Project Guide

Tracking of an Autonomous Robot

Roland Hostettler

Department of Electrical Engineering and Automation
Aalto University, Finland

November 23, 2018

Contents

1 Introduction

The aim of this project is to develop an algorithm for tracking an autonomous robot by using smartphone sensors. The robot, a DiddyBorg rover-type robot, is programmed to follow a black line. On top of the robot, a strong permanent magnet is installed (Figure 1). By measuring the magnetic field using magnetometers, we can localize the source of this magnetic field, and hence we can determine the robot's position. The same approach can, for example, be used to track metallic objects such as vehicles or submarines (Birsan, 2003; Wahlström, 2010).

Modern smartphones are well equipped with different sensors such as accelerometers and gyroscopes but also magnetometers (digital compasses). The magnetometers are traditionally used for navigation purposes, that is, finding the direction toward (magnetic)

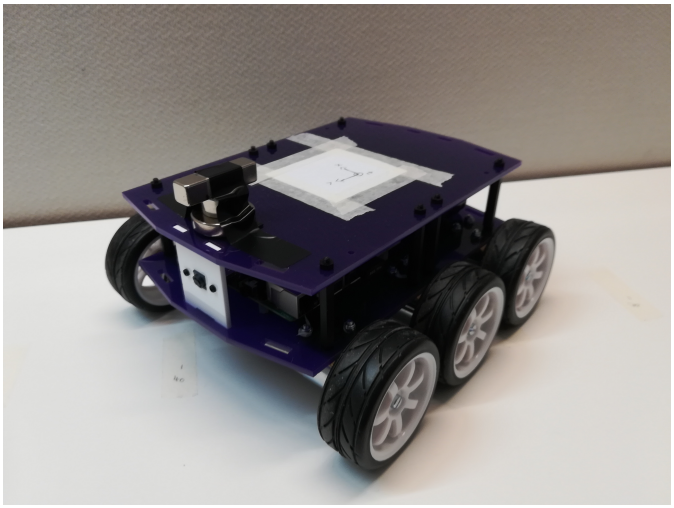

Figure 1: DiddyBorg robot with permanent magnet on top.

north. They essentially measure the magnetic vector field, which is dominated by the earth's magnetic field that points northwards. However, small local magnetic sources cause an additional field that is additive to the earth's magnetic field and by suitable modeling, the two fields, and thus the two sources, can be separated. This is what we exploit in this project to localize and track the robot.

The project consists of two parts. In the first part, we develop and verify the sensor model that describes the relationship between the magnetometer measurements and the position of the robot. This includes the following steps:

- Derivation of the sensor model,
- estimation of the model parameters, and
- model validation.

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 parts, you will need to take measurements using the smartphone and the robot and you will need to allocate time for this. 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 based on the peer review of the intermediate report. The final grade of the project work will be based on the final report as well as the comments from the peer-review. On the course homepage, you can find more information about booking lab time for measurements, the peer review, deadlines, and the grading criteria.

2 Part I: Sensor Modeling

2.1 Sensor Model

The magnetometer measures the three-dimensional magnetic field vector, that is, for every sensor reading, we obtain a measurement $\boldsymbol{y}_n$ that is a 3×1 vector. This measurement is a superposition of the earth's magnetic field in the sensor's local coordinate frame and the vector field created by the permanent magnet on the robot. The vector field of the permanent magnet can be modeled as a *magnetic dipole* where the location of the dipole coincides with the robot's position. The magnetometer also suffers from a slowly-varying, but significant bias. We also assume that the sensor (phone) is stationary throughout the project. Hence, our model should include the following components:

- a static component modeling the earth's field, denoted $\boldsymbol{B}_G$ (a 3×1 vector),
- a dipole field due to the permanent magnet,
- an additive bias, denoted $\boldsymbol{b}_m$ (a 3×1 vector).

As long as the sensor is stationary, that is, it is not moved or rotated and for our tracking purposes, the earth's magnetic field as well as the bias can be considered an offset term. Hence, we can define the static offset to be

$$\boldsymbol{B}_0 = \boldsymbol{B}_G + \boldsymbol{b}_m. \tag{1}$$

What remains is to model the contributions of the dipole field.

Tasks:

1. Find the model of the magnetic vector field (the flux density) $\boldsymbol{B}(\boldsymbol{p})$ for a dipole with dipole moment $\boldsymbol{m}_T$ at a position $\boldsymbol{p}$ relative to the dipole's location. Here, $\boldsymbol{B}(\boldsymbol{p})$ is a 3×1 vector that depends on $\boldsymbol{p}$, the position with respect to the dipole's location, and $\boldsymbol{m}_T$ is a 3×1 vector. Note that you may drop the constant scaling factor $\mu_0/(4\pi)$ from your model for simplicity (this will just result in a scaled $\boldsymbol{m}_T$).

