Z3373093
William Johnston

Report for Project 2

Part 0

Laser data

The laser data is stored in a .mat file which contains the struct with the laser data. The struct has 3 elements:

- N – representing the number of laser scans
- Ranges – the relevant scan data
- Times – the time each scan was taken

For each scan there are 361 range measurements, 1 measurement for every 0.5 degrees in the field of view which is 180 degrees. Each measurement represents the distance between the laser scanner and the nearest object in that direction. From this we can determine all the objects and their distances from the robot in the field of view of the laser scanner. Additionally, an intensity measurement is included with the range measurement, for determining the reflectiveness of the object. Figure 1 below shows how these measurements can be turned into useful information for interpretation. Figure 2 below plots an example of this data for demonstration.

```
% Using masks to isolate range and intensity information from measurement
mask1FFF = uint16(2^13-1);
maskE000 = bitshift(uint16(7),13)   ;
intensities = bitand(scan,maskE000);
ranges     = single(bitand(scan,mask1FFF))*0.01;

%create array of angles for conversion to cartesian coordinates
angles = [0:360]'*0.5* pi/180 ;

%convert ranges to cartesian coordinates
X = -cos(angles).*ranges;
Y = sin(angles).*ranges;

%find pixels with high reflectivity
ii = find(intensities~=0);
```

Figure 1: Pseudo-code showing how laser scan data is interpreted

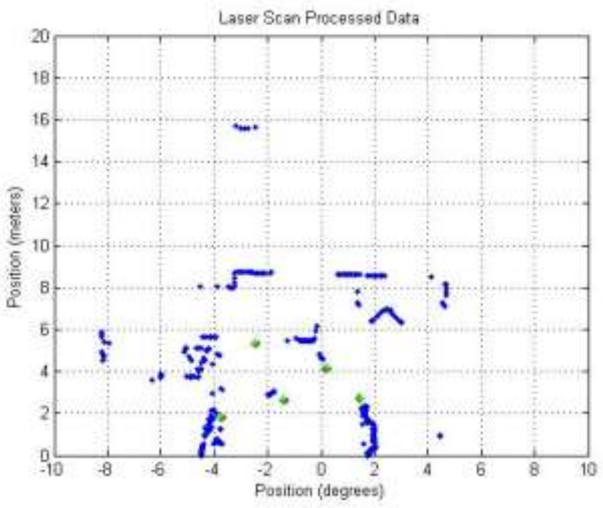
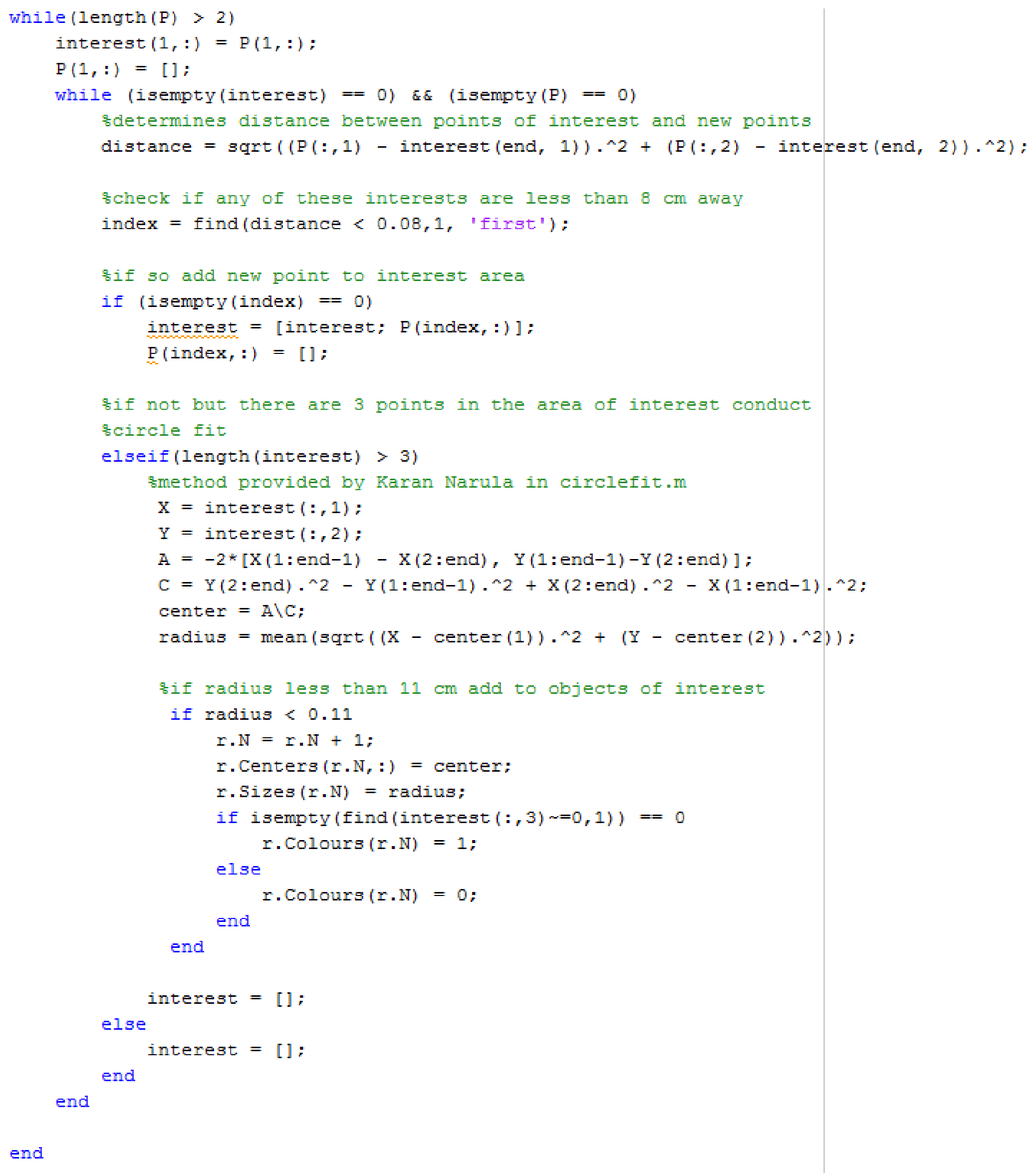


