Aalto University, School of Electrical Engineering
ELEC-E8740 - Basics of sensor fusion

Final Report

Basics of Sensor Fusion Project
Group 21

Date: 10.01.2021

Zhixin Cui 882781
Jingsheng Chen 914837
Yun Hua 914730

Table of Contents

1. Abstract

The aim of this project is to develop an algorithm for tracking an autonomous robot by using a set of sensors, and the main work is doing processing and sensor fusion of the data. The robot is equipped with an inertial measurement unit (IMU), an infra-red detector, a motor controller, and a camera module. The IMU can record the acceleration and angular velocity information of the robot, and the camera can be positioned by scanning the QR code. By using the filtering algorithm to fuse the IMU and camera data, we can complete the dynamic tracking of the robot's position.
Keyword: Autonomous robot, Sensor fusion, IMU, QR code, camera

2. Brief introduction

The main objective of this project is to develop a robot tracking system given a DiddyBorg rover-type robot with some sensors. The robot is equipped with an inertial measurement unit (IMU), which is a combination of accelerometer, gyroscope, and magnetometer. In addition to the IMU, the robot is also equipped with an infra-red detector, a motor controller, and a camera module.

The IMU will measure the acceleration as well as the angular rate of the robot from three orthogonal body axis. The accelerometer measurements and the gyroscope measurements (angular velocity) from IMU are combined to obtain the acceleration in the inertial frame. The velocity and the position of the robot are then readily obtained by integration of the acceleration, and it can be also given by the motor directly. The camera system will detect several predefined rectangles that contain unique QR codes with known positions.

The first part of this project is developing a sensor model with estimated parameters. In the second part of this project, the derivated sensor model will be combined with a sequential estimation algorithm in order to estimate the position and attitude of the robot. In the third part, we will develop a tracking algorithm and fuse the IMU and camera data to obtain our final robot tracking system.

3. Part I: Sensor modeling

3.1 Task One

Task 1a. One row in the dataset means the detected data in one timestep. The dataset contains 12 columns. The first column is system time. The second to the fourth column is the accelerator g force data. The fifth to the sixth column is the accelerator angles data. The seventh to the ninth columns is the gyroscope angular velocity data. And the tenth to the twelfth column is the magnetometer data.

	0	1	2	3	4	5	6	7	8	9	10	11
0	1604404754.93610	0.00464	-0.00647	1.02419	-0.09731	-0.03662	-0.06463	0.04380	0.06722	-0.36787	0.94563	0.58930
1	1604404754.99924	0.00451	-0.00586	1.02407	-0.02842	-0.02941	0.21537	-0.37620	-0.17778	-0.36554	0.94695	0.58652
2	1604404755.06101	0.00342	-0.00586	1.02370	-0.06128	0.03291	0.18037	-0.09620	-0.07278	-0.36554	0.94695	0.58652
3	1604404755.12264	0.00610	-0.00439	1.02285	0.01960	-0.09076	0.49537	-0.97120	-0.24778	-0.37153	0.94402	0.59106
4	1604404755.18425	0.00695	-0.00708	1.02700	-0.18359	-0.03504	-0.16963	1.05880	0.13722	-0.37153	0.94402	0.59106
5	1604404755.24585	0.00317	-0.00281	1.02090	0.27240	0.04711	0.11037	0.18380	-0.07278	-0.36700	0.94636	0.59047
6	1604404755.30747	0.00476	-0.00500	1.02370	0.02612	-0.04327	-0.09963	0.07880	0.13722	-0.37153	0.94607	0.59164
7	1604404755.36910	0.00647	-0.00854	1.02358	-0.21578	-0.14089	-0.13463	0.32380	0.06722	-0.37153	0.94607	0.59164
8	1604404755.43072	0.00610	-0.00488	1.02297	-0.00771	0.00494	0.00537	0.14880	0.06722	-0.36685	0.94212	0.59383
9	1604404755.49233	0.00634	-0.00525	1.02102	-0.16569	0.09352	0.07537	-0.16620	-0.03778	-0.36685	0.94212	0.59383
10	1604404755.55394	0.00561	-0.00305	1.02358	0.05301	-0.24277	-0.20463	0.11380	-0.00278	-0.36802	0.93920	0.58462
11	1604404755.61555	0.00488	-0.00537	1.02431	-0.03761	-0.05206	0.14537	-0.23620	-0.10778	-0.36977	0.94388	0.59544
12	1604404755.67781	0.00769	-0.00634	1.02370	-0.08936	-0.20647	-0.51963	0.84880	0.27722	-0.36977	0.94388	0.59544
13	1604404755.74088	0.00573	-0.00598	1.02419	-0.35461	-0.09674	0.11037	-0.16620	-0.00278	-0.36787	0.94358	0.59120
14	1604404755.80379	0.00537	-0.00659	1.02334	-0.00119	-0.07742	0.00537	-0.16620	0.03222	-0.36787	0.94358	0.59120

