbf{p}_1 &= \mathbf{q}_0, & \mathbf{p}_j &= \frac{(t_{\mathrm{gps},j} - s_{j-1})\mathbf{q}_j + (t_{\mathrm{gps},j} - s_j)\mathbf{q}_{j-1}}{s_j - s_{j-1}}, \\ \mathbf{p}_{n_{\mathrm{gps}}} &= \mathbf{q}_{n_{\mathrm{gps}}} & & (j = 2, \ldots, n_{\mathrm{gps}} - 1). \end{aligned} \tag{14}$$

The values $\mathbf{p}_j$ are the smoothed values. In Figure 2(a) the (blue) circles are the $\hat{\mathbf{p}}_j$ ($j = 1 \ldots, 10$), the (dark green) "$\times$" symbols are the $\mathbf{q}_j$ ($j = 0 \ldots, 10$), and the (red) "+" symbols are the smoothed values $\mathbf{p}_j$ ($j = 1, \ldots, 10$). The sequence of smoothed values $\mathbf{p}_j$ is regular enough to perform spline interpolation (red curve in Figure 2(a), all curves in Figure 2(b)). Even the time derivatives of the spline-interpolated $\mathbf{p}_j$ (giving the approximate velocities deduced from GPS positions) do not show oscillations at the sampling frequency of the GPS signal. The approximate velocities are shown in Figure 2(c).

(3.9) The procedure transforming the sequence of $\hat{\mathbf{p}}_j$ into $\mathbf{p}_j$ through Equation (14) can be applied (repeatedly) to the sequence of $\mathbf{p}_j$ again. The results will becomes increasingly smooth but will also increasingly tend to "cut corners" in the path. The short (`Matlab`/`octave`) function `mpsmooth(t,p,n)` applies the smoothing procedure `n` times to the values `p` at times `t`.

(3.10) The procedure in paragraph (3.8) is a simple approximation of a low-pass filter that decreases the amplitude of high frequencies exponentially (highest frequency would be sampling frequency, lowest frequency would be time of entire trajectory). The function `fftsmooth(y,options)` performs a similar smoothing operation for profiles with uniform time spacing. The function `fftsmooth` is faster if one performs the equivalent of many smoothing oprations of the form (14). For example, if `ahat` is the variable for the acceleration measurement $\hat{\mathbf{a}}$, then `a=fftsmooth(ahat,'halfdecay',1/200);` produces a smoothed version `a` that is roughly the same as the result of `a=mpsmooth(t,ahat,1000)`. (Smoothing $\hat{\mathbf{a}}$ may be a useful pre-processing step before extracting gravitational acceleration to determine the upward direction.)

3.5 Assimilation of GPS values into sensor-based trajectory

(3.11) The standard solution for assimilating measurements into a state trajectory obtained from a linear model is the Kalman filter. It chooses the optimal

balance between the state produced by the model and the indirect information about the state due to the measurements. Optimality is based on the assumption that both, model and measurements are disturbed by random errors. An estimate of the covariance of these errors (a measure for their size) is generated as part of the procedure. Generalizations to models with nonlinearity (which is introduced by the rotation $\mathbf{R}$ in this problem) work as long as one starts from a good initial guess. A separate section (Section 4) explains the implementation of the Kalman filter and the Kalman smoother. This section gives a "naive" version that is a generalization of the approach adopted in the prior solution for attitude adjustment. The generalization permits us to include an arbitrary number of additional measurements.

(3.12) In the given problem the spacing of the time points t_k for the sensor measurements is uniform such that we have the time increment

$$\Delta_t = t_k - t_{k-1} (= 0.01\text{s}).$$

After smoothing and interpolating we have the same time resolution also for the GPS data. We list all equations regarding the state $\mathbf{p}(t_k) \in \mathbb{R}^3$, $\mathbf{v}(t_k) \in \mathbb{R}^3$ and $\mathbf{R}(t_k) \in \mathbb{R}^{3\times 3}$ in a single system:

$$\frac{1}{\Delta_t}\left[\mathbf{p}(t_k) - \mathbf{p}(t_{k-1})\right] = \mathbf{v}(t_k) + \{\mathbf{c}_p(t_k)\} \tag{15}$$

$$\frac{1}{\Delta_t}\left[\mathbf{v}(t_k) - \mathbf{v}(t_{k-1})\right] = \mathbf{R}(t_k)\hat{\mathbf{a}}(t_k) + \mathbf{g} + \{\mathbf{c}_v(t_k)\} \tag{16}$$

$$\frac{1}{\Delta_t}\left[\mathbf{R}(t_k) - \mathbf{R}(t_{k-1})\right] = \hat{\mathbf{\Omega}}(t_k)\mathbf{R}(t_k) + \\ + \left\{\mathbf{J}_m(\hat{\mathbf{m}}(t_k))^T \mathbf{c}_m(t_k) + \mathbf{J}_g(\hat{\mathbf{a}}(t_k))^T \mathbf{c}_g(t_k)\right\} \tag{17}$$

$$\mathbf{R}(t_k)\hat{\mathbf{m}}(t_k) = \mathbf{n} \tag{18}$$

$$\mathbf{R}(t_k)\hat{\mathbf{a}}(t_k) = \frac{|\hat{\mathbf{a}}(t_k)|}{|\mathbf{g}|}\mathbf{g} \tag{19}$$

$$\mathbf{p}(t_k) = \mathbf{p}_{\text{gps}}(t_k) \tag{20}$$

$$\mathbf{v}(t_k) = \mathbf{v}_{\text{gps}}(t_k) \tag{21}$$

$$\mathbf{R}(t_k)^T\mathbf{R}(t_k) = \mathbf{I}_3 \tag{22}$$

(The terms in curly brackets will be explained below.) In (16) $\mathbf{g}$ is the gravitational acceleration. In (18) $\mathbf{n}$ is the magnetic north defined in (3). In (20) and (21) $\mathbf{p}_{\text{gps}}(t_k)$ and $\mathbf{v}_{\text{gps}}(t_k)$ are the positions and velocities obtained by GPS (smoothed and interpolated using (14)). In (22) $\mathbf{I}_3$ is the 3×3 identity matrix (see (5)).

(3.13) Note that (15)–(22) adopted the approximation corresponding to an implicit Euler integration: on all right-hand sides we insert the values at the current time t_k instead of t_{k-1}.

(3.14) If we ignore the terms in curly brackets then the only unknowns are $\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$ and $\mathbf{R}(t_k)$ such that the system (15)–(22) is overdetermined. The updating rules (15)–(17) and the additional measurements (18)–(22) will not be perfectly consistent (due to measurement noise and errors in the previous states).

