Introduction

Project 2 aims to utilise the robot's depth camera and internal odometry to determine where it is within a given map of the environment, which is known as robot localisation. This demonstrates how a robot practically estimates its position in the world through its sensor and odometry. The localisation of robots is a fundamental component in achieving them to be completely autonomous [1]. Therefore, it is important to accurately estimate the position of the robot. This report will explore estimating the position of the robot accurately using the internal odometry and depth sensor.

Environment Setting

This project is done using the gazebo virtual world simulation, comparing where the robot thinks it is and its actual position. The robot's depth camera and its internal odometry are used in a known environment. The environment setup here is a 4x4 rectangle where its bottom left corner is at (0,0). Therefore, the identified landmarks are at coordinates (0,0), (4,0), (0,4), and (4,4).

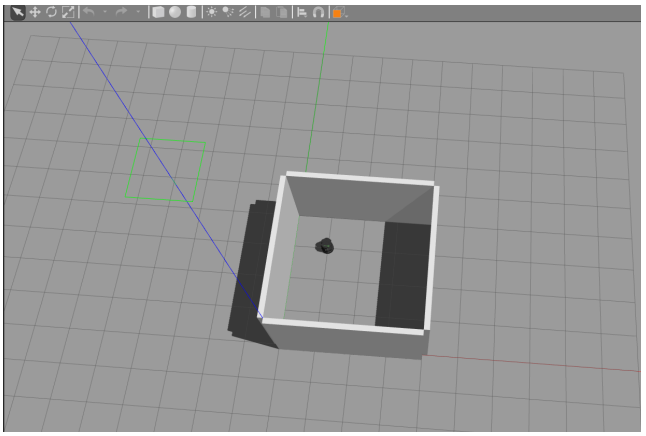


Fig. 1. 4x4 square environment

Project implementation

The robot will detect the distance to the walls, and get the position estimate using the information obtained. The depth camera is used by subscribing to the topic "/camera/depth/image_raw". The node subscribes to the depth image from the camera. This image represents the distances between the camera and the objects in its field of view, with each pixel's value corresponding to a distance.

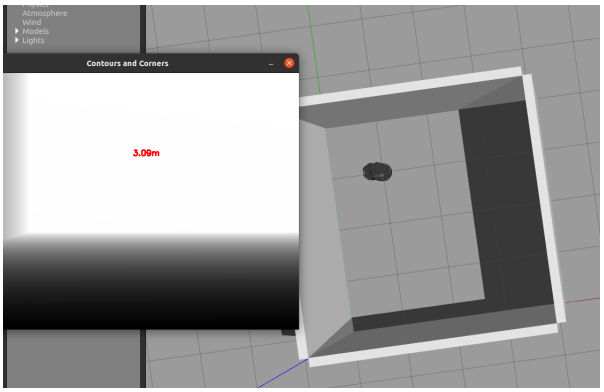


Fig. 2. Distance measurement in x direction

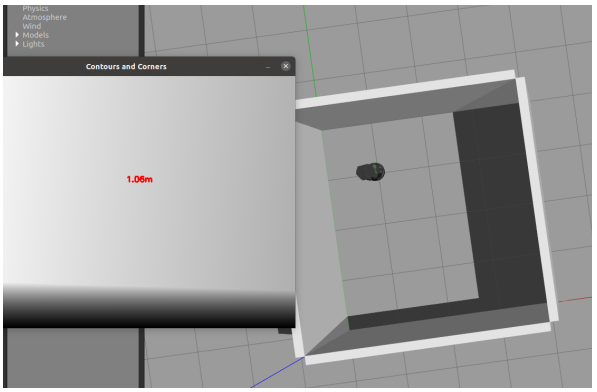


Fig. 3. Distance measurement in y direction

The robot position estimation could be done using the distance measurement from the walls. The distance measurement is used since the size of the environment (wall lengths) is known.
Since the robot moves not only horizontally and vertically but also at different angles, the trigonometry calculation is implemented here. For example, if the robot is facing the top wall, the distance to the vertical walls (parallel to the y-axis) x_distance = distance to the wall measured * cos (yaw) and for the distance to the horizontal walls (parallel to the x-axis) y=distance = distance to the wall measured * sin(yaw). These measured

distances can be subtracted from the wall (4m - measured distance) or used themselves to get the position of the robot.
The corners are also detected with the depth camera. To get the yaw estimation, a vector from the robot to the detected corner and another vector from the robot to the known corners ((0,0), (0,4), (4,0), (4,4)) are used. The difference between the angles is calculated and averaged to get the yaw estimation.