Figure 1. One sight of the data

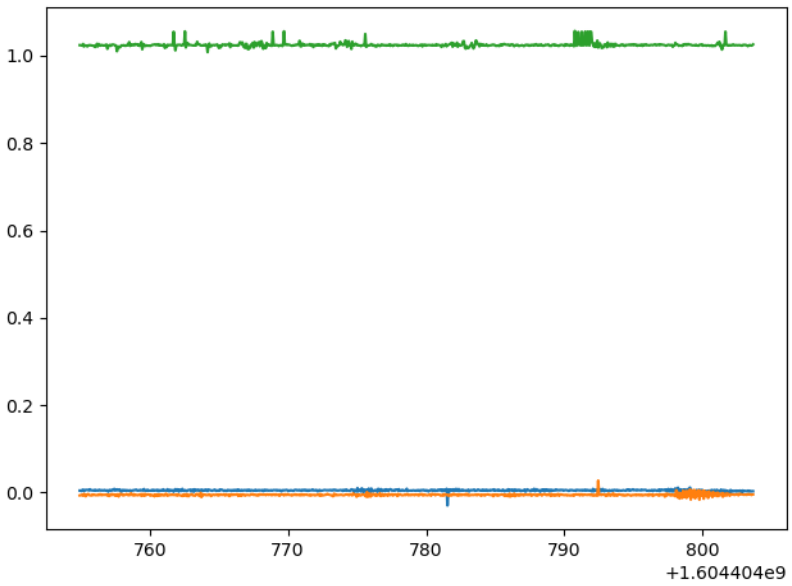


Figure 2. Accelerator g force data

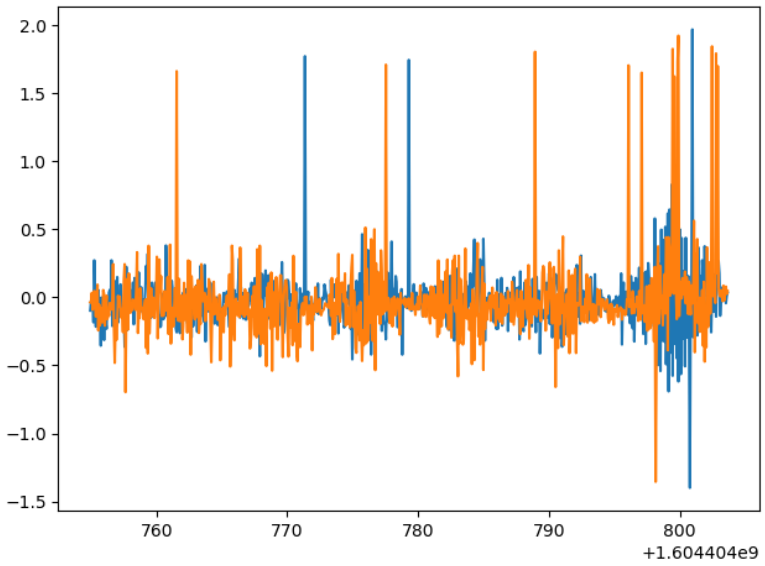


Figure 3. Accelerator angles data

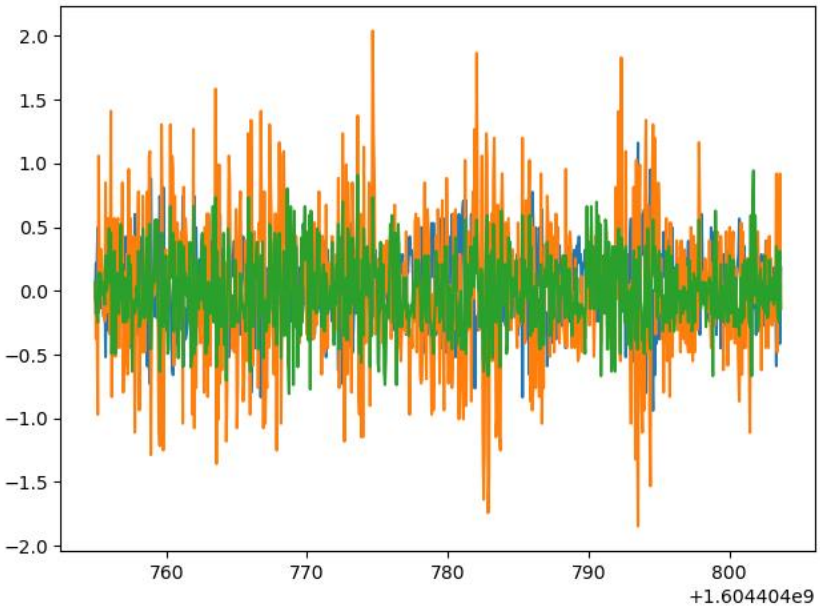


Figure 4. Gyroscope angular velocity data

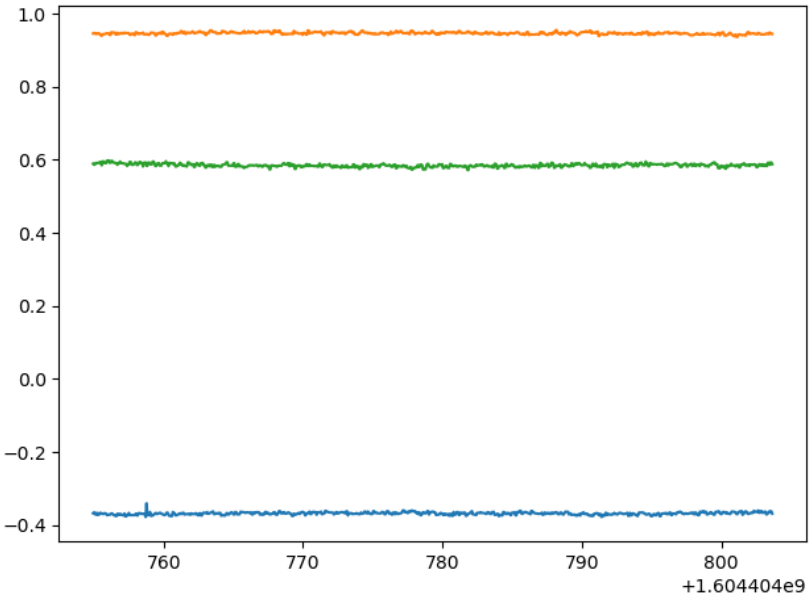


Figure 5. Magnetometer data

The data shown above indicates that the sample time is about 0.06 seconds. All sensors' raw data is relatively small, and it is possible to extract the columns of the data to do further data processing.

Task 1b. Assume gyroscope data were read in the order of x-axis, y-axis, z-axis, then the bias of x-axis is 0.008983018018018006. The bias of y-axis is 0.016876891891891902, and the bias of z-axis is -0.0012995945945945963.

Task 1c. The variance matrix of measurement noise is showed as below:

	0	1	2
0	0.08498	0.00000	0.00000
1	0.00000	0.32679	0.00000
2	0.00000	0.00000	0.09397

Figure 6. variance matrix of the IMU measurement noise

3.2 Task Two

Task 2a. Plots of data are shown below:

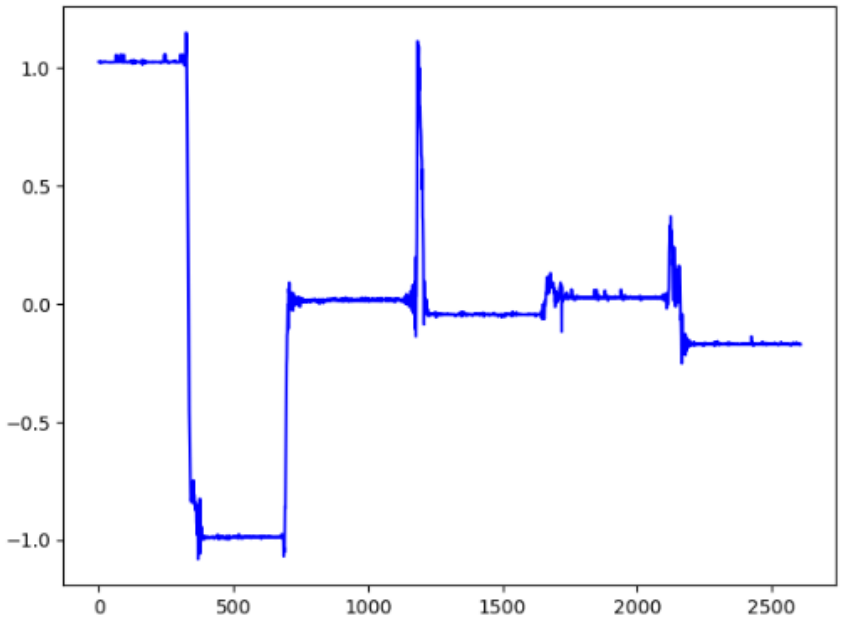


Figure 7. Accelerator of g force in z-axis

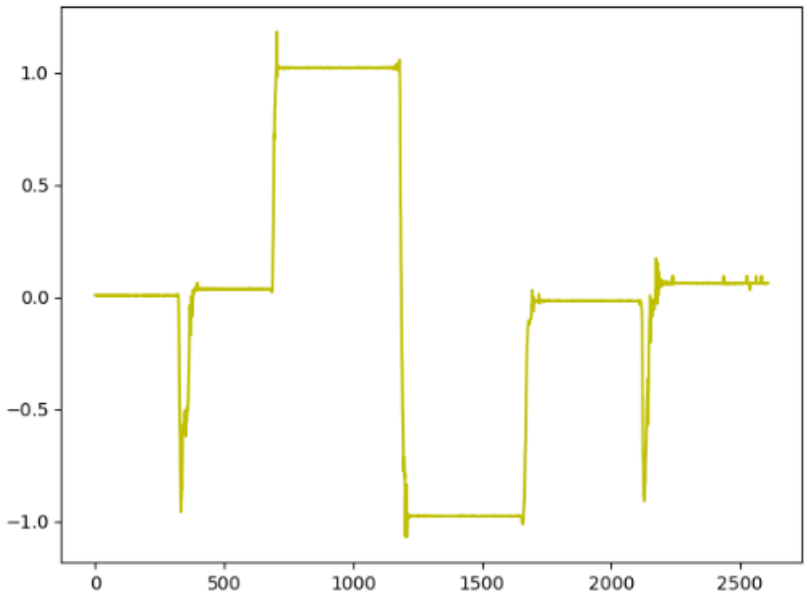


Figure 8. Accelerator of g force in x-axis

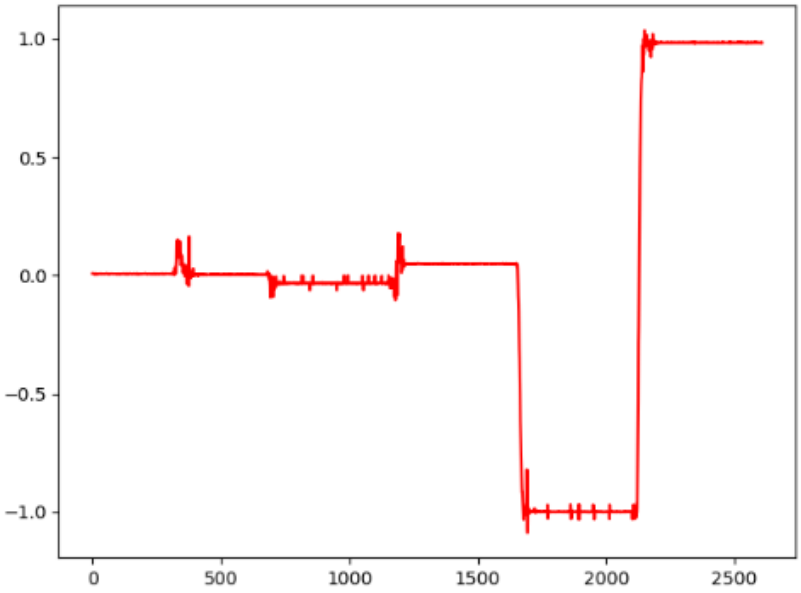


Figure 9. Accelerator of g force in y-axis

Through these plots of data, three facts are observed.
First is that robot were rotated 5 times and the data gives 6 measurement of gravity in 6 different directions.
Second is that there is bias in the measurement result and it may be caused by XYZ axis of the robot is not exactly same with the IMU's.
Third is that, according to the order provided by Readme.txt, +y direction appear earlier than -y direction, but the IMU reading is minus at first, this mean that robot's y-axis is reverse to IMU's y axis. (But here comes a question, as showed in the figure, the robot's coordinate is a right-hand coordinate, which means that coordinate of IMU is not right hand coordinate.)

The gains and biases for each body axis are showed below:

```
Kx = {float64} 0.10222257295746508
Ky = {float64} -0.10134584250611851
Kz = {float64} 0.10293077655709967
```

Figure 10. Gains for x, y, z-axis

```
Bx = {float64} 0.022346460356587772
By = {float64} -0.007794543173043789
Bz = {float64} 0.01971690527156622
```

Figure 11. Bias for x, y, z-axis

The reason why Ky and By is minus has been argued in the third fact.

