# **Coursework Report**

## IMAT2721 Intelligent Robotics

Erva Karabulut P2860091

# Introduction

Localization is a crucial process in robotics to determine the robot’s position and orientation. The Bayes Filter predicts the new state of the robot based on past control inputs and past measurements. When the filter receives sensor measurements it updates its prediction. Kalman Filter (KF) is derived from the Bayes Filter, however it only works on linear environments, additionally this project is based on a non-linear environment. To address this, Extended Kalman Filter (EKF) linearises non-linear equations to make them suitable to use KF and work in a non-linear environment which is suitable for this project.

The goal of the project is to implement EKF to simulate robot localization. There are components of the robot used, to achieve this goal: a state vector which has 4 states at time t and represents the robot’s state, a covariance matrix which represents uncertainty, sensor inputs from a Speed sensor, a Gyro sensor, a Global Navigation Satellite System (GNSS) sensor which offer noisy information of the robot’s position and motion.

# Process model of EKF

Process model predicts changes in the robot’s state accounting dynamics of the robot and control inputs.

Process model consists of:

* **State Vector(xt)**

The robot has a state vector that models robot’s motion, it is represented as:

State vector incudes 4 states at time t, which are:



xt,yt: Robot’s position in 2D space.

Φt: Robot’s orientation (yaw angle).

vt: Robot’s velocity.

* **State Covariance Matrix ()**

To add uncertainty to the state its covariance matrix ()’s initial state at time t=0 can be modelled as:

diyagram, tipografi, tasarım içeren bir resim

Açıklama otomatik olarak orta güvenilirlik düzeyiyle oluşturuldu

where  is a variance

* **Motion Model with Noise**

The process is modelled as non-linear motion of the robot and a noise:



* **Gaussian Noise Covariance Matrix**

“Motion Model with Noise” equation is where is used to account uncertainty to the measurements as a zero-mean Gaussian white noise with covariance matrix () which means wx=N(0,Q) where can be represented as:

ekran görüntüsü, çizgi, diyagram, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* **Non-linear Motion Function**

The robot’s non-linear motion function is modelled as:

metin, yazı tipi, çizgi, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

is the **control input**, hence velocity is from control input and represents angular velocity.

Additionally, *Δt*  is the time interval.

* **State Transition Matrix**

Moreover, the state transition matrix (F) is derived as the Jacobian of the motion function. It is used to linearize the non-linear motion model and is defined as:

yazı tipi, metin, çizgi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Observation model of EKF

Observation model corrects the outputs of the process model based on two-dimensional (x-y) position measurements from GPS (GNSS).

Components of the observation model

* **Sensor Observation Vector:**

The actual observations from sensors at time t are defined as:

yazı tipi, metin, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

* **Observation Function**

Observation Function creates a link between the robot’s state and observed measurements, it is given as:

yazı tipi, metin, sayı, numara, simge, sembol içeren bir resim

Açıklama otomatik olarak oluşturuldu

* **Sensor Observation Model (zt)**

GPS (GNSS) sensor observation with accounting noise is represented as:



* **Gaussian Noise Covariance Matrix (R)**

 wz equals to a zero-mean Gaussian white noise with covariance matrix (R) which means . The observation noise covariance matrix is modelled as:

metin, yazı tipi, sayı, numara, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* **Jacobian Matrix**

Since the observation model is non-linear, Jacobian matrix of the observation function is used to linearise the model, it is defined as:

metin, yazı tipi, beyaz, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Compact formula of EKF

The EKF algorithm can be summarised in two steps:

1. **Predict**

The EKF algorithm predicts the robot’s next state using the Process Model:



Uncertainty gets to be added as the process noise covariance matrix (Q), where F is the Jacobian matrix of the motion function:



1. **Update**

The EKF algorithm updates the predicted state based on observation function h(x) and its Jacobian matrix (H).

yazı tipi, diyagram, beyaz, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Uncertainty is added with observation function noise covariance matrix (R). Additionally, K represents Kalman Gain which determines the weight given to the robot’s measurements. Update step is modelled as:

metin, yazı tipi, beyaz, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Experimental Analysis of EKF Localization

To evaluate the results of the EKF algorithm, testing scenarios were applied to main parameters that affect accuracy of localisation. These main parameters include:

1. **Noise Ratio Added to Control Input**

Uncertainty is added to control inputs (velocity and angular velocity) with Gaussian noise, it has a direct impact on process noise covariance matrix (Q).

1. **Process Noise Covariance (Q)**

Process noise covariance matrix affected by the noise ratio, models uncertainty in the process model. Moreover, process noise covariance is used in the prediction step of the EKF algorithm.

1. **Noise Ratio Added to Observation**

This noise ratio adds uncertainty to GNSS (sensor) measurements. Noise ratio has a dynamic impact on the observation noise covariance (R) based on the reliability of the signals it receives from sensors.

1. **Observation Noise Covariance (R)**

Observation noise covariance matrix represents uncertainty in the sensor measurements.