The additional terms in curly brackets are the constraint forces necessary to satisfy all equations simultaneously. The terms $\mathbf{c}_p(t_k) \in \mathbb{R}^3$, $\mathbf{c}_v(t_k) \in \mathbb{R}^3$, $\mathbf{c}_m(t_k) \in \mathbb{R}^3$ and $\mathbf{c}_g(t_k) \in \mathbb{R}^3$ are additional unknowns such that (15)–(21) (ignoring orthogonality condition (22) because it is nonlinear) becomes a linear system of 27 equations with 27 unknowns ($\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$, $\mathbf{R}(t_k)$, $\mathbf{c}_p(t_k)$, $\mathbf{c}_v(t_k)$, $\mathbf{c}_m(t_k)$ and $\mathbf{c}_g(t_k)$). The two matrices $\mathbf{J}_m$ and $\mathbf{J}_g$ in (17) are the pre-factors of $\mathbf{R}(t_k)$ in the constraints from the magnetometer and gravity, (18) and (19):

$$\mathbf{J}_m(\hat{\mathbf{m}}) = \hat{\mathbf{m}}^T \otimes \mathbf{I}_3, \quad \mathbf{J}_g(\hat{\mathbf{a}}) = \hat{\mathbf{a}}^T \otimes \mathbf{I}_3.$$

Both are 3×9 matrices (using the convention that the rotation matrix $\mathbf{R}(t_k)$ is treated as a column-wise 9×1 vector when solving the linear system (15)–(21)). The `Matlab`/`octave` commands generating these *Kronecker product* matrices are (calling $\hat{\mathbf{m}}$ =`mag` and $\hat{\mathbf{a}}$ =`acc`)

```
Jm=kron(mag(:)',eye(3));
Jg=kron(acc(:)',eye(3));
```

(3.15) The solution to this full system satisfies all additional constraints (18)–(21) exactly. Hence, this solution is not directly useful because $\mathbf{p}(t_k)$ and $\mathbf{v}(t_k)$ will be determined entirely by the GPS signal. However, we can use the resulting constraint forces $\mathbf{c}_p$, $\mathbf{c}_v$, $\mathbf{c}_m$ and $\mathbf{c}_g$ (pre-multiplied by small factors) to correct drift of the states as obtained from (15)–(17) *without* constraint forces (the terms in curly brackets) and ignoring the additional constraints (18)–(21).

(3.16) This results in the folowing overall procedure for updating $\mathbf{p}$, $\mathbf{v}$ and $\mathbf{R}$:

1. **(Given)** States at previous time t_{k-1}: $\mathbf{p}(t_{k-1})$, $\mathbf{v}(t_{k-1})$, $\mathbf{R}(t_{k-1})$
   (Given) Measurements at current time: $\mathbf{p}_{\text{gps}}(t_k)$, $\mathbf{v}_{\text{gps}}(t_k)$ (smoothed and interpolated), $\hat{\mathbf{a}}(t_k)$, $\hat{\boldsymbol{\Omega}}(t_k)$ (connected to $\hat{\boldsymbol{\omega}}(t_k)$ via (2)), $\hat{\mathbf{m}}(t_k)$ (corrected using (4)).
2. Compute constraint forces $\mathbf{c}_p(t_k)$, $\mathbf{c}_v(t_k)$, $\mathbf{c}_m(t_k)$, $\mathbf{c}_g(t_k)$ by solving the linear system (15)–(21) for the variables $\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$, $\mathbf{R}(t_k)$, $\mathbf{c}_p(t_k)$, $\mathbf{c}_v(t_k)$, $\mathbf{c}_m(t_k)$, $\mathbf{c}_g(t_k)$ (but ignore $\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$ and $\mathbf{R}(t_k)$ for now).
3. Compute what the new state $\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$, $\mathbf{R}(t_k)$ would be when we ignore the constraint forces. That is, solve (15)–(17) for $\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$, $\mathbf{R}(t_k)$, ignoring the terms in curly brackets. Call this intermediate result $\mathbf{p}_{\text{int}}$, $\mathbf{v}_{\text{int}}$, $\mathbf{R}_{\text{int}}$.

4. Get the new state:

$$\begin{aligned}\mathbf{p}(t_k) &= \mathbf{p}_{\text{int}} + g_p\mathbf{c}_p(t_k)\\ \mathbf{v}(t_k) &= \mathbf{v}_{\text{int}} + g_v\mathbf{c}_v(t_k)\\ \mathbf{R}(t_k) &= \mathbf{R}_{\text{int}} + g_m\mathbf{J}_m(\hat{\mathbf{m}}(t_k))\mathbf{c}_m(t_k) + g_g\mathbf{J}_g(\hat{\mathbf{a}}(t_k))\mathbf{c}_g(t_k)\end{aligned}$$

   where the correction factors g_p, g_v g_m and g_g are chosen non-negative numbers (see comments in paragraph (3.18)). In addition compute the acceleration $a(t_k)$ by

$$\mathbf{a}(t_k) = \frac{1}{\Delta_t}[\mathbf{v}(t_k) - \mathbf{v}(t_{k-1})].$$

5. Correct matrix $\mathbf{R}(t_k)$ back to orthogonality (see comments in paragraph (3.17)).

(3.17) To project the rotation $\mathbf{R}$ back to orthogonality two `matlab`/`octave` functions are provided: `[Rorth,flag,cR]=Matrix2Orthogonal(R)` finds the nearest orthogonal matrix for an arbitrary matrix `R` (reliable, not guaranteed to find a solution for matrices far away from orthogonality, slow). The other outputs indicate success (`flag`) and the magnitude and direction of the correction (`cR`). The function

```
[Rorth,cR]=ApproxOrthproj(Rprev,cprev,R)
```

which needs an initial guess `Rprev` and `cprev` is fast but gives good approximation for nearly orthogonal matrices.[2] A good initial guess for `Rprev` and `cprev` is the corresponding result from the previous time step.

The entire problem can be formulated equivalently using a quaternion $\mathbf{q}$ representing $\mathbf{R}$. However, then the equations (16), (17), (18) and (19) become nonlinear in the unknown $\mathbf{q}$.

(3.18) The factors g_p, g_v, g_m and g_g have to be chosen balancing the uncertainty of each constraint. Since the GPS data is only for rough guidance to avoid drift, the factors g_p and g_v have to be small. In the demonstration we chose $g_p = g_v = 0.05$. For the factors g_m and g_g we chose

$$g_m = \begin{cases} 1 & \text{if } |\hat{\boldsymbol{\omega}}| < 1.5 \\ 0.5 & \text{if } |\hat{\boldsymbol{\omega}}| \geq 1.5 \end{cases}, \qquad g_g = \begin{cases} 0.7 & \text{if } 7 < |\hat{\mathbf{a}}| < 10.3 \\ 0.2 & \text{if } |\hat{\mathbf{a}}| \geq 10.3 \\ 0 & \text{otherwise} \end{cases},$$

guided by the heuristics of the prior solution for attitude adjustment.

[2]The function `Matrix2Orthogonal` projects its input onto the nonlinear surface of orthogonal matrices. This is an iterative procedure. The function `ApproxOrthproj` performs a single step of this iteration.

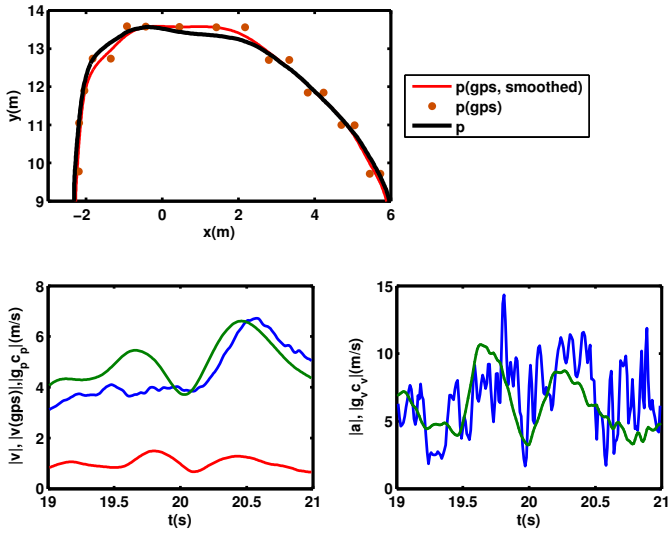


Figure 3: Segment of computational results following the steps in paragraph (3.16) for file 4 (`RIDE_2`). (top) (x, y)-components of path $\mathbf{p}(t)$, GPS data (points), smoothed and interpolated GPS data (red) and trajectory $\mathbf{p}(t)$ (black); (bottom-left) segment of speed time series, $|\mathbf{v}(t)|$ (blue), $|\mathbf{v}_{\text{gps}}(t)|$ (smoothed and interpolated from GPS, green), correction $|g_p \mathbf{c}_p(t)|$ (red); (bottom-right) segment of acceleration time series, $|\mathbf{a}(t)|$ (blue), $|g_v \mathbf{c}_v(t)|$ (green).