2. Rewrite the model such that it is linear in the unknown magnetic dipole moment $\boldsymbol{m}_T$, that is, determine the position-dependent matrix $\boldsymbol{H}(\boldsymbol{p})$ such that

$$\boldsymbol{B}(\boldsymbol{p}) = \boldsymbol{H}(\boldsymbol{p})\boldsymbol{m}_T. \tag{2}$$

3. Formulate the complete measurement model for the 3×1 vector measurements $\boldsymbol{y}_n$ of the magnetometer, in terms of the offset (1), the dipole field (2), and additive measurement noise $\boldsymbol{r}_n$. This can be written as a linear model

$$\boldsymbol{y}_n = \boldsymbol{G}_n(\boldsymbol{p}_n)\boldsymbol{\theta} + \boldsymbol{r}_n, \tag{3}$$

with the parameters $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{B}_0^\mathsf{T} & \boldsymbol{m}_T^\mathsf{T} \end{bmatrix}^\mathsf{T}$.

2.2 Estimation of the Model Parameters

We have now derived a model for the magnetic field measured by the magnetometer. The model depends on two types of parameters: First, it depends on the position $\boldsymbol{p}$ of the dipole, which is the property that we are going to exploit for localization and tracking. Second, it also depends on the magnetic offset $\boldsymbol{B}_0$ and the dipole moment $\boldsymbol{m}_T$, which we need to determine before we can use the model for tracking. The latter parameters are what we are going to estimate in this part (i.e., we are going to calibrate the model).

The parameters can be estimated by placing the robot at different positions (at least two different positions — why?) and measuring the magnetic field. Then, given the different positions $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N$, the model is linear in $\boldsymbol{\theta}$ and we can estimate $\boldsymbol{\theta}$.

To collect the measurement data (we will refer to this dataset as the *calibration data*), follow these steps.

> *Important: Be careful not to keep the permanent magnet too close to the smartphone for an extended period of time. Permanent exposure to strong magnetic fields may destroy the magnetometer.*

1. Place the phone on a level surface. The point where you place it will be the origin of the coordinate system for these measurements.

2. Place the robot some distance away from the phone, not too far (the magnetic field of the dipole will be too weak) and not too close either (the dipole approximation of the magnetic field does not hold anymore). For simplicity, make sure that the coordinate systems of the phone and the robot are aligned (i.e., only introduce a translation between the origins of the coordinate systems, but no rotation, see Figure 2). Write down the coordinates of the robot in terms of the phone's coordinate system (including the z-coordinate).

3. Start the *Sensor Fusion* app on the phone and select "Log Data". Ensure that the checkbox "Log" is checked (and "Stream" is unchecked), then select the "Magnetic Field (raw)" sensor and start logging by pressing "Start". The sensor measures at a frequency of 10 Hz (i.e., you get 10 measurements per second), so by measuring for about 5 s–10 s per position, you should get enough datapoints for that position. Finish logging by pressing the "Stop" button.

4. Repeat steps 2–3 for a couple of different robot positions. Make sure that you have stopped the measurements before moving the robot. Once you have moved the robot to its new position, restart the measurement.

For each measurement, the app generates a file on the phone. You can transfer those files to your computer by opening the file manager and navigating to the phone's main folder, where you should see a couple of files named `sensorLog_*.txt`. Select the files and transfer them to your computer (e.g., using USB, Bluetooth, or e-mail), where we

(a) (b)

Figure 2: Illustration of the smartphone and robot coordinate systems. The origin of the phone's coordinate system is in the middle of the phone, while the robot's coordinate system's origin is where the magnet is located.

will analyze them in Matlab. You can load the calibration data into Matlab using the `load_data()`-function (see MyCourses, use `help load_data` to see a description of the function's interface). *Once you have successfully transferred the files to your computer, delete them on the phone.*

Using the calibration data, we can now estimate the magnetic offset as well as the dipole moment (i.e., estimate $\boldsymbol{\theta}$).

Tasks:

1. Collect a calibration dataset as described above.
2. Implement the linear least squares estimator that uses the data you just gathered to estimate $\boldsymbol{\theta}$.