Task 2b. The difference of the true value and the estimated value are given in figure 12. Note that the result is in the same order as Readme.txt showed.

	0	1	2
0	0.04449	-0.05962	9.94479
1	0.31063	-0.03193	-9.61797
2	9.98016	0.33282	0.11659
3	-9.57253	-0.47854	-0.44404
4	-0.19659	9.86691	0.23511
5	0.58155	-9.69843	-1.67790

Figure 12. Difference of the true and estimated value

It is not totally the same as what we expect. There is still bias in the data. It could be better with further calibration.

3.3 Task Three

Task 3a. The measurement of distance and pixels in the terminal can be visualized in the figure below.

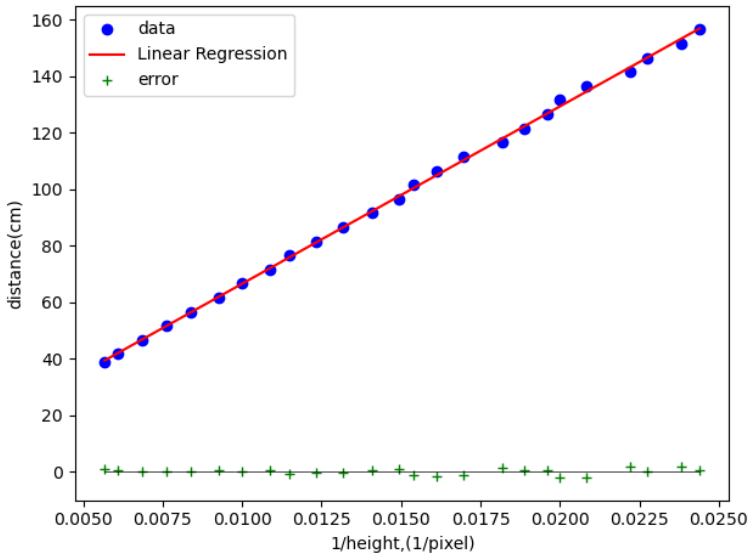


Figure 13. Relationship of distance and height of QR-code

The result of gradient and bias are shown below.

```
gradient is :  6285.201718390626
bias is :  3.6828589487804493
```

Figure 14. Gradient and bias

Task 3b. The focal length in pixel can be calculated from the following equation.

$$x_3 = \frac{h_0 * f}{h} + b$$

$$40cm + 1.6cm + 5cm = \frac{11.5cm * f}{146pixel} + 3.683$$

$$f = 544.85pixel$$

3.4 Task Four

The first column in the file is the total distance in cm. The second column is the time interval of each 40cm in second. Therefore, the speed can be calculated by 40cm divided by time interval. The result is shown in Table 1.

