Robot Probabilistic Localization using Redundant Landmarks

João Clemente 98212

May 5, 2024

0.1 Introduction

This project aims to develop a localization system tailored for a robot navigating environments featuring multiple landmarks/beacons. It involves implementing the Extended Kalman Filter alongside suitable motion and observation models. The primary output of our simulation will include the successive estimation of the robot's positions along its trajectory. Maximizing beacon utilization is prioritized, though potential beacon detection failures require strategies for handling missing information within the observation function. The ensuing report will detail the implementation of this simulation utilizing the Extended Kalman Filter and delve into discussions on various simulation parameters and their impacts.

0.2 Methodology

0.2.1 Initialization

Firstly, the initialization of the variables that will be used in the simulation is performed.

- **N** - number of beacons (default = 4)
- **Dt** - sensors sampling time interval (default = 1 second)
- **r** - radius of the robots wheels (default = 0.15 meters)
- **L** - separation/distance of the wheels according to the kinematic model (default = 1 meter)
- **Vn** - uncertainty (sigma) in the linear velocity to be imposed on the robot (input) (default = 0.1 m/s)
- **Wn** - uncertainty (sigma) in the angular velocity to be imposed on the robot (input) (default = 0.1 m/s)
- **V** - average velocity of the robot (default = 5 m/s)

All of them are optional and if not passed the default values will be used.

0.2.2 Robot trajectory

Using the coordinates obtained from the *BeaconDetection* function, which is useful for simulating beacon placements that are necessary for navigation, the expected trajectory of the robot is determined. The function provides the x and y coordinates affected by observational noise, mimicking real-world sensor uncertainties. The trajectory calculation starts with setting initial parameters like the robot's starting position (P) and observational noise. Initially, the robot's path is linearly plotted between successive beacons to depict a simple route.

To achieve a smoother and more realistic trajectory, Piecewise Cubic Hermite Interpolating Polynomial (*PCHIP*) is applied. This interpolation method calculates intermediate points between beacons based on their distance, the robot's velocity (V), and the time step (Dt), ensuring a detailed and smooth path for the robot's movement from one beacon to the next.

0.2.3 Velocities at each trajectory point

To enable the robot to navigate its environment effectively, it is crucial to determine the velocities required at each trajectory point. These velocities dictate how the robot transitions from one pose to the next, influencing both its speed and orientation adjustments over time.

Therefore, the *calculateVelocities* function processes each pair of successive points from the *true_points* matrix, which contains the x, y coordinates and orientation angle (θ) for each point along the path.

To compute both the linear and the angular velocities, the following method was used:

Linear velocity: $V_n = \frac{\sqrt{(X_{i+1}-X_i)^2+(Y_{i+1}-Y_i)^2}}{Dt} \times \mathit{randn()}$
Angular velocity: $W_n = \frac{\theta_{i+1}-\theta_i}{Dt} \times \mathit{randn()}$

Since, in realistic robotic applications, measurements and controls are susceptible to various sources of noise, it was added to both linear and angular velocities.

0.2.4 Extended Kalman Filter

The implementation of the Extended Kalman Filter (EKF) required the correct adaptation from the lectures' example, since it differed in the number of beacons *N*.

Generate data for EKF

The function generate_ekf is crucial for preparing the simulation, creating realistic sensor data and control inputs that simulate the robot's environment interaction. It generates noise-free and noisy control inputs, noisy landmark observations, and captures the robot's true path. Additionally, the process includes steps to manage data inconsistencies and randomize landmark detection sequence, enhancing the algorithm's robustness and realism.

Prediction with EKF

The ekf_N_localization function manages the robot's state estimation, processing each trajectory step using noisy inputs and observations. It begins by initializing state and covariance matrices, which track the robot's estimated position, orientation, and associated uncertainties. The prediction phase uses these inputs to estimate the robot's next state, incorporating process and observation noise models. Observational data are then integrated, adjusting the state estimate based on the difference between actual and expected landmark observations. This refinement uses computed innovation covariances and Kalman gains to merge the new data with prior estimates effectively.

0.3 Results

0.3.1 Expected Robot Trajectory

As previously discussed, the difference between linearized and pchip path generation is shown by the left and middle figures of 1. Besides, the middle and right plots 1 depict the effect of different sampling times. For a higher *Dt* the number of points between consecutive beacons is lower.