Hints:

- It may be useful to first take a measurement with a stationary phone where you freely move the robot in the phone's neighborhood and then plot the signal in Matlab just to try out the measurement setup and see how the moving robot affects the magnetometer.
- We get measurements at L robot positions, and at each position $l = 1, \ldots, L$, we have measured N_l 3-dimensional measurements. Hence, there are a total of $3\sum_{l=1}^{L} N_l$ datapoints and the final measurement vector $\boldsymbol{y}$ should be $(3\sum_{l=1}^{L} N_l) \times 1$.
- To construct the overall measurement matrix $\boldsymbol{G}$ the Matlab function `repmat()` is helpful.

- Since there is no dynamic variation in the measurement signals at this point, we can use a sample estimator to estimate the covariance matrix $\boldsymbol{R}_n$. To do this, you can use the Matlab command `cov()` for each of the individual measurement positions.

2.3 Model Verification

Having calibrated the model, we can now verify it. We do this be evaluating the ability of localizing a static target using the estimated parameters. That is, having estimated $\boldsymbol{\theta}$, we now try to estimate $\boldsymbol{p}$ using $\hat{\boldsymbol{\theta}}$, which is a nonlinear least squares problem. For this purpose, we need a new dataset (the *validation data*), that is collected independently of the calibration data.

Hence, you first need to collect a second set of measurement data in the same way as you collected the calibration data in the previous section (again, do not rotate the robot). Try a few different positions, also positions different from the ones that you used in the last section. Remember to write down the ground truth positions. Note that if you have moved the phone, or if a significant amount of time (several hours) has passed since you collected the calibration data, you also need to re-collect a new calibration dataset (due to changes in the environment and the magnetometer bias).

Tasks:

1. Collect a validation dataset as described above.

2. Derive the Jacobian matrix of the measurement model with respect to the unknown position $\boldsymbol{p}_n = \begin{bmatrix} p_n^x & p_n^y & p_n^z \end{bmatrix}^\mathsf{T}$. The task is to localize the robot in the x-y-plane only and p_n^z can be assumed constant and known (i.e., you only need to derive the Jacobian with respect to p_n^x and p_n^y).

3. Implement one or more nonlinear least squares estimator(s) (e.g., gradient descent, Gauss–Newton, or Levenberg–Marquardt) and try to localize the target. Note that you have to do this for every validation position individually. Also compute the contour lines of the cost function in the x-y-plane.

4. Investigate and comment the localization power of the model. How accurate are the estimated positions? How far away from the sensor are you able to accurately estimate the position of the target? What do the contour lines tell us about the ability to localize the robot?

3 Part II: Tracking

In the first part of the project, we derived the sensor model, developed an estimator for the unknown model parameters, and verified that the model is indeed useful for localization purposes. In this second part, we extend these results and develop the models and algorithm necessary for tracking the robot when moving. A prerequisite for this is

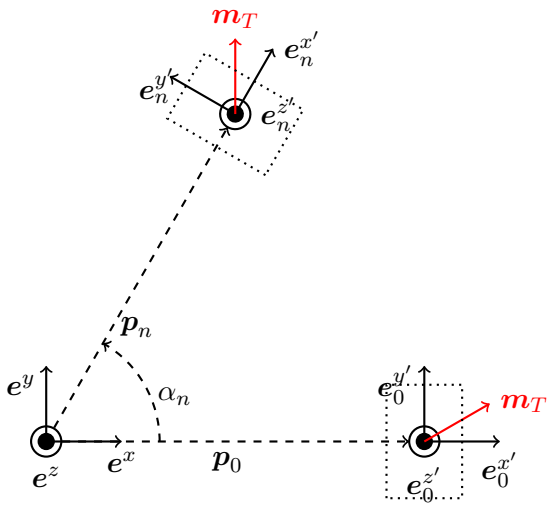

Figure 3: Illustration of the global (phone) coordinate system spanned by the unit vectors $\{\boldsymbol{e}^x, \boldsymbol{e}^y, \boldsymbol{e}^z\}$, the robot's initial coordinate system ($\{\boldsymbol{e}_0^x, \boldsymbol{e}_0^y, \boldsymbol{e}_0^z\}$), and the coordinate system at t_n ($\{\boldsymbol{e}_n^x, \boldsymbol{e}_n^y, \boldsymbol{e}_n^z\}$). The red vector illustrates the magnetic dipole moment $\boldsymbol{m}_T$, which is rotated by the rotation α_n of the robot.

that all the tasks in Part I 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.

3.1 Tracking Using a Single Sensor

In Section 2.1, the basic sensor model for localizing a static dipole using magnetometers was developed. Next, a dynamic model that describes the motion of the robot has to be chosen and a small adjustment to the measurement model has to be made to account for robot rotations.