Distance(cm)	Time interval(s)	Speed(cm/s)
40	3.08	12.98701299
80	6.59	6.069802731
120	6.54	6.116207951
160	6.89	5.805515239
200	6.29	6.359300477
240	6.48	6.172839506
280	6.64	6.024096386

Table 1. Speed Calculation

4. Part II
4.1 Task five

Task 5a

In task 5 we need to localize the robot, i.e. to calculate three parameters: p^x, p^y, ψ, which are the position in x and y coordinate and orientation of the robot.
By measuring each QR code we can get two equations, which formed $g(x)$ in the measurement model:

$$h_i = \frac{h_0 f}{\sqrt{(s_i^x - p_i^x)^2 + (s_i^y - p_i^y)^2}}$$
$$C_{x,i} = f \tan\left(\arctan((s_i^y - p_i^y)/(s_i^x - p_i^x)) - \psi\right)$$

Therefore, we need at least three equations to calculate three unknowns, which means the minimum number of different QR-codes is 2.

Task 5b

We have the positions of multiple QR codes measured by the camera and the global position of the QR codes. To calculate global coordinate of robot, we used the Gauss-Newton method to process multiple sets of data collected by the camera, and obtained a relatively accurate position through multiple iterations.

Algorithm 2 Gauss–Newton Algorithm

Require: Initial parameter guess $\hat{\mathbf{x}}^{(0)}$, data $\mathbf{y}$, function $\mathbf{g}(\mathbf{x})$, Jacobian $\mathbf{G_x}$
Ensure: Parameter estimate $\hat{\mathbf{x}}_{\text{WLS}}$
1: Set $i \leftarrow 0$
2: **repeat**
3: Calculate the update direction

$$\Delta\mathbf{x}^{(i+1)} = (\mathbf{G}_\mathbf{x}^\mathsf{T}(\hat{\mathbf{x}}^{(i)})\mathbf{R}^{-1}\mathbf{G}_\mathbf{x}(\hat{\mathbf{x}}^{(i)}))^{-1}\mathbf{G}_\mathbf{x}^\mathsf{T}(\hat{\mathbf{x}}^{(i)})\mathbf{R}^{-1}(\mathbf{y} - \mathbf{g}(\hat{\mathbf{x}}^{(i)}))$$

4: Calculate

$$\hat{\mathbf{x}}^{(i+1)} = \hat{\mathbf{x}}^{(i)} + \Delta\mathbf{x}^{(i+1)}$$

5: Set $i \leftarrow i + 1$
6: **until** Converged
7: Set $\hat{\mathbf{x}}_{\text{WLS}} = \hat{\mathbf{x}}^{(i)}$

Figure 15. Gauss–Newton Algorithm

The Jacobin matrix:

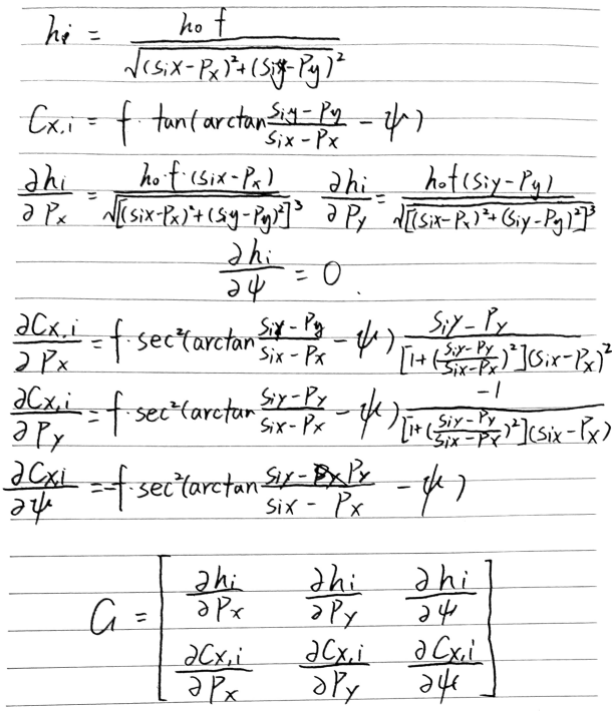


Figure 16. Jacobin matrix of g(x)

The values of p^x, p^y, ψ, gradually converge to the smallest value of the loss function through the Gauss Newton algorithm:

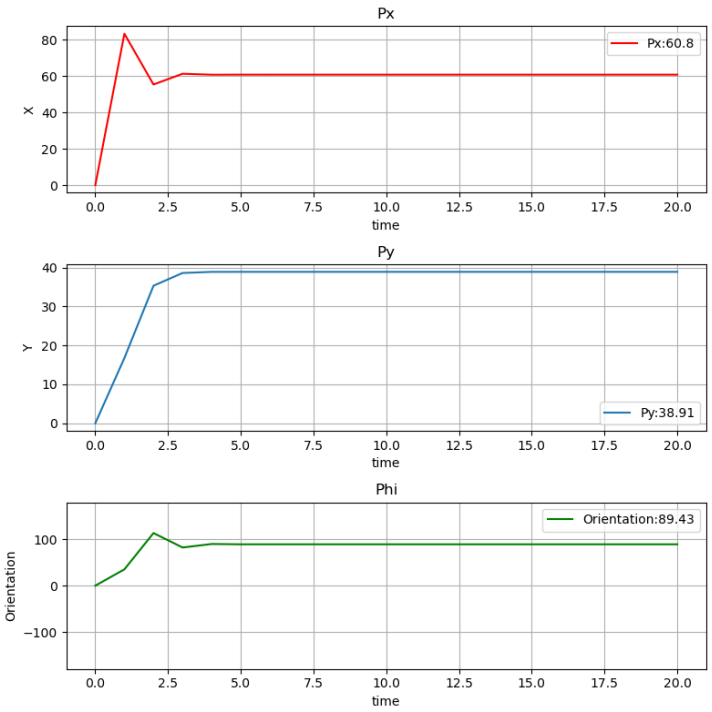


Figure 17. converge result of $\widehat{p^x}$, $\widehat{p^y}$, $\hat{\psi}$

And final result:

$$\widehat{p^x}=60.8$$

$$\widehat{p^y} = 38.91$$
$$\hat{\psi} = 89.43$$

Which is very close to the real global position of robot:

$$p^x = 60$$
$$p^y = 39$$
$$\psi = 90$$

And the positions of robot are as follow:

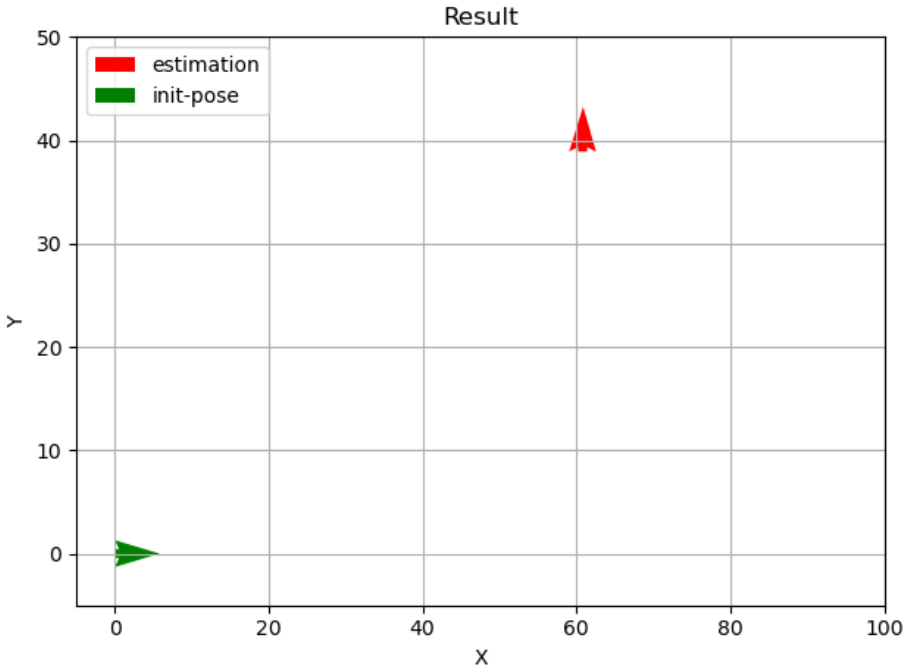


Figure 18. result position of robot

5.Part III
5.1 Task six

Task 6a

In Task 6, we choose Quasi-Constant Turn Model as our dynamic model to model the motion of the robot:

$$\dot{p}^x(t) = v(t)\cos(\varphi(t))$$
$$\dot{p}^y(t) = v(t)\sin(\varphi(t))$$
$$\dot{\varphi}(t) = \omega_{\text{gyro}}(t)$$

The state is now:

$$\mathbf{x}(t) = \begin{bmatrix} p^x(t) & p^y(t) & \varphi(t) \end{bmatrix}^{\mathsf{T}}$$