(3.19) Figure 3 shows a few representative segments of $\mathbf{p}(t)$, $\mathbf{v}(t)$ and $\mathbf{a}(t)$ and the components of the computation. As one can see in the top panel, the trajectory follows the (smoothed) GPS data closely despite the small prefactor $g_p = 0.05$. The smallness of g_p implies that the corrections are small compared to the speed $|\mathbf{v}(t)|$ (bottom-left panel). In contrast, the acceleration signal is composed of two signals of approximately equal magnitude: the correction $|g_v \mathbf{c}_v(t)|$ is of similar magnitude as the other part of $\mathbf{a}(t)$, $\mathbf{R}(t)\hat{\mathbf{a}}(t) + \mathbf{g}$ (the difference between blue and green curves in the bottom-right panel). In short, the low-frequency components of the resulting acceleration are determined by the GPS data, the high-frequency components are determined by the accelerometer.

(3.20) Figure 3 does not show the attitude $\mathbf{R}(t)$ (a tripod of orthogonal unit vectors). An animation `Ride2Segment.mp4` is included for a visualisation of $\mathbf{R}$ (along a segment of file 4 (`RIDE_2`).

3.6 Summary

(3.21) The method proposed in this section is a generalisation of the prior solution presented by the problem provider. This generalisation incorporates the GPS to avoid drift.

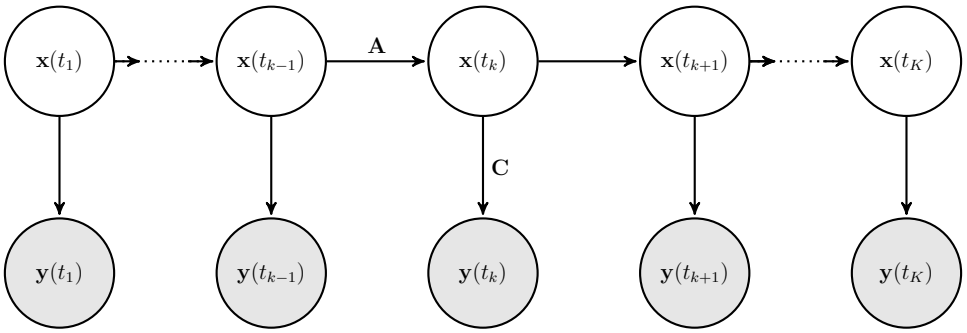


Figure 4: A graphical model of the statistical dependencies in a first-order Hidden Markov Model. The state at time t_k is denoted by $\mathbf{x}(t_k)$; the corresponding observation by $\mathbf{y}(t_k)$. The arrows indicate the statistical dependencies.

(3.22) However, the correctness (or at least plausibility) of the attitude is difficult to check without test data with known attitude. The combination of the update rule based on the gyroscope and the additional conditions from the magnetometer (18) and the approximate gravity (19) result in large corrections $g_m\mathbf{c}_m(t)$ and $g_g\mathbf{c}_g(t)$.

Strictly speaking, the constraint on gravity (19) is *redundant* as it follows from (16) if velocities incorporate information from GPS data (position or velocity).

4 Kalman filter based assimilation of all measurement data simultaneously

4.1 Background

(4.1) The presented problem fits into the framework of *Hidden Markov Models* (HMMs). A first-order Hidden Markov Model (HMM) consists of a time-series of state variables $\mathbf{x}(t_k)$, each of which is statistically dependent only on the state at the previous time-step $\mathbf{x}(t_{k-1})$ (the order indicates how many previous time-steps a state is dependent on), and an observation $\mathbf{y}(t_k)$ of each state, which is dependent only on the state $\mathbf{x}(t_k)$ for the same time-step. The state $\mathbf{x}(t_k)$ is called hidden (or *latent*) because the observation $\mathbf{y}(t_k)$ may provide only some components (or a function) of the state $\mathbf{x}(t_k)$ Figure 4 shows a diagram of a typical first-order HMM, which may be represented by the following expressions:

$$\mathbf{x}(t_k) = \mathbf{A}\mathbf{x}(t_{k-1}) + \boldsymbol{\epsilon}(t_k) \tag{23}$$

$$\mathbf{y}(t_k) = \mathbf{C}\mathbf{x}(t_k) + \mathbf{e}(t_k) \tag{24}$$

where $\boldsymbol{\epsilon}_n$ and $\mathbf{e}_n$ are noise variables. The matrices $\mathbf{A}$ and $\mathbf{C}$ are assumed to be known (but may depend on time t).

Where the variables of a first order HMM are continuous, the noise is Gaussian-distributed and the relationships between states, and between the states and the observations, are linear, then the HMM is known as a Kalman Filter [1, 2]. The Kalman Filter estimates the state at time t_k dependent on all the observations up to, and including, time t_k using an efficient recursive algorithm. Where the data are being processed offline (as they are in this case), all the observations, including those in the "future", may be included in the state estimation, a process known as a Kalman Smoother.

(4.2) Figure 5 shows the typical outcome of the application of a Bayesian Kalman Smoother to estimate the position of an object at 1 sec intervals (the red line) based on GPS locations recorded at 10 sec intervals (the blue line). The grey ellipses indicate the degree of uncertainty in the estimates (two standard deviations).

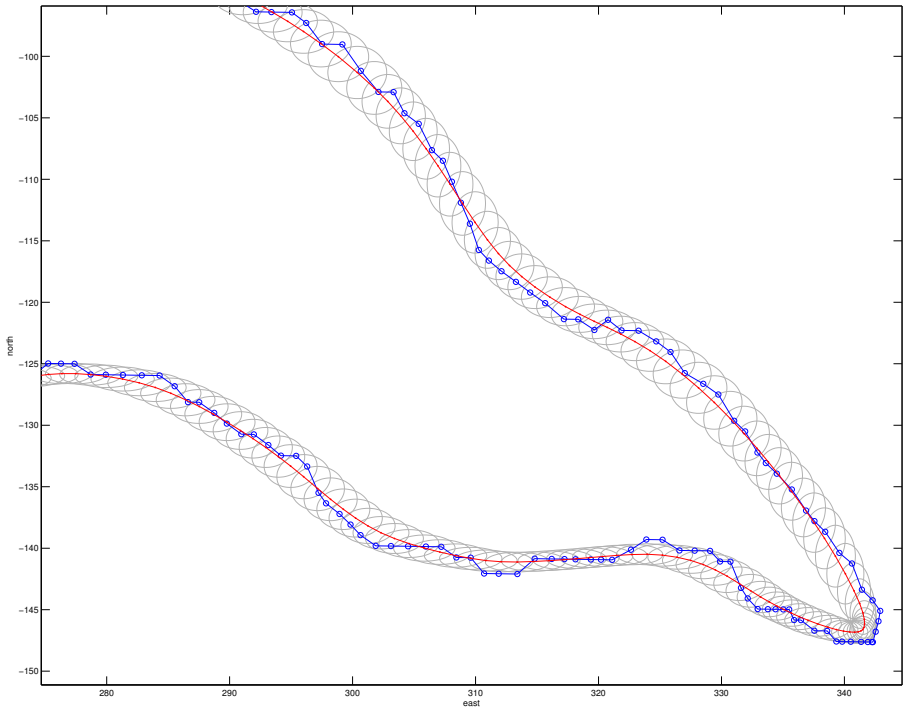


Figure 5: The results of using a Kalman Filter/Smoother against the GPS sensor data. In this model the noise is assumed to be heavier-tailed than Gaussian. The GPS track is shown in blue, the estimated track in red and the grey ellipses indicate the uncertainty in the estimates (2 standard deviations).