In particular, the measurement model derived in the first part assumed that the coordinate systems of the robot and the sensor are aligned and that only a translation is applied between the two. This implies that the magnetic dipole moment $\boldsymbol{m}_T$ is estimated in the sensor's coordinate system, which in this case is equal to the robot's coordinate system. However, in practice, the dipole moment is constant with respect to the robot's coordinate system. Hence, when rotating the robot, its coordinate system rotates relative to the global (sensor) coordinate system and so does the dipole moment $\boldsymbol{m}_T$. This in turn means that the magnetic field measured by the sensor is a dipole field caused by the rotated moment $\boldsymbol{m}_T'$. These relationships are illustrated in Figure 3, where the subscripts

0 and n denote the reference frame during model parameter estimation and any arbitrary time t_n, respectively.

Tasks:

1. Chose a dynamic model to model the motion of the robot (e.g., Wiener velocity model, quasi-constant turn model). 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.

2. Modify the sensor model developed in Part I to take the robot rotation into account. Depending on the dynamic model chosen above, the rotation angle is determined in different ways. The dynamic model together with the modified sensor model now define your complete state-space model.

3. Choose and implement a suitable filtering algorithm for the state-space model derived in Tasks 1 and 2.

4. Take measurements and validate the tracking algorithm (see Section 3.3 for instructions on how to take measurements of the moving robot). Start by evaluating the method based on the simple test track provided and then move on to more complex scenarios if time permits (see Section 3.3 for details). Since there is no ground truth available in this case (unless you devise a good strategy to find a ground truth), you have to make enough convincing experiments to show that the algorithm actually works. How accurate can you track the robot? How far from the sensor can the robot be tracked?

Hints:

- The rotation matrix for a counterclockwise rotation of the 3×1 vector $\boldsymbol{a}$ by an angle α around the z-axis is given by

$$\boldsymbol{C}(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  and thus, the rotated vector is

$$\boldsymbol{a}' = \boldsymbol{C}(\alpha)\boldsymbol{a}.$$

- Due to the choice of the coordinate systems (Figure 2), a robot heading of $\pi/2$ corresponds to a motion parallel to the global coordinate system's y-axis. In this case, the rotation angle α is zero, and thus, this offset has to be taken into account when calculating the rotation angle α from the heading of the robot.
- 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.

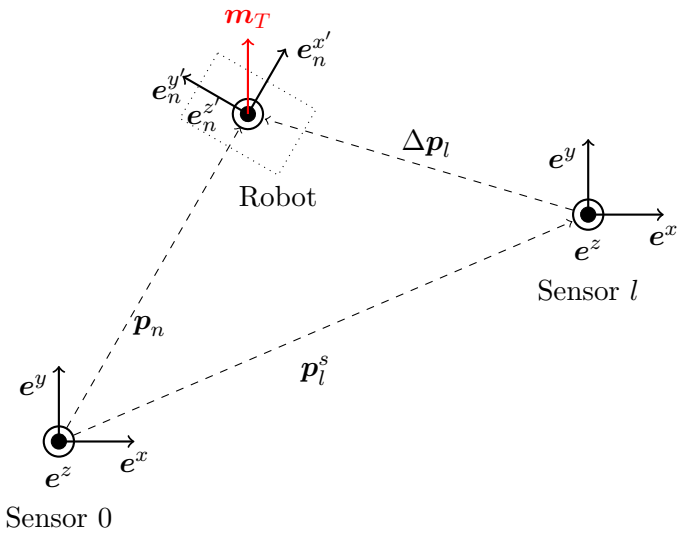

Figure 4: Illustration of the measurement setup for multiple sensors. The lth sensor is located at position $\boldsymbol{p}_l^s$ and the dipole model describes the vector field based on the relative positions $\Delta\boldsymbol{p}_l$ of the dipole $\boldsymbol{p}_n$ and the sensor $\boldsymbol{p}_l^s$.

3.2 Tracking Using Multiple Sensors*

As shown in the first part, the range of a single sensor is quite limited and hence, we can only track rather small movements of the robot. However, by combining multiple sensors we can overcome this and cover a larger area. Since adding more sensors to the problem does not affect the movement of the robot, the dynamic model does not change, and the only thing that we have to change is the sensor model. Instead of having measurements of the 3D vector field at one point in space, we now get 3D vector field measurements at as many points as we have sensors.

Assuming that one of the sensors coincides with the origin of the coordinate system (reference sensor), it is easiest to make sure that the coordinate systems of the other sensors are aligned with the coordinate system of the reference sensor and that only translations exist between the different sensors.