**Effects Parameters Have on Each Other**

High process noise (Q), increases reliability on sensor (GNSS) measurements meanwhile high observation noise (R), reduces reliability on sensor (GNSS) measurements. Levels of both observation noise covariance (R) and process noise covariance (Q), directly affect Kalman Gain. Additionally, Kalman Gain determines the reliability of each sensor measurement.

# **Visualization of Trajectories in EKF Localization**

The visualization illustrates how noise affects localization of the robot and reliability of performance of the EKF algorithm.

In this project the following trajectories are included:

* **True Trajectory (Red):** The actual path of the robot.
* **Observed Trajectory (Green):** The path of the robot which is obtained from noisy sensor measurements. Furthermore, observation noise covariance (R) has an impact on observed trajectory.
* **Estimated Trajectory (Blue):** This trajectory is estimated by the EKF algorithm with impacts of process noise (Q) and observation noise (R).

**Experiment 1**

In this experiment, the EKF algorithm was tested with following default parameter values:

* **Process Noise Covariance (Q):**

[0.15, 0.15, deg2rad(1.0), 1.2]

* **Observation Noise Covariance (R):**

[1.5, 1.5]

* **Noise Ratios Added to Control Input:**

[1.20, np.deg2rad(45.0)]

* **Noise Ratios Added to Observation:**

[1.0, 1.0]

**Experiment 1 Results Analysis**

(see Figure 1 in Appendix A)

* **Observed Trajectory (Green)**

Observed trajectory (green) is highly deviated from true trajectory (red) due to added noise in sensor measurements i.e. in observation covariance (R).

* **Estimated Trajectory (Blue)**

Estimated trajectory (blue) follows the true trajectory(red) closer than observed trajectory(green) which demonstrates that the EKF algorithm can correct noisy observations of sensor measurements to some extent.

**Next Step**

To achieve closer results to the ideal EKF algorithm, high deviations of observed trajectory(green) should be reduced. To address this, noise ratios added to observation and observation covariance matrix (R) values should be reduced. This will also indirectly cause estimated trajectory (blue) to get closer to true trajectory(red).

# **Experiment 2**

In this experiment, the EKF algorithm was tested with following parameter values:

* **Process Noise Covariance (Q):**

[0.15, 0.15, np.deg2rad(1.0), 1.2]

* **Observation Noise Covariance (R):**

[0.5, 0.5]

* **Noise Ratios Added to Control Input:**

[1.20, np.deg2rad(45.0)]

* **Noise Ratios Added to Observation:**

[0.5, 0.5]

**Experiment 2 Results Analysis**

(see Figure 2 in Appendix A)

* **Observed Trajectory (Green)**

Observed trajectory (green) is significantly closer to true trajectory (red) comparing to Experiment 1. This improvement happened due to reduction in observation noise covariance and noise ratios added to observation.

* **Estimated Trajectory (Blue)**

Estimated trajectory (blue) is closer to true trajectory (red) as well. Reduction in observation noise covariance (R) had a direct effect on Kalman Gain which caused increase in reliability to sensor measurements for the EKF algorithm.

# Conclusion

This project gave me a good understanding of the EKF algorithm. I understood components of EKF, relationship between those components, workflow of EKF and so on. Experiments on the EKF algorithm showed me how different parameters significantly affect accuracy of the results.

In addition, I improved my skills in different areas such as Python programming, visualising results of algorithms with matplotlib, critically analysing results of algorithms and algorithm design. The experience I gained with this project taught me the importance of the EKF algorithm in robot localization and gave me a solid foundation for my further studies in intelligence robotics.

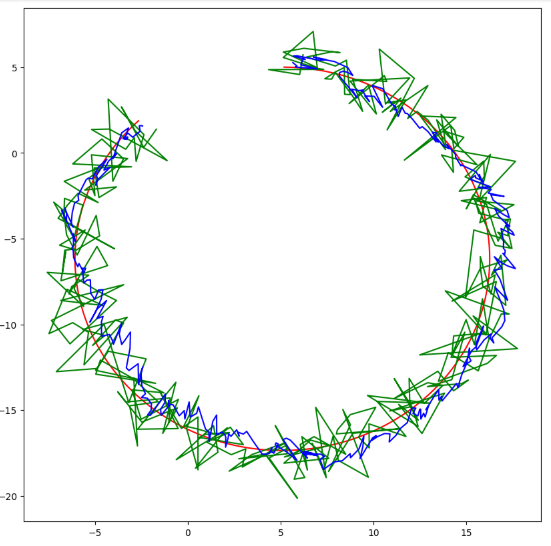
# Reference

*W3schools.com* (no date) *W3Schools Online Web Tutorials*. Available at: https://www.w3schools.com/python/default.asp (Accessed: 27 January 2025).

(No date) *Python numpy tutorial (with Jupyter and colab)*. Available at: https://cs231n.github.io/python-numpy-tutorial/ (Accessed: 27 January 2025).

# Appendix A

**Figure 1**



**Figure 2**