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 (1,3) with yaw=0. Then, this project tries to get the best estimation of the robot's position using odometry and depth sensor fusion. It predicts the robot's new position based on its previous state.

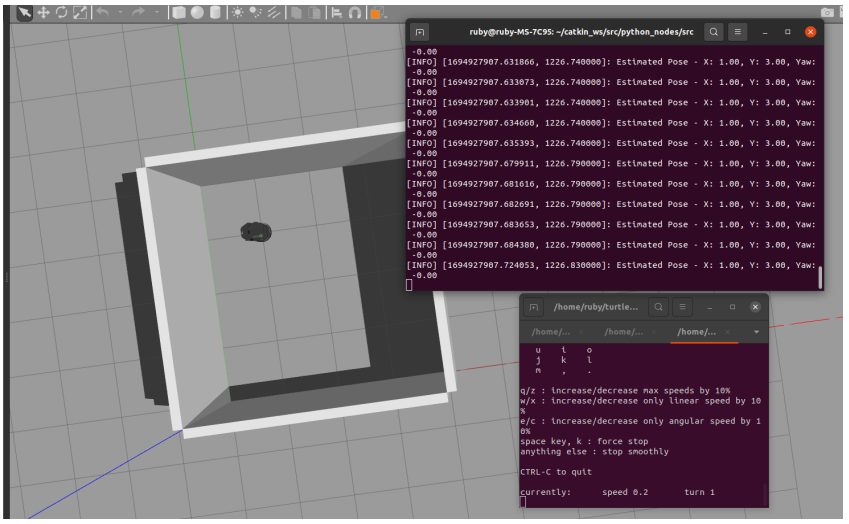


Fig. 4. Odometry without noise

In the simulation world, the odometry data is accurate without any noise, so I manually introduced the noise to the odometry data. To add noise to odometry data, Gaussian noise is added. Now the odometry data is noisy and this could be adjusted by using the sensor fusion.

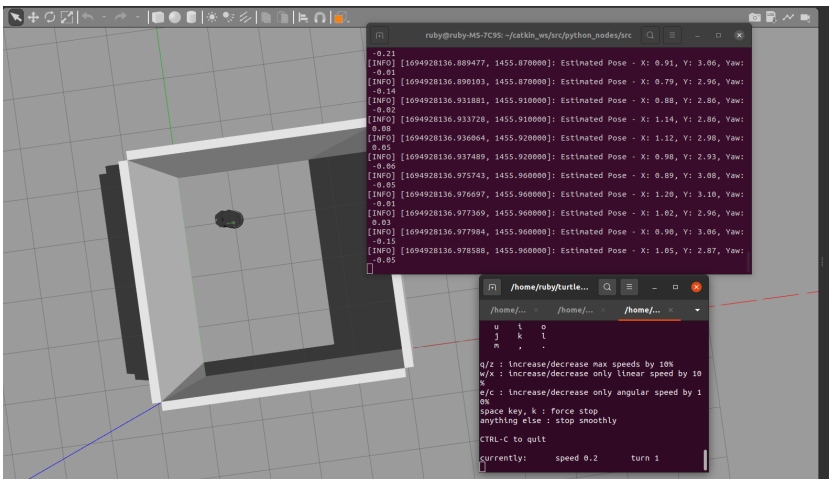


Fig 5. Odometry with noise

Kalman filter is used to combine the noisy odometry data with the measurements from the depth sensor to produce a more accurate and stable estimate of the robot's position. Furthermore, the use of pnp solver for this project was impossible as it has a lack of contours. As the Kalman filter works with Gaussian noise and a non-complex environment [1], I believe the use of the Kalman filter is appropriate. The odometry data is used for pose prediction and depth camera data is used for correction. The Kalman gain is set to determine how much weight to give to the odometry data prediction and the depth camera data. In this project 0.8 is used to give more

weight to the depth camera readings as odometry data is noisy. Then the pose estimation prediction is updated using the depth camera data and the Kalman gain. This process is repeated in a loop continuously. The Kalman filter results in noise reduction which is introduced in odometry data and improved accuracy compared to data from individual sensors.