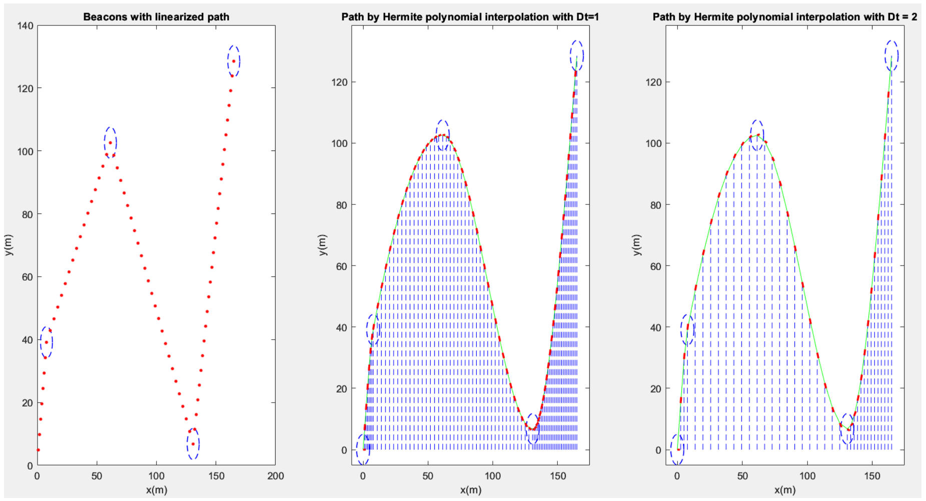


Figure 1: Left and Middle plot represent the difference between Linearization and Hermite polynomial interpolation path generation, respectively. Middle and Right figures translate the sampling time difference (*Dt=1* and *Dt=2*, respectively).

Different beacons also affect the path generation, as displayed in the figure 2.

0.3.2 Velocities at each trajectory point

In figure 3, the difference between linear and angular velocities for each trajectory point with and without noise can be noticeable. In the 3a the number of beacons N is 8 while in the 3b N is 4.

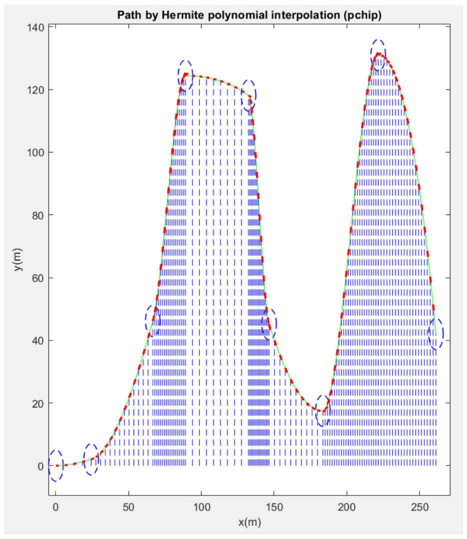
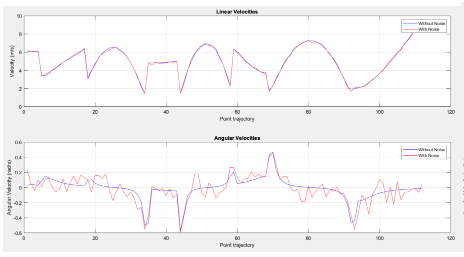
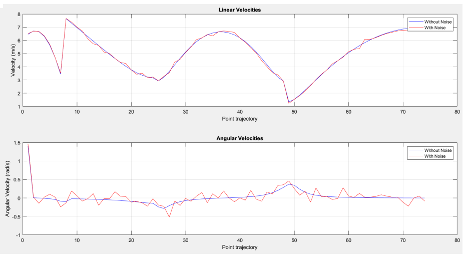


Figure 2: Path by Hermite polynomial interpolation, for N=8 beacons



(a) Velocities for Number of beacons = 8



(b) Velocities for Number of beacons = 4

Figure 3: Comparison of linear and angular (non-noisy and noisy) velocities for different numbers of beacons

0.3.3 Extended Kalman Filter

The figure 4 shows a simulation with N=4. It can be seen that, the real path obtained from the *PCHIP* method, in red, and the estimated result from the Extended Kalman Filter, in blue, do not superimpose. A little offset is noticed alongside all the path, specially between the origin and the first beacon that the robot encounters.

0.4 Conclusion

To sum up, this report has demonstrated that utilizing the Extended Kalman filter for robot localization closely matched the anticipated poses. The expected

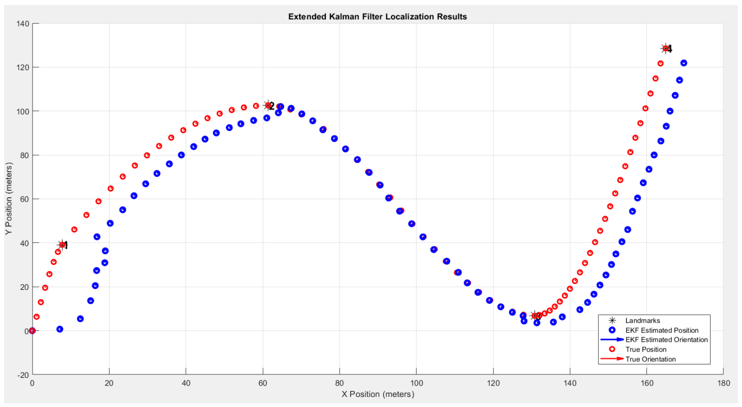


Figure 4: Plot of the expected EKF path prediction (red) alongside with the real EKF prediction (blue)

trajectory of the robot was effectively deduced by incorporating the distance between beacons, the robot's average velocity, and the sampling time interval. Additionally, trajectory points were generated under varying conditions of the sampling interval and beacon count, allowing observation of the impact of these variables on the trajectory. Velocities at each point were computed with and without noise inclusion. Notably, the Extended Kalman filter's prediction showed a significant initial offset in the robot's trajectory from the starting point to the first beacon, comparing to the rest of the path. Although the complexities of deriving accurate Extended Kalman Filter estimations precluded velocity calculations for various robot kinematic models like Differential drive and Tricycle, the project largely addressed the key objectives.