Figure 2: Visual representation of laser data.

The laser data is stored as uint 16, which is a 16-bit unsinged integer. Uint 16 has a max size of 65,535 which is large enough for use with the laser scans while uint 8 only has a range of 0 to 255 which is too low. The first 12 bits of the uint 16 define the range in centimetres, while the last 4 bits represent the intensity of the object. An object is said to be reflective if this intensity is above 0.

Feature Extraction

For feature extraction we have a set definition for our objects of interest, they must have a diameter between 5 and 20 cm. For determining if these objects, the pixels for each range measurement must be organised into clusters or groups to represent an object. This is done by stepping through the pixels from the laser scanner and determining if the next pixel is within 8 cm of the next pixel. Once each group is defined a circle is fitted to the group in order to determine the diameter of the object. If the object fits within the diameter bounds it is saved as an object of interest. Figure 3 below provides the pseudo – code for this process.



```
while(length(P) > 2)
    interest(1,:) = P(1,:);
    P(1,:) = [];
    while (isempty(interest) == 0) && (isempty(P) == 0)
        %determines distance between points of interest and new points
        distance = sqrt((P(:,1) - interest(end, 1)).^2 + (P(:,2) - interest(end, 2)).^2);

        %check if any of these interests are less than 8 cm away
        index = find(distance < 0.08,1, 'first');

        %if so add new point to interest area
        if (isempty(index) == 0)
            interest = [interest; P(index,:)];
            P(index,:) = [];

        %if not but there are 3 points in the area of interest conduct
        %circle fit
        elseif(length(interest) > 3)
            %method provided by Karan Narula in circlefit.m
             X = interest(:,1);
             Y = interest(:,2);
             A = -2*[X(1:end-1) - X(2:end), Y(1:end-1)-Y(2:end)];
             C = Y(2:end).^2 - Y(1:end-1).^2 + X(2:end).^2 - X(1:end-1).^2;
             center = A\C;
             radius = mean(sqrt((X - center(1)).^2 + (Y - center(2)).^2));

             %if radius less than 11 cm add to objects of interest
              if radius < 0.11
                  r.N = r.N + 1;
                  r.Centers(r.N,:) = center;
                  r.Sizes(r.N) = radius;
                  if isempty(find(interest(:,3)~=0,1)) == 0
                      r.Colours(r.N) = 1;
                  else
                      r.Colours(r.N) = 0;
                  end
              end

            interest = [];
        else
            interest = [];
        end
    end

end
```