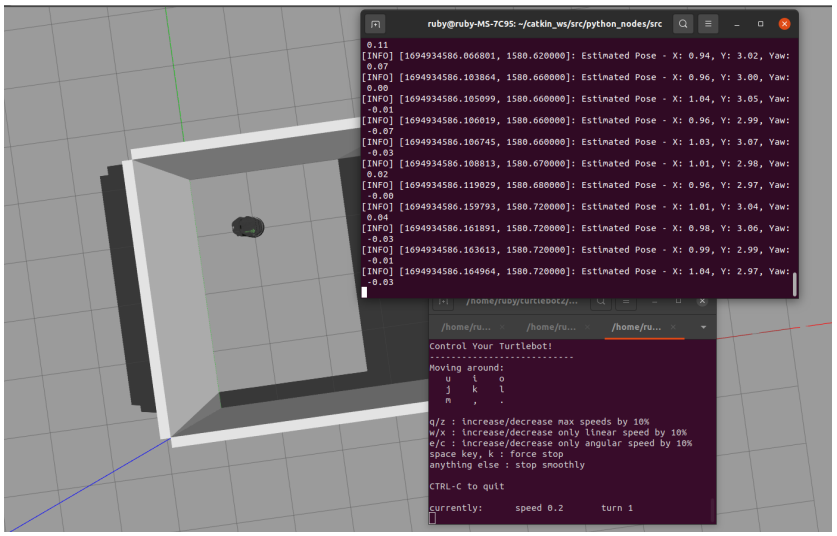


Fig 6. Pose estimation after Kalman filter

Results

The sample 10 results (at (1,3) and yaw=0) before applying and after applying the Kalman filter are shown in Fig.7.

Before applying the Kalman filter:

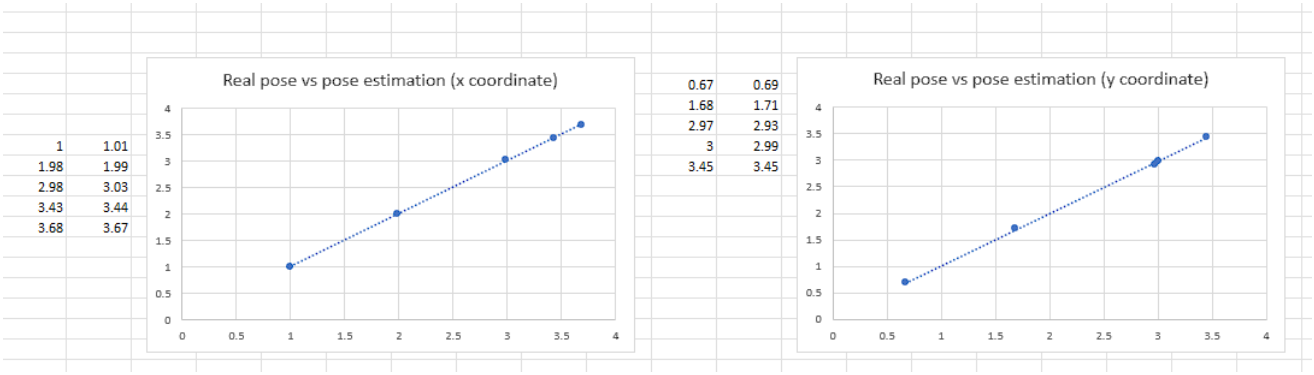
The pose estimation accuracy is measured by determining if the pose is accurate within the 0.1 range. 20 values were used to get the accuracy and it was 70%.

After applying the Kalman filter:

Same as before applying the Kalman filter, the 0.1 range is used with 20 values. The accuracy was 90%. This shows that the implementation of the Kalman filter has reduced noise and improved the accuracy of the pose estimation.

