EE 183DA, Winter 2018
Lab 3-4: State Estimation and Motion Planning of a Mobile Robot

Jonathan Chang, Michael Warren, Nick Bruce

3/13/2018

1 Introduction

A mobile robot is a common application that requires a multitude of techniques in state estimation, controllability, and motion planning, all which requires contending with noise sensors and forces that cannot be necessarily predicted beforehand. This project uses a robot body printed on paper, so as to minimize the effect of friction, a function of floor material and speed, but the relative looseness in which the electronics sit may have introduced additional noise.

We are given two laser sensors to detect distances at an orthogonal distance, to the front and right, as well as an IMU sensor. For ideal controllability, we would have ideally liked at least 4 range finders at various angles, 2 on each side. For instance, 2 sensors cannot deterministically predict the position of the robot when sensors are pointed at the same wall. Nevertheless, we attempted to contend with the shortcomings of our design. More time would have been preferred to tune parameters and integrate various parts of our road map.

2 Simulation

In many projects, physical testing with actual apparatus may be impractical or dangerous. While those are not concerns for this project, simulation, if done right, should reduce time to completion. Therefore, the simulation should be poised to model the reality of the experiment, by using Gaussian noise where necessary and having a proper separation of simulation and model components. We designed our simulation with this in mind.

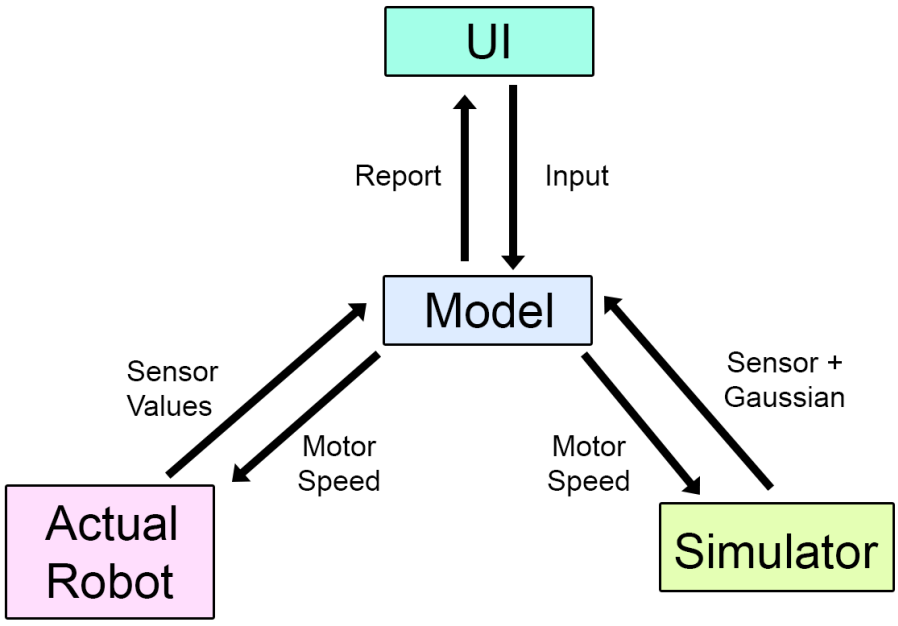

Figure 1: Organizational flowchart.

In order to prevent duplicate work, the model should act like an ideal switch, such that trading between the *simulator* and *actual robot* involves minimal preparation. This could be achieved through event programming, and designing a general interface, such that the model itself remains constant regardless of which module passes it sensor values.

3 User interface

Although not required for this class, a little bit of design could greatly improve work flow and possibly end results. For example, having a USB cable connected to the robot at all times was something we wanted to avoid, to prevent an extra force affecting end results. We also cannot rely on local network security; therefore, the robot should act as its own access point (although WiFi mode was used for testing). Thankfully, this is a feature on the *ESP8266*. A combination of Javascript and Python was chosen for their browser integration without much work, such that we could have a good reporting and controlling scheme for feedback purposes.