Figure 3: Pseudo-code showing feature extraction process

Z3373093
William Johnston

Data Association

In order to ensure that an identified pole was assigned to the correct known pole a data association matrix was used. For each identified pole the distance between the identified and known poles was determined. All of these values were placed into a matrix in order to determine the best match for each identified pole. An example of this matrix is shown below.

Let a, b, c, d and e represent detected poles with 1, 2, 3, 4, and 5 represent identified poles with D being the distance between them.

$$\begin{matrix} D_{a1} & D_{a2} & D_{a3} & D_{a4} & D_{a5} \\ D_{b1} & D_{b2} & D_{b3} & D_{b4} & D_{b5} \\ D_{c1} & D_{c2} & D_{c3} & D_{c4} & D_{c5} \\ D_{d1} & D_{d2} & D_{d3} & D_{d4} & D_{d5} \\ D_{e1} & D_{e2} & D_{e3} & D_{e4} & D_{e5} \end{matrix}$$

Once the matrix is filled it is able to be used for data association. For each detected pole the relevant row is scanned to determine the know pole that is closest. Once this pole is determined, the known pole's column is also scanned to ensure that none of the other detected landmarks are a better fit for this pole. Once all the detected landmarks have their best match, the distance is then checked between each pair to ensure it is less than 50 cm, the threshold for this project.

Part 1

The EKF localiser method is utilised in this project for updating the platform's position. The EKF uses the expected and measured range and bearing in order to determine the update step. The EKF relies on a process model in order to determine the next position of the platform.

Process Model

The Process model used for determining the next position of the platform is as follows:

$$\begin{pmatrix} X\ Position_{next} \\ Y\ Position_{next} \\ Heading_{next} \end{pmatrix} = \begin{pmatrix} X\ Position_{current} \\ Y\ Position_{current} \\ Heading_{current} \end{pmatrix} + dT \begin{pmatrix} Speed \times \cos(Heading_{next}) \\ Speed \times \sin(Heading_{next}) \\ \Delta\ Heading \end{pmatrix}$$

Initial Values

Several values for the EKF are assumed at the beginning of the process. The potential noises in the measurements are included in these values. These potential noises are assumed to be:

• Noise in Gyroscope	= 0.035
• Noise in Speed	= 0.3
• Noise in range	= 0.25
• Noise in Bearing	= 0.0087

The position of the platform is also assumed at the beginning of the simulation. The beginning position of the platform was assume to be XY(0,0) with a heading of 90°.

The EKF also relies on several matrices which have to be determined or initialised at the start of the program. These include the covariance matrix P and the noise matrix R. The noise matrix remains the same throughout the simulation as the potential noise does not change. The noise matrix is:

$$\begin{pmatrix} 4 \times Noise\ in\ Range^2 & 0 \\ 0 & 4 \times Noise\ in\ Bearing^2 \end{pmatrix}$$

The covariance matrix is initialised at the beginning of the simulation, but also requires and update after each iteration of the EKF. The initialisation process is as follows:

$$Qi = \begin{matrix} 0.001 & 0.001 & 0.0003 \end{matrix}$$

Qi is due to the level of uncertainty in the process model, the uncertainty in X, Y and the Heading.

$$Pu = \begin{matrix} 0.09 & 0.001225 \end{matrix}$$

Pu is due to the level of uncertainty in the noise for speed and gyroscope.

$$J = \begin{matrix} 1 & 0 & -dT \times speed \times \sin(heading) \\ 0 & 1 & -dT \times speed \times \cos(heading) \\ 0 & 0 & 1 \end{matrix}$$

$$Ju = \begin{matrix} dT \times \cos(Heading) & 0 \\ dT \times \sin(Heading) & 0 \\ 0 & dT \end{matrix}$$

