



**MTE 544 Lab 2**

**Prof. M Biglarbegian**

**MTE 544 – Autonomous Mobile Robots**

**Due: March 13th, 2020**

By:

Reza Rajan – 20599340

Cameron Blair – 20614860

# Preamble

This lab focuses on mapping and state estimation using a Turtlebot’s sensors, an external indoor positioning system (IPS) and an Extended Kalman Filter (EKF). Specifically, for mapping, the lab aims at mapping to an occupancy grid, using laser scan data, and Bresenham’s line plotting algorithm, in conjunction with a log odds update. The state estimation portion is based on implementing an EKF on position data modified with additive gaussian noise, requiring the engineer to test various noise covariance matrices to produce suitable state predictions.

# Mapping

The requirements of the mapping portion of the lab are detailed as follows:

1. Create a line plotting algorithm using Bresenham’s line plotting algorithm;
2. Implement a log odds update algorithm and then;
3. Using Kinect sensor data, construct a 2D occupancy grid of the area being mapped.

## Assumptions

1. The robot follows a non-holonomic, two-wheel differential-drive model;
2. The IPS data used for positioning is sufficiently accurate such that filtering of data is not required.

## 

## Motion Model and Control Input

The robot is assumed to follow the motion model:



Where is the timestep, and is the input vector defined as:

With being the forward speed of the robot and being the rotational speed.

The EKF requires that this motion model is linearized. This is done by taking the Jacobian, which results in

## Measurement Model

The measurement model represents how the robot’s positional data is transformed into a sensor reading. In this lab, the sensor used was the IPS, which directly measured the position. The measurement model is therefore

## Mapping Results

With the above, the mapping algorithm is implemented by functionalizing each part of the requirements and executing them sequentially. The results of the mapping are shown in three phases:

A picture containing flying, aircraft, sky, outdoor

Description automatically generated

Figure 1 - Initial Mapping Stage (1/3)

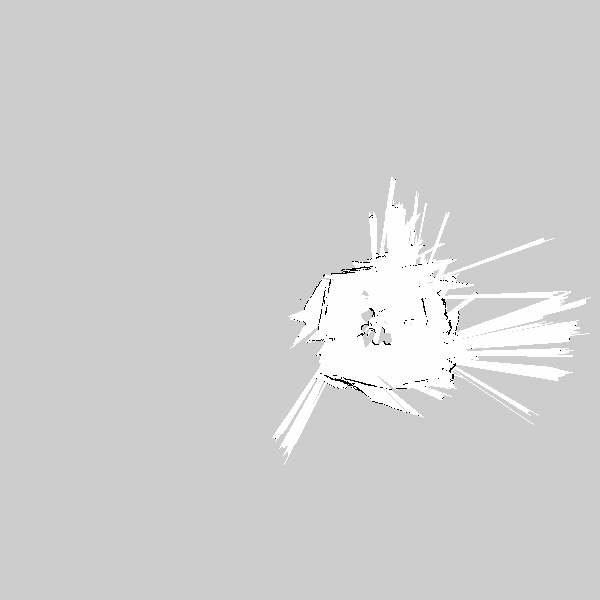


Figure 2 - Second Mapping Stage (2/3)

A close up of a flower

Description automatically generated

Figure 3 - Final Mapping Stage (3/3)

A picture containing indoor, white, photo

Description automatically generated

Figure 4 - Mapping Location

In Figure 1 to Figure 3, note that the image has been cropped. The resolution of the true occupancy grid contains empty space, which is out of bounds for the IPS, but is required due to positioning offset.

A close up of a logo

Description automatically generated

Figure 5 - Mapping Using Odometer Readings

As a point of comparison, the same mapping was performed using the odometer reading from the Turtlebot only; Figure 5 shows that odometer readings result in a less noisy, and thicker boundaries than with the maps created using the IPS readings. A potential reason for this may include the fact that greater processing power is required for IPS mapping, compared to odometer mapping. The algorithms are run in a VirtualBox environment, which severely limits the map-rendering capabilities of the system.

# State Estimation

This portion of the lab requires the following:

1. Design an EKF algorithm for the robot’s state estimation;
2. Create an algorithm for drawing the EKF covariance ellipse;
3. Introduce additive gaussian noise with a standard deviation of 10 cm to the IPS data;
4. Using wheel odometer readings and the degraded IPS data, perform EKF-based state estimation;
5. Tune the noise covariance matrices to produce suitable results for the filter.

As such, the code for each of these requirements has been functionalized and the steps have been performed sequentially. Note that an assumption made for ‎iv above is that the sensor data must be *fused*, i.e. the IPS provides position data, while the wheel odometer provides linear and angular velocities.

## EKF Parameters

The following are the parameters used in the EKF:

Measurement Noise Covariance Matrix:

This parameter is set based on the injected noise, with standard deviation of 10 cm (0.1m), and thus, variance of 0.01 m.

Process Noise Covariance Matrix:

This parameter should be tuned, so as a semi-random starting point, this is set to the equivalent of the measurement noise covariance matrix.

State Covariance Matrix, :

Since there is a large uncertainty about the system, this is initially set to represent the worst conditions, i.e. a covariance of 1.

Timestep, *dt*:

The timestep is set based on the time between the EKF function calls; when both position and odometer data is available, the timestep is set to the temporal difference in function calls. In this manner, the timestep is specifically dynamic, and does not require an assumption of data transmission times, though this creates additional processing requirements.

## State Estimation Results

The following results are based on an EKF-based state estimation, performed simultaneously with mapping *using* the IPS:

A screenshot of a computer

Description automatically generated

Figure 6 - State Estimation Results; Red: Degraded Position; Yellow: True Position

A screenshot of a computer

Description automatically generated

Figure 7 - State Estimation Results; Magenta: Estimated Position; Yellow: True Position

A screenshot of a computer

Description automatically generated

Figure 8 - State Estimation Results; Magenta: Estimated Position; Red: Degraded Position

A screenshot of a computer

Description automatically generated

Figure 9 - State Estimation Results; Red: Degraded Position; Magenta: Estimated Position; Yellow: True Position

As seen in the results, the estimated results track slightly closer to the true position than the degraded position. Particularly, where the EKF truly performs well is in its ability to filter the jitter in the degraded position data (Figure 8). Note that the covariance ellipses are shown in the figures, at the end of each path. They are calculated at a 95% confidence interval, and appears almost circular, indicating little covariance in the data.