4.2 The model for the presented problem

(4.3) For the sensor data, we define the hidden (or *latent*) state $\mathbf{x}(t_k)$ to represent the truth, i.e. the actual location and orientation of the sensor in global

coordinates; we are only able to observe this state indirectly, through the readings from the sensor. The state at time t_k, $\mathbf{x}(t_k)$, is a vector composed of the following five elements, all defined with respect to the global coordinate system:

$$\mathbf{x}(t_k) = \begin{pmatrix} \mathbf{p}(t_k) \\ \mathbf{v}(t_k) \\ \mathbf{v}(t_{k-1}) \\ \mathbf{R}(t_k) \\ \mathbf{R}(t_{k-1}) \end{pmatrix} \tag{25}$$

where $\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$ and $\mathbf{R}(t_k)$ define the position, velocity and rotation matrix (to move from global coordinates to sensor coordinates) for the current time-step (see Table 2 in paragraph (3.1.1, but note that this is the inverse convention: the rotation $\mathbf{R}$ in this section is the transpose of the rotation $\mathbf{R}$ in Section 3). These are the truths that we wish to estimate. The $\mathbf{v}(t_{k-1})$ and $\mathbf{R}(t_{k-1})$ are included because they enter the observations at time t_k. This inclusion ensures that this is a first-order system.

(4.4) We simplify the model (15)–(17) to a constant velocity model, so that

$$\mathbf{p}(t_k) = \mathbf{p}(t_k) + \mathbf{v}(t_k)\Delta t + noise \tag{26}$$
$$\mathbf{v}(t_k) = \mathbf{v}(t_{k-1}) + noise \tag{27}$$
$$\mathbf{R}(t_k) = \mathbf{R}(t_{k-1}) + noise \tag{28}$$

where Δt is the time interval between observations. This linear evolution is represented by the (fixed) *transmission* matrix $\mathbf{A}$ (see figure 4 and (23)):

$$\mathbf{x}(t_k) = \begin{pmatrix} \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_9 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_9 & \mathbf{0} \end{pmatrix} \mathbf{x}(t_{k-1}) + \boldsymbol{\epsilon}(t_k) \tag{29}$$

where $\mathbf{I}_d$ represents the $d \times d$ identity matrix (remember that $\Delta t = t_k - t_{k-1} = 0.01\,\mathrm{s}$ in our case).

(4.5) The observation at time-step t_k, $\mathbf{y}(t_k)$, is a vector composed of the outputs of the sensors:

$$\mathbf{y}(t_k) = \begin{pmatrix} \mathbf{l}(t_k) \\ \hat{\mathbf{a}}(t_k) \\ \hat{\boldsymbol{\omega}}(t_k) \\ \hat{\mathbf{m}}(t_k) \end{pmatrix} \in \mathbb{R}^{12} \tag{30}$$

where $\mathbf{l}(t_k)$, $\hat{\mathbf{a}}(t_k)$, $\hat{\boldsymbol{\omega}}(t_k)$ and $\hat{\mathbf{m}}(t_k)$ are the GPS location, accelerometer, gyroscope and magnetometer readings respectively. The hat over the variable names indicates that they are in sensor coordinates (this convention is

different from Section 3). These variables are recorded at 100Hz. The GPS location is in global coordinates and is recorded only at 1Hz. The model for the observations (excluding the noise) is

$$\mathbf{l}(t_k) = \mathbf{p}(t_k) \tag{31}$$

$$\hat{\mathbf{a}}(t_k) = \mathbf{R}(t_k)\left[\frac{\mathbf{v}(t_k) - \mathbf{v}(t_{k-1})}{\Delta t} - \mathbf{g}\right] \tag{32}$$

$$\hat{\boldsymbol{\omega}}(t_k) = \mathbf{J}_{\boldsymbol{\omega}}\mathbf{R}(t_k)^T\mathbf{R}(t_{k-1})/\Delta t \tag{33}$$

$$\hat{\mathbf{m}}(t_k) = \mathbf{R}(t_k)\mathbf{n}. \tag{34}$$

In (33) the operation $\mathbf{J}_{\boldsymbol{\omega}}$ extracts the appropriate three entries of a 3×3 matrix to recover $\hat{\boldsymbol{\omega}}$ from $\hat{\boldsymbol{\Omega}}$:

$$\mathbf{J}_{\boldsymbol{\omega}}\mathbf{M} = \begin{pmatrix} M_{2,3} \\ M_{3,1} \\ M_{1,2} \end{pmatrix}, \text{ such that } \mathbf{J}_{\boldsymbol{\omega}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{35}$$

if we treat $\hat{\boldsymbol{\Omega}}$ as a 9×1 vector. Note that (31)–(34) are identical to (15)–(18) and (20) (ignoring the terms in curly brackets and re-arrranging for the observations instead of the state at the new time).

(4.6) There are a number of issues associated with this approach:

1. The relationship between $\mathbf{y}(t_k)$ and $\mathbf{x}(t_k)$ is not linear, that is, there is no matrix $\mathbf{C}(t_k)$ such that $\mathbf{y}(t_k) = \mathbf{C}(t_k)\mathbf{x}(t_k) + \textit{noise}$.
2. The rotation matrices (the $\mathbf{R}(t_k)$) must be constrained so that they remain valid rotation matrices, i.e. orthonormal.
3. It is not clear that the noise is Gaussian. In other trials, GPS noise has been seen to be heavier-tailed than Gaussian [4]. It is well-known that Gaussian distribution estimation is badly affected by the presence of outliers [5].

There are non-linear versions of the Kalman Filter, for example, the extended Kalman Filter and unscented Kalman Filter [10, 9]. None of the team are familiar with these techniques and, since the time interval between observations is very small, it was decided to use a standard, linear Kalman Filter/Smoother iteratively. In the interests of expediency, the GPS locations were spline-interpolated to 100Hz (using `interp1` immediately, in contrast to `mpsmooth` used in Section 3.4), making the observations $\mathbf{l}(t)$ available at every time step t_k.

4.3 Iterative procedure

(4.7) *Linearisation* We can treat the problem by applying the linear techniques iteratively. Given a reference trajectory $\mathbf{x}_{\text{ref}}(t_k)$, we use the model for the output (31)–(34) to obtain the output $\mathbf{y}_{\text{ref}}(t_k)$ that would correspond to

the reference trajectory $\mathbf{x}_{\text{ref}}(t_k)$. Then the true state $\mathbf{x}(t_k)$ and the true observation $\mathbf{y}(t_k)$ are treated as small deviations from their reference values:

$$\mathbf{x}(t_k) = \mathbf{x}_{\text{ref}}(t_k) + \Delta\mathbf{x}(t_k)$$
$$\mathbf{y}(t_k) = \mathbf{y}_{\text{ref}}(t_k) + \Delta\mathbf{y}(t_k).$$

These deviations $\Delta\mathbf{x}$ (unknown) and $\Delta\mathbf{y}$ (known) now fit into the linear framework (23), (24). The transmission matrix $\mathbf{A}$ is as given in (29). The emission matrix $\mathbf{C}$ depends on the reference state $\mathbf{x}_{\text{ref}}(t_k)$ (and, thus, on t_k). Section 4.4 will describe how the initial reference $\mathbf{x}(t_k)$ is chosen. Section 4.7 gives the details how it is updated in subsequent iterations.