J and Ju are the jacobian matrices for the process model

Once all the initial matrices are filled the Q matrix can be determined for the determination of the covariance matrix.

$$Q = Qi + Qu \qquad where\ Qu = Ju \times Pu \times Ju'$$

$$P = J \times P \times J' + Q$$

EKF Process

For the EKF process an H matrix must be determined for each landmark at each iteration. The H matrix is determined from the position of the landmark and the position of the platform. The H matrix is as follows:

$$H = (\begin{matrix} \frac{x_L - x}{\sqrt{(x_L - x)^2 + (y_L - y)^2}} & \frac{y_L - y}{\sqrt{(x_L - x)^2 + (y_L - y)^2}} & 0 \\ \frac{y_L - y}{(x_L - x)^2 + (y_L - y)^2} & \frac{x - x_L}{(x_L - x)^2 + (y_L - y)^2} & -1 \end{matrix}$$

From the H matrix we can then update the position of the platform and the covariance, the determination of this is as follows:

$$S = R + H \times P \times H'$$

$$K = P \times H' \times iS$$

$$X_{new} = X + K \times z$$

Update of position where z is the difference between expected and measured values.

$$P = P - P \times H' \times iS \times H \times P$$

Update of covariance matrix P.

The following figures 4, 5, 6, and 7 show the results of the EKF process at varying stages of the platforms path.