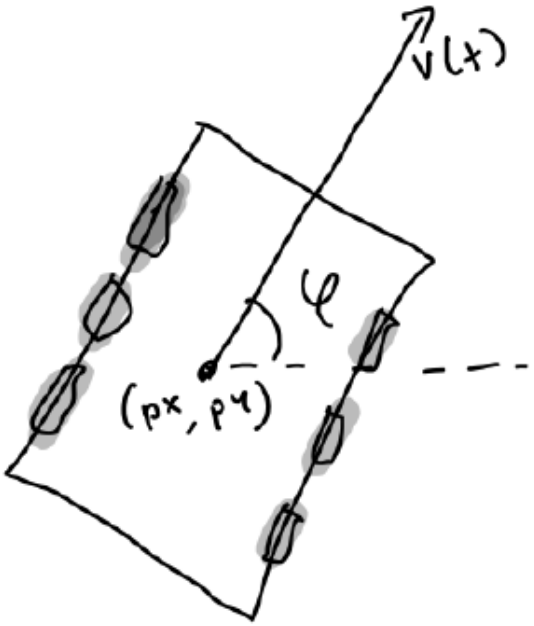
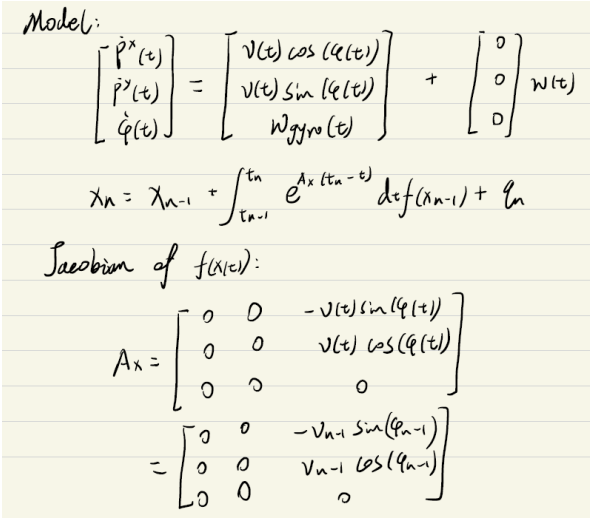
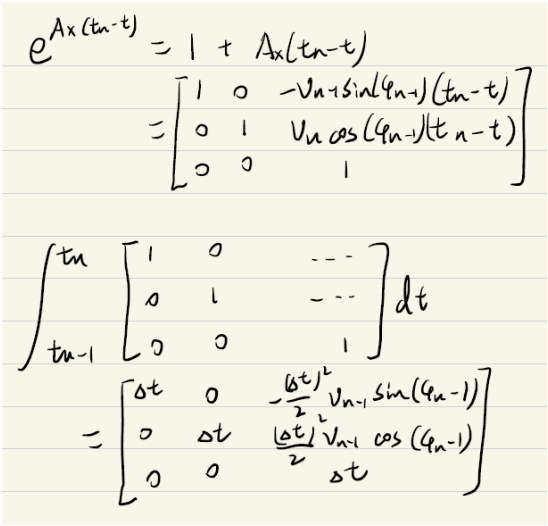


Figure 19. Quasi-Constant Turn Model

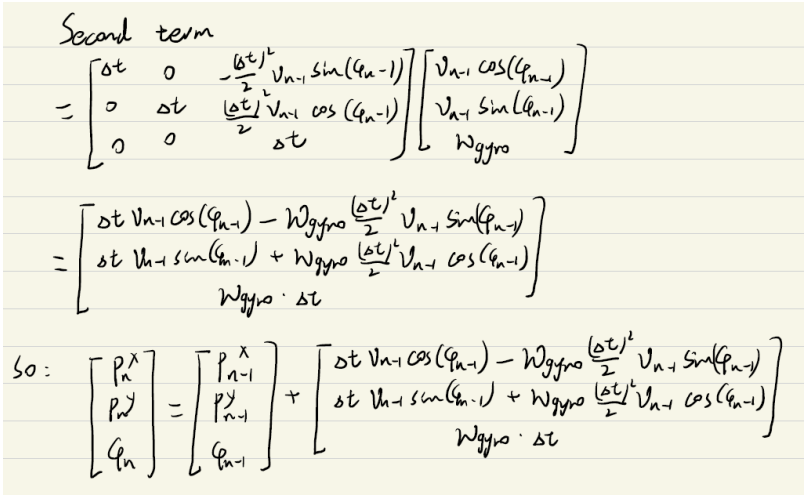
And then, as the equation is nonlinear, we doing the discretization by using Euler–Maruyama discretization:



(a)



(b)



(c)

Figure 19. Model discretization

Task 6b&c

Then we do the dead-reckoning in python based on speed measurements from the motor control and turn rate measurements from the gyroscope. The results are:

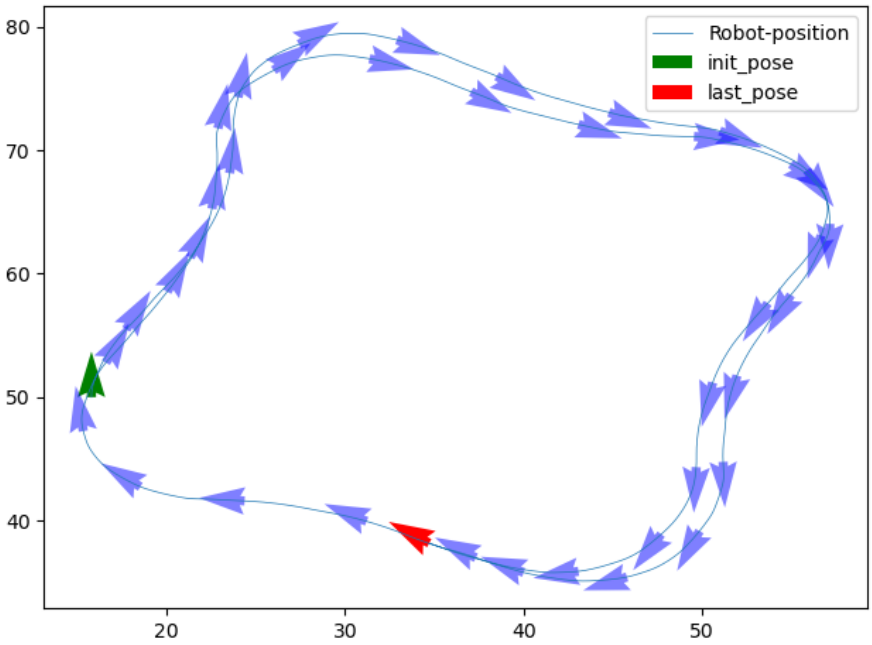


Figure 20. Result of dead-reckoning

Compare with the track in Figure6 of project guide, it looks similar.

Figure 21. Figure6 of project guide

5.2 Task seven

The IMU sensors alone may not be sufficient to perform localization accurately. Therefore, we need to incorporate camera measurements into the filtering algorithm. In this task, we choose EKF filter to do the fuse of the camera and IMU data.

Algorithm 1 Extended Kalman Filter

1: Initialize $\hat{\mathbf{x}}_{0|0} = \mathbf{m}_0$, $\mathbf{P}_{0|0} = \mathbf{P}_0$
2: **for** $n = 1, 2, \ldots$ **do**
3: Prediction (time update):

$$\hat{\mathbf{x}}_{n|n-1} = \mathbf{f}(\hat{\mathbf{x}}_{n-1|n-1})$$
$$\mathbf{P}_{n|n-1} = \mathbf{F}_x \mathbf{P}_{n-1|n-1} \mathbf{F}_x^{\mathsf{T}} + \mathbf{Q}_n$$

4: Measurement update:

$$\mathbf{K}_n = \mathbf{P}_{n|n-1} \mathbf{G}_x^{\mathsf{T}} (\mathbf{G}_x \mathbf{P}_{n|n-1} \mathbf{G}_x + \mathbf{R}_n)^{-1}$$
$$\hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n (\mathbf{y}_n - \mathbf{g}(\hat{\mathbf{x}}_{n|n-1}))$$
$$\mathbf{P}_{n|n} = \mathbf{P}_{n|n-1} - \mathbf{K}_n (\mathbf{G}_x \mathbf{P}_{n|n-1} \mathbf{G}_x + \mathbf{R}_n) \mathbf{K}_n^{\mathsf{T}}$$

5: **end for**

Figure 22. Extended Kalman Filter

We choose the point when all of the IMU, motor and camera's timestep star to collect data as start point, and use IMU and motor data as model prediction and use camera data as measurement update.
The results are as follow:

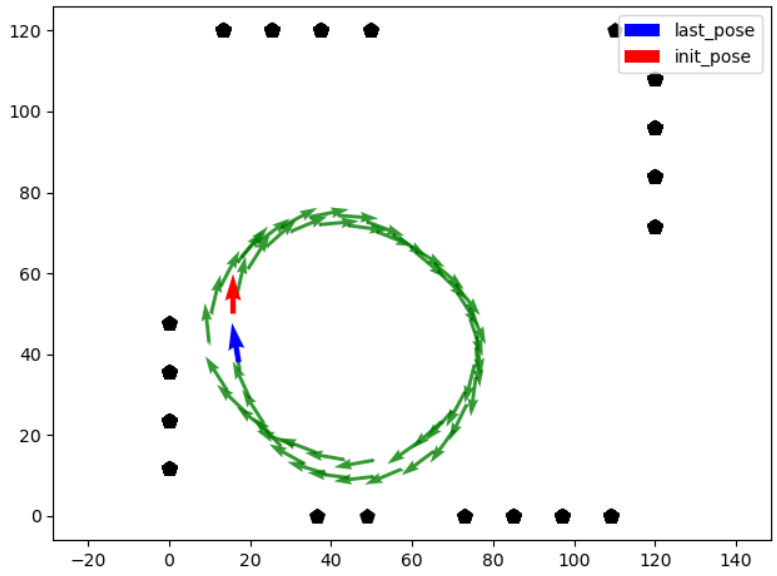


Figure 23. Result of EKF

6.Conclusion

In this project work, firstly, we are familiar with the calibration, correction of deviation and use of sensor modules such as IMU and camera. In the second part, we completed the static localization of the robot by using the camera to read the QR code. After that, we used separate IMU and motor method and IMU combined with camera to complete the dynamic tracking of the robot position.

This project work is a comprehensive application of what we learned in class.

In the process of localization of the static robot, we need to process many sets of redundant data and find the value with the smallest error. Therefore, we use the Gauss Newton method to converge to the point with the smallest loss function through multiple iterations. And for the dynamic tracking of the robot, we chose the Quasi-Constant Turn Model, which is a nonlinear and time-continuous model. But the data we sampled is dispersed, so we use the Euler–Maruyama discretization method to discretize the model. Finally, in addition to using the IMU and motor to predict the position of the robot, we also want to use the camera to measure and update the position of the robot. Therefore, we also need a filtering algorithm. Here, we use the EKF algorithm to fuse the two sets of data to make it more accurately track the position of the robot.