I was working on the project, and after returning from dinner today, I found that my computer had turned off itself. Despite several attempts at restarting, it kept shutting down a few seconds after rebooting. Since I use dual-booting and couldn't access my files, I decided to proceed with writing the report… The images in the report are manually sketched as I was unable to take screenshots, and my video submission also includes the Kalman filter sensor integration from Project 2 🙁.

Introduction

Project 2 focuses on utilising the robot's depth camera and its internal odometry to implement Extended Kalman Filter (EKF) simultaneous localisation and mapping (SLAM). There are two goals when implementing EKF SLAM: to determine the robot's position within a predefined environmental map, a process known as robot localization, and to create a map of the environment using identifiable landmarks. The challenge of performing simultaneous localisation and mapping (SLAM) has a long and rich history, spanning over two decades. EKF-SLAM, in particular, has been historically significant in the field and continues to be relevant. It is especially recognised for its ability to close loops, a capability stemming from its maintenance of correlations between distant landmarks [1].

Environment setting

This project is done using the gazebo virtual world simulation, comparing where the robot thinks it is and its actual position. Also, comparing where the actual landmarks are and where the estimated landmarks are. The robot's depth camera and its internal odometry are used in a simulation environment.

Due to the computer issues I mentioned above, I was unable to capture a screenshot of the simulation world utilised for this project. As an alternative, I took a photo from Project 2 (Fig. 1) and manually sketched the setup of the environment (Fig. 2).

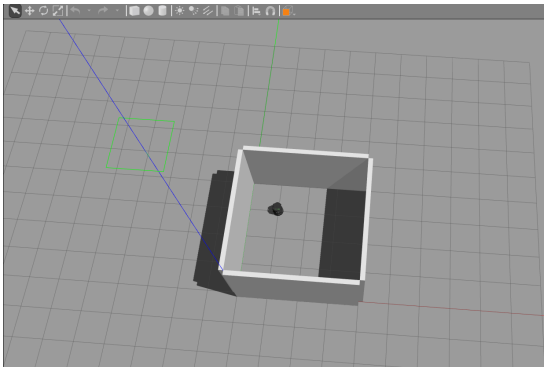

Fig. 1. Gazebo simulation world

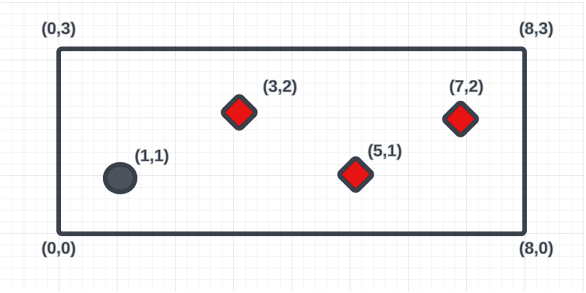

Fig. 2. Gazebo simulation world used for project 3

The environment setup here is an 8x3 rectangle where its bottom left corner is at (0,0). Therefore, the identified landmarks' corners are at coordinates (0,0), (8,0), (0,3), and (8,3). Additionally, I added three different cones which are in diamond red in Fig. 2. They are placed in (3,2), (5,1), and (7,2).

Project implementation

The robot will detect the distance to the walls and cones using depth sensor data. The depth camera is used by subscribing to the topic "/camera/depth/image_raw". The node subscribes to the depth image

from the camera. The distance information from the depth sensor is used in the correction phase of EKF later on.
Odometry determines the path taken by a robot by interpreting data gathered from various sensors, considering both its initial position and its recorded travel path to approximate its present location [2]. The internal odometry for our project is set to the known starting point, in this case (1,1) with yaw=0. This can determine the current position of the robot. Since the simulation world is used for this project, the Gaussian noise is added to the odometry data like Project 2. This odometry information is used in the prediction phase of EKF later on.

The robot's current orientation, combined with the angular position of a landmark relative to it, enables the computation of the landmark's position from the robot's perspective. This process involves trigonometric calculations that transform polar coordinates (distance and angle) into Cartesian coordinates (x, y). To get the y-coordinate of a landmark, the robot's existing y-coordinate is added to the product of the landmark's distance and the sine of the observed angle (sin(angle)). Similarly, the x-coordinate is derived by adding to the robot's current x-coordinate the product of the landmark's distance and the cosine of the angle (cos(angle)).
To implement the transformation of landmark positions to global coordinates, the robot's current position in the global frame is utilised. This involves using the transformation matrices that factor in the robot's position and orientation.

