Localization with the Extended Kalman Filter

Kaanthi Pandhigunta, Sean Wu

*Abstract*—This paper details the implementation of an EKF localization filter for the case study of a bot traversing a 10x10 m square on an open field.

1. Introduction

A bot equipped with a lidar sensor, IMU, and GPS was made to traverse a 10x10 meter square path located at the center of an open field. A pylon was placed at the center of the square, which serves as a known location for EKF localization to be performed. An image of the experimental setup is shown in Figure 1. Data was sampled at 10 Hz. This case study uses data provided in 2020\_2\_26\_\_17\_21\_59\_filtered.csv.



Fig. 1. Image of data collection with bot and pylon

1. Motion Model
2. Vectors

A 5 degree of freedom state vector in the global coordinate frame was defined as shown in Equation 1. Here, and are position and velocity in the East (+) and West (-) direction. Elements and are position and velocity in the North (+) and South (-) direction. is positive counterclockwise and 0 towards East.

The control input is defined as shown in Equation 2. Here, is the lateral acceleration or in the local coordinate frame from IMU, is the forward acceleration or in the local coordinate frame from IMU, and is the yaw velocity of the robot. IMU measurement is positive clockwise.

1. Kinematics

Because the robot is non-holonomic, assume there is no lateral slip. Therefore, the acceleration in the y direction ay’,t is 0. The x and y position are defined by Equations 3-4.

The velocities vx and vy are defined by Equations 5-6. Note that the global y velocity still depends on the forward x’ acceleration not lateral y’ acceleration.

Once Equations 5-6 are plugged into Equations 3-4, we get Equations 7-8.

The yaw angle of the bot is given by Equation 9.

1. Linearizing the Motion Model

The state transition probability function can be approximated as linear with added Gaussian noise, as shown in Equation 10-11.

The Gx matrix can be calculated by taking derivatives of the kinematics equations with respect to each element in the state vector. The resulting matrix is shown in Equation 12.

The Gu matrix is calculated by taking derivatives of the kinematics equations with respect to the control inputs. There are only two columns, however, because there is no lateral slip, so ay is 0. The resulting matrix is shown in Equation 13.

1. Measurement vector

The measurement vector, which contains data obtained from the lidar is shown in Equation 14. The equations are referenced to (5, -5) which is the location of the pylon. Here, ylidar is positive in the local forward direction, and xlidar is positive in the local left direction.

The measurement vector only affects the first, second, and last elements of the state vector. Thus, the H matrix is created to isolate just those elements of the state vector, as shown in Equation 15.

1. Extended Kalman Filter

The Kalman filter algorithm takes in the inputs , , , and .

* 1. Prediction step

Equations 16 were used in the prediction step. The estimated state is recalculated using the kinematics equations at each step. The covariance matrix is recalculated as well.

* 1. Correction step

The correction step includes the Kalman gain, the updated state, and the updated covariance matrix. The prediction and correction steps were both implemented for each time stamp.

* 1. Sensor description and covariance matrix

The lidar sensor and the IMU measurements are assumed to be independent, so the initial covariance matrices are identity matrices.

1. Results
   1. Estimated path

The Kalman filter was applied to all lidar data. The estimated path that was outputted by the filter is shown in Figure 2. The state estimates of the position over time were smoothed to closely match the lidar data and gave an improved position resolution as compared to the GPS data.

Chart

Description automatically generated

Fig. 2. Plot with estimated path from EKF

* 1. Tracking error

The tracking error of the estimated path was found by finding the shortest distance between the path and the perfect square at each point on the estimated path. The error is plotted as a function of time in Figure 3. The root mean squared error was found to 0.227. For all times, the error was under 0.6 meters.

Chart, histogram

Description automatically generated

Fig. 3. Tracking error of estimated path

* 1. Yaw plot

The yaw data from the state estimates is plotted in Figure 4. The angles shown in the figure match the angles of a square, which is roughly the path traversed by the bot.

Chart, histogram

Description automatically generated

Fig. 3. Estimated robot heading (yaw) over time

* 1. Covariance ellipses

The confidence ellipse was plotted at multiple points on

the estimated path using the 2x2 xy portion of the covariance matrix as shown in Figure 5 using established methods. [1]

Diagram

Description automatically generated

Fig. 5. Covariance ellipses plotted on estimated path

* 1. Covariance matrix elements as a function of time

Each element of the state estimate covariance matrix is plotted as a function of time in Figure 6.

Diagram, engineering drawing

Description automatically generated

Fig. 6. Plot of covariance matrix elements

The covariance drops dramatically on the diagonal elements, showing that the filter improves the confidence in the state over time.

References

1. C. Schelp, *An Alternative Way to Plot the Covariance Ellipse,* [*https://carstenschelp.github.io/2018/09/14/Plot\_Confidence\_Ellipse\_001.html*](https://carstenschelp.github.io/2018/09/14/Plot_Confidence_Ellipse_001.html)*,* Accessed Oct. 5, 2022.