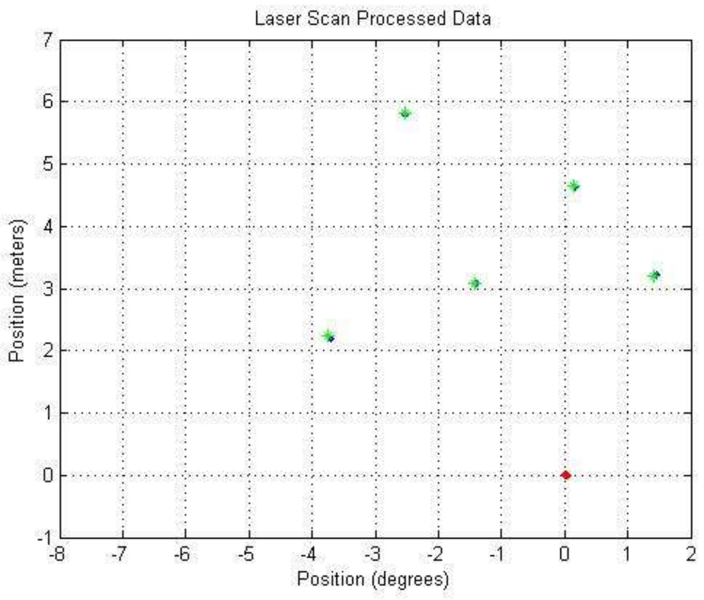


Figure 4: Plot of EKF at 10% of simulation

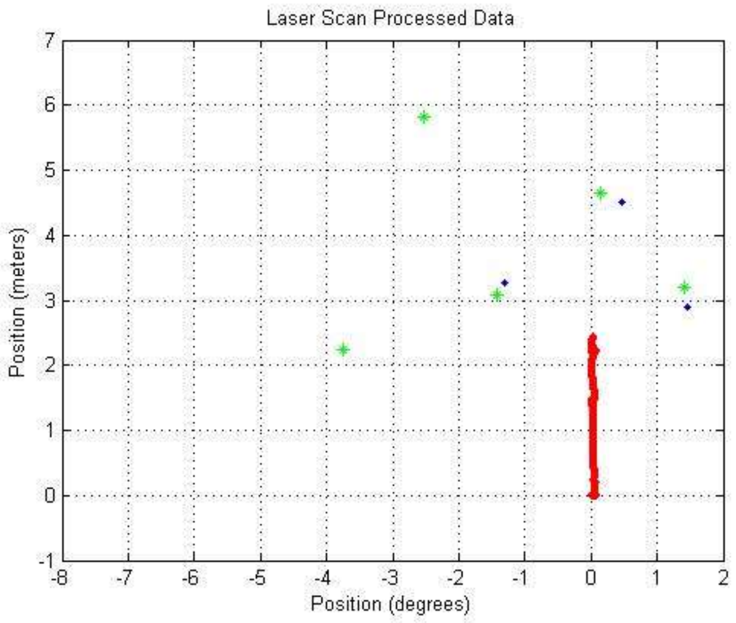


Figure 5: Plot of EKF at 30% of simulation

Z3373093
William Johnston

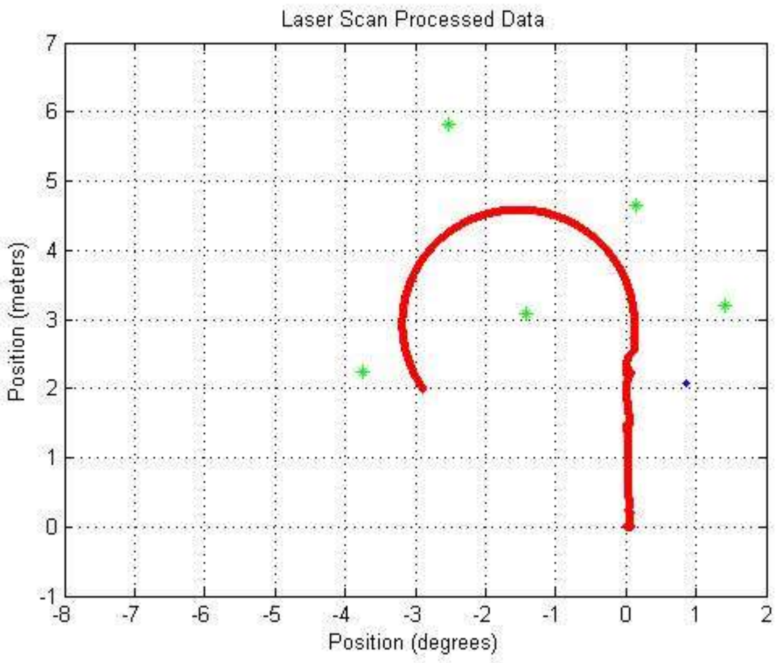


Figure 6: Plot of EKF at 60% of simulation

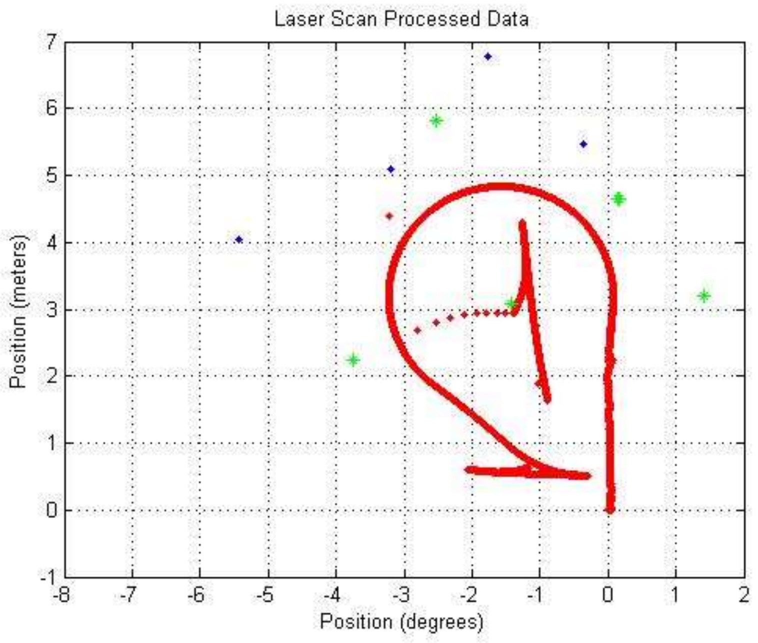


Figure 7: Plot of EKF at End of simulation

Part 2

For Part 2 the bias, which was previously estimated in part 1, is determined as part of the EKF process in part 2. To achieve this several changes had to be made to the process model, the matrices used to determine the covariance matrix and the H matrix. The updated process model is as follows:

$$\begin{pmatrix} X\ Position_{next} \\ Y\ Position_{next} \\ Heading_{next} \\ Bias_{next} \end{pmatrix} = \begin{pmatrix} X\ Position_{current} \\ Y\ Position_{current} \\ Heading_{current} \\ Bias_{current} \end{pmatrix} + dT \begin{pmatrix} Speed \times \cos(Heading_{next}) \\ Speed \times \sin(Heading_{next}) \\ \Delta\ Heading \\ 0 \end{pmatrix}$$