The next step in the implementation is integrating these global coordinates into the EKF for SLAM. The EKF keeps a state vector that includes the robot's position, orientation, and the positions of all landmarks. EKF has two phases: the prediction phase and the correction phase. During the prediction phase, based on the noisy odometry data, the EKF predicts the future state of the robot and the stationary landmarks. The Jacobian matrices are implemented here to get linear approximation from the non-linear motions.
When a landmark is detected again, the observation is updated, where its newly measured position is compared with its predicted position in the state vector, highlighting the importance of accurate distance and angle measurements. The EKF calculates a Kalman gain, which is a measure of new observations versus prior predictions, assigning more weight to more reliable sensor data. This is also used in Project 2 to integrate the depth sensor data and odometry data. Simultaneously, there's an update to the covariance matrix, which reflects the estimated uncertainty in the state vector.

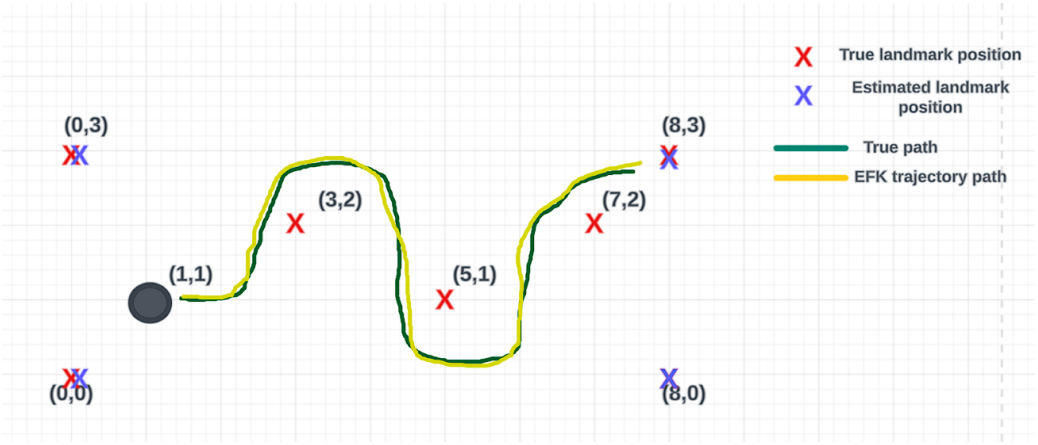

Fig. 3. Brief sketch of the grid plotting implementation

To visualise the data and the effectiveness of the implementation, various elements were plotted on a grid. Since getting a screenshot was not possible, I manually sketched the grid and the position implementation to illustrate the setup as described in Fig. 3. This included the actual landmarks as they are physically located in the environment, and their estimated positions as determined by the EKF. Additionally, both the real path of the robot and the EKF-projected trajectory were plotted. This visual representation helps in comparing the actual and estimated positions of landmarks, providing a clear illustration of the accuracy of the SLAM process. It also allows for an evaluation of the robot's real path against the refined trajectory calculated by the EKF. It provides insights into the precision and reliability of the localisation and mapping process in dynamic environments.

Fig. 4 shows the successful localisation using the Kalman filter in Project 2, using depth sensor and odometry data. The EKF used in Project 3 differs from the KF used in Project 2 in its ability to handle non-linear systems. By linearising the system around the current estimate and computing the Jacobian matrices for the process and measurement functions, the EKF could approximate non-linear dynamics and measurements, which the standard KF cannot do. However, it is similar in the way that it predicts and updates the cycle.

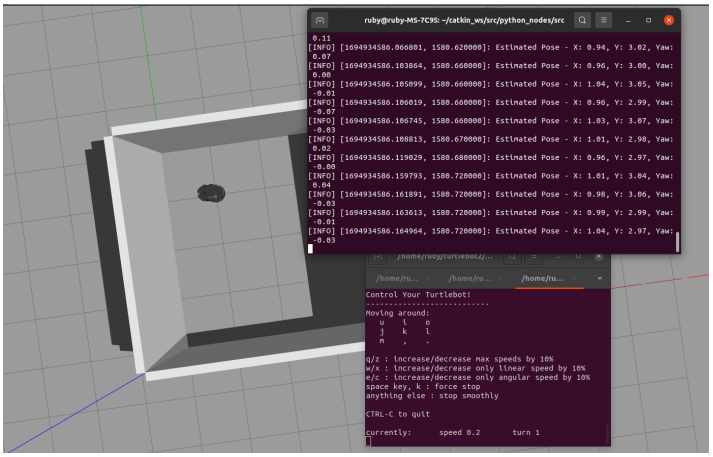

Fig. 4. Pose estimation after Kalman filter in Project 2

Evaluation

When I tested a single run, the landmarks estimate and EKF trajectory path was pretty accurate compared to real positions and path. Due to the inability to assess the performance and run multiple trials, I was unable to include graphs in this report. However, if I could make it work, I would plot two graphs. The first would illustrate the positions of both the real and estimated landmarks. The second graph would compare the real path with the EKF trajectory path. To obtain accurate data for the real path and trajectory path, my strategy would involve navigating the system to precise points, such as (2,2) and (3,3), and then comparing the recorded coordinates.
The graph below is one I replotted following feedback from Project 2. Similar graphs would have been plotted for Project 3 if I could record multiple points. Furthermore, I plan to conduct multiple tests in different environmental setups.