Before Kalman filter	After Kalman filter
(2.09, 3.04), Yaw: 0.07	(2.01, 3.03), Yaw: -0.08
(2.07, 3.10), Yaw: -0.02	(1.92, 3.07), Yaw: 0.08
(2.01, 2.94), Yaw: 0.07	(1.99, 2.95), Yaw: -0.06
(1.90, 3.08), Yaw: 0.11	(2.02, 2.97), Yaw: -0.08
(1.85, 2.94), Yaw: -0.10	(2.03, 2.03), Yaw: 0.01
(1.88, 2.92), Yaw: 0.05	(1.89, 2.99), Yaw: 0.01
(1.92, 2.97), Yaw: 0.06	(2.05, 2.94), Yaw: -0.08
(1.95, 3.18), Yaw: 0.09	(1.94, 2.94), Yaw: -0.01
(1.84, 2.98), Yaw: -0.10	(1.98, 2.98), Yaw: -0.04
(2.00, 2.99), Yaw: 0.10	(2.01, 3.01), Yaw: -0.02

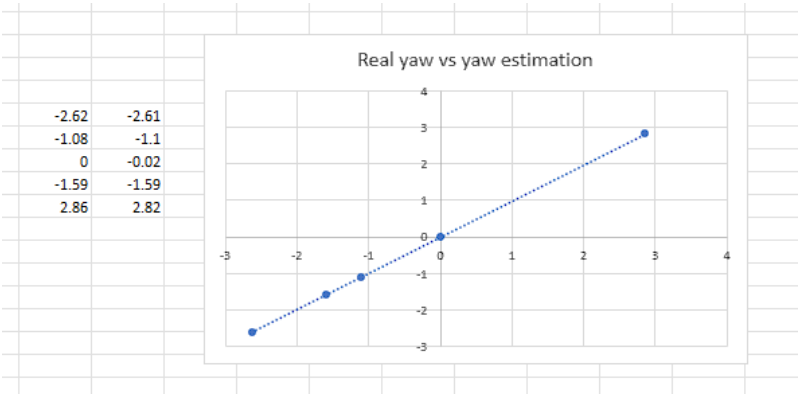
Fig.7. localisation values sample 10 at (2,2) yaw=0



1	1.01
1.98	1.99
2.98	3.03
3.43	3.44
3.68	3.67

0.67	0.69
1.68	1.71
2.97	2.93
3	2.99
3.45	3.45

Fig.8. real pose vs pose estimation x and y



-2.62	-2.61
-1.08	-1.1
0	-0.02
-1.59	-1.59
2.86	2.82

Fig.9. real yaw vs yaw estimation

Inaccuracy in map/ obstacle

When the obstacle is placed in the environment or when the map is changed to a different size, it results in wrong localisation and I believe this is because of the wrong distance measurement from the depth sensor reading due to the static environment, knowing exact positions of existing landmarks.

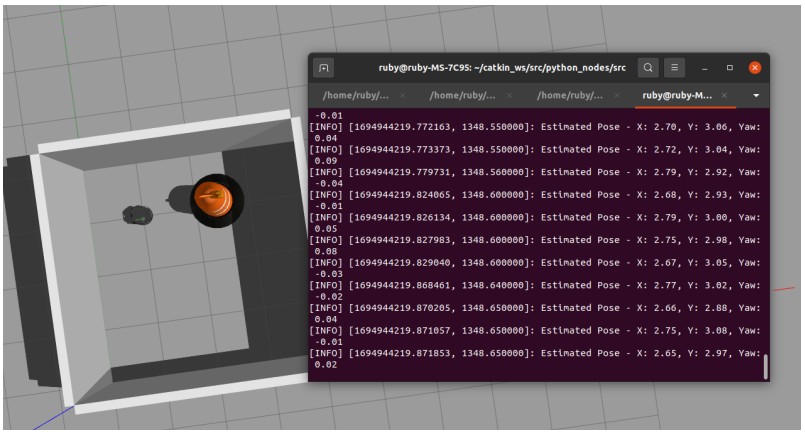


Fig.10 Obstacle placement

Conclusion

In conclusion, this project explores an approach to get the best pose estimate using odometry and depth camera. Measuring the distance to the walls using the depth camera and correcting the odometry data using Kalman filter resulted in noise reduction and improved accuracy.

References

[1] Manasee*, P. Sudheesh and M. Jayakumar, “Indoor Robot Localisation using Kalman Filter, Indian Journal of Science and Technology”, 2016, DOI: 10.17485/ijst/2016/v9i38/101945

[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/polym14102019