Next, remember that the position in the dipole model derived in Part I is actually the relative position between the sensor and the dipole. Hence, when tracking the dipole with sensors that are not located at the origin, the dipole model needs to be reformulated such that it takes the translation of the sensor into account and such that the measurements of all the sensors are expressed in the global coordinate system. The geometry of this setup is illustrated in Figure 4.

Note: It is not required to complete this part (Section 3.2) to pass the project. However, it might affect the grade.

Tasks:

1. Extend the sensor model for an arbitrary number of sensors taking the above considerations into account.

2. Take measurements and validate the tracking algorithm using multiple sensors. Start by considering the simple trajectory before moving towards more complex (larger) tracks. Does using multiple sensors improve localization accuracy? If so, where is the accuracy the highest? Does it also improve coverage (i.e., can the robot be tracked in larger areas as hypothesized)? Are there any conditions that actually make tracking worse compared to the single sensor case?

Hints:

- In this case, you need to calibrate $\boldsymbol{B}_0$ for each sensor individually. You can do this either using the method derived in Part I or by realizing that in the absence of the robot (i.e., with the robot being far enough away), the only thing measured by the sensor is $\boldsymbol{B}_0$.
- Unfortunately, the clocks of the mobile phones are not synchronized well enough, that is, the timestamps you obtain from the data are offset with respect to each other. Hence, before you can process the data, you need to make sure that the clocks are synchronized.
  A simple way of doing this is by not only recording the magnetometer but also the accelerometer signal in the data recording app. Then, before you start the robot, you can tap the table a few times, which should be picked up by the accelerometers in all the sensors. In the computer, you can determine the clock offset by finding the taps in the accelerometer signal (e.g., by visually comparing the two signals or by using the cross-correlation, see `help xcorr` in Matlab; the `load_data()` function has support for loading the accelerometer data into Matlab, see `help load_data`). Once you have determined the clock offset, choose one clock as your reference clock and use Matlab's `interp1()` function to interpolate the data of all the other sensors to the same timebase.

3.3 Measurements

For tracking the robot, it is necessary to estimate the model parameters (bias $\boldsymbol{B}_0$ and dipole moment $\boldsymbol{m}_T$) before every measurement session. This is due to the fact that the geomagnetic field is slowly varying, metallic structures in the surrounding area have changed, and the magnetometer bias has changed. To re-estimate the model parameters, proceed as in Part I (see Section 2.2).

Once the model parameters are estimated, you can collect measurement data from the moving robot for evaluating the tracking algorithm. The robot is programmed to follow a black line on a white background and it uses its camera to detect the line and follow it. Note that since the camera looks ahead of the robot, it will not follow the line exactly, but rather use the line as a rough guide on where to go. Hence, the line does not

correspond to the true robot trajectory. Additionally, other black objects in the field of view may distract the robot.

In the lab area, there is one predefined test track, a (rather small) elliptical track, which is suitable for evaluating the tracking algorithm using one sensor. Once you are able to track the robot on this track, you are free to experiment with other and more complex trajectories (also for the multiple sensor case). Contact the teacher or teaching assistant for more information.

After starting the robot, it will be in idle mode and line following can be activated and deactivated by using a green paper. To take measurements with the moving robot, follow these steps:

1. Switch the robot on by flipping the switch between the motors on the bottom of the robot. It takes about 10–15 s for the robot to boot.
2. Place the robot on top of the line such that the line is in the field of view of the camera.
3. Start the measurement using the data logger app on the phone.
4. Hold the card with the green paper in front of the robot's camera at a distance of about 5–10 cm to make the robot start following the line. *Closely monitor the robot whenever it is moving to prevent it from falling off the table or bumping into other obstacles.*
5. To stop the robot, again hold the card with the green face in front of the camera.
6. When done taking measurements, flip the switch on the bottom to turn the robot off.

4 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[1]:

- 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,
- description of the estimation method(s) used for parameter estimation, validation, and tracking,

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

- the results,
- conclusions and/or a summary,
- references (if applicable).

For the intermediate report, not all of the above items might apply yet. Use your own judgment about what you see necessary to include, based on the criteria outlined in the beginning.

References

Birsan, M. (2003). Non-linear Kalman filters for tracking a magnetic dipole. Technical Report DRDC-ATLANTIC-TM-2003-230, Defence R&D Canada, Dartmouth, NS, Canada.

Wahlström, N. (2010). Target tracking using maxwell's equations. Master's thesis, Linköping University.