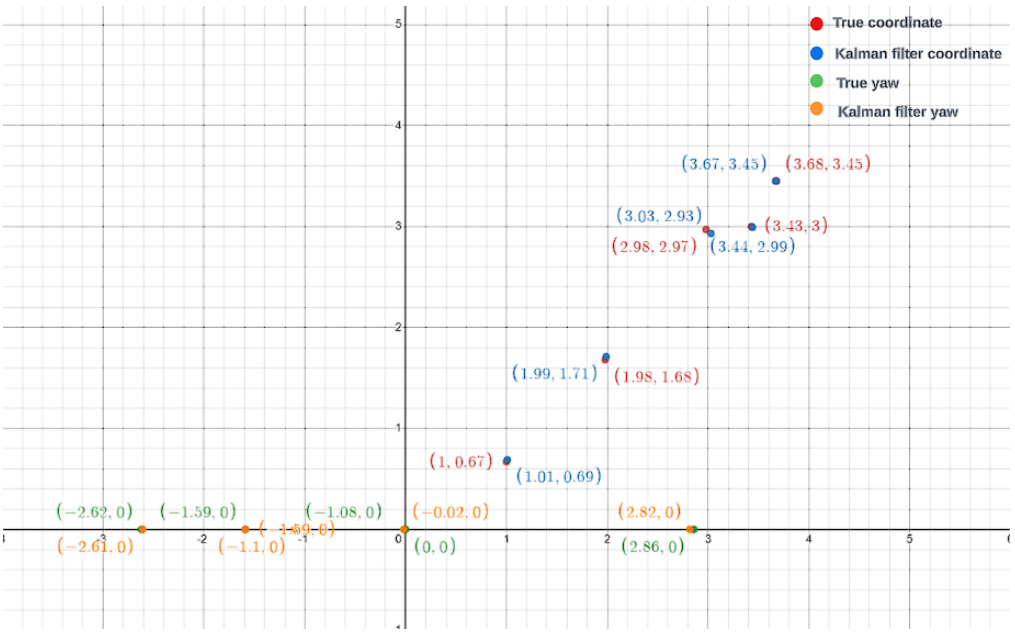

Fig. 5. Position plot for project 2 (which will be similar to project 3)

More or less landmarks

As I could not test introducing a new landmark in the environment, I assume that it will significantly impact the system's trajectory estimation. Once the new landmark is detected, the EKF's state vector, which includes the positions of all landmarks, should be updated to incorporate its coordinates. Initially, there will be uncertainty about the new landmark's position, leading to the EKF to rely more on depth sensor measurements for its localisation. After repeated observations, this uncertainty will disappear, and the trajectory estimation will stabilise. Furthermore, I assume adding more landmarks will reduce the overall system's uncertainty eventually, providing more reference points. However, this will be only possible when the new landmark is accurately detected and updated. I plan to test the introduction of new landmarks and conduct evaluations once the computer issue is resolved. On the other hand, I assume introducing fewer landmarks into an environment could potentially increase the overall uncertainty in the system's trajectory estimation. With fewer reference points, the Extended Kalman Filter (EKF) may rely more heavily on the robot's odometry data, which can accumulate errors over time. This could lead to less accurate localisation and pose estimation.

Conclusion

In conclusion, this project explores an approach to implement the EKF in robotic localisation and mapping, integrating depth sensor data and odometry. Even though I was unable to properly evaluate the system's efficiency due to the computer issue, I am planning to retest it once it is fixed and plot graphs for a proper comparison.

References

[1] Axel Barrau, Silvere Bonnabel, “An EKF-SLAM algorithm with consistency properties”, 2015, available: https://arxiv.org/abs/1510.06263
[2] Mengshen Yang, Xu Sun, Fuhua Jia, Adam Rushworth, Xin Dong, Sheng Zhang, Zaojun Fang, Guilin Yang, Bingjian Liu, “Sensors and Sensor Fusion Methodologies for Indoor Odometry: A Review”, 2019, DOI: https://doi.org/10.3390/polym1410201