For the inclusion of bias the Qi, J, Ju, and H matrix had to be adapted. The new matrices are as follows:

$$Qi = \begin{matrix} 0.001 & 0.001 & 0.0003 & \left(\frac{\pi \times dT}{648000}\right)^2 \end{matrix}$$

$$J = \begin{matrix} 1 & 0 & -dT \times speed \times \sin(heading) & 0 \\ 0 & 1 & -dT \times speed \times \cos(heading) & 0 \\ 0 & 0 & 1 & -dT \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$Ju = \begin{matrix} dT \times \cos(Heading) & 0 \\ dT \times \sin(Heading) & 0 \\ 0 & dT \\ 0 & 0 \end{matrix}$$

$$H = (\begin{matrix} \frac{x_L - x}{\sqrt{(x_L - x)^2 + (y_L - y)^2}} & \frac{y_L - y}{\sqrt{(x_L - x)^2 + (y_L - y)^2}} & 0 & 0 \\ \frac{y_L - y}{(x_L - x)^2 + (y_L - y)^2} & \frac{x - x_L}{(x_L - x)^2 + (y_L - y)^2} & -1 & 0 \end{matrix}$$

The following figures 8, 9, 10, and 11 show the results of the EKF process at varying stages of the platforms path.

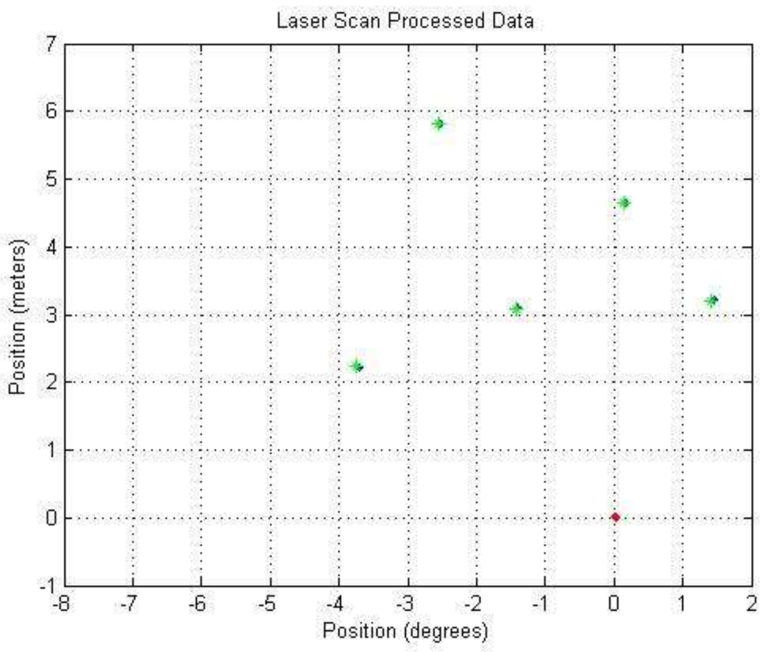


Figure 8: Plot of EKF with Bias at 10% of simulation

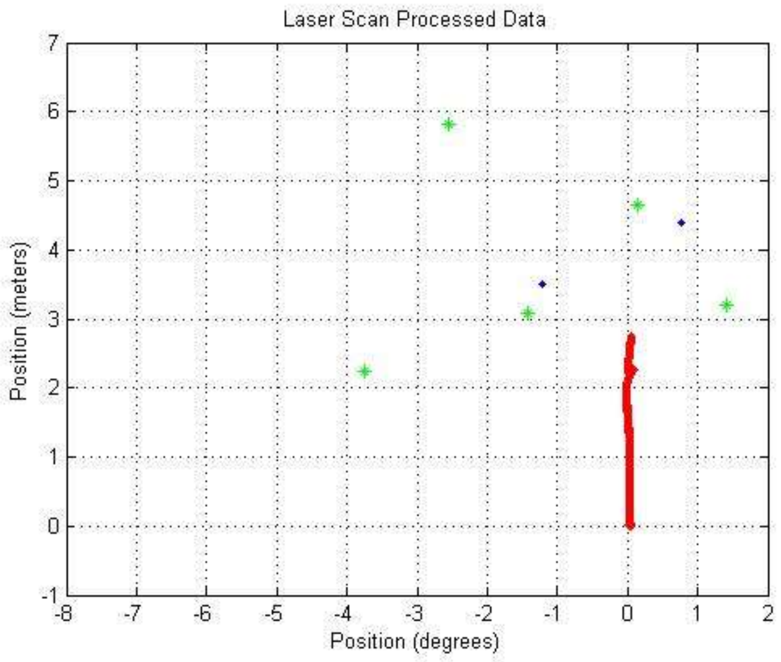


Figure 9: Plot of EKF with Bias at 30% of simulation

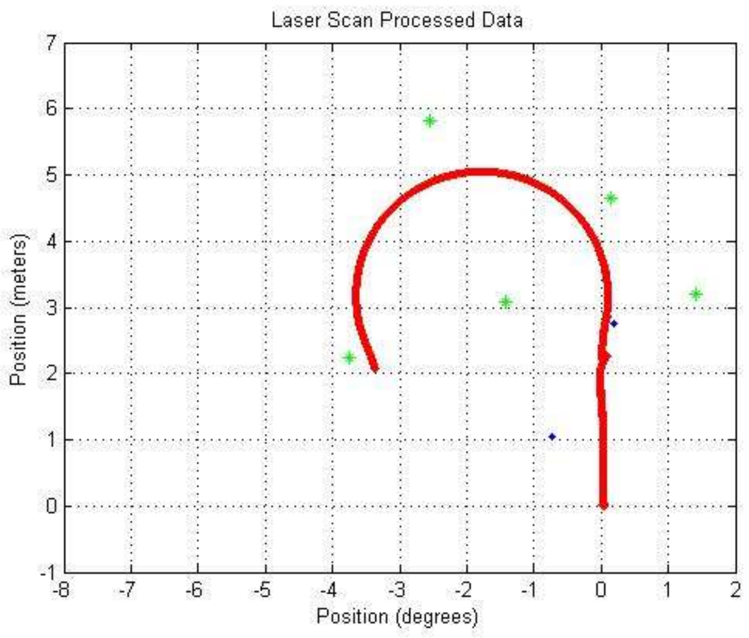


Figure 10: Plot of EKF with Bias at 60% of simulation

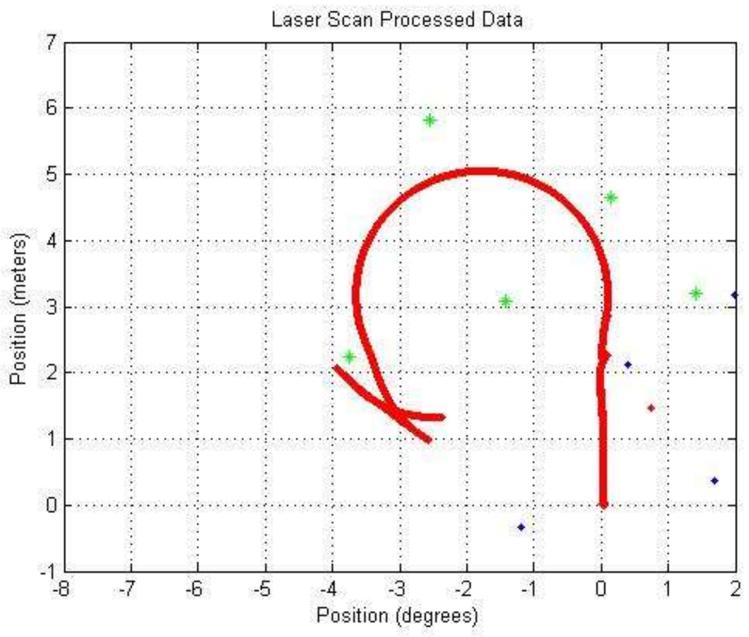


Figure 11: Plot of EKF with Bias at End of simulation