(4.8) *Linearised emission matrix* At each time-step an emission matrix $\mathbf{C}(t_k)$ is constructed based on a linearisation of the observation model (31)–(34) with respect to the state variables $\mathbf{p}(t_k)$, $\mathbf{v}(t_k)$, $\mathbf{v}(t_{k-1})$, $\mathbf{R}(t_k)$ and $\mathbf{R}(t_{k-1})$. The matrix $\mathbf{C}$, which depends on time, consists of 5 columns (each column is itself a matrix consisting of several columns, see below for explanations of the entries), $\mathbf{C}_{\mathbf{p}} \in \mathbb{R}^{12\times 3}$, $\mathbf{C}_{\mathbf{v}(t_k)} \in \mathbb{R}^{12\times 3}$, $\mathbf{C}_{\mathbf{v}(t_{k-1})} \in \mathbb{R}^{12\times 3}$, $\mathbf{C}_{\mathbf{R}(t_k)} \in \mathbb{R}^{12\times 9}$, $\mathbf{C}_{\mathbf{R}(t_{k-1})} \in \mathbb{R}^{12\times 9}$:

$$\mathbf{C}_{\mathbf{p}} = \begin{pmatrix} \mathbf{I}_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{C}_{\mathbf{v}(t_k)} = \begin{pmatrix} 0 \\ \frac{1}{\Delta t}\mathbf{R}_{\text{ref}}(t_k) \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{C}_{\mathbf{v}(t_{k-1})} = \begin{pmatrix} 0 \\ \frac{-1}{\Delta t}\mathbf{R}_{\text{ref}}(t_k) \\ 0 \\ 0 \end{pmatrix}$$
$$\mathbf{C}_{\mathbf{R}(t_k)} = \begin{pmatrix} 0 \\ (\cdot)\left[\frac{\mathbf{v}_{\text{ref}}(t_k)-\mathbf{v}_{\text{ref}}(t_{k-1})}{\Delta t} - \mathbf{g}\right] \\ \mathbf{J}_{\boldsymbol{\omega}}(\cdot)^T\mathbf{R}_{\text{ref}}(t_{k-1})/\Delta t \\ (\cdot)\mathbf{n} \end{pmatrix}, \quad \mathbf{C}_{\mathbf{R}(t_{k-1})} = \begin{pmatrix} 0 \\ 0 \\ \mathbf{J}_{\boldsymbol{\omega}}\mathbf{R}_{\text{ref}}(t_k)^T(\cdot)/\Delta t \\ 0 \end{pmatrix}. \tag{36}$$

Terms with the subscript ref are reference values. In the iterative procedure described in sections 4.6 and 4.7, their values will be the state estimates from the previous iteration.

The second to fourth entries of $\mathbf{C}_{\mathbf{R}(t_k)}$ are 3×9 matrices, operating on a 9×1 vector $\mathbf{r}$ by first reshaping $\mathbf{r}$ into a 3×3 matrix $\mathbf{R}$ and then performing the indicated operation on $\mathbf{R}$. The matrices can be obtained by the `matlab`/`octave` commands

```
CRtk2=kron(dvg',eye(3));
CRtk3=Jom*reshape(permute(reshape(kron(rref1/dt,eye(3)),[3,3,3,3]),...
   [1,4,2,3]),9,9);
CRtk4=kron(north',eye(3));
```

if `dvg` is the second term (in square brackets) in $\mathbf{C}_{\mathbf{R}(t_k),2}$, `rref1` is $\mathbf{R}_{\text{ref}}(t_{k-1})$, `dt` is Δt, `Jom` is the 3×9 matrix representation of $\mathbf{J}_{\boldsymbol{\omega}}$ in (35), and `north` is the magnetic north vector $\mathbf{n}$. Similarly, the third entry in $\mathbf{C}_{\mathbf{R}(t_{k-1})}$ is obtained by

```
CRtkm1=Jom/dt*kron(eye(3),rref0');
```

where `rref0` is $\mathbf{R}_{\text{ref}}(t_k)$.

Concatenating all columns of $\mathbf{C}$ the linearized model for the observations is

$$\Delta\mathbf{y}(t_k) = \mathbf{C}(\mathbf{x}_{\text{ref}}(t_k))\Delta\mathbf{x}(t_k), \quad \text{where}$$
$$\mathbf{C}(\mathbf{x}_{\text{ref}}(t_k)) = \begin{bmatrix}\mathbf{C}_{\mathbf{p}} & \mathbf{C}_{\mathbf{v}(t_k))} & \mathbf{C}_{\mathbf{v}(t_{k-1}))} & \mathbf{C}_{\mathbf{R}(t_k))} & \mathbf{C}_{\mathbf{R}(t_{k-1}))}\end{bmatrix}$$

(4.9) *Updating the reference iteratively* As the deviations $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$ satisfy approximately a linear model, application of the linear algorithm (described in Section 4.6) results in an estimate for the mean and covariance of the unknown $\Delta\mathbf{x}$. Then we can update the reference trajectory to the new and (hopefully) more accurate reference

$$\mathbf{x}_{\text{ref,new}}(t_k) = \mathbf{x}_{\text{ref,old}}(t_k) + \Delta\mathbf{x}(t_k) \tag{37}$$

and update $\mathbf{y}_{\text{ref}}(t_k)$ using (31)–(34) accordingly. Then we repeat the linear algorithm with the linearised emission matrix $\mathbf{C}$ in the new reference $\mathbf{x}_{\text{ref,new}}(t_k)$. Over multiple iterations of the estimation process it is anticipated that the linearisation will gradually converge.

4.4 Initial guesses

Before the iteration can be started we need an initial reference trajectory $\mathbf{x}_{\text{ref}}(t_k)$. The position and velocity component for the initial $\mathbf{x}_{\text{ref}}(t_k)$ are taken from the (spline-interpolated) GPS data $\mathbf{l}(t_k)$ and its time derivative (see Section 3.4, how to extract this data). The rotation components of $\mathbf{x}(t_k)$ assume that the sensor is initially pointed in the direction of travel. Alternatively, results obtained from the methods in Section 3.5 can be used as initial reference.

4.5 Expectation (E) Maximisation (M) iteration

Assuming that the disturbances in (23) and (24) have a Gaussian distribution, we get Gaussian (also called *normal*) distributions for $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$, too:

$$\Delta\mathbf{x}(t_k) \sim \mathcal{N}\Big(\Delta\mathbf{x}(t_k) \,|\, \mathbf{A}\Delta\mathbf{x}(t_{k-1}), \mathbf{\Phi}\Big) \tag{38}$$

$$\Delta\mathbf{y}(t_k) \sim \mathcal{N}\Big(\Delta\mathbf{y}(t_k) \,|\, \mathbf{C}(\mathbf{x}_{\text{ref}}(t_k))\Delta\mathbf{x}(t_k), \mathbf{\Psi}\Big) \tag{39}$$

(read, for example, the first as "$\Delta\mathbf{x}(t_k)$ has normal distribution with mean $\mathbf{A}\Delta\mathbf{x}(t_{k-1})$ and covariance matrix $\mathbf{\Phi}$"). We implement the Expectation Maximisation (EM) algorithm [7, 8] to estimate the unknown means $\boldsymbol{\mu}(t_k)$ and covariance matrices $\mathbf{\Phi}$ for the states $\Delta\mathbf{x}(t_k)$ at the time steps t_k ($k = 2 \ldots K$). The procedure works iteratively, alternating the E step and the M step described in the following subsections.

4.6 E step

(4.10) Using the forward-backward algorithm [3], also known as the Baum-Welch algorithm [6], (for a tutorial see [5]), the E step is calculated in two parts: after the forward sweep (from t_0 to t_K) we arrive at a sequence of means $\boldsymbol{\mu}^\alpha(t_k)$ and covariances $\boldsymbol{\Sigma}^\alpha(t_k)$ for the probability distributions of $\mathbf{x}(t_k)$ taking into account only the observations $\mathbf{y}(t_1), \ldots, \mathbf{y}(t_k)$. The superscript α indicates that these are "forward" results, depending only on past and current observations.

(4.11) *Initial guesses* The E steps starts with an initial guess $\boldsymbol{\mu}^\alpha(t_0) \in \mathbb{R}^{27}$ (a guess for mean of the state $\mathbf{x}$ at t_0), a guess for $\boldsymbol{\Sigma}(t_0) \in \mathbb{R}^{27\times 27}$, and guesses for the covariance matrices $\boldsymbol{\Phi} \in \mathbb{R}^{27\times 27}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{12\times 12}$ (estimates for the covariances of the state $\mathbf{x}$ and the observation $\mathbf{y}$). At the initial iterate $\boldsymbol{\mu}^\alpha(t_0)$ is zero. This corresponds to starting on the reference trajectory $\mathbf{x}_{\text{ref}}(t_0)$ as defined in Section 4.4.

The initial matrix $\Sigma(t_0)$ is a diagonal matrix, with the location and velocity diagonal elements set to 10 and the rotation elements set to 0.1. The matrix Φ is initially the identity matrix $\mathbf{I}_{27}$, and the matrix $\boldsymbol{\Psi}$ is a diagonal matrix, with 10 for the location, 1 for the acceleration, 0.05 for the gyro and 0.05 for the magnetometer (numbers in the covariance matrix Ψ are guided by the tolerances of the measurements in the units used for computation).

While these initial values are fairly arbitrary, from the second iterate onward the initial values will be provided by the M step (see (55)–(58)).

(4.12) *Forward sweep* At the first time step we set

$$\mathbf{P}(t_0) = \boldsymbol{\Sigma}(t_0) \tag{40}$$

$$\boldsymbol{\mu}^\alpha(t_1) = \boldsymbol{\mu}^\alpha(t_0) + \mathbf{G}(t_1)\Big(\mathbf{y}(t_1) - \mathbf{C}(t_1)\boldsymbol{\mu}^\alpha(t_0)\Big), \tag{41}$$

$$\boldsymbol{\Sigma}^\alpha(t_1) = \Big(\mathbf{I} - \mathbf{G}(t_1)\mathbf{C}(t_1)\Big)\mathbf{P}(t_0). \tag{42}$$

($\boldsymbol{\Sigma}^\alpha(t_0) = \boldsymbol{\Sigma}(t_0)$, and see (46) for definition of $\mathbf{G}$). For times t_k with $k > 1$ we set:

$$\mathbf{P}(t_{k-1}) = \mathbf{A}\boldsymbol{\Sigma}^\alpha(t_{k-1})\mathbf{A}^{\mathrm{T}} + \boldsymbol{\Phi} \tag{43}$$

$$\boldsymbol{\mu}^\alpha(t_k) = \mathbf{A}\boldsymbol{\mu}^\alpha(t_{k-1}) + \mathbf{G}(t_k)\Big(\mathbf{y}(t_k) - \mathbf{C}(t_k)\mathbf{A}\boldsymbol{\mu}^\alpha(t_{k-1})\Big) \tag{44}$$

$$\boldsymbol{\Sigma}^\alpha(t_k) = \Big(\mathbf{I} - \mathbf{G}(t_k)\mathbf{C}(t_k)\Big)\mathbf{P}(t_{k-1}). \tag{45}$$

For each time step $k \geq 1$ the matrix $\mathbf{G}(t_k)$, known as the Kalman gain matrix, is defined by

$$\mathbf{G}(t_k) = \mathbf{P}(t_{k-1})\mathbf{C}(t_k)^{\mathrm{T}}\Big(\mathbf{C}(t_k)\mathbf{P}(t_{k-1})\mathbf{C}(t_k)^{\mathrm{T}} + \boldsymbol{\Psi}\Big)^{-1} \tag{46}$$

Remember that the definition of the emission matrix $\mathbf{C}(t_k)$ at each step requires references values $\mathbf{x}_{\text{ref}}(t_k)$ that are in the first iteration chosen as described in Section 4.4. In later iterations we have estimates for the means, which we can use: $\mathbf{x}_{\text{ref}}(t_k) = \boldsymbol{\mu}(t_k)$ from the previous iteration.
In a modification of this classical algorithm, after each time step we extract the components of $\boldsymbol{\mu}^\alpha(t_k)$ corresponding to $\mathbf{R}(t_k)$ and $\mathbf{R}(t_{k-1})$ (components 10 to 18 and 19 to 27 of $\boldsymbol{\mu}$) and re-orthogonalise them (using the function `Matrix2Orthogonal` provided). This correction is applied after each step k.

(4.13) *Backward sweep* The backwards sweep starts at the final time t_K. Let us define as $\mathbf{Y}$ the matrix $(\mathbf{y}(t_1), ..., \mathbf{y}(t_K))$ of ordered observations. The backwards sweep produces means $\boldsymbol{\mu}(t_k) \in \mathbb{R}^{27}$ and covariances $\boldsymbol{\Sigma}(t_k) \in \mathbb{R}^{27\times 27}$ for the *posterior distribution* of $\mathbf{x}(t_k)$ (note that the missing α indicates that these quantities depend on all observations, past and future). At time t_K we start with

$$\boldsymbol{\mu}(t_K) = \boldsymbol{\mu}^\alpha(t_K) \tag{47}$$
$$\boldsymbol{\Sigma}(t_K) = \boldsymbol{\Sigma}^\alpha(t_K). \tag{48}$$

Then, for k decreasing from $K-1$ to 0 we define

$$\mathbf{J}(t_k) = \boldsymbol{\Sigma}(t_k)\mathbf{A}^{\mathrm{T}}\mathbf{P}(t_k)^{-1} \tag{49}$$
$$\boldsymbol{\mu}(t_k) = \boldsymbol{\mu}^\alpha(t_k) + \mathbf{J}(t_k)\Big(\boldsymbol{\mu}(t_{k+1}) - \mathbf{A}\boldsymbol{\mu}^\alpha(t_k)\Big) \tag{50}$$
$$\boldsymbol{\Sigma}(t_k) = \boldsymbol{\Sigma}^\alpha(t_k) + \mathbf{J}(t_k)\Big(\boldsymbol{\Sigma}(t_{k+1}) - \mathbf{P}(t_k)\Big)\mathbf{J}(t_k)^{\mathrm{T}}. \tag{51}$$

Similar to the forward sweep the components of $\boldsymbol{\mu}(t_k)$ corresponding to $\mathbf{R}(t_k)$ and $\mathbf{R}(t_{k-1})$ are re-orthogonalised (for example, using the function `Matrix2Orthogonal`) after each step k.

(4.14) For the M (maximisation) step the following expectations are required:

$$\langle \mathbf{x}(t_k) \,|\, \mathbf{Y} \rangle = \boldsymbol{\mu}(t_k) \tag{52}$$
$$\langle \mathbf{x}(t_k)\mathbf{x}(t_{k-1})^{\mathrm{T}} \,|\, \mathbf{Y} \rangle = \mathbf{J}(t_{k-1})\boldsymbol{\Sigma}(t_k) + \boldsymbol{\mu}(t_k)\boldsymbol{\mu}(t_{k-1})^{\mathrm{T}} \tag{53}$$
$$\langle \mathbf{x}(t_k)\mathbf{x}(t_k)^{\mathrm{T}} \,|\, \mathbf{Y} \rangle = \boldsymbol{\Sigma}(t_k) + \boldsymbol{\mu}(t_k)\boldsymbol{\mu}(t_k)^{\mathrm{T}} \tag{54}$$

(read, for example, $\langle \mathbf{x}(t_k) \,|\, \mathbf{Y} \rangle$ as "expectation of $\mathbf{x}(t_k)$ given observations $\mathbf{Y}$").

4.7 M step

(4.15) The maximisation step updates the initial guesses that have entered the E step at time t_0 and the covariance matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ (expectation dependencies on $\mathbf{Y}$ have been omitted to aid readability):

$$\boldsymbol{\mu}^\alpha(t_0) = \langle \mathbf{x}(t_1) \rangle \tag{55}$$

$$\mathbf{\Sigma}(t_0) = \langle \mathbf{x}(t_1)\mathbf{x}(t_1)^{\mathrm{T}} \rangle - \langle \mathbf{x}(t_1) \rangle \langle \mathbf{x}(t_1) \rangle^{\mathrm{T}} \tag{56}$$

$$\begin{aligned} \mathbf{\Phi} = & \frac{1}{K-1} \sum_{k=2}^{K} \Big[\langle \mathbf{x}(t_k)\mathbf{x}(t_k)^{\mathrm{T}} \rangle - \mathbf{A} \langle \mathbf{x}(t_{k-1})\mathbf{x}(t_k)^{\mathrm{T}} \rangle \\ & - \langle \mathbf{x}(t_k)\mathbf{x}(t_{k-1})^{\mathrm{T}} \rangle \mathbf{A}^{\mathrm{T}} + \mathbf{A} \langle \mathbf{x}(t_{k-1})\mathbf{x}(t_{k-1})^{\mathrm{T}} \rangle \mathbf{A}^{\mathrm{T}} \Big] \end{aligned} \tag{57}$$

$$\begin{aligned} \mathbf{\Psi} = & \frac{1}{K} \sum_{k=1}^{K} \Big[\mathbf{y}(t_k)\mathbf{y}(t_k)^{\mathrm{T}} - \mathbf{C}(t_k) \langle \mathbf{x}(t_k) \rangle \mathbf{y}(t_k)^{\mathrm{T}} \\ & - \mathbf{y}(t_k) \langle \mathbf{x}(t_k) \rangle^{\mathrm{T}} \mathbf{C}(t_k)^{\mathrm{T}} + \mathbf{C}(t_k) \langle \mathbf{x}(t_k)\mathbf{x}(t_k)^{\mathrm{T}} \rangle \mathbf{C}(t_k)^{\mathrm{T}} \Big]. \end{aligned} \tag{58}$$

The means $\boldsymbol{\mu}(t_k)$ generated in (52) will be used to create the new reference trajectory $\mathbf{x}_{\mathrm{ref}}(t_k)$ in the next iteration: $\boldsymbol{\mu}(t_k)$ is inserted as $\Delta\mathbf{x}(t_k)$ in (37).

4.8 Initial results

(4.16) Figure 6 shows very preliminary results of using five iterations of the Kalman Filter/Smoother against a short sequence of the data. It is interesting to note that the estimated uncertainty in the easterly direction is considerably bigger than in either of the other two directions (standard deviation approximately 1.7 metres as opposed to 0.6 m and 0.2 m).

5 Wavelet smoothing and tests of Android phone

5.1 Wavelet smoothing

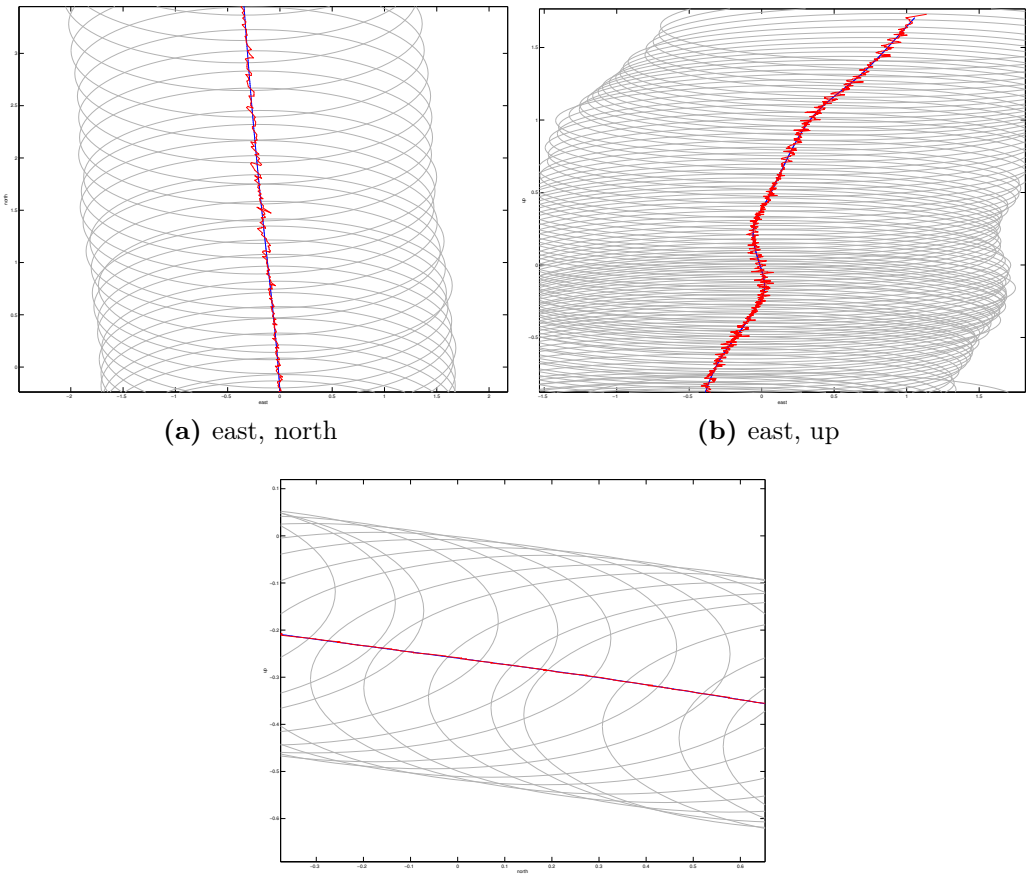
(5.1) Since the sensor is believed to output rather noisy data, it may be useful to pre-process the raw datasets by smoothing them. One of the best-known methods for smoothing a noisy signal is based on the wavelet transform. This technique is nowadays widely used in many fields such as signal analysis, econometrics or image processing.

The wavelet transform uses a projection of a signal onto an orthonormal set of components. It provides us a technique for time frequency localization, with many similarities to the Fourier transform. However, since wavelets are localized in time, they can distinguish local events at different moments in time.

(5.2) Any signal can be represented as a sequence:

$$f(t) = S_J + D_{J-1} + \cdots + D_1,$$

where J is the approximation level such as 2^J, which is the maximum scale sustainable by the length of our signal. Then, S is so called *father wavelet* (representing smoothed signal) and D is *mother wavelet* (representing noise; when its subscript increases, the frequency of the noise increases as well). A



(a) east, north

(b) east, up

(c) north, up

Figure 6: Initial results from the Kalman Filter/Smoother, showing the interpolated GPS track (in blue), the track estimated from all the sensor data (in red) and the uncertainty (as grey ellipses) as one standard deviation; the latter are drawn for every 10th observation. For clarity, only a short sections of the track are displayed. In each plot the scales of the two axes are the same.

one level higher approximation of the signal is obtained by decomposing the actual refined signal into approximation and details and then by subtracting the latter part, i.e. $S_{J+1} = S_J - D_{J+1}$. Figure 7 presents an example of wavelet smoothing.

As mentioned earlier in 2.5, in our case the sensor outputs three vectors: linear acceleration, angular velocity and magnetic field. Even when the data include some noise, without pre-processing it can be filtered out by other methods, e.g., by Kalman Filter (see 4). It is not clear which of those require smoothing, if any. We strongly recommend testing different approximation of all three signals for further research.

(5.3) During the week we have also examined several scientific articles. Although there are many of them, none was describing exactly the same challenge we were facing. On of the papers, published in Robotica [11], describes

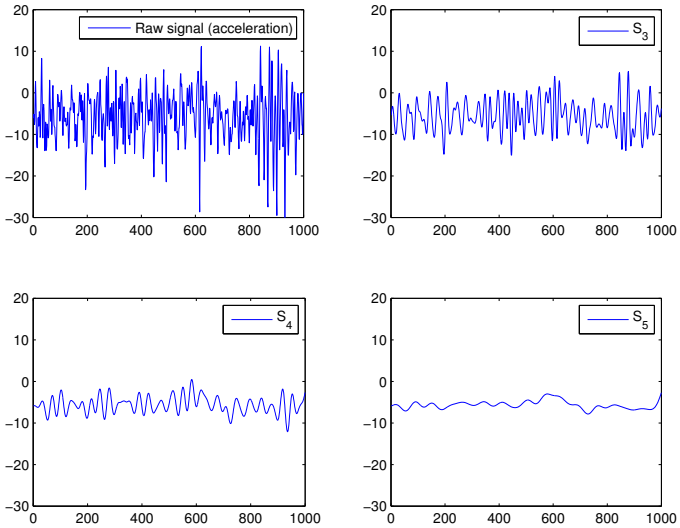


Figure 7: Example of wavelet smoother performed on linear acceleration of the sensor with respect to x axis. With every next level of approximation, the signal gets smoother.

tracking robots. The authors state that noise generated by a sensor during movement is strictly dependent on acceleration. Therefore, they suggest de-noising the data in two steps: the first - using wavelet transform same as described above. The second, called variable thresholding method, is according to their study a way to filter out velocity-dependent signal. We have run a similar analysis and had **not** found such a relationship. The datasets from the sensor located on a cyclist's helmet do not seem to have any velocity-dependent noise.

5.2 Android device as a sensor

(5.4) Modern smartphones are often equipped with an Android operating system and sensors such as a magnetometer, a gyroscope, an accelerometer and GPS. Therefore, we can use these built-in sensors to track the movement of the smartphone.

(5.5) Current technologies deliver software which can be used to analyse and depict the position of a smartphone in real time. One possible Android application is **M**LConnect. It sends raw values of the magnetometer, gyroscope and accelerometer to Matlab using a Wi-Fi connection. One needs to install MLConnect applications on the Android device, use the API provided on `http://mlconnect.chschmid.com` on the computer and connect both devices to the same network. An example of how to read the values of the available sensors is provided at the mentioned website.

(5.6) The Android device can be used as an inertial measurement unit which sends

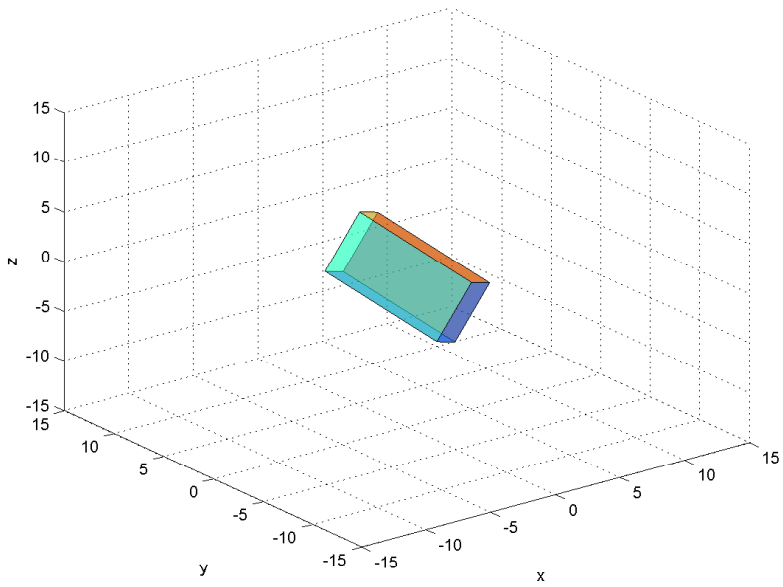


Figure 8: Orientation of the phone based on the gyroscope data.

the data in real time to (for example) Matlab using a Wi-Fi connection. Figure 8 shows the result of an application of this set of tools. Integrated data from the gyroscope were used to obtain an approximate orientation. Accelerometer data increases the precision (exploiting that gravity produces an upward acceleration). These computations can be performed in real time. Tests have shown that provided setup can be used to check and improve solution found in 3.5.

6 Conclusions and suggestions for next steps

(6.1) There are several ways to incorporate the GPS data and avoid drift. For example, a simple generalization of the algorithm provided by the problem presenter achieves this.

(6.2) A major problem (as had already been pointed out by the problem presenter) is to ensure that the attitude (that is, head orientation) is tracked correctly. At the moment, the only way to validate the attitude is a check for plausibility: if one represents the attitude as a rotation matrix $\mathbf{R}$ and the current velocity as $\mathbf{v}$ then $\mathbf{w} = \mathbf{R}^T\mathbf{v}$ is a vector describing the attitude relative to the current direction of motion. Implausible ranges of $\mathbf{w}$ (visible in some parts of the animations) point to problems. However, even plausible ranges of $\mathbf{w}$ are no guarantee for correct attitude. Thus, it is impossible to determine the true effectiveness of the proposed solutions with regard to accuracy of the resulting speed and accelerations because of this uncertainty

in the attitude. In particular, it is hard to distinguish what degree of correctness can be achieved by improving the algorithms without a reference case to benchmark the algorithms.

(6.3) One method would be to generate synthetic data which conform to our understanding of the underlying dynamics and which have known noise models. However, the only true test is to record sensor readings in a tightly controlled environment, where the actual location and attitude of the sensor is known at all times over a reasonable period of time.

Bibliography

[1] Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the American Society for Mechanical Engineering, Series D, Journal of Basic Engineering*, 82:35–45.

[2] Kalman, R. and Bucy, R. (1961). New results in linear filter and prediction theory. *Journal of Basic Engineering*, 83:95–108.

[3] Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

[4] Christmas, J., not yet published.

[5] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer, New York.

[6] Baum, L. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, 3:1–8.

[7] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

[8] McLachlan, G. and Krishnan, T. (1997). *The EM algorithm and extensions*. John Wiley & Sons Inc, New York.

[9] Wan, E.A. and van der Merwe, R. (2000). The Unscented Kalman Filter for Nonlinear Estimation. *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC) 2000*.

[10] Julier, S.J. and Uhlmann, J.K. (1997). A New Extension of the Kalman Filter to Nonlinear Systems. *Proceedings of the 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls (AeroSense)*.

[11] W. Seo, S. Hwang, J. Park, J. Lee (2013). Precise outdoor localization with a GPS-INS integration system. Robotica, 31, pp 371-379.