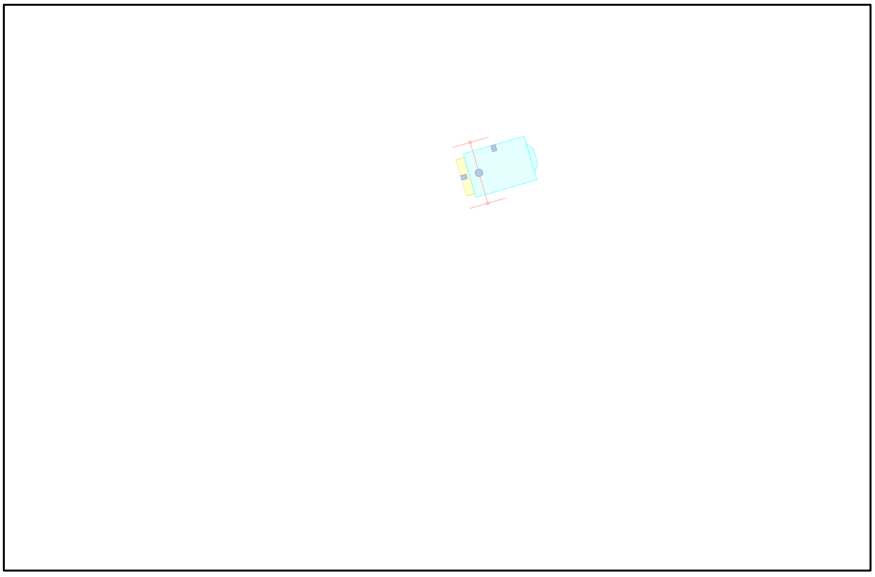

Figure 2: Used to test estimation iterations with keyboard.

4 Model

To model the simulation, we use the *Extended Kalman Filter (EKF)*, since we have a non-linear function. The EKF requires various parameters, such as noise and variance matrices, that may be hard to determine, whether experimentally or mathematically. The observational matrix, R, could be determined by measuring noise without input at start, or consulting sensor specifications. For example, a glance through documention finds

sensor	notes
Adafruit GY VL53LOX Laser sensors	0.03-0.12 ranging accuracy error.
FEETECH FS90R Continuous servo	100-130 max rpm, no load.
MPU-9250	± 0.005 non-linearity.

Table 1: Sensor errors and limits

5 Estimate state

The *Extended Kalman Filter (EKF)* prediction step requires that

$$\hat{x}_t = f(x_{t-1}, u_t, w_t) \tag{1}$$

We use the previous state, which comprises $[x, y, \theta, v]$, and sensors as inputs. The sensors are a, the acceleration in the forward direction in robot frame of reference measured by the accelerometer; and $\dot{\theta}$, the change in angle since the last reading measured by the gyro.

By approximation, the arc length,

$$\alpha \approx \frac{v + a}{2} \tag{2}$$

Since $\alpha = r\dot{\theta}$, the turn radius,

$$r = \frac{\alpha}{\dot{\theta}} \tag{3}$$

To prevent large calculations approaching infinite, and division by zero errors, we consider two cases separately. If $\dot{\theta} > 0.1$, we consider that the robot is turning. Otherwise, the robot is going relatively straight.

If the robot is turning, we need to find the center of rotation. We know the r and α, so we apply a transform to find the point a distance of r to the right relative to the robot state. This is a rotation of the relative position $-r$ by θ, then a translation by x and y. The result is

$$p_{center} = \begin{bmatrix} -rcos\theta + x \\ -rsin\theta + y \end{bmatrix} \tag{4}$$

We must then rotate by α about p_{center}. By applying the transformation matrix again, we end up in

$$p_{new} = \begin{bmatrix} rcos\theta cos\dot{\theta} - rsin\theta sin\dot{\theta} - rcos\theta + x \\ rsin\theta cos\dot{\theta} + rsin\dot{\theta} cos\theta - rsin\theta + y \end{bmatrix} \tag{5}$$

By trigonometric identities,

$$p_{new} = \begin{bmatrix} r(cos(\theta + \dot{\theta}) - cos\theta) + x \\ r(sin(\theta + \dot{\theta}) - sin\theta) + y \end{bmatrix} \tag{6}$$

On the other hand, if the robot goes nearly straight, then a transformation matrix with the original position as origin, and the robot travels by some α leaves

$$p_{new} = \begin{bmatrix} -\alpha sin\theta + x \\ \alpha cos\theta + y \end{bmatrix} \tag{7}$$

In either case,

$$\theta_{new} = \theta + \dot{\theta} \tag{8}$$

$$v_{new} = v + a \tag{9}$$

Therefore, in the turning condition, the state estimation matrix Jacobian becomes

$$A = \begin{bmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} & \frac{\partial f_x}{\partial \theta} & \frac{\partial f_x}{\partial v} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} & \frac{\partial f_y}{\partial \theta} & \frac{\partial f_y}{\partial v} \\ \frac{\partial f_\theta}{\partial x} & \frac{\partial f_\theta}{\partial y} & \frac{\partial f_\theta}{\partial \theta} & \frac{\partial f_\theta}{\partial v} \\ \frac{\partial f_v}{\partial x} & \frac{\partial f_v}{\partial y} & \frac{\partial f_v}{\partial \theta} & \frac{\partial f_v}{\partial v} \end{bmatrix} \tag{10}$$

The equations are

$$\frac{\partial f_x}{\partial x} = 1 \tag{11}$$

$$\frac{\partial f_x}{\partial y} = 0 \tag{12}$$

$$\frac{\partial f_x}{\partial \theta} = r(sin\theta - sin(\theta + \dot{\theta})) \tag{13}$$

$$\frac{\partial f_x}{\partial v} = \frac{cos(\theta + \dot{\theta}) - cos\theta}{2\dot{\theta}} \tag{14}$$

$$\frac{\partial f_y}{\partial x} = 0 \tag{15}$$

$$\frac{\partial f_y}{\partial y} = 1 \tag{16}$$

$$\frac{\partial f_y}{\partial \theta} = r(cos(\theta + \dot{\theta}) - cos\theta) \tag{17}$$

$$\frac{\partial f_y}{\partial v} = \frac{sin(\theta + \dot{\theta}) - sin\theta}{2\dot{\theta}} \tag{18}$$

$$\frac{\partial f_\theta}{\partial x} = \frac{\partial f_\theta}{\partial y} = \frac{\partial f_\theta}{\partial v} = 0 \tag{19}$$

$$\frac{\partial f_\theta}{\partial \theta} = 1 \tag{20}$$

$$\frac{\partial f_v}{\partial x} = \frac{\partial f_v}{\partial y} = \frac{\partial f_v}{\partial \theta} = 0 \tag{21}$$

$$\frac{\partial f_v}{\partial v} = 1 \tag{22}$$

In a straight line, the equations are

$$\frac{\partial f_x}{\partial x} = 1 \tag{23}$$

$$\frac{\partial f_x}{\partial y} = 0 \tag{24}$$

$$\frac{\partial f_x}{\partial \theta} = 0 \tag{25}$$

$$\frac{\partial f_x}{\partial v} = \frac{-cos\theta}{2} \tag{26}$$

$$\frac{\partial f_y}{\partial x} = 0 \tag{27}$$

$$\frac{\partial f_y}{\partial y} = 1 \tag{28}$$

$$\frac{\partial f_y}{\partial \theta} = 0 \tag{29}$$

$$\frac{\partial f_y}{\partial v} = \frac{sin\theta}{2} \tag{30}$$

$$\frac{\partial f_\theta}{\partial x} = \frac{\partial f_\theta}{\partial y} = \frac{\partial f_\theta}{\partial v} = 0 \tag{31}$$

$$\frac{\partial f_\theta}{\partial \theta} = 1 \tag{32}$$

$$\frac{\partial f_v}{\partial x} = \frac{\partial f_v}{\partial y} = \frac{\partial f_v}{\partial \theta} = 0 \tag{33}$$

$$\frac{\partial f_v}{\partial v} = 1 \tag{34}$$

We can also describe an input Jacobian matrix, such that

$$B = \begin{bmatrix} \frac{\partial f_x}{\partial \dot{\theta}} & \frac{\partial f_x}{\partial a} \\ \frac{\partial f_y}{\partial \dot{\theta}} & \frac{\partial f_y}{\partial a} \\ \frac{\partial f_\theta}{\partial \dot{\theta}} & \frac{\partial f_\theta}{\partial a} \\ \frac{\partial f_v}{\partial \dot{\theta}} & \frac{\partial f_v}{\partial a} \end{bmatrix} \tag{35}$$

In the turning condition, the equations are

$$\frac{\partial f_x}{\partial \dot{\theta}} = -\frac{v+a}{2\dot{\theta}^2}(cos(\theta+\dot{\theta}) - cos\theta) - \frac{v+a}{2\dot{\theta}} sin(\theta+\dot{\theta}) \tag{36}$$

$$\frac{\partial f_x}{\partial a} = \frac{1}{2\dot{\theta}}(cos(\theta+\dot{\theta}) - cos\theta) \tag{37}$$

$$\frac{\partial f_y}{\partial \dot{\theta}} = -\frac{v+a}{2\dot{\theta}^2}(sin(\theta+\dot{\theta}) - sin\theta) + \frac{v+a}{2\dot{\theta}} cos(\theta+\dot{\theta}) \tag{38}$$

$$\frac{\partial f_y}{\partial a} = \frac{1}{2\dot{\theta}}(sin(\theta+\dot{\theta}) - sin\theta) \tag{39}$$

$$\frac{\partial f_\theta}{\partial \dot{\theta}} = 1 \tag{40}$$

$$\frac{\partial f_\theta}{\partial a} = 0 \tag{41}$$

$$\frac{\partial f_v}{\partial \dot{\theta}} = 0 \tag{42}$$

$$\frac{\partial f_v}{\partial a} = 1 \tag{43}$$

In a straight line,

$$\frac{\partial f_x}{\partial \dot{\theta}} = 0 \tag{44}$$
$$\frac{\partial f_x}{\partial a} = \frac{-cos\theta}{2} \tag{45}$$
$$\frac{\partial f_y}{\partial \dot{\theta}} = 0 \tag{46}$$
$$\frac{\partial f_y}{\partial a} = \frac{sin\theta}{2} \tag{47}$$
$$\frac{\partial f_\theta}{\partial \dot{\theta}} = 0 \tag{48}$$
$$\frac{\partial f_\theta}{\partial a} = 1 \tag{49}$$
$$\frac{\partial f_v}{\partial \dot{\theta}} = 1 \tag{50}$$
$$\frac{\partial f_v}{\partial a} = 0 \tag{51}$$

w is a time invariant value.

6 Estimate state by input

Similarly, given left and right inputs, we could define the motor speed as approximately

$$V_L = \gamma_{i,L} * V_{max} \tag{52}$$
$$V_R = \gamma_{i,R} * V_{max} \tag{53}$$

where V_{max} is some function of the maximum RPM found in the specs, and γ_i are ratios of inputs over the maximum allowable value. Suppose that turning left is the positive angle, and let r be the turn radius of the left wheel and w be the width of the robot, then

$$\begin{aligned} \frac{V_L}{r} &= \frac{V_R}{r+w} \\ \frac{r+w}{r} &= \frac{V_R}{V_L} \\ \frac{w}{r} &= \frac{V_R - V_L}{V_L} \\ r &= \frac{V_L w}{V_R - V_L} \end{aligned} \tag{54}$$

The radius from the center of the robot is

$$\begin{aligned} r* &= w\left(\frac{V_R}{V_R - V_L} + \frac{1}{2}\right) \\ &= w\left(\frac{V_L + 0.5(V_R - V_L)}{V_R - V_L}\right) \\ &= \frac{w}{2}\left(\frac{V_R + V_L}{V_R - V_L}\right) \end{aligned} \tag{55}$$

Since arc length at the center of the robot is

$$\alpha = \frac{V_R + V_L}{2} \tag{56}$$

the change in angle is

$$\begin{aligned} \dot{\theta} &= \frac{\alpha}{r*} \\ &= \frac{V_R - V_L}{w} \end{aligned} \tag{57}$$

With r, α, θ, $\dot{\theta}$, x, and y, we proceed with the rest of the calculations as the same as before, by finding the center and then rotating about it using the rotation and translation matrices.

7 Estimate covariance

Covariance estimate is

$$P_t = AP_{t-1}A + B\Gamma_{t-1}B + Q \tag{58}$$

Since the actual state is unknown, the state covariance matrix is initialized as identity,

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{59}$$

Input covariance is also unknown, so we scale the input covariance matrix by the maximum value of input so that its relative weight is accurate. For the gyro, maximum rpm is about 100, with some load assumed; therefore, given that axle length is $8.4cm$ and wheel circumference is $\pi * 4.9cm$, it can make a full circle in a ms. $\dot{\theta}_{max} \approx 0.001 * (\text{period})$. Given $100rpm$, the

maximum speed is about $0.0257 cm/ms$. The acceleration can be limited by the control algorithm. We can assume that the maximum is a tenth, making $\dot{v}_{max} \approx 0.00257 * (\text{period})$.

$$\Gamma = \begin{bmatrix} 0.00101 & 0 \\ 0 & 0.00257 \end{bmatrix} \tag{60}$$

8 Innovation

In the innovation step, we measure the actual state such that

$$z_t = h(\hat{x}_t, v_t) + y \tag{61}$$

where y is the residual.

v_t depends on the variances of the sensors. h depends on the wall that the laser sensors projects on. Using the estimated state $\hat{x}_t$, we calculate for the wall of shortest distance by the quadrant under which the sensors face. The quadrant was found to be

$$\begin{aligned} p &= \text{floor}\left[\frac{\theta}{\pi/4}\right] + 2 \\ map &= [4, 1, 2, 3] \\ q &= p[map] \end{aligned} \tag{62}$$

Depending on the wall the sensors project on, we can determine either x or y. Therefore, if the two sensors project on adjacent walls, a position is absolutely determinable. Otherwise, we need to fuse it with the a projection from the estimated state. Let f and r be the distance measurements, and h and w be height and width respectively. Then, for the front sensor

wall	position	$\delta\theta$
front	$(fsin\theta$, ? $)$	$(fcos\theta$, ? $)$
top	$(?, h - fcos\theta)$	$(?, fsin\theta)$
left	$(w + fsin\theta$, ? $)$	$(fcos\theta$, ? $)$
bottom	$(?, -fcos\theta)$	$(?, fsin\theta)$

Table 2: Positions determined by front sensor.

For the right sensor

wall	position	$\delta\theta$
front	$(-rcos\theta, ?)$	$(rsin\theta, ?)$
top	$(?, h - fsin\theta)$	$(?, -rcos\theta)$
left	$(w - rcos\theta, ?)$	$(rsin\theta, ?)$
bottom	$(?, -fsin\theta)$	$(?, -rcos\theta)$

Table 3: Positions determined by right sensor

We take an average measured by the two sensors, if the axis repeats, and fill in the unknown values by projecting from the estimated state.

9 Kalman gain

To complete the update step, we must find the Kalman gain. First, we find the residual y by computing the differences between the measured and estimated states. Then, the steps are

$$S = HPH^T + R \tag{63}$$

$$K = \frac{PH^T}{S} \tag{64}$$

$$x = \hat{x} + Ky \tag{65}$$

$$P_t = (I - KH)P_{t-1} \tag{66}$$

10 Finding initial conditions

While it is possible to place the robot in a predetermined location at each trial, this introduces human error and makes the system less robust. It is useful to have a procedure in which the robot could determine initial conditions. We use a *variation of angles* method to calibrate. The steps are

1. We measure the initial distance from the front sensor and angle.
2. We turn by some small known $\delta\theta$.
3. We measure a second distance f_2 from the front sensor and angle θ_2.
4. We calculate the least angle along the current wall and go back to it.

5. We turn 180deg and measure the opposite site.

By calibrating as such, we could also estimate observation noise, and angle at shortest distances to determine the box size, and box orientation from magnetic North.

Calculation involves basic trigonometric facts. We assume in most circumstances, turning a small angle allows the robot to project on the same wall. Therefore, let $\delta\theta = \theta_2 - \theta_1$ as measured by the magnetometer. Using the law of cosines,

$$x^2 = f_1^2 + f_2^2 - 2f_1 f_2 cos\delta\theta \tag{67}$$

where x denotes the length of wall between the two angles. Now, using the law of sines, we note that

$$\frac{x}{sin\delta\theta} = \frac{f_1}{\beta} \tag{68}$$

where β