Project Guide

Tracking of an Autonomous Robot

Muhammad Emzir, Sakira Hassan

Department of Electrical Engineering and Automation
Aalto University, Finland

December 7, 2020

Contents

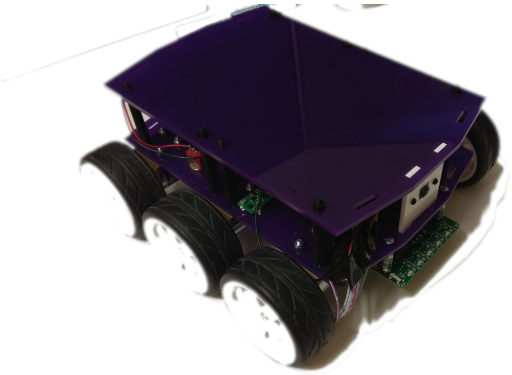

Figure 1: DiddyBorg robot with camera module and IR detector.

1 Introduction

The aim of this project is to develop an algorithm for tracking an autonomous robot by using a set of sensors. The robot, a DiddyBorg rover-type robot, is programmed to follow a black line inside a closed area surrounded by walls. 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. However, as time increasing, the error will be accumulated, and hence the deviation from the actual position grows. The camera system will detect several predefined rectangles which contain unique QR codes with known position attached in the wall. This measurements can be used to correct the position estimate obtained by twice integration.

The sensors are connected to the main computer which is a Raspberry Pi. The Raspberry Pi system is responsible to handle all sensor measurement preprocessing and logging. It can also be used to transfer the recorded measurements data to other means.

The project consists of two parts. In the first part, we develop and verify the sensor model for the IMU and camera system. This includes the following steps:

- Derivation of the sensor model,
- Estimation of the model parameters.

In the second part of the project, we combine the sensor model developed in the first part with a dynamic model and a sequential estimation algorithm to obtain our final robot tracking system.

The project contains both theoretical and practical parts. In the practical part, you will be given measurement data (record log files) from different sensors. **You can find the information of log file for each task separately in Appendix 6.3.5**. You

also need to write two reports: one intermediate report on the first part and a final report which includes the results from both parts as well as improvements of the first part. The final grade of the project work will be based on the final report. On the course homepage, you can find more information about the peer review, deadlines, and the grading criteria [1].

2 Part I: Sensor modeling

2.1 Sensor model

The robot is equipped by IMU, IR detection, camera systems, and motor control. The IR detection is used to detect a line pattern in the floor for tracking purposes, and it comes with built in calibration.

2.1.1 Inertial measurement unit

In this section, we will describe the sensor model of IMU. Inside an IMU, each individual sensor has the following linear equation; see for example [1, 2, 3].

$$\mathbf{y} = \mathbf{G}\mathbf{x} + \mathbf{b} + \mathbf{r}$$

The acceleration measurement in the body axis is can be written as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}^{-1} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}, \tag{1}$$

where k_i, b_i, r_i, are the gain, bias, and noise, respectively in $i = \{x, y, z\}$ axis.

As the case of the accelerometers, the gyroscope also experiences the earth angular velocity, which may be used as a reference for calibration. However, for the case of MEMS gyroscope, the earth angular velocity is so small that it is buried in the sensor noise.

From the IMU measurement record log, we can determine the variance matrix of the IMU measurement noise. This matrix will be used later for tracking purpose.

2.1.2 Static IMU experiment

We can measure the IMU reading of a static robot using the python script provided in Appendix 6.3.1. We can obtain the bias of the gyroscope by placing the robot in a static position for a period of time. Then, we average the gyroscope readings to obtain the bias.

Task 1a. From the measurement record log file visualize the data. Read the Appendix 6.3.1 in order to understand what each column represents. What do you observe? Summarize what you have understood and write it down in your report.

[1] Part of this guide is based on the previous year guide written by Dr. Roland Hostettler.

Task 1b. Determine the bias of the IMU sensors and write down the result in your report. At least gyroscope measurement should be reported.

Task 1c. Determine the variance of the IMU measurement noises and write down the result in your report. You can also report the covariance matrices of the IMU sensors.

2.1.3 IMU calibration (accelerometer calibration)

To calibrate the accelerometer, we can depend on the assumption that the earth gravitational is fixed on a static object and we use the following simple procedure for every robot body axis:

- Turn on the robot, and run the python script for IMU (see Appendix).
- Place the robot in a firm horizontal support.
- Use a timer to record the log reading at each body axis, For example, 30s or 60s.
- Record the acceleration reading for the robot in up position, a_u in that direction.
- Rotate the robot 180° in the selected axis, and record the acceleration reading for the robot in up position, a_d.
- Calculate the gain for the selected axis as

$$k_i = \frac{a_u - a_d}{2g}.$$

  Here, g is the gravity.
- Calculate the bias b_i for the selected axis By

$$b_i = \frac{a_u + a_d}{2}.$$

Task 2a. Plot the data from the measurement record log file. What did you observe? Is there any difference? Why it is so? Determine the gain k_i and bias b_i for each body axis $i = \{x, y, z\}$. Write down the results in your report.

Note that, the x-y plane of IMU sensor is in opposite direction from the x-y plane of the robot and camera sensor. That is, the accelerometer measurements a_x and a_y in the log file is actually $-a_x$ and $-a_y$, respectively.

In Task 2a, we obtain the gain k_i and bias b_i for each body axis $i = \{x, y, z\}$ for the accelerometer. If we plugin these values in Equation (1), then we have our calibrated model. We will verify this in the following task.

Task 2b. In the previous task, we observed gravitational acceleration in 6 different orientations at different timepoints. First, take a subset of data from the measurement log file by choosing acceleration in one of these orientations (say $+x$ *axis*) where you have positive/negative acceleration in one direction and almost no acceleration in other orientations. Now, try to solve the Equation (1) with the parameters you estimated in previous task. You already have the acceleration measurements $\boldsymbol{y} = [y_1, y_2, y_3]^T$ in $\{x, y, z\}$ axes, respectively from the data. You can assume that the true value of

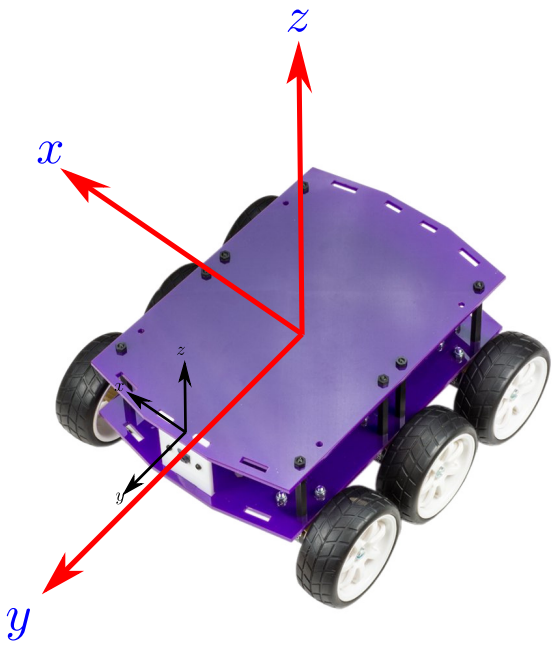

Figure 2: Illustration of the robot and camera coordinate systems given by the left hand axis rule. Notice the camera coordinate system is parallel to the robot coordinate system

$\boldsymbol{x} = [1, 0, 0]^T$ for example, if you choose *+x axis*. If you choose *-y axis*, then your true value will be $\boldsymbol{x} = [0, -1, 0]^T$ and so on. Determine the difference between the true value and the estimated value. What do you expect? Write your answer in your report.

Note that, the accelerations a_x, a_y, and a_z are given in gravitational unit. For example, if $a_x = 1$ in the log file, then it represents $9.8\, ms^2$ acceleration in x direction.

Optional. The gain factor k_i and the bias b_i are not enough here. We also need to consider the misalignment parameters as discussed in [1] (See Equation 3). Ideally, the sensor sensitivity axes should be orthogonal, but due to the impreciseness in the construction of the IMU this is seldom the case. Thus, the primary goal of a calibration is to estimate the misalignment, scaling and bias parameters, i.e.,

$$\theta = [b_x, b_y, b_x, k_x, k_y, k_z, \theta_{yz}, \theta_{zy}, \theta_{xz}], \tag{2}$$

and minimize the cost function provided in Equation (9) of Ref. [1]. We can obtain new dataset in 14 different orientation as described in the paper. Using this data, one can use linear regression methods to estimate the θ that minimizes the errors. This is an optional task. The data for this task can be provided on request.

2.1.4 Camera module calibration

In this section, we will use a simple method to determine the distance of the robot relative to a specific wall. Assuming that the geometry of the confined area and the QR-code

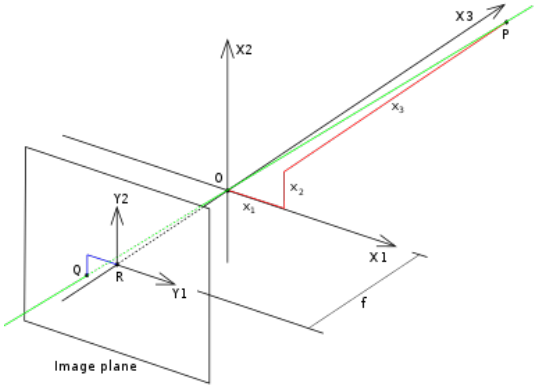

Figure 3: Illustration of a pinhole projection.

rectangles to be detected are fixed. Using additional measurement from the IMU, we can use the distance measurements we obtained from the camera to adjust the position estimate using the IMU only.

To determine the distance and attitude of the QR-codes to the camera, we can use the pinhole projection rule,which was first proposed by Brunelleschi at the early of the fifteenth century. It is a simplified mathematical model that describes the relationship between the coordinates of an object in a three dimensional space and its projection, assuming that the camera aperture is so tiny and there are no lenses used to focus the light; see Figure 3. Under this assumption, we have the following relation [4, Section 1.1.1]:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -\frac{f}{x_3} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3}$$

The image coordinate point (y_1, y_2) is given in pixel unit. At the sensor measurement logs, you will receive y_1 and y_2 in pixel units, as well as the unique number associated to each QR-code. We can convert these values into the position of the robot in the inertial frame. To do this, we need to estimate the focal length of the camera in pixel unit.

- Turn on the robot and run the python script for camera module (see Appendix).
- Wait until camera module ready.
- Prepare one QR-code only for the calibration.
- Place tape measure below the robot facing perpendicular to the wall with QR-code as close as possible until the QR-code is detected, and printed in the terminal. Write down the actual distance of the camera lens to the wall. You do not need to write the terminal QR code reading as they are stored in log file.
- Increase one or two cm, and hold for several seconds. Repeat this step until the QR-code cannot be read by the robot.
- Using the log file reading and Equation (3), we can determine the focal length f.

To get an insight of the focal length, plot one over the height against the recorded distance; you should get nearly linear relation; see Fig 4 . Then you can use the standard

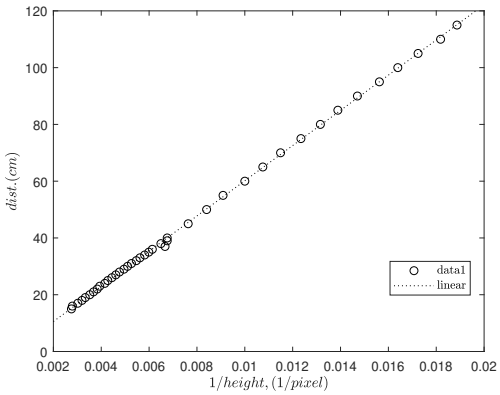

Figure 4: Relation between distance of QR-codes from the camera and detected height.

linear least-square regression to obtain the gradient and bias. Notice that the gradient k that you get is a multiplication of the QR-code length (cm) and the "actual" focal length in pixel.'

For each QR code detected, you can estimate the horizontal distance x_3 and direction ϕ from the QR-code center point to the robot using (3). Let y_1, y_2 are the center point of the QR-code, and h is the detected height of QR-code, all given in pixels. If the actual height of QR-code is given by h_0, then we can measure the robot distance x_3 and heading ϕ using

$$x_3 = \frac{h_0 f}{h} + b, \tag{4}$$

where b is a bias and

$$\phi = \arctan(\frac{y_1}{f}). \tag{5}$$

Task 3a. There are two columns in the measurement record log file. The first represents the measured distance (in cm) and the second column represents the height (in pixel) measured from the terminal. You need to plot the data as described above and determine the gradient and bias. Write down the results in your report. Note that, you also need to consider the distance of the camera from the surface of the robot which is provided in *readme.txt* file with the data.

Task 3b. Determine the focal length in pixel from the Equation (4), given that the height h_0 of the QR-code is 11.5 cm.

2.1.5 Motor control

We can determine the speed of the robot. For this, we will use the script in Appendix 6.3.3. First, take a measuring tape to measure the distance and a stopwatch or timer to record

each time lapse. Then, prepare the measuring tape in the floor in any direction, for example, 200cm. Now, place the robot at the beginning of the measuring tape. The robot will start moving forward if we run the script *MotorControl.py*. Now, we can record the time at every 40cm, for example, as the robot moves forward.

Task 4 There are two columns in the measurement record log file. The first represents the distance (in cm) measured from the measuring tape and the second column represents the measured time (in s). You need to determine the speed of the robot. Write down the results in your report. Note that, you need to determine the distance interval from the given log file.

3 Part II: Localization

Localization is the process of estimating the position of a robot with respect to its environment. It is a fundamental process in autonomous driving vehicles and mobile robots to localize themselves globally for further decision making. In this part, we are interested to estimate the robot position and attitude using a camera sensor.

Now that we have both IMU and camera ready, we can use both of them to localize the robot position. This will serve as a validation to the measurement model of the camera derived before. Notice that although the camera axis is parallel to robot axis, the center position of the robot differs to the camera lens position. In the robot local axis, the camera lens position is fixed, but in global axis we will need to take account the direction of the robot in the global coordinate. The rotation matrix for a counterclockwise rotation of the 2×1 vector $\boldsymbol{a}$ by an angle α around the z-axis is given by

$$\boldsymbol{R}(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{6}$$

and thus, the rotated vector is

$$\boldsymbol{a}' = \boldsymbol{R}(\alpha)\boldsymbol{a}. \tag{7}$$

We will now place the robot with camera facing one of the walls. Make sure that the camera is able to detect as many as QR-codes possible (See Figure 5). We may need to use a smaller size QR-code if the camera does not able to detect more than one QR-code at the same time. You can check the Appendix 6.3.2 on how to adjust the python script for this case. We measure the exact global position of the robot for a reference. Also, we measure the QR-codes position in a global coordinate.

We assume that the global position of each QR-code is (s_i^x, s_i^y). If the global position and heading of the robot are (p_i^x, p_i^y, ψ), then for each QR-code, the measurement model for camera can be defined as

$$\begin{aligned} d_i &= \sqrt{(s_i^x - p_i^x)^2 + (s_i^y - p_i^y)^2} \\ \phi_i &= \arctan((s_i^y - p_i^y)/(s_i^x - p_i^x)) - \psi \end{aligned} \tag{8}$$

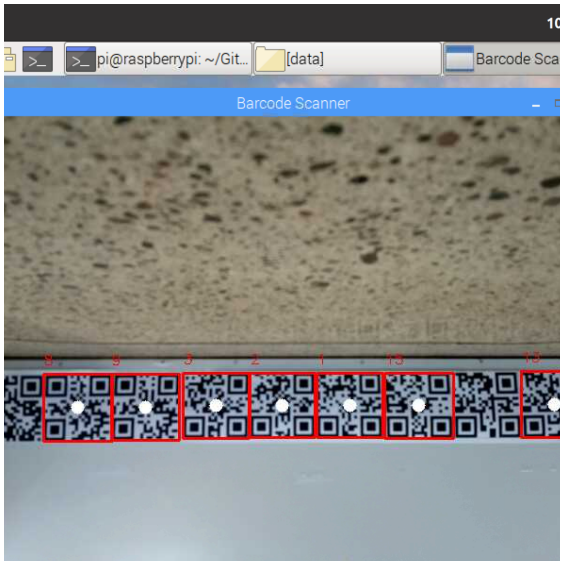

Figure 5: Detected QR-codes by CameraModule script.

If we know the exact distance d_i and the heading ϕ_i of each QR-code, then we can estimate p_i^x, p_i^y, ψ. In general, these are not known. However, since we know the focal length f of the camera, we can recover d_i and ϕ_i using Equation (4) and (5). The reading from the camera gives the height h_i and the center position $(C_{x,i}, C_{y,i})$ of each QR-code in the image plane and the true height h_0 of the QR-code is known i.e. 11.5cm, we can get

$$\begin{aligned} d_i &= \frac{h_0 f}{h_i} \\ \phi_i &= \arctan(\frac{C_{x,i}}{f}) \end{aligned} \tag{9}$$

Therefore, we have

$$\begin{aligned} h_i &= \frac{h_0 f}{\sqrt{(s_i^x - p_i^x)^2 + (s_i^y - p_i^y)^2}} \\ C_{x,i} &= f \tan\left(\arctan((s_i^y - p_i^y)/(s_i^x - p_i^x)) - \psi\right). \end{aligned} \tag{10}$$

We can write Equation (10) in the matrix notation

$$\mathbf{y} = \mathbf{g}(\boldsymbol{x}). \tag{11}$$

Task 5a. Describe the relation between the QR-codes global coordinates and the robot's static position in your report; see Figure 7 as a reference. What is the minimum number of different QR-codes that are needed to estimate the position and attitude of the robot in global coordinate?

Task 5b. Next, use nonlinear (weighted) least square technique to estimate the position and heading of the robot. To do this, you need to derive the Jacobian of measurement model that you choose and the measurement variance matrix for the camera module.

Hints

- Try to identify how many qr codes are detected by the camera at each time step from the record log file.
- The global position of the qr codes are given in *qr_code_position_in_global_coordinate.csv* file. Also, note that, the true position of the robot with respect to the frame/wall is given.
- You need to consider the focal length f that you found in **Task 3** to correct the distance in the measurement log file.
- The measurement model $\mathbf{g}(\boldsymbol{x})$ is defined in Equation (8) or Equation (10). Choose appropriate Jacobian of measurement model. For this, check the lecture 4, slide 18 as a starting point.
- Choose an appropriate nonlinear optimization method to estimate the position and heading of the robot with respect to the frame (i.e., the global position). See lecture 4 and 5.

4 Part III: Tracking

In **Part I** of the project, we derived the sensor model. In **Part II**, we developed an estimator for the unknown model parameters, and verified that the model is indeed useful for tracking purposes. In this **Part III** , we extend these results and develop the models and algorithm necessary for tracking the robot when moving. A prerequisite for this is that all the tasks in Part I and Part II have been completed successfully. In this part, the tasks are to:

- develop a dynamic model,
- adjust the sensor model,
- implement a filtering algorithm,
- validate the algorithm.

In **Part II**, we have tracked a static robot. In practice, we are interested in tracking a mobile robot. Thus, we need to know the position, heading, velocity, acceleration of the robot at time step t. There are two solutions to track a mobile robot. First is to start from a known position and track the robot's position locally provided that a dynamic model is given. This process is known as dead-reckoning. The other is to use an external sensor that measures the robot's position globally. See Lecture 6 for details. In this section, we will use either 2D wiener velocity model or quasi-constant turn model as our motion model.

In our setup, the robot is programmed to follow a black line on a white background. We have a predefined test track shown in Figure 6, a semi-elliptical track, in a 121.5cm x 121.5cm squared boundary area. Using IMU and camera sensors, you will build a nonlinear filter to estimate the robot's global position and heading.

Figure 6: The predefined semi-elliptical tracking path.

4.1 Using IMU only

In this section, we will use gyroscope and motor controller to estimate the position and heading of the robot. The robot's position and heading can be determined from the wheel rotation measurements. Our robot is a six-wheeled robot. We have configured our *MotorControl.py* script that allows the same amount of input voltage to each wheel on each side. Therefore, the speed of the robot will be a function of the pulse width modulation (PWM) input given to each side. In the calibration, we use 30% PWM. This determines the linear gain between PWM input to the actual robot speed in straight movement. From the log file, we can get the PWM inputs of both sides of the robot. Then, we can estimate the current velocity of the robot.

Task 6a. Choose a dynamic model to model the motion of the robot. The robot moves in two dimensions and hence, a two dimensional model is needed. Once you have chosen a suitable motion model, discretize it using an appropriate discretization method. The inputs to the motor in terms of pulse width modulation are recorded which gives the velocity and the gyroscope gives the turning rate. You should use these inputs in your dynamic model.

Task 6b. Do dead-reckoning (i.e., prediction) based on speed measurements from motor control and turn rate measurements from gyroscope.

Task 6c. Compare your dead-reckoning result with the predefined track in Figure 6.

Hints:

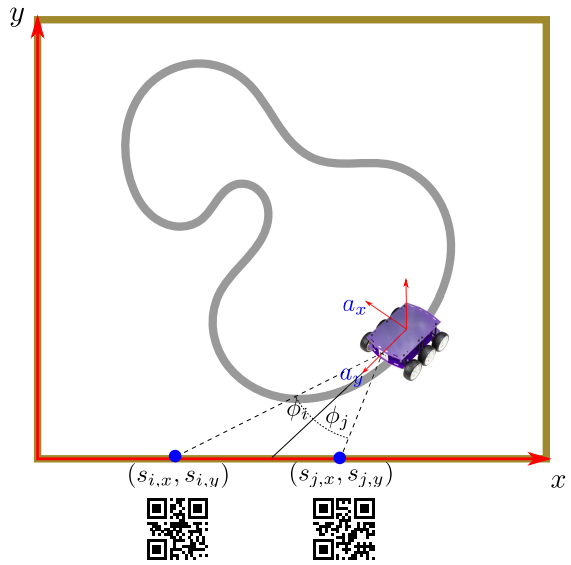

Figure 7: Illustration of the global coordinate system, QR-codes with known positions in global coordinate, and the robot local acceleration. When the robot camera heads to the y axis, the global angle of the robot equal to zero.

- See the course slides for dynamic models. Also make sure that the sign of the gyroscope measurement is correct and that its units are converted to radians.
- We will use the gyroscope to measure the heading.
- You can consider the velocity to be the average of input pulse applied to each wheel. If you are interested to build up more complicated model, you can consider the diameter and width of the wheels as well as the distance between right and left wheels. The diameter and width of each wheel are 65mm and 25mm, respectively. The difference between the left and right wheels is 180mm.

4.2 Tracking with both IMU and Camera

The IMU sensors alone may not be sufficient to perform localization accurately. Therefore, we need to incorporate camera measurements into the filtering algorithm. The camera sensor detects the landmarks (QR-codes). Each QR-code is in a known position on the global coordinate which is given in *qr_code_position_in_global_coordinate.csv* file. This information will be used in the filtering algorithm. Thus, we can estimate the global position of the mobile robot. Notice that at each time-stamp, there could be more than one QR-code detected. Furthermore, timestamp from camera log files might differ totally with the timestamp you get from IMU.

Task 7. Use discretization and nonlinear filter to estimate the robot position and heading in global coordinate.

Hint:

- The model and filtering algorithms may be quite sensitive to the tuning parameters (spectral density of the process noise and measurement noise). For the process noise, think about what it actually represents physically and relate it to the robots motion. For the measurement noise, you may estimate it based on the model parameter estimation data and possibly add some margin to account for modeling errors.

5 Reporting

To report the technical results of a project is an important skill. Reporting should be done using concise and accurate language, including enough detail such that someone with the same education and background can understand, interpret, and reproduce your work. Your derivations, results, and interpretations should be backed up by data, illustrations, and so forth. Furthermore, also make sure that you answer and discuss the questions raised in this guide.

Your project report should include at least the following[2]:

- An abstract that briefly summarizes what you have done and what the results are,
- a brief introduction to the project and the problem that you are solving,
- derivation of the model,
- calibration procedures that are used (if any).
- description of the estimation method(s) used for parameter estimation, validation, and tracking,
- the results,
- conclusions and/or a summary,
- references (if applicable).

For the intermediate report, not all of the above items might apply yet. As a guide, you could include a brief description of the robot and sensor models, explanatory data analysis of the log file, and plans to implement localization algorithm and tracking.

6 Appendix

In this section we will describe how to connect to the robot platform via SSH, and how to run the simulation.

6.1 Template codes

Template codes for Part II and Part III are available at

`https://github.com/puat133/DiddyBorg_Sensor_Fusion.`

Also note that scripts described in this section can also be found in the aforementioned link.

[2]You are free to use whatever structure you prefer, as long as it is consistent and logical.

6.2 Connecting to the robot

To connect to the robot, you could use SSH connection either directly via terminal, or using VNC (preferred one).

- Turn on the robot using switch located at the bottom of the robot.
- Find the IP address of the robot. Check the MAC address written in the upper part of the robot. Then use an ip scanner to find the associated IP address. If you facing a difficulty to find the IP address, you could also connect a monitor to the robot via an HDMI port located behind the camera (unscrew top part of the robot first). Then connect usb keyboard to robot's USB ports. Then execute `ifconfig`.
- Once you found the IP address of the robot, you can connect to the robot via SSH, with user name:`pi`, password: `pipipipi`. You can also automate this process as below:

```
ssh-keygen -t rsa -b 4096
ssh-copy-id pi@your_raspberry_ip_address
```

- The necessary files are located in

```
~/Git/DiddyBorg_Sensor_Fusion
```

  We have also prepared a set of QR-codes in `~/Git/DiddyBorg_Sensor_Fusion/QR-codes` folder. Please pull from `Git` at least once when you just receive the robot to make sure that you have the unmodified version of program.

6.3 Modules

To run the module change the directory to `~/Git/DiddyBorg_Sensor_Fusion/data`. All log files will be stored in this folder. There are three files that you need to run the robot properly:

- `IMU.py`
- `CameraModule.py`
- `MotorControl.py`

These files stored in `~/Git/DiddyBorg_Sensor_Fusion/DiddyBorg_python`. To check options available for each of these files, execute:

```
python3 ../DiddyDiddyborg_python/module_xxx.py --help
```

Pressing `Escape` button will immediately stop the python scripts, except on `CameraModule.py` while not showing any video output. Ideally, you should execute `IMU.py` command until it is ready, and then `CameraModule.py`. Once the camera is ready, then execute `MotorControl.py`.

You can modify (and encougared!) these files to accommodate your need. Should you need to reset the configuration to default, you can always pull from `Git`.

```
git pull
```

6.3.1 IMU

After changing the directory to `~/Git/DiddyBorg_Sensor_Fusion/data`, to collect reading from IMU, you can run

```
python3 ../Diddyborg_python/IMU.py
```

You can also specify the output file name using `--output=some_files.csv`. Default sampling times is 0.05, and you can modify it with `--sampling=0.1`.

The IMU log files column are, the Timestamp in ms, linear acceleration in x,y,z axis given in gravity unit, roll and pitch angle from accelerometer in degree, gyroscope x,y,z in degree/s, and magnetometer field strength in x,y,z axis in Gauss unit.

6.3.2 Camera module

To collect reading from IMU, you can execute the following command

```
python3 ../Diddyborg_python/CameraModule.py
```

You can also specify the output file name using `--output=some_files.csv`. If you have access to X-server when connecting to the robot (using VNC, for example), you can also specify `--show` to show the video stream from the camera module. You can also specify the QR-code length in cm using `--qrlength=xx` where the input is given in cm. The IMU log files column are, the Timestamp in ms, QR-code number, center position $(C_{x,i}, C_{y,i})$ of QR-code in pixel, width and height of QR-code in pixel, raw distance from camera to QR-code in cm, raw attitude of the QR-code relative to the camera in degree.

6.3.3 Motor control

To run the motor

```
python3 ../Diddyborg_python/MotorControl.py
```

You can also specify the output file name using `--output=some_files.csv`. The IMU log files column are, the Timestamp in ms, first and second inputs as a percentage of pulse width modulation (PWM). The input signal is between 0-1.

6.3.4 Line sensor

Line Sensor has to be calibrated when its tried in a different material for every first time once its calibrated for the black line not necessary for following events. To calibrate it for the black line click the calibration button once and while the LEDs blink robot should be waved like the arrow showed in the following picture. Make sure all five sensors are exposed to both regions (black and white). The mode indicator lighting in the module stands for inverse detected region power on. Make sure the mode indicator is off for whole operation.

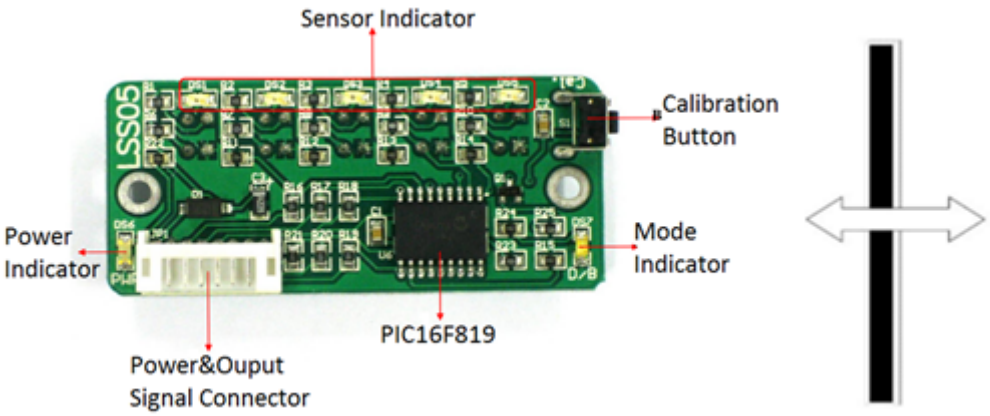

Figure 8: IR Line Detector

6.3.5 Data & experiment checklist

No.	Task	File location	Notes
Part I			
1	Static IMU experiment	data/task1/	To check the static bias and the covariance.
2	IMU calibration	data/task2/	To determine the gain and the bias of the accelerometers.
3	Camera module calibration	data/task3/	To determine the focal length and bias.
4	Motor control	data/task4/	To determine the speed of the robot.
Part II			
5	Localization	data/task5/	To estimate the position and attitude of the robot.
Part III			
6	Tracking with IMU	data/task6/	To develop a tracking algorithm.
7	Tracking with IMU and camera	data/task6/	To develop a tracking algorithm.

References

[1] U. Qureshi and F. Golnaraghi, "An algorithm for the in-field calibration of a MEMS IMU," IEEE Sensors Journal, vol. 17, no. 22, pp. 7479–7486, nov 2017.

[2] P. Patonis, P. Patias, I. N. Tziavos, D. Rossikopoulos, and K. G. Margaritis, "A fusion method for combining low-cost IMU/magnetometer outputs for use in applications on mobile devices," Sensors, vol. 18, no. 8, p. 2616, aug 2018.

[3] K. Papafotis and P. P. Sotiriadis, "MAG.i.c.AL. – a unified methodology for magnetic and inertial sensors calibration and alignment," IEEE Sensors Journal, pp. 1–1, 2019.

[4] D. A. Forsyth and J. Ponce, Computer Vision: A Modern Approach, 2nd ed. Pearson, 2011.