Robot Probabilistic Localization using Redundant Landmarks

Tiago Silva Pereira, 98360

Introduction

The task involves localizing a robot in an environment with multiple landmarks (or beacons) along a predefined path. These landmarks have known positions, and the robot is equipped with sensors to detect and differentiate the beacons, measuring their distance and orientation relative to its current position with a certain degree of uncertainty. However, the system should be adaptable to function even with partial information. We employed Extended Kalman Filter, incorporating appropriated motion and sensor models. The goal is that we can continually update the robot's estimated position by integrating sensor measurements with the predicted state using EKF

Functions

In this project we used some different functions:

- pchip_path
- linear_path
- velocities
- EKF_Generate
- motionmodelDD_TRI
- ekf
- motionmodel
- plot_paths
- plot_velocities
- GetConv
- plot_ekf_data
- plot_model_velocities
- jacobi
- sensormodel

Linear_path

Calculate the number of steps that the robot is going to do between point A and B

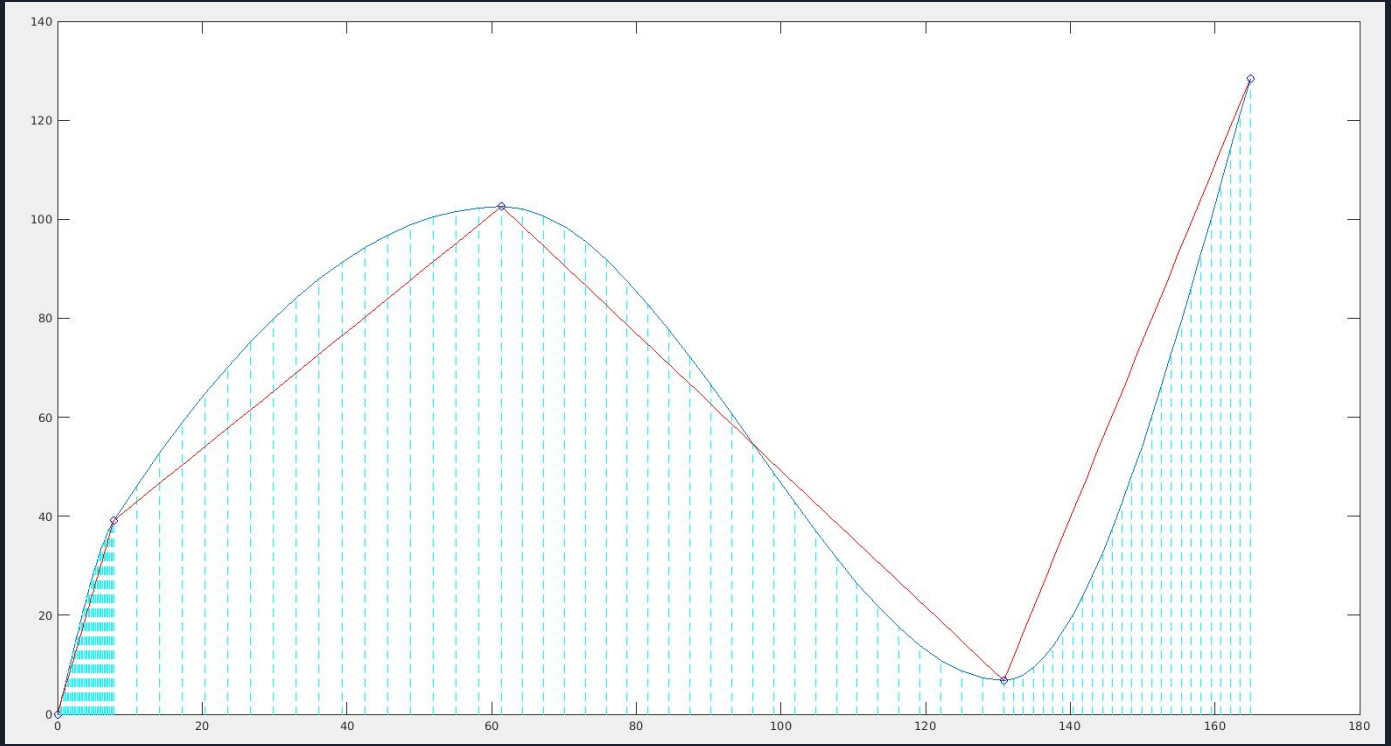
$$\frac{\sqrt{(x\_(2) - x\_(1))^2 + ((y\_(2) - y\_(1)^2}}{(Dt * Vn)}$$

pchip_path

Calculates the smother path possible, using the Hermite polynomial interpolation.

It uses the x’s coordinates and uses the pchip matlab function to calculate the y coordinates.

plot_paths



velocities

This function is going to calculate the linear and angular velocitie between a pair of beacons.

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\arcsin \frac{y_{t+1} - y_{t-2}}{d}$$

$$v_t = \frac{d_t}{Dt} * (1 + NOISE)$$

$$w_t = \frac{\Delta\theta_t}{Dt} * (1 + NOISE)$$

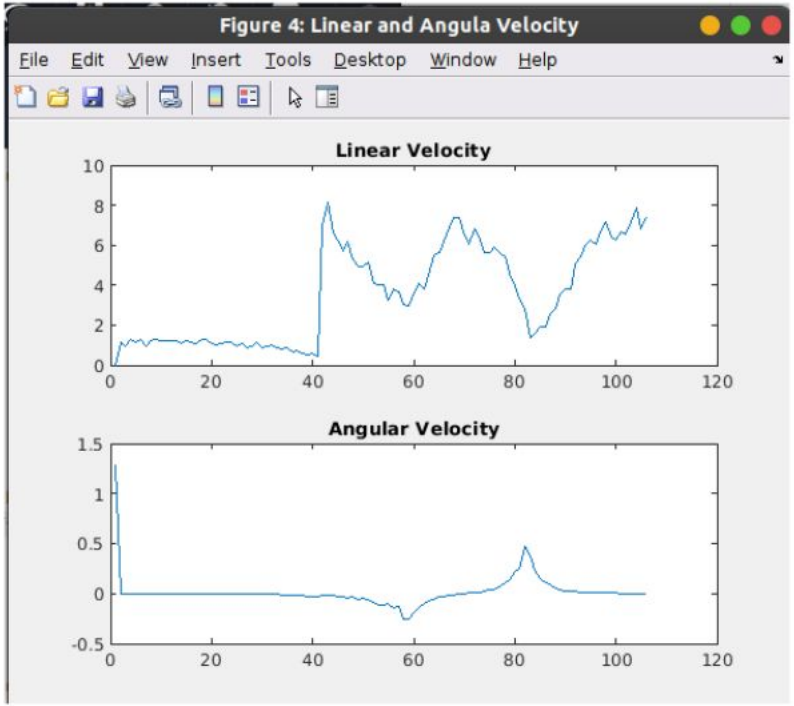


Figure 4: Linear and Angula Velocity
File Edit View Insert Tools Desktop Window Help
Linear Velocity
10
8
6
4
2
0
0 20 40 60 80 100 120
Angular Velocity
1.5
1
0.5
0
-0.5
0 20 40 60 80 100 120

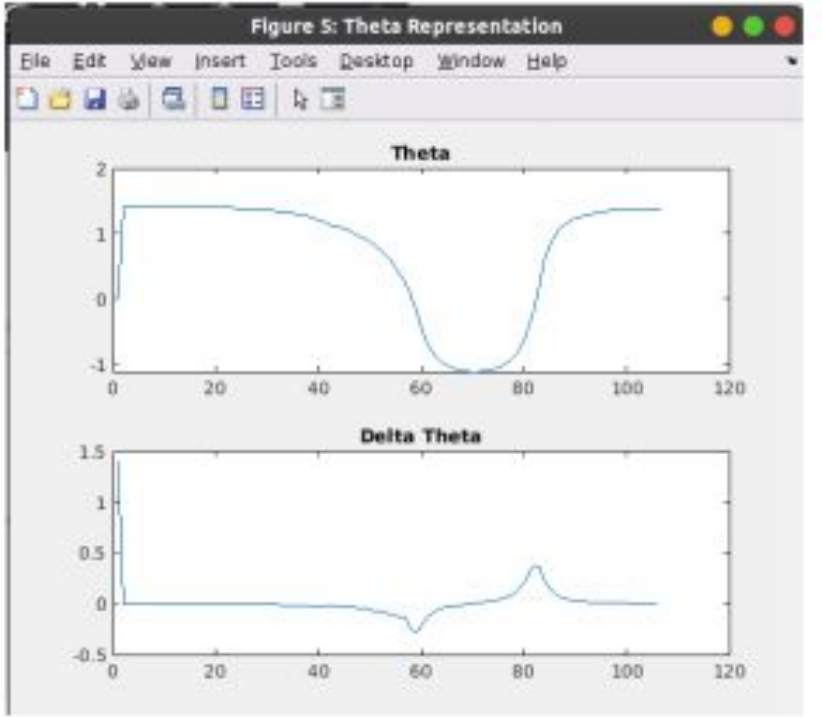


Figure 5: Theta Representation
File Edit View Insert Tools Desktop Window Help
Theta
2
1
0
-1
0 20 40 60 80 100 120
Delta Theta
1.5
1
0.5
0
-0.5
0 20 40 60 80 100 120

ekf

This functions is one of the most important in our program and be composed in several steps.

1. Motion Model
2. Uncertainty of the robot measures
3. Calculate the Jacobian of the sensor model
4. Measurement Error and Error covariance
5. Calculus of the estimation position and orientation

Motion Model

$$x_{nextstate} = x_{state} + (V_{in}(1) + D_n(1)) * t * cos(\theta_{state})$$

$$y_{nextstate} = y_{state} + (V_{in}(1) + D_n(1)) * t * sin(\theta_{state})$$

$$\theta_{nextstate} = \theta_{state} + (V_{in}(2) + D_n(2)) * t$$

Uncertainty

We need to calculator the uncertainty, using the matrix of the predicted covariance and noise.

-

$$Jfx * P_t * Jfx' + Jfw * Q * Jfw'$$

$$Jfx = \begin{bmatrix} 1 & 0 & -\delta_t * input * sin(est_\theta \\ 0 & 1 & -\delta_t * input * cos(est_\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$Jfw = \begin{bmatrix} -\delta_t * input * cos(est_\theta & 0 \\ -\delta_t * input * sin(est_\theta & 0 \\ 0 & \delta_t \end{bmatrix}$$

Jacobian

$$\begin{bmatrix} a & b & 0 \\ c & d & -1 \end{bmatrix}$$

$$a = \frac{x - xym(1)}{\sqrt{(x - xym(1))^2 + (y - xym(2))^2}}$$

$$b = \frac{y - xym(2)}{\sqrt{(x - xym(1))^2 + (y - xym(2))^2}}$$

$$c = \frac{y - xym(2)}{(\frac{(y-xym(2))^2}{(x-xym(2))^2} + 1)(x - xym(1)^2)}$$

$$d = \frac{1}{(\frac{(y-xym(2))^2}{(x-xym(2))^2} + 1)(x - xym(1)^2)}$$

Measurement Error and Error Covariance

$$S = J * P_t * J' + R$$

$$K = P_t * J' * S^{-1}$$

Estimated position and orientation

$$x_{t+1} = x_t + K * (error)$$

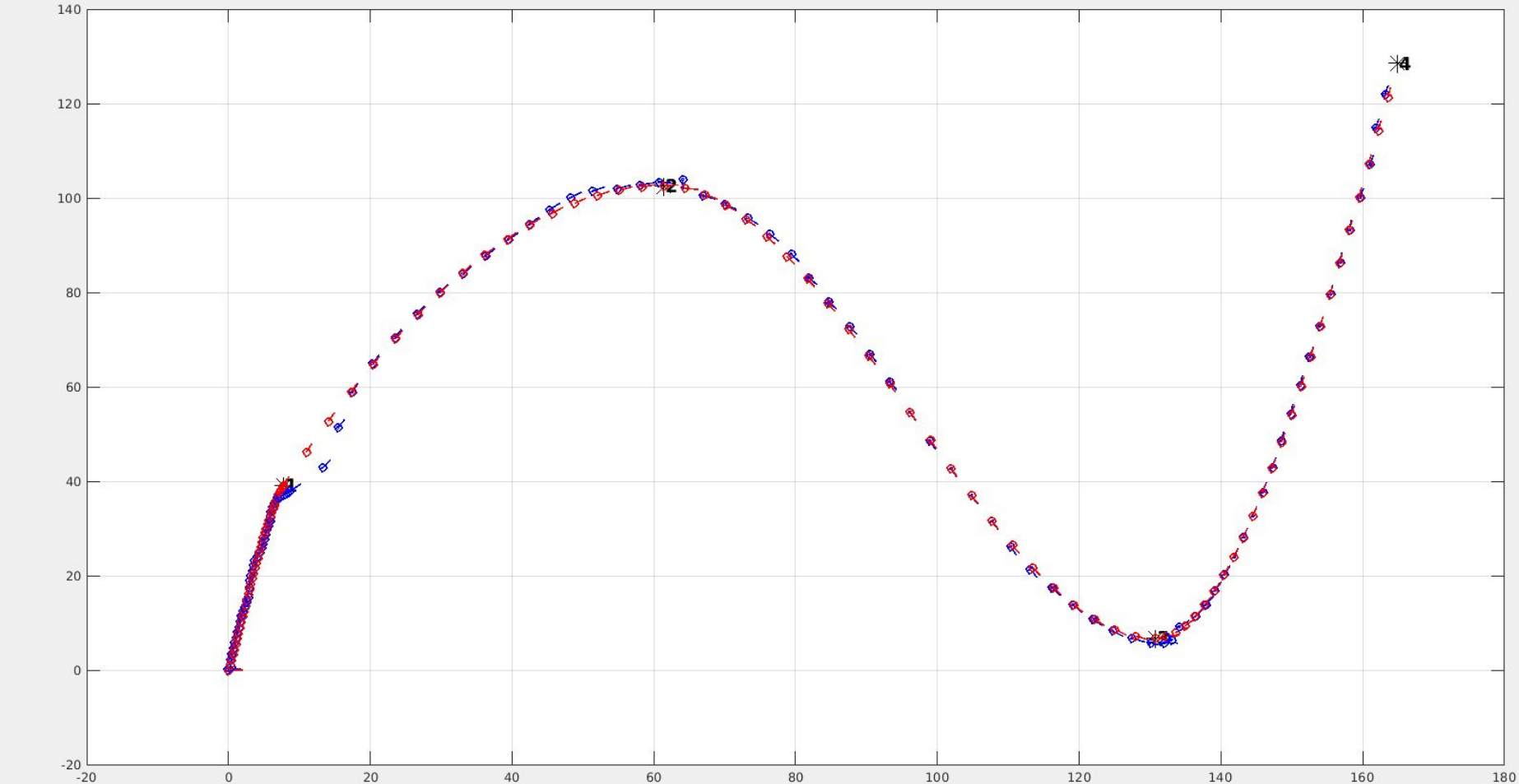
$$P_{t+1} = (I - K * J) * P_t$$

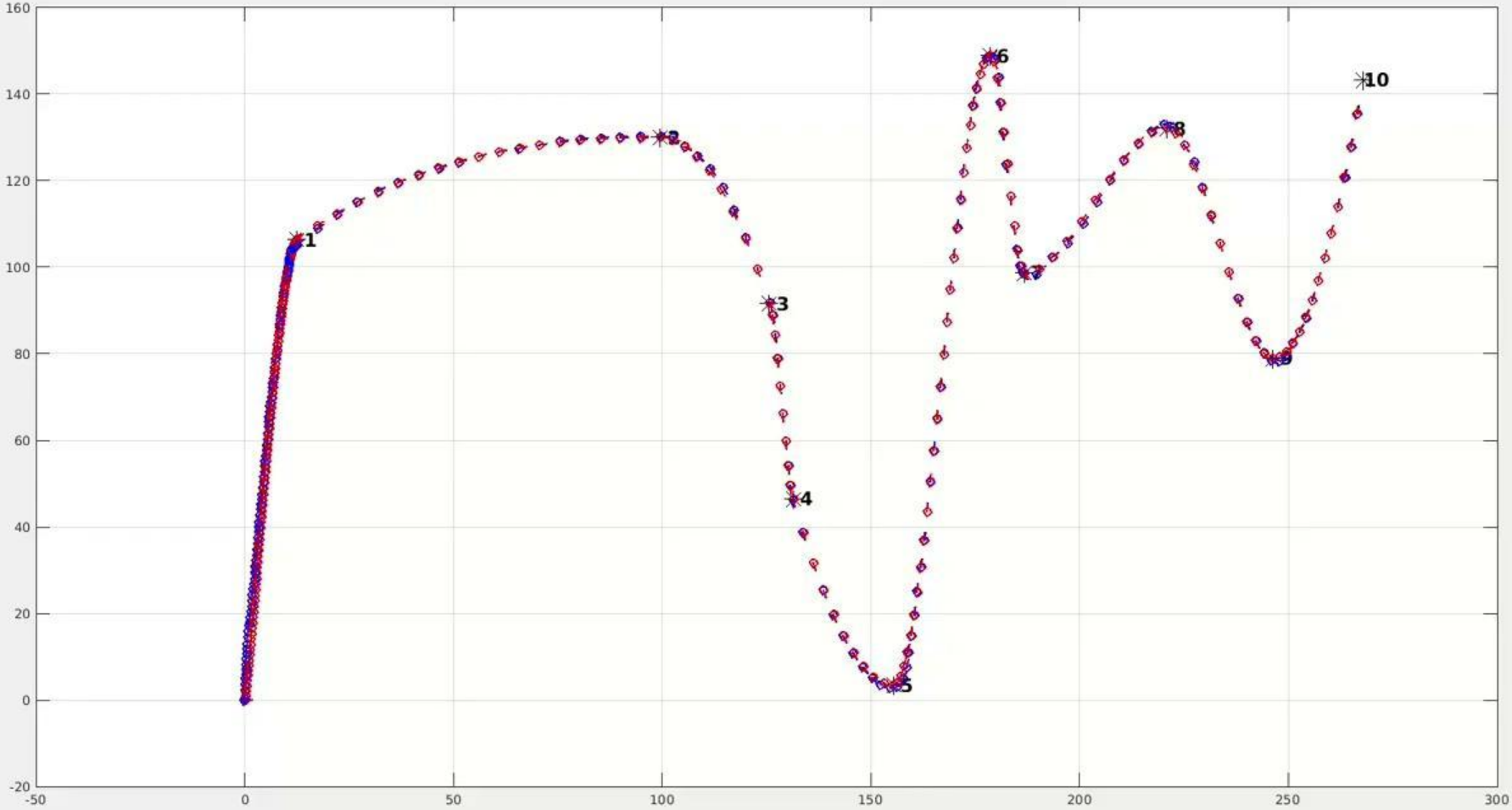
EKF_generate

This function is crucial for our project as it generates and prepares the values used in the Extended Kalman Filter (EKF).

In the first part, two variables assist the prediction, while others verify the accuracy of the Kalman Filter. We start by generating the true robot poses, the true state used in the EKF, followed by creating the observation data.

In the second part, the function obtains the true and estimated positions, the range and bearing of the beacons at all times, and the control input. We can compare the estimated values and the true position of the robot. The estimation is generally accurate, but discrepancies in orientation appear near the tops and downs defined by the beacons.


140
120
100
80
60
40
20
0
-20
-20
0
20
40
60
80
100
120
140
160
180
1
2
3
4


160
140
120
100
80
60
40
20
0
-20
-50
0
50
100
150
200
250
300
1
2
3
4
5
6
7
8
9
10

motionmodel_DD_TRI

To obtain the linear and angular velocities of the virtual DD and Tricycle robots, we need to take in account the following formulas

$$DD_{rightwheel} = \frac{v + w * \frac{L}{2}}{r}$$

$$DD_{leftwheel} = \frac{v - w * \frac{L}{2}}{r}$$

$$TRI_{velocity} = v - w * L$$

$$TRI_{\alpha} = arcsin(\frac{w * L}{TRI_{velocity}})$$

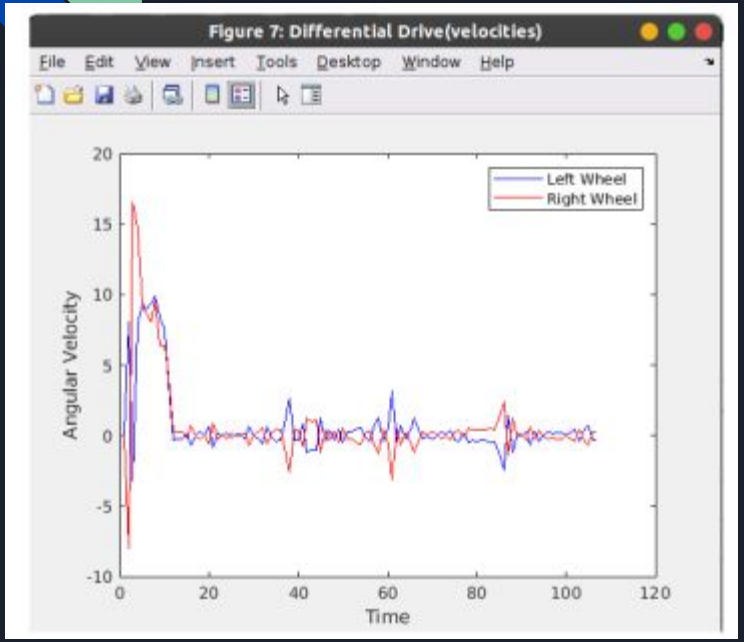


Figure 7: Differential Drive(velocities)
File Edit View Insert Tools Desktop Window Help
Left Wheel
Right Wheel
Angular Velocity
Time
20
15
10
5
0
-5
-10
0
20
40
60
80
100
120

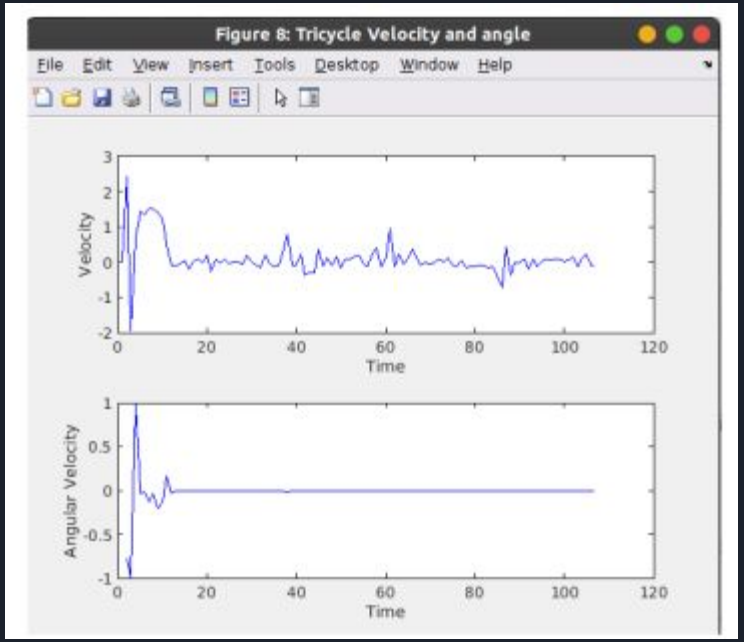


Figure 8: Tricycle Velocity and angle
File Edit View Insert Tools Desktop Window Help
Velocity
Time
3
2
1
0
-1
-2
Angular Velocity
Time
1
0.5
0
-0.5
-1
0
20
40
60
80
100
120

Errors between real and predicted

By calculating the norm of the difference between the values of the predicted states and the real positions, we can find the errors between them.

.	Average Error	Smallest Error	Maximum Error	Deviation
x	0.214343	0	1.503143	0.210749
y	0.294070	0	1.726984	0.337704
θ	0.072853	0	1.430293	0.198748

Conclusion

This project provided valuable insights into the Extended Kalman Filter (EKF) for predicting a robot's position. Despite its robustness, the EKF is not infallible, with discrepancies arising from sensor noise and movement uncertainty. These limitations were evident in our results.

Though not perfect, the EKF remains a powerful tool for estimating robot positions. The similarities in results for Direct Drive and Tricycle robots are due to the similar velocity equations, with primary differences being the Tricycle's longer length and third wheel.

In conclusion, while the EKF is useful for robot localization, recognizing and addressing its limitations is essential for improving accuracy.