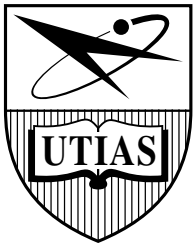
State Estimation for Aerospace Vehicles

– AER1513 Course Assignments –

A LEAST-SQUARES TRILOGY



Timothy David Barfoot
Institute for Aerospace Studies
University of Toronto
4925 Dufferin Street
Toronto, Ontario
Canada M3H 5T6
<tim.barfoot@utoronto.ca>

AER1513-003, Rev: 2.2
September 8, 2020

Contents

List of Figures

Notation

a	Symbols in this font are real scalars.
$\mathbf{a}$	Symbols in this font are real column vectors.
$\mathbf{A}$	Symbols in this font are real matrices.
$\mathcal{A}$	Symbols in this font are sigma-points used in the Unscented Transformtion, except $\mathcal{N}$, $\mathcal{F}$, and $\mathcal{O}$, which have special meanings.
$p(\mathbf{a})$	The probability density of $\mathbf{a}$.
$p(\mathbf{a}\|\mathbf{b})$	The probability density of $\mathbf{a}$ given $\mathbf{b}$.
$\sim \mathcal{N}(\mathbf{a}, \mathbf{B})$	Normally distributed with mean $\mathbf{a}$ and covariance $\mathbf{B}$.
$\mathcal{O}$	Observability matrix.
$(\cdot)_k$	The value of a quantity at timestep k.
$(\cdot)_{k_1:k_2}$	The set of values of a quantity from timestep k_1 to timestep k_2, inclusive.
$\mathbf{a}^{(m)}$	Superscript (m) is the index of a sample drawn from a density.
$\underrightarrow{\mathcal{F}}_a$	A vectrix representing a reference frame in three dimensions.
$\underrightarrow{a}$	A vector quantity in three dimensions.
$(\cdot)^\times$	The cross-product operator which produces a skew-symmetrix matrix from a column.
$\mathbf{1}$	The identity matrix.
$\mathbf{0}$	The zero matrix.

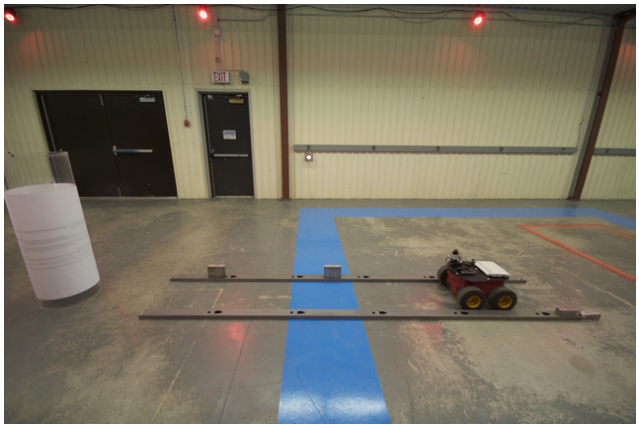
Assignment 1

'Giant Glass of Milk Dataset'

In this first assignment we'll investigate a linear one-dimensional problem consisting of a robot driving back and forth in a straight line. We wish to estimate the position of the robot along this line throughout its ≈ 280 m journey. We'll use the batch linear-Gaussian estimator to fuse speed measurements coming from wheel odometry with range measurements coming from a laser rangefinder.

1.1 Experimental Setup

The setup consists of a mobile robot driving back and forth between two fixed rails as depicted in Figure 1.1. The robot, the rails, and the large cylinder were equipped with reflective markers and tracked using a ten-camera motion capture system. This motion capture system is able to provide the position of each reflective marker to within a few



(a) Mobile robot driving between rails. Robot has an odometer to measure translational speed and a laser rangefinder to measure range to the large cylinder. Both sensors are noisy.

(b) Vicon motion capture lab. Ten cameras work together to track markers on the robot and provide groundtruth position/orientation.

Figure 1.1: Setup for Dataset 1.

millimeters. Position estimates from this motion capture system are considered to be quite a bit more accurate than estimates based on the robot's onboard sensors and thus it serves as the benchmark/groundtruth in our experiment.

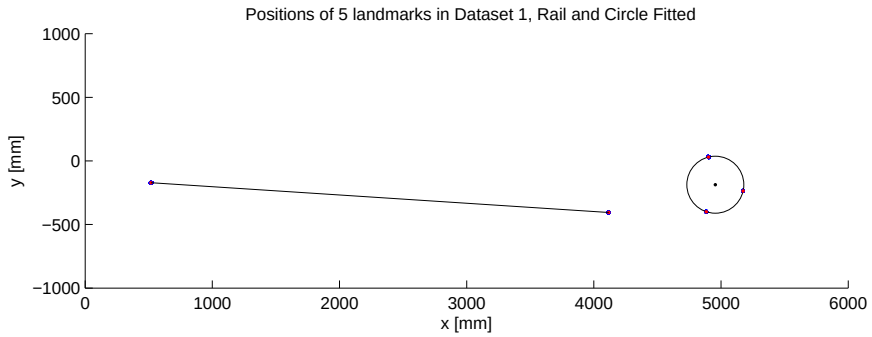


Figure 1.2: Circle and Rail for Dataset 1.

There were two reflective markers placed on the rail to the robot's right and three along the circumference of the large cylinder. The positions of these were measured using the motion capture system for 20 minutes and then averaged. A circle was fit to the points on the cylinder and its center found. A reference line was drawn between the two points on the rail. The output of these steps can be seen in Figure 1.2.

The mobile robot was driven back and forth for 20 minutes and three streams of data were logged:

– laser rangefinder scans consisting of 681 range measurements spread over a 240° horizontal field of view centered on straight ahead (Hokuyo URG-04LX sensor), logged at 10 Hz

– robot's speed based on a wheel odometry, logged at 10 Hz

– ground position of a marker on the laser rangefinder origin, logged at 70 Hz

Figure 1.3 shows the robot's path over the 20 minute trial. We see that the robot's path is very straight and consistent. We may therefore think of this as a one-dimensional estimation problem for this assignment.

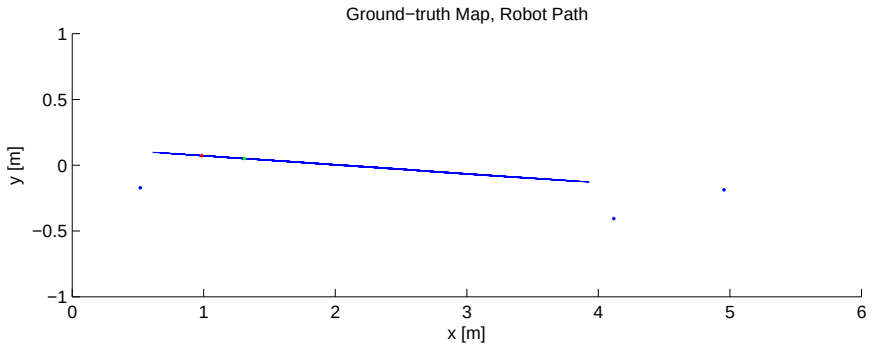


Figure 1.3: Robot Path for Dataset 1. Two points on rail and cylinder center shown for reference.

As with most real datasets, our data streams do not arrive synchronously or with evenly-spaced timesteps. Figure 1.4 shows the variability in the sampling periods of the nominally 10 Hz laser scans. For our purposes, we will

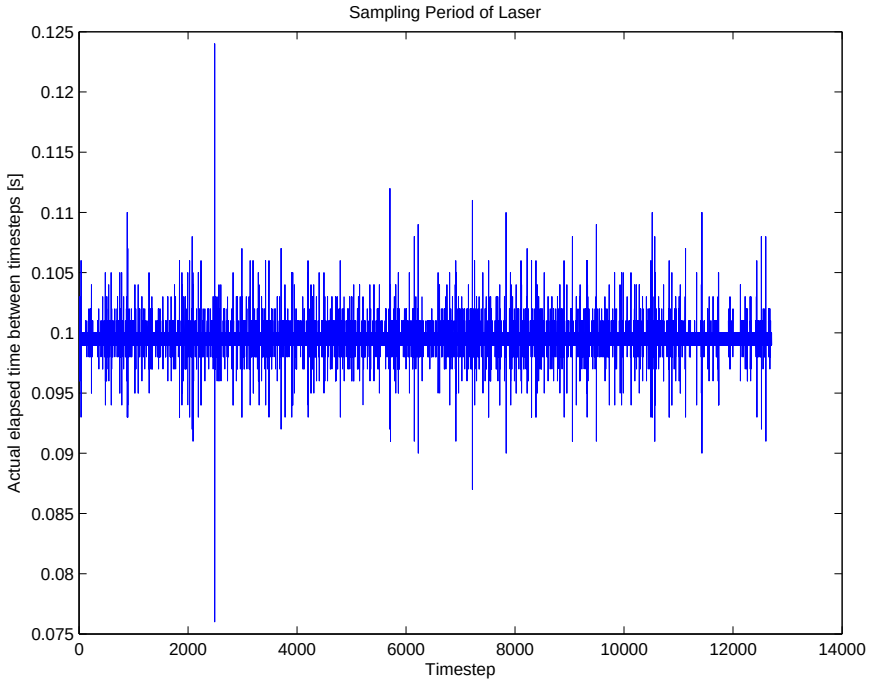


Figure 1.4: Sampling periods of the 10 Hz laser rangefinder scans. We see that the period does hover around 0.1 seconds.

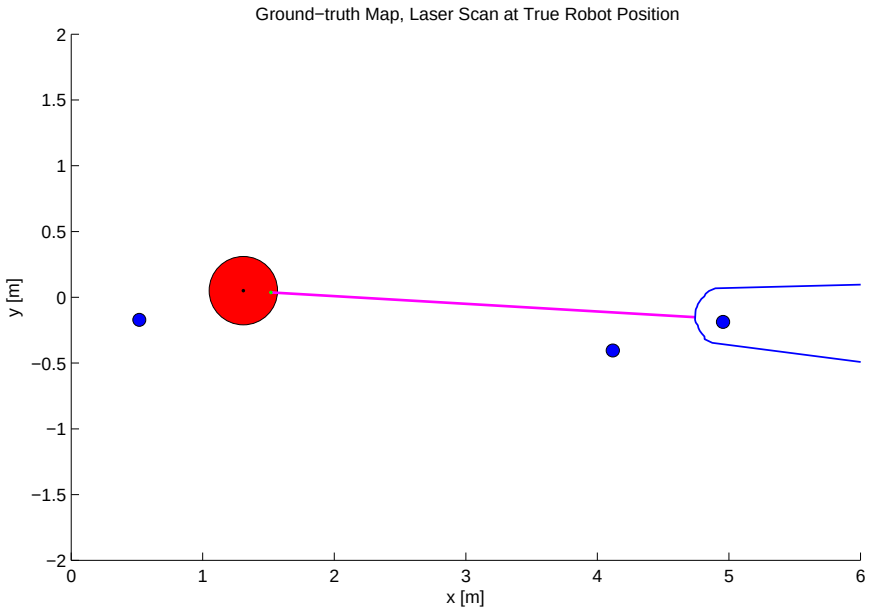


Figure 1.5: The purple line shows a range measurement from the robot's laser scanner to the cylinder's closest edge. The cylinder's radius is added to this to get the actual range to the cylinder.

assume that the data was acquired with a uniform 0.1 second sampling period. To avoid having to deal with the speed measurements and groundtruth data arriving asynchronously with the laser rangefinder scans, these measurements were linearly interpolated at the laser rangefinder timestamps. Thus, the data to be used in this assignment is synchronous with a uniform sampling period. Moreover, to avoid having to process the raw laser rangefinder scans, the range between the robot and the cylinder was extracted from each scan by using the cylinder's circular shape. Figure 1.5 shows how a range measurement was fit to a laser scan.

Another important issue worth mentioning is the noise on the sensor readings. Because we have a very accurate motion capture system, we may use the groundtruth positions of everything to determine the true robot-to-cylinder range readings and the true robot speed at each timestep. Figure 1.6 shows histograms of the errors in the range and speed measurements for the 20 minute dataset.

The file containing the final processed data for this assignment is called `dataset1.mat`, which is a Matlab binary file. You can view the contents by typing

```
load dataset1.mat
who
```

at the Matlab (or Octave) command prompt. The following seven Matlab variables will be listed:

`t` : a 12709×1 array containing the data timestamps [s]

`x_true` : a 12709×1 array containing the true position, x_k, of the robot (one-dimensional, along the rail) [m]

`l` : the scalar true position, x_c, of the cylinder's center (distance along the rail) [m]

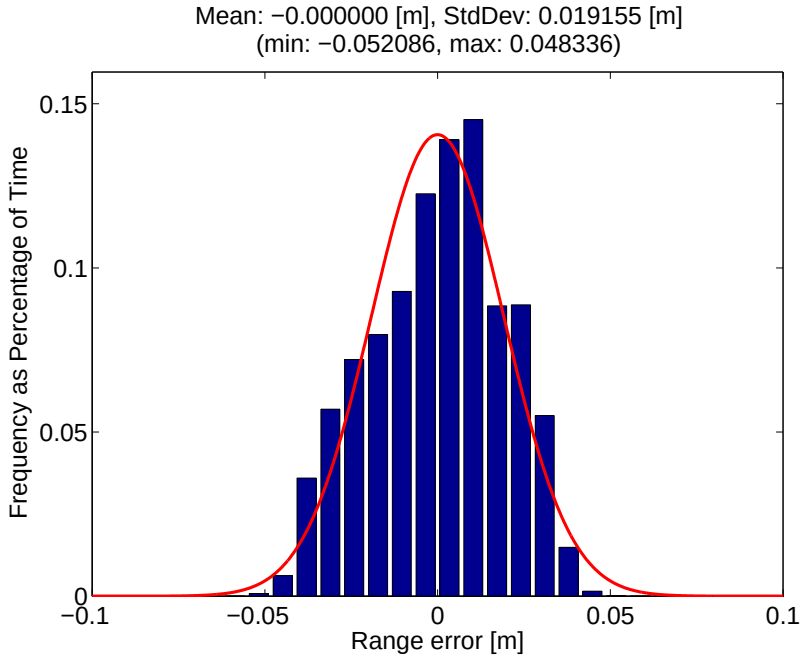
`r` : a 12709×1 array containing the range, r_k, between the robot and the cylinder's center as measured by the laser rangefinder sensor [m]

`r_var` : the variance of the range readings (based on groundtruth) [m^2]

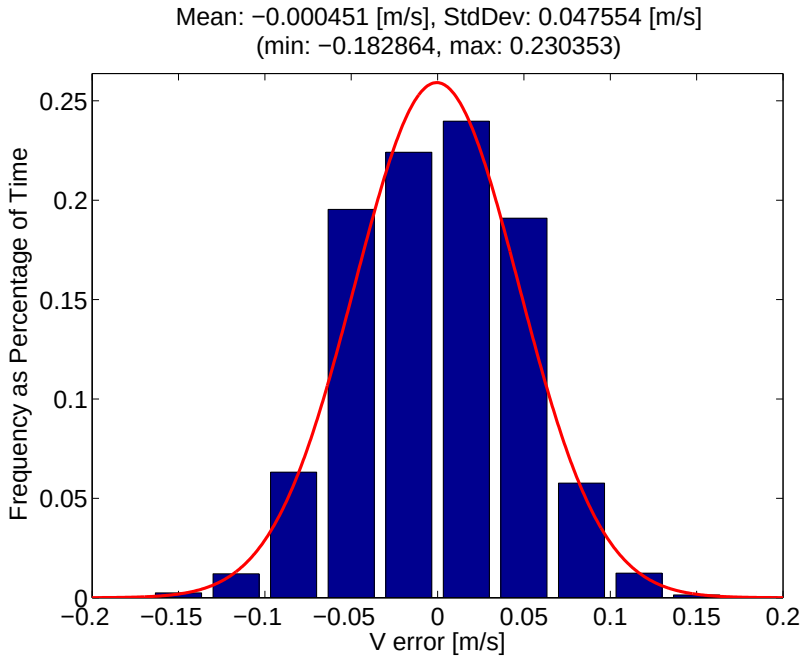
`v` : a 12709×1 array containing the speed, u_k, of the robot as measured by the robot's odometer [m/s]

`v_var` : the variance of the speed readings (based on groundtruth) [m^2/s^2]

The main idea is to estimate the robot's position (along the rails) without using `x_true` and then use this as a benchmark to gauge performance.



(a) Range errors.



(b) Speed errors.

Figure 1.6: Histograms of sensor errors. Red curves are Gaussians with standard deviation fit to the data. Both range and speed have close to zero-mean errors with a spread.

1.2 Motion and Observation Models

For this simple one-dimensional problem, the vehicle and sensor models are quite straightforward:

$$\text{motion:} \quad x_k = x_{k-1} + Tu_k + w_k \tag{1.1a}$$

$$\text{observation:} \quad y_k := x_c - r_k = x_k + n_k \tag{1.1b}$$

where x_k is the robot's position along the rail, u_k is the robot's speed (derived from odometer), w_k is the process noise, T is the sampling period, r_k is the range to the cylinder (derived from the laser rangefinder), x_c is the position of the cylinder's center (along the rail), and n_k is the exteroceptive sensor noise. We make a small transformation to the observation model to create a different sensor output, $y_k = x_c - r_k$. Both the motion and observation models are linear.

1.3 Assignment

For the assignment we'll use the batch linear-Gaussian algorithm to estimate the robot's position using both the odometry and laser measurements. You might wonder, why do we need to use both? In this simple example we could use either datastream to estimate the robot's position. Figure 1.7 shows the errors we would obtain if we use each datastream individually. We see the odometry error grows without bound over time (probably due to a very small bias). The position error derived from range remains bounded, but in a real situation, we might not have so much range data; we'll look at using only a small portion of this to effectively correct the odometry data.

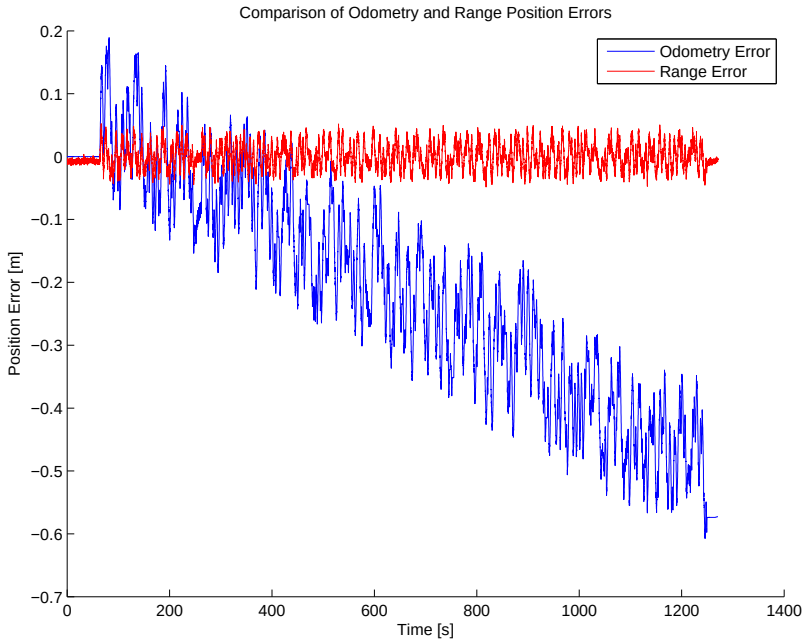


Figure 1.7: Estimation errors based on using either wheel odometry or laser range (and landmark position). Note that the odometry error is growing without bound over time.

Answer the following questions in a short typeset report (marks in the margin - 20 total):

3 1. Based on the data above, is the assumption of zero-mean Gaussian noise reasonable? What values of the variances, σ_q^2 and σ_r^2, should we use?

$$w_k \sim \mathcal{N}(0, \sigma_q^2), \qquad n_k \sim \mathcal{N}(0, \sigma_r^2) \tag{1.2}$$

3 2. Write out an expression for the batch linear-Gaussian objective function that we will seek to minimize:

$$J\left(x_{1:K} \,|\, u_{1:K}, y_{1:K}\right), \tag{1.3}$$

where K is the index of the maximum time.

4 3. Derive an expression for the optimal position estimates,

$$x_{1:K}^* = \arg\min_x \; J\left(x_{1:K} \,|\, u_{1:K}, y_{1:K}\right) \tag{1.4}$$

using the method of least squares.

4 4. This dataset has $K = 12709$. It may be computationally intractable to brute-force solve for all K positions using a batch method. Make a plot of the sparsity pattern of the left-hand side of the linear system of equations whose solution minimizes the objective function and comment on the pattern. Propose a method to solve this linear system that exploits the sparsity pattern to be as efficient as possible.

6 5. Write a Matlab or Python script to solve for the robot positions (mean and covariance) at all K timesteps. We will experiment with only using the laser rangefinder reading at a subset of the timesteps,

$$r_\delta, r_{2\delta}, r_{3\delta}, \ldots, r_K, \tag{1.5}$$

but will always use all of the odometry speed measurements. You will need to modify the objective function and expression for the optimal position estimates to handle this situation when

$$\delta = 1, 10, 100, 1000. \tag{1.6}$$

In words, there will be four cases where we use every, every 10th, every 100th, every 1000th laser rangefinder reading. Make the following figures and write some short comments about your findings. Note, the true robot position, x_k, is contained in the `x_true` array.

(a) Plot the following quantities on the same axes:
- the error, $x_k^* - x_k$ vs. t_k, as a solid line.
- the uncertainty envelope, $\pm 3\sigma_{x_k}$ vs. t_k, as two dotted lines. Note, the variance, $\sigma_{x_k}^2$, can be extracted from the least-squares solution (see the notes).

Make one plot for each value of δ.

(b) Make a histogram of the errors $x_k^* - x_k$, for each value of δ. How does this correspond to the predicted uncertainty from the least-squares formulation?

Attach your code in an Appendix.

Assignment 2

'Lost in the Woods Dataset'

In this second assignment we'll investigate a nonlinear two-dimensional problem consisting of a robot driving amongst a forest of tubes. We wish to estimate the position/orientation of the robot throughout its $\approx$ 360 m traverse. We'll use the recursive extended Kalman filter to fuse speed measurements coming from wheel odometry with range/bearing measurements coming from a laser rangefinder.

2.1 Experimental Setup

The setup consists of a mobile robot driving amongst a forest of plastic tubes as depicted in Figure 2.1. The robot and the tubes were equipped with reflective markers and tracked using a ten-camera motion capture system. This

(a) Mobile robot driving amongst a forest of tubes. Robot has an odometer to measure translational/rotational speeds and a laser rangefinder to measure range/bearing to the cylindrical landmarks. Both sensors are noisy.

(b) Vicon motion capture lab. Ten cameras work together to track markers on the robot and provide groundtruth position/orientation.

Figure 2.1: Setup for Dataset 2.

motion capture system is able to provide the position of each reflective marker to within a few millimeters. Position estimates from this motion capture system are considered to be quite a bit more accurate than estimates based on the robot's onboard sensors and thus it serves as the benchmark/groundtruth in our experiment.

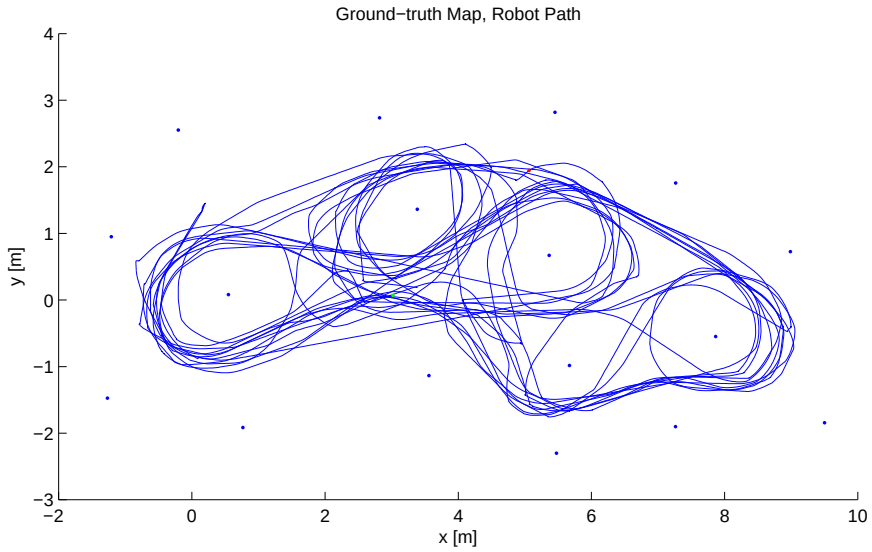


Figure 2.2: Robot path and 17 landmarks for Dataset 2.

There were 17 reflective markers placed on the tubular landmarks. The positions of these were measured using the motion capture system for 20 minutes and then averaged. The output of these steps can be seen in Figure 2.2.

The mobile robot was driven around for 20 minutes and three streams of data were logged:

- laser rangefinder scans consisting of 681 range measurements spread over a 240° horizontal field of view centered on straight ahead (Hokuyo URG-04LX sensor), logged at 10 Hz
- robot's speed based on a wheel odometry, logged at 10 Hz
- ground position of a marker on the laser rangefinder origin, logged at approximately 40 Hz

Figure 2.2 shows the robot's path over the 20 minute trial. We see that the robot's path is very twisted and looping. We must estimate the robot's position and orientation as it moves in this two-dimensional environment.

As with most real datasets, our data streams do not arrive synchronously or with evenly-spaced timesteps. Figure 2.3 shows the variability in the sampling periods of the nominally 10 Hz laser scans. For our purposes, we will assume that the data was acquired with a uniform 0.1 second sampling period. To avoid having to deal with the speed measurements and groundtruth data arriving asynchronously with the laser rangefinder scans, these measurements were linearly interpolated at the laser rangefinder timestamps. Thus, the data to be used in this assignment is synchronous with a uniform sampling period. Moreover, to avoid having to process the raw laser rangefinder scans, the range/bearing to each plastic tube was determined using groundtruth data. Figure 2.4 shows how a range measurement was fit to a laser scan.

Another important issue worth mentioning is the noise on the sensor readings. Because we have a very accurate motion capture system, we may use the groundtruth positions of everything to determine the true range/bearing

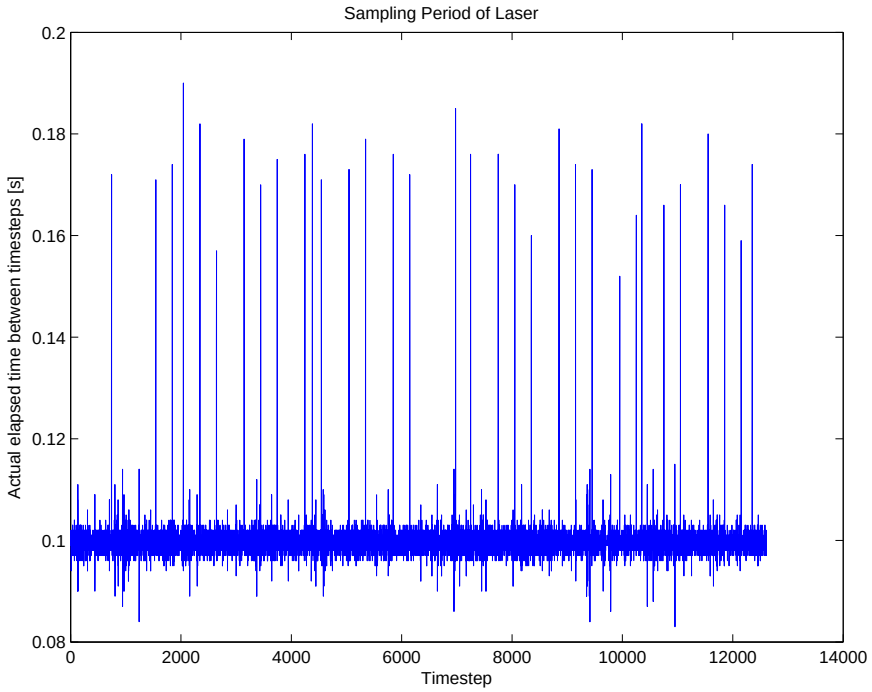


Figure 2.3: Sampling periods of the 10 Hz laser rangefinder scans. We see that the period does hover around 0.1 seconds.

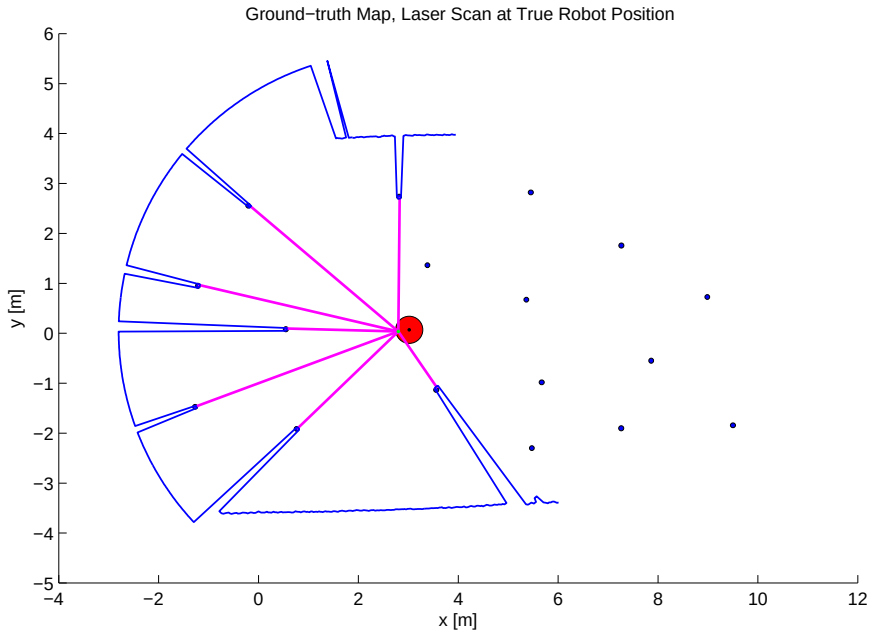


Figure 2.4: The purple line shows a range measurement from the robot's laser scanner to the cylinder's closest edge. The cylinder's radius is added to this to get the actual range to the cylinder.

readings and the true robot translational/rotational speeds at each timestep. Figure 2.5 shows histograms of the errors in the range/bearing for the 20 minute dataset. Figure 2.6 shows histograms of the errors in the translational/rotational speeds for the 20 minute dataset.

The file containing the final processed data for this assignment is called `dataset2.mat`, which is a Matlab binary file. You can view the contents by typing

```
load dataset2.mat
who
```

at the Matlab (or Octave) command prompt. The following 15 Matlab variables will be listed:

`t` : a 12609×1 array containing the data timestamps [s]

`x_true` : a 12609×1 array containing the true x-position, x_k, of the robot [m]

`y_true` : a 12609×1 array containing the true y-position, y_k, of the robot [m]

`th_true` : a 12609×1 array containing the true heading, θ_k, of the robot [m]

`true_valid` : a 12609×1 array containing a flag indicating if the groundtruth data is 'valid' or not (a value of 1 means it is valid and a value of 0 means it was interpolated due to some data dropouts)

`l` : a 17×2 array containing the true xy-positions, (x_l, y_l), of the 17 landmarks [m]

`r` : a 12609×17 array containing the range, r_k^l, between the robot's laser rangefinder and each landmark's center as measured by the laser rangefinder sensor [m] (a range of 0 indicates the landmark was not visible at that timestep)

`r_var` : the variance of the range readings (based on groundtruth) [m^2]

`b` : a 12609×17 array containing the bearing, ϕ_k^l, to each landmark's center in a frame attached to the laser rangefinder, as measured by the laser rangefinder sensor [rad] (if the range is 0 it indicates the landmark was not visible at that timestep and thus the bearing is meaningless)

`b_var` : the variance of the range readings (based on groundtruth) [rad^2]

`v` : a 12609×1 array containing the translational speed, v_k, of the robot as measured by the robot's odometers [m/s]

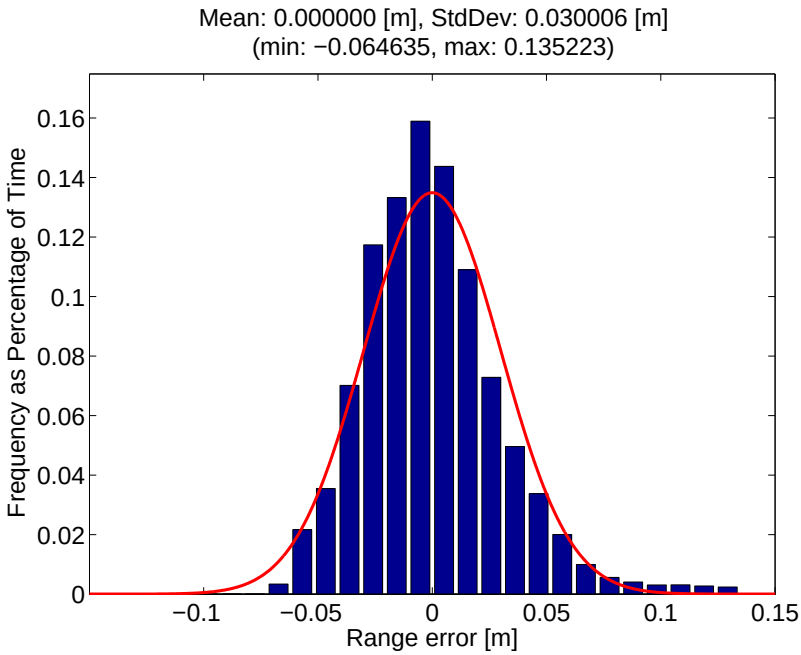
`v_var` : the variance of the translational speed readings (based on groundtruth) [m^2/s^2]

`om` : a 12609×1 array containing the rotational speed, ω_k, of the robot as measured by the robot's odometers [rad/s]

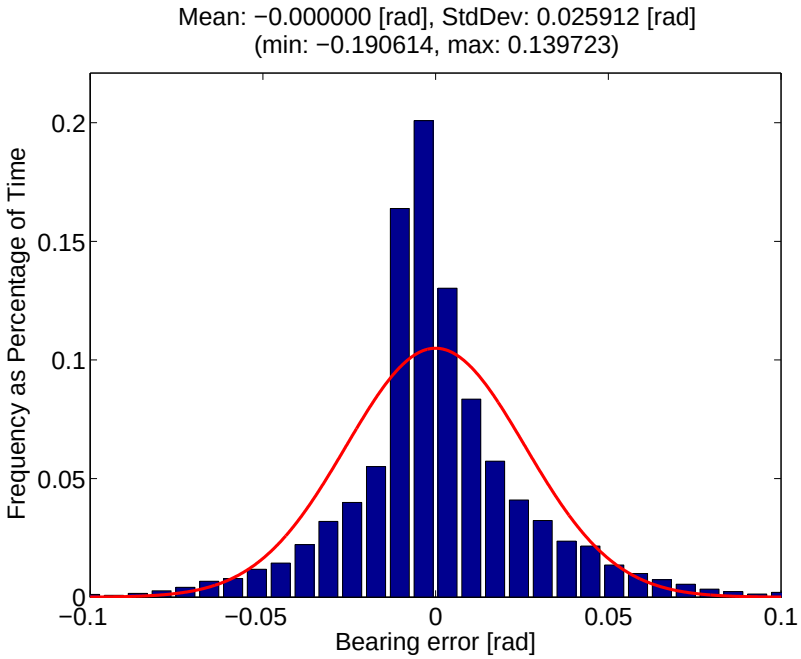
`om_var` : the variance of the rotational speed readings (based on groundtruth) [rad^2/s^2]

`d` : the distance, d, between the center of the robot and the laser rangefinder [m]

The main idea is to estimate the robot's position without using `x_true`, `y_true`, and `th_true` and then use these as a benchmark to gauge performance.

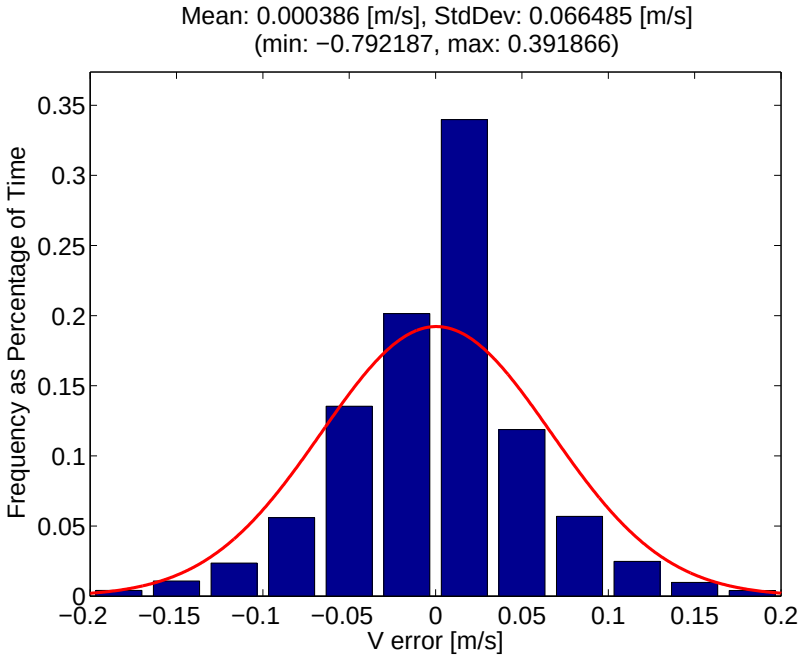


(a) Range errors.

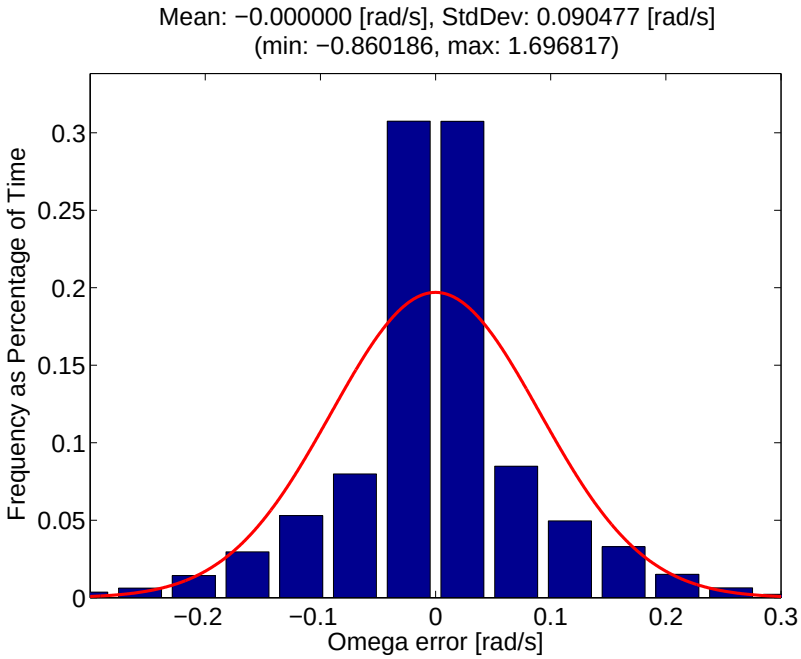


(b) Bearing errors.

Figure 2.5: Histograms of sensor errors. Red curves are Gaussians with standard deviation fit to the data. Both range and bearing have close to zero-mean errors with a spread.



(a) Translational speed errors.



(b) Rotational speed errors.

Figure 2.6: Histograms of sensor errors. Red curves are Gaussians with standard deviation fit to the data. Both translational and rotational speeds have close to zero-mean errors with a spread.

2.2 Motion and Observation Models

For this nonlinear estimation problem, use the following vehicle and sensor models:

$$\text{motion:} \quad \underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix}}_{\mathbf{x}_{k-1}} + T \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\underbrace{\begin{bmatrix} v_k \\ \omega_k \end{bmatrix}}_{\mathbf{u}_k} + \mathbf{w}_k \right) \tag{2.1a}$$

$$\mathbf{h}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k)$$

$$\text{observation:} \quad \underbrace{\begin{bmatrix} r_k^l \\ \phi_k^l \end{bmatrix}}_{\mathbf{y}_k^l} = \underbrace{\begin{bmatrix} \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \\ \text{atan2}(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix} + \mathbf{n}_k^l}_{\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^l)} \tag{2.1b}$$

where (x_k, y_k, θ_k) is the robot's position/orientation, (v_k, ω_k) is the robot's translational/rotational speed (derived from odometers), $\mathbf{w}_k$ is the process noise, T is the sampling period, (r_k^l, ϕ_k^l) is the range/bearing to landmark l (derived from the laser rangefinder), (x_l, y_l) is the position of the center of landmark l, and $\mathbf{n}_k^l$ is the exteroceptive sensor noise. The distance between the robot center and the laser rangefinder is d (the laser is at the front of the robot). Note that at each timestep, k, there are multiple exteroceptive measurements (e.g., in Figure 2.4 there are 7 purple lines). Both the motion and observation models are nonlinear.

2.3 Assignment

For the assignment we'll use the extended Kalman filter to estimate the robot's position using both the odometry and laser measurements. You might wonder, why do we need to use both? In this example we could just use the wheel odometry to estimate the robot's position. Figure 2.7 shows the robot's path as estimated using only wheel odometry. It is very different that the true path in Figure 2.2; in general the difference between the two paths will grow without bound. Although there are a lot of laser-based range/bearing measurements, there are some timesteps when there are less than two such landmarks measurements. Figure 2.8 shows a histogram of the number of landmarks seen simultaneously. In fact, there are 436 out of 12609 timesteps when less than 2 landmarks are observed. This means that it is impossible to solve for the robot's position/orientation from the laser measurements alone for some timesteps. Thus, in this situation, we actually need both the odometry and laser measurements to create a reasonable estimator.

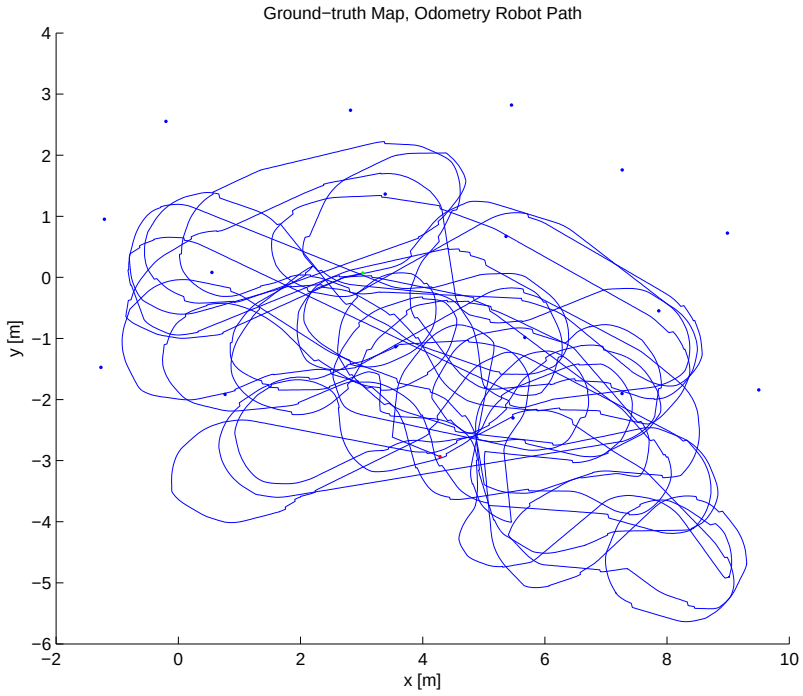


Figure 2.7: Robot path for dataset2 as computed using only wheel odometry (no laser rangefinder measurements).

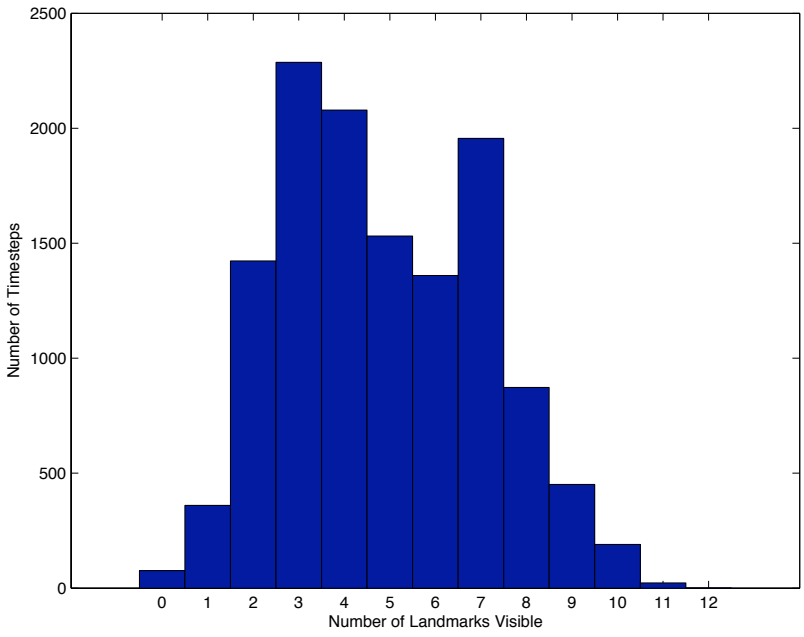


Figure 2.8: Histogram of number of landmarks seen simultaneously.

Answer the following questions in a short typeset report (marks in the margin - 25 total):

3 1. Based on the data above, is the assumption of zero-mean Gaussian noise reasonable? What values of the variances, $\mathbf{Q}_k$ and $\mathbf{R}_k^l$, should we use?

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \qquad \mathbf{n}_k^l \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k^l) \tag{2.2}$$

4 2. Write out expressions for all the Jacobians (of the motion and observation models) that are required in the extended Kalman filter.

4 3. Modify and write out the equations of the extended Kalman filter, modified to handle a variable number of exteroceptive measurements at each timestep.

10 4. Write a Matlab or Python script to compute an estimate the robot poses, $\hat{\mathbf{x}}_k = (\hat{x}_k, \hat{y}_k, \hat{\theta}_k)$, using the extended Kalman filter. Your code should be able to artificially limit itself to only use range/bearing landmark observations for which the range is less than a user-defined threshold, $r_{\max}$ [m]. Note, the true robot position, $\mathbf{x}_k = (x_k, y_k, \theta_k)$, is contained in the `x_true`, `y_true`, and `th_true` arrays. Make the following figures and write some short comments about your findings.

   (a) Use $\hat{\mathbf{x}}_0 = \mathbf{x}_0, \hat{\mathbf{P}}_0 = \text{diag}\{1, 1, 0.1\}$ as the initial condition (i.e., the estimator mean starts with the right answer). Plot the following quantities on the same axes:
   - the error, $\hat{x}_k - x_k$ vs. t_k, as a solid line.
   - the uncertainty envelope, $\pm 3\sigma_{x_k}$ vs. t_k, as two dotted lines. Note, the variance, $\sigma_{x_k}^2$, can be extracted from the diagonal of the covariance matrix, $\hat{\mathbf{P}}_k$.

   Make one plot for each value of $r_{\max} = 5, 3, 1$. Repeat for y and θ. That's 9 plots total.

   (b) Repeat (a) with $\hat{\mathbf{x}}_0 = (1, 1, 0.1)$, i.e., a poor initial condition. See how far you can push the initial condition away and still have your estimator converge.

   (c) Repeat (a) where you evaluate all the Jacobians at the *true robot state*, $\mathbf{x}_k$, instead of the estimated robot state, $\hat{\mathbf{x}}_k$. This is the CRLB version of the EKF. Contrast this to (a).

4 5. Make an animation of your EKF estimate for the case $r_{\max} = 1$ and $\hat{\mathbf{x}}_0 = \mathbf{x}_0, \hat{\mathbf{P}}_0 = \text{diag}\{1, 1, 0.1\}$. It should display the following information:

   - true landmarks positions; static dots (black)
   - true robot position, (x_k, y_k); a moving dot (blue)
   - estimated robot position, $(\hat{x}_k, \hat{y}_k)$; a moving dot (red)
   - 3-sigma covariance ellipse (use the top-left 2×2 of $\hat{\mathbf{P}}_k$); this should be centered on the estimated robot position; we want to see that the true robot position stays within this ellipse (but it probably won't); a moving ellipse (red)

   Hand this in as a movie file and make sure it can play in VLC.

Attach your code in an Appendix.

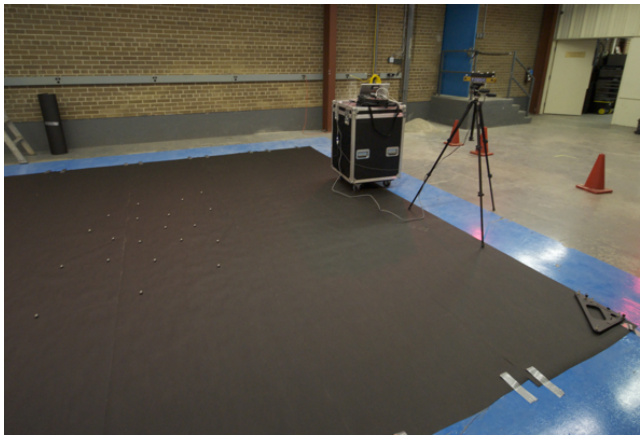
Assignment 3

‘Starry Night Dataset’

In this third assignment we’ll investigate a nonlinear three-dimensional problem consisting of a vehicle equipped with a stereo camera and inertial measurement unit (IMU) flying above a map of point features. We wish to estimate the position/orientation of this vehicle throughout its ≈ 44.3 m traverse. We’ll use the batch Gauss-Newton method to fuse speed measurements from the IMU with point measurements from the stereo camera.

3.1 Experimental Setup

The setup consists of a sensor head being translated/rotated above a set of reflective markers as depicted in Figure 3.1. The sensor head included a stereo camera and IMU rigidly attached to each other. Both the sensor head and the reflective markers were tracked using a ten-camera motion capture system. This motion capture system is able

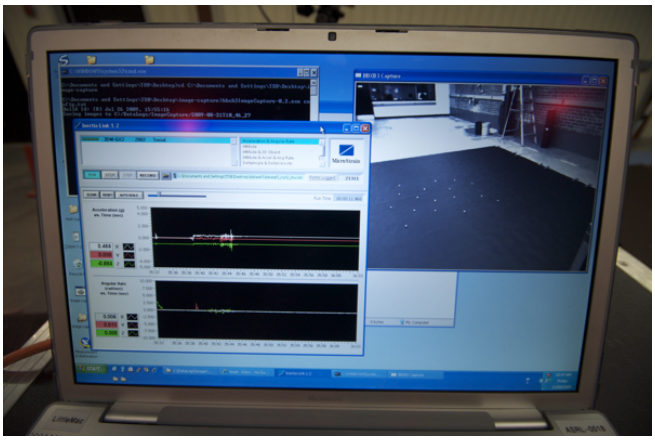


(a) A sensor head on the end of a tripod was translated/rotated above a map of point landmarks (white dots on the black floor).

(b) Vicon motion capture lab. Ten cameras work together to track markers on the sensor head and provide groundtruth position/orientation.

Figure 3.1: Setup for Dataset 3.

(a) The sensor head has an IMU to measure translational/rotational speeds and a stereo camera to measure the positions of point landmarks. Both sensors are noisy.



(b) Laptop screen shows what the stereo camera sees (left image only) as well as the IMU data capture software.

Figure 3.2: Sensor head.

to provide the position of each reflective marker to within a few millimeters. Position estimates from this motion capture system are considered to be quite a bit more accurate than estimates based on the sensor head and thus it serves as the benchmark/groundtruth in our experiment.

The sensor head was translated/rotated at the end of a tripod for approximately 3 minutes and three streams of data were logged:

- stereo image pairs, each image 640×480 resolution, logged at approximately 15 Hz
- the sensor head's velocity based on IMU measurements logged at approximately 100 Hz
- groundtruth position and orientation of a set of markers on the sensor head, logged at approximately 40 Hz

Figure 3.3 shows the sensor head's path over the 3 minute trial. We see that its path is very twisted and looping. We must estimate the sensor head's position and orientation as it moves in this three-dimensional environment.

The 20 landmarks used in this dataset were reflective markers placed on a black background. The positions of these were measured using the motion capture system for 3 minutes and then averaged. The output of these steps can be seen in Figure 3.3.

As with most real datasets, our data streams do not arrive synchronously or with evenly-spaced timesteps. Figure 3.4 shows the variability in the sampling periods of the nominally 15 Hz stereo image capture. Because of the variability in the camera sampling period, we cannot assume a uniform sampling rate. However, as with the other datasets, the velocities derived from the IMU data and the groundtruth values have been interpolated at the image times. Moreover, to avoid having to process the raw stereo images, stereo features were extracted from the images and associated with the landmarks using groundtruth data. Figure 3.5 shows how an example stereo image pair, and the extracted stereo features. Figure 3.6 and 3.7 show histograms of the IMU and stereo camera sensor errors based on groundtruth position/orientation.

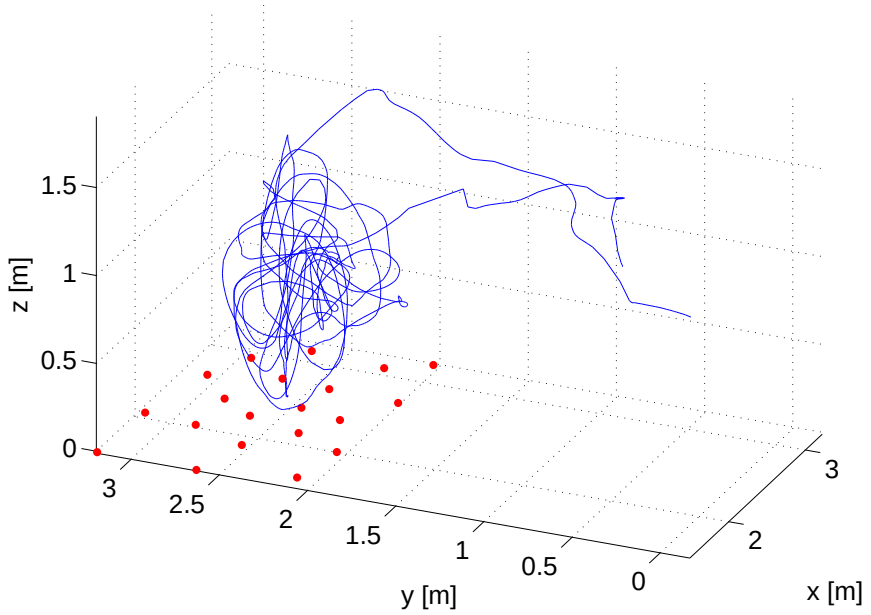


Figure 3.3: Groundtruth sensor head trajectory and 20 landmarks for Dataset 3.

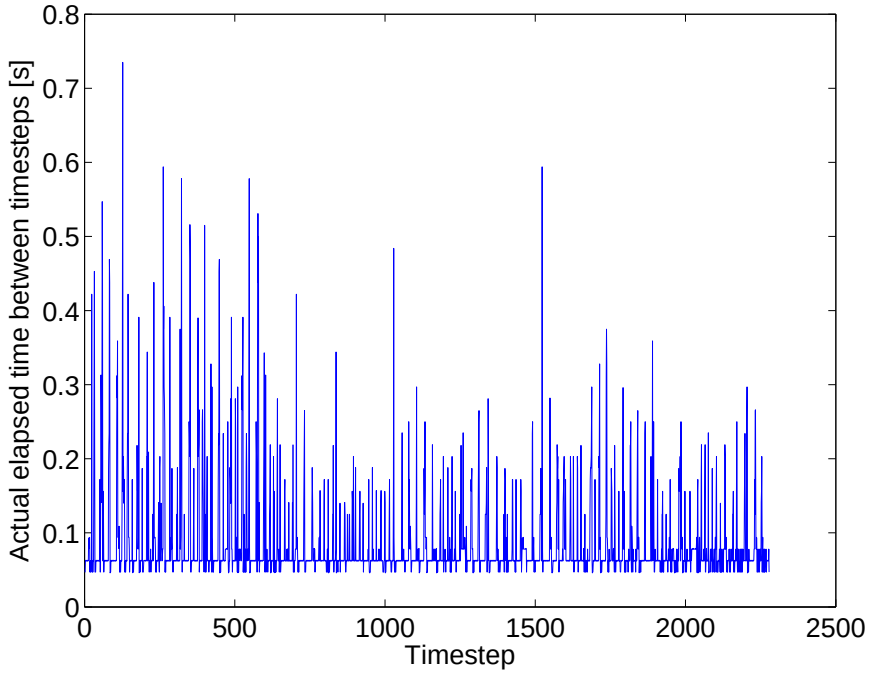


Figure 3.4: Sampling periods of the 15 Hz stereo images. We see that the nominal period is around 0.065 s, but the logging is not very consistent.

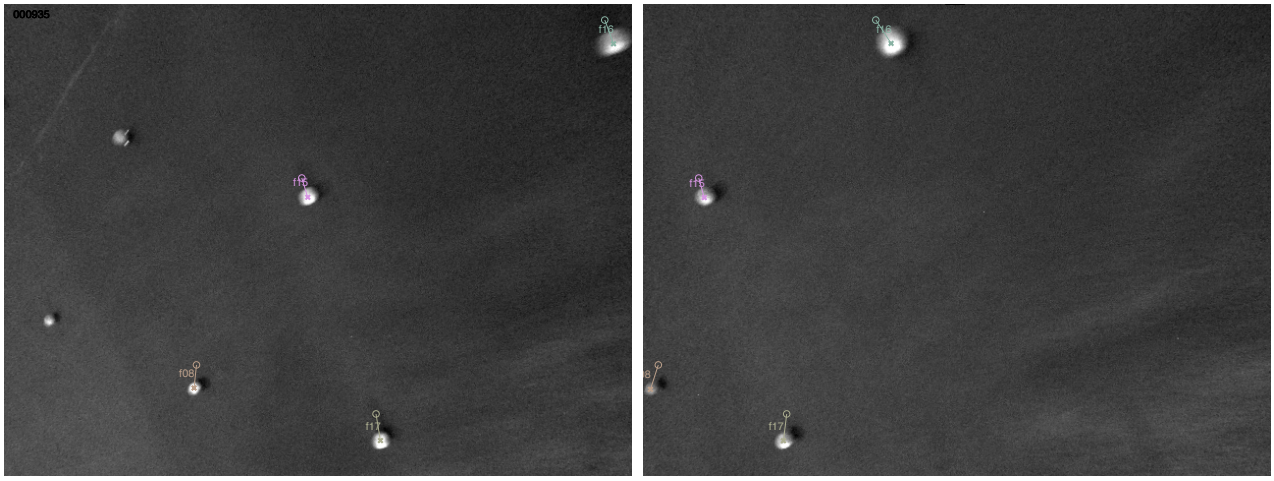
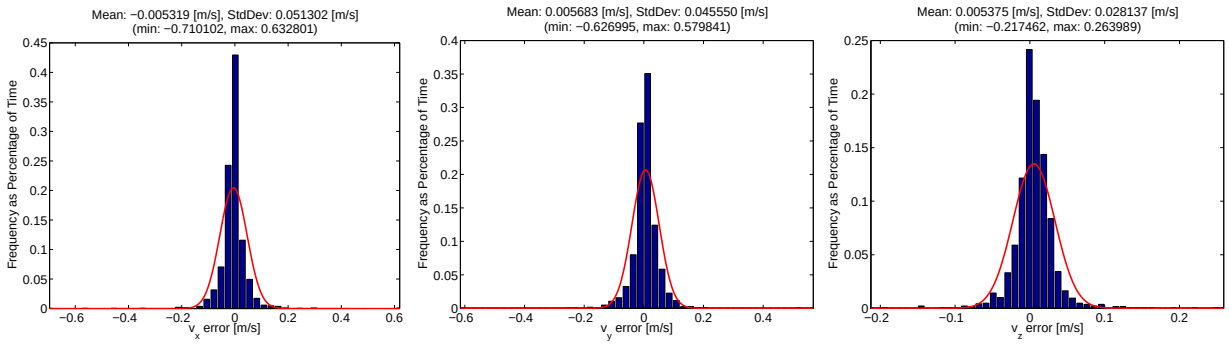
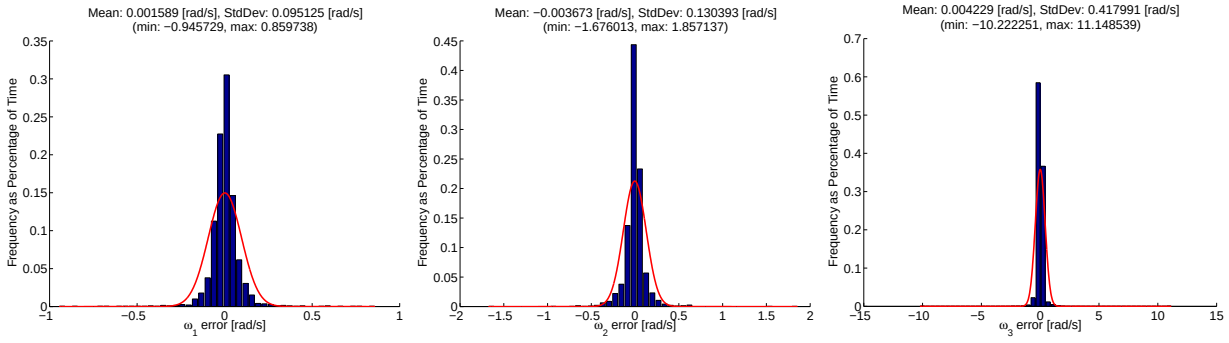


Figure 3.5: A sample pair of stereo images. The 'x' is the measurement, the 'o' indicates the predicted position using the groundtruth position and orientation.



(a) Translational speed errors.



(b) Rotational speed errors.

Figure 3.6: Histograms of IMU sensor errors. Red curves are Gaussians with standard deviation fit to the data. Both translational and rotational speeds have close to zero-mean errors.

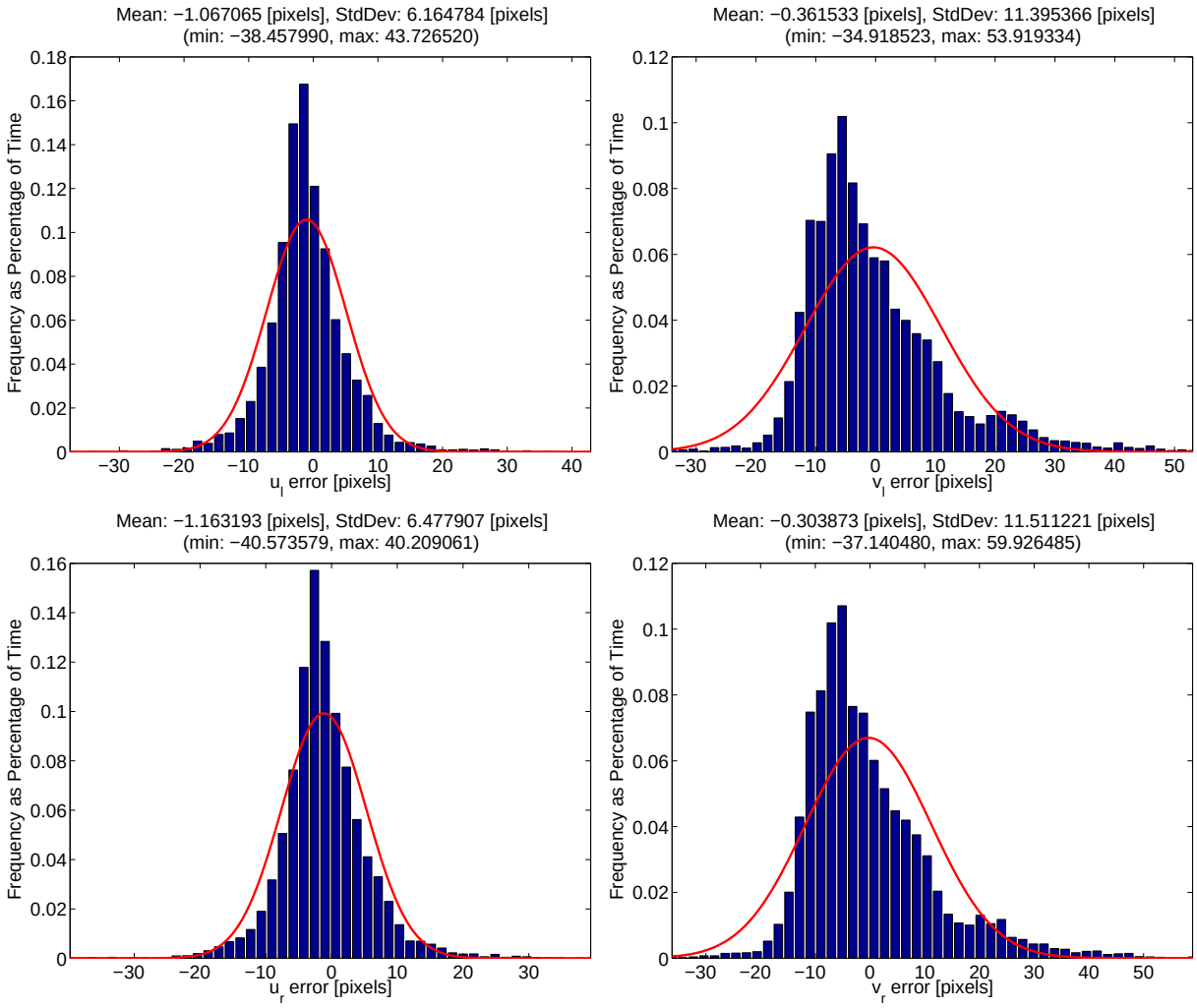


Figure 3.7: Histograms of stereo camera sensor errors. Red curves are Gaussians with standard deviation fit to the data.

The file containing the final processed data for this assignment is called `dataset3.mat`, which is a Matlab binary file. You can view the contents by typing

```
load dataset3.mat
who
```

at the Matlab (or Octave) command prompt. The following 17 Matlab variables will be listed:

`theta_vk_i` : a $3 \times K$ matrix where the k^{th} column is the axis-angle representation of the groundtruth value of $\mathbf{C}_{v_k i}$. Use this vector as $\boldsymbol{\psi}_k$ in (3.3c) to recover the rotation matrix.

`r_i_vk_i` : a $3 \times K$ matrix where the k^{th} column is the groundtruth value of $\mathbf{r}_i^{v_k i}$ [m]

`t` : a $1 \times K$ matrix of time values $t(k)$ [s]

`w_vk_vk_i` : a $3 \times K$ matrix where the k^{th} column is the measured rotational velocity, $\boldsymbol{\omega}_{v_k}^{v_k i}$ [rad/s]

`w_var` : a 3×1 matrix of the computed variances (based on groundtruth) of the rotational speeds [rad^2/s^2]

`v_vk_vk_i` : a $3 \times K$ matrix where the k^{th} column is the measured translational velocity, $\mathbf{v}_{v_k}^{v_k, i}$ [m/s]

`v_var` : a 3×1 matrix of the computed variances (based on groundtruth) of the translational speeds [m^2/s^2]

`rho_i_pj_i` : a 3×20 matrix where the j^{th} column is the position of feature j, $\boldsymbol{\rho}_i^{p_j i}$ [m]

`y_k_j` : a $4 \times K \times 20$ array of observations, $\mathbf{y}_k^j$ [pixels]. All components of `y_k_j(:,k,j)` will be -1 if the observation is invalid.

`y_var` : a 4×1 matrix of the computed variances (based on groundtruth) of the stereo measurements [pixels^2]

`C_c_v` : a 3×3 matrix giving the rotation from the vehicle frame to the camera frame, $\mathbf{C}_{cv}$

`rho_v_c_v` : a 3×1 matrix giving the translation from the vehicle frame to the camera frame, $\boldsymbol{\rho}_v^{cv}$ [m]

`fu` : the stereo camera's horizontal focal length, f_u [pixels]

`fv` : the stereo camera's vertical focal length, f_v [pixels]

`cu` : the stereo camera's horizontal optical center, c_u [pixels]

`cv` : the stereo camera's vertical optical center, c_v [pixels]

`b` : the stereo camera baseline, b [m]

3.2 Motion and Observation Models

3.2.1 Reference Frames

We need to define some reference frames for this problem. Figure 3.8 shows these reference frames. The frames are briefly described as follows:

- $\underrightarrow{\mathcal{F}}_i$: The inertial frame. Our estimate of the vehicle's translation/rotation will be computed in this frame. Additionally, the positions of the points on the floor, P_j, will be provided in this frame.
- $\underrightarrow{\mathcal{F}}_v$: The vehicle frame. This will be the main reference frame attached to the sensor head. We will be estimating the position/orientation of this frame with respect to the inertial frame. For convenience we choose to use a frame attached to the IMU for this frame.
- $\underrightarrow{\mathcal{F}}_c$: The camera frame. Since the stereo camera and the IMU can't be in the same location, we need an extra frame attached to the camera. For convenience this is attached to the left camera in the stereo pair. The translation/rotation between the vehicle frame and the camera frame, $\underrightarrow{\rho}^{vc}$ and $\mathbf{C}_{cv}$, are fixed and determined by calibration in advance.

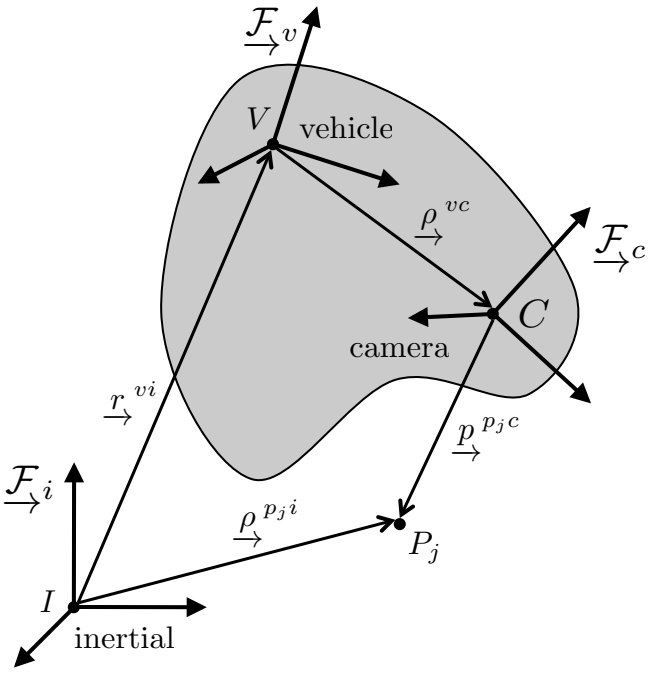


Figure 3.8: Reference frames used in Dataset 3.

3.2.2 Motion Model

This motion model we will use is very similar to the one derived by Barfoot (2017). It has been modified to account for a variable sampling period:

$$T_k := t(k) - t(k-1) \; . \tag{3.1}$$

We define the discrete-time motion model to be

$$\mathbf{r}_i^{v_k i} = \mathbf{r}_i^{v_{k-1} i} + \mathbf{C}_{v_{k-1} i}^T \mathbf{d}_{v_{k-1}}^{v_k v_{k-1}}, \tag{3.2a}$$

$$\mathbf{C}_{v_k i} = \boldsymbol{\Psi}_{v_k v_{k-1}} \mathbf{C}_{v_{k-1} i}, \tag{3.2b}$$

where k is the discrete-time index. The state of the system at timestep k is

$\mathbf{r}_i^{v_k i}$: Translation of vehicle frame with respect to inertial frame, expressed in inertial frame,
$\mathbf{C}_{v_k i}$: Rotation of vehicle frame with respect to inertial frame.

The quantities $\mathbf{d}_{v_{k-1}}^{v_k v_{k-1}}$ and $\boldsymbol{\Psi}_{v_k v_{k-1}}$ are given by

$$\mathbf{d}_{v_{k-1}}^{v_k v_{k-1}} := \mathbf{v}_{v_{k-1}}^{v_{k-1} i} T_k + \delta \mathbf{d}_k, \tag{3.3a}$$

$$\boldsymbol{\Psi}_{v_k v_{k-1}} := \cos\psi_k \mathbf{1} + (1 - \cos\psi_k) \left(\frac{\boldsymbol{\psi}_k}{\psi_k}\right) \left(\frac{\boldsymbol{\psi}_k}{\psi_k}\right)^T - \sin\psi_k \left(\frac{\boldsymbol{\psi}_k}{\psi_k}\right)^\times, \tag{3.3b}$$

$$\boldsymbol{\psi}_k := \boldsymbol{\omega}_{v_{k-1}}^{v_{k-1} i} T_k + \delta \boldsymbol{\psi}_k, \tag{3.3c}$$

$$\psi_k := |\boldsymbol{\psi}_k|. \tag{3.3d}$$

The interoceptive measurements (and associated noise variables) come from the IMU in this case:

$\mathbf{v}_{v_k}^{v_k i}$: Translational velocity of vehicle frame with respect to inertial frame, expressed in vehicle frame
$\boldsymbol{\omega}_{v_k}^{v_k i}$: Rotational velocity of vehicle frame with respect to inertial frame, expressed in vehicle frame
$\delta \mathbf{d}_k$: Translational process noise
$\delta \boldsymbol{\psi}_k$: Rotational process noise

3.2.3 Observation Model

The observation model, $\mathbf{g}(\cdot)$, is a nonlinear function that projects points expressed in the left camera frame into the camera's image coordinates. The position of a point on the floor, P_j, expressed in the camera frame, $\underrightarrow{\mathcal{F}}_c$, is denoted $\mathbf{p}_{c_k}^{p_j c_k}$ and is given by

$$\mathbf{p}_{c_k}^{p_j c_k} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{C}_{cv} \left(\mathbf{C}_{v_k i} \left(\boldsymbol{\rho}_i^{p_j i} - \mathbf{r}_i^{v_k i} \right) - \boldsymbol{\rho}_v^{cv} \right), \tag{3.4}$$

where $\left\{ \mathbf{r}_i^{v_k i}, \mathbf{C}_{v_k i} \right\}$ is the 'state' of the vehicle, $\{\boldsymbol{\rho}_v^{cv}, \mathbf{C}_{cv}\}$ is the known translation/rotation from the vehicle frame to the camera frame, and $\mathbf{p}_i^{p_j i}$ is the known position of P_j in the inertial frame.

The observation model, $\mathbf{g}(\cdot)$, projects $\mathbf{p}_{c_k}^{p_j c_k}$ into the rectified images of an axis-aligned stereo camera:

$$\mathbf{y}_k^j := \mathbf{g}\left(\mathbf{p}_{c_k}^{p_j c_k}\right) = \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix} = \frac{1}{z}\begin{bmatrix} f_u x \\ f_v y \\ f_u(x-b) \\ f_v y \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \\ c_u \\ c_v \end{bmatrix} + \delta\mathbf{n}_k^j, \tag{3.5}$$

where the exteroceptive measurement and associated measurement noise are

- $\mathbf{y}_k^j$: The pixel coordinates of the point, P_j, projected into the left and right images of the stereo camera, (u_l, v_l) and (u_r, v_r), respectively
- $\delta\mathbf{n}_k^j$: Measurement noise

This calibrated, parallel stereo camera model, has the following fixed calibration parameters:

- c_u, c_v : The horizontal and vertical optical center from the top left of the image [pixels]
- f_u, f_v : The horizontal and vertical focal length [pixels]
- b : The camera baseline (i.e., distance between the two centers of projection) [m]

3.3 Assignment

We'll use the batch Gauss-Newton method to estimate the sensor head's position/orientation using both the IMU and stereo camera measurements. As usual, you should ask yourself do we really need both? Figure 3.9 shows what the sensor head's path looks like using only the interoceptive measurements (i.e., the IMU). A quick visual comparison with the groundtruth path in Figure 3.3 shows this is way off. And, although there are a lot of sensor camera measurements, there are many timesteps when there are less than three such observations. Figure 3.10 shows a histogram of the number of landmarks seen simultaneously. In fact, there are over 650 out of 1900 timesteps when there are less than three landmarks observed. This means that it is impossible to solve for the sensor head's position/orientation from the stereo camera measurements alone for many timesteps. Thus, in this situation, we actually need both the IMU and stereo camera measurements to create a reasonable estimator.

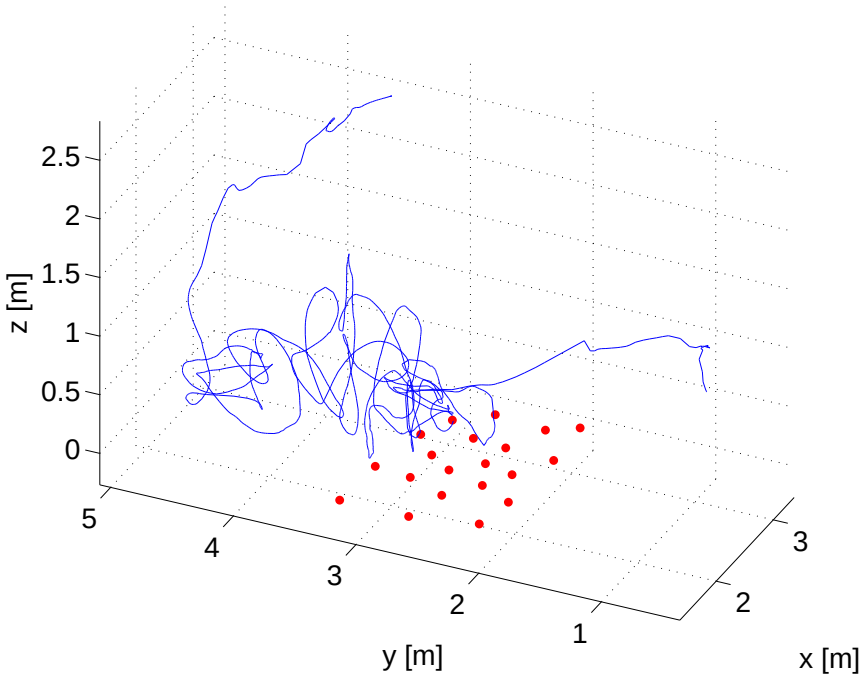


Figure 3.9: Sensor head path for Dataset 3 as computed using only IMU readings (no stereo camera measurements).

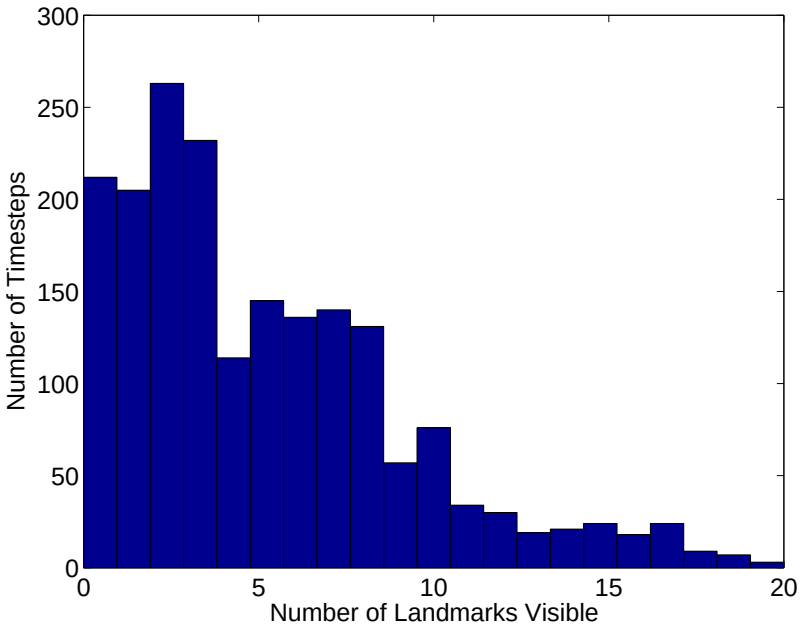


Figure 3.10: Histogram of number of landmarks seen simultaneously.

Answer the following questions in a short typeset report (marks in the margin - 25 total):

3 1. Based on the data above, is the assumption of zero-mean Gaussian noise reasonable? What values of the variances, $\mathbf{Q}_k$ and $\mathbf{R}_k^j$, should we use? Pay attention to the fact that we have a variable sampling period.

$$\begin{bmatrix} \delta\mathbf{d}_k \\ \delta\boldsymbol{\psi}_k \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \qquad \delta\mathbf{n}_k^j \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k^j) \tag{3.6}$$

5 2. Write out an expression for the batch nonlinear least-squares objective function that we will seek to minimize using the Gauss-Newton method:

$$J\left(\bar{\mathbf{x}}_{k_1:k_2}\right) := \frac{1}{2}\mathbf{e}(\bar{\mathbf{x}}_{k_1:k_2})^T\mathbf{T}^{-1}\mathbf{e}(\bar{\mathbf{x}}_{k_1:k_2}), \tag{3.7}$$

where $\bar{\mathbf{x}}_{k_1:k_2}$ is the set of all poses from timestep k_1 to timestep k_2, i.e., $\bar{\mathbf{x}}_{k_1:k_2} = \left\{\bar{\mathbf{r}}_i^{v_{k_1}i}, \bar{\mathbf{C}}_{v_{k_1}i}, \ldots, \bar{\mathbf{r}}_i^{v_{k_2}i}, \bar{\mathbf{C}}_{v_{k_2}i}\right\}$. Note that we want to be able to handle only a window of poses in order to try different cases below.

4 3. Write out the Gauss-Newton algorithm to find

$$\bar{\mathbf{x}}_{k_1:k_2}^* = \arg\min_x J\left(\bar{\mathbf{x}}_{k_1:k_2}\right). \tag{3.8}$$

3 4. Make a plot of the number of visible landmarks at each given timestep, i.e., M_k vs. t_k. Use green dots for those timesteps for which there are at least three visible landmarks and red dots otherwise. We'll use this to interpret our results later.

10 5. Write a Matlab or Python script to implement your Gauss-Newton algorithm above, keeping k_1 and k_2 general. Note, the groundtruth sensor head position and orientation, $\left\{\mathbf{r}_i^{v_k i}, \mathbf{C}_{v_k i}\right\}$ are contained in the `r_i_vk_i` and `theta_vk_i` arrays.

(a) BATCH: First apply GN to the entire interval $k_1 = 1215$ to $k_2 = 1714$. To initialize the GN algorithm, you will need an initial guess for $\bar{\mathbf{x}}_{k_1:k_2}$. For this use the groundtruth at timestep k_1 and then dead reckon using only the interoceptive measurements and the motion model up to timestep k_2. This should look like a portion of Figure 3.9. Note you are estimating 3000 states since each timestep has 6 scalar states in all.

The estimation errors at timestep k can be computed according to

$$\text{translational errors:} \quad \delta\mathbf{r}_k = \begin{bmatrix} \delta r_{x,k} \\ \delta r_{y,k} \\ \delta r_{z,k} \end{bmatrix} := \bar{\mathbf{r}}_i^{v_k i^*} - \mathbf{r}_i^{v_k i},$$

$$\text{rotational errors:} \quad \delta\boldsymbol{\theta}_k^\times = \begin{bmatrix} \delta\theta_{x,k} \\ \delta\theta_{y,k} \\ \delta\theta_{z,k} \end{bmatrix}^\times := \mathbf{1} - \bar{\mathbf{C}}_{v_k i}^* \mathbf{C}_{v_k i}^T.$$

Plot the following:

- the error, $\delta r_{x,k}$ vs. t_k, as a solid line.

  - the uncertainty envelope, $\pm 3\sigma_{r_{x,k}}$ vs. t_k, as two dotted lines. Note, the variance, $\sigma^2_{r_{x,k}}$, can be extracted from the last iteration of the GN least-squares solution (see the notes).

  Repeat for $\delta r_{y,k}$, $\delta r_{z,k}$, $\delta\theta_{x,k}$, $\delta\theta_{y,k}$, and $\delta\theta_{z,k}$. That's 6 plots total.

(b) SLIDING WINDOW I: Start by apply GN to the interval $k_1 = k$ to $k_2 = k + \kappa$, where $k = 1215$ and $\kappa = 50$. Store only the pose from timestep k. Then repeatedly slide the window along by making $k = 1216, 1217, \ldots, 1714$, solving a GN problem at each k and storing only the pose from timestep k each time. To initialize GN, use a similar procedure to the BATCH case, but dead reckon from your most recent stored pose (and groundtruth only for the first pose, $k = 1215$). Make the same plots as in (a).

(c) SLIDING WINDOW II: Repeat (b) with $\kappa = 10$.

Comment on the differences in accuracy and computational effort between the BATCH and SLIDING WINDOW cases. Discuss any trends you see between your error plots and the plot you made in question 4. Attach your code in an Appendix.

Bibliography

Barfoot, T. D. (2017). "State Estimation for Robotics". Cambridge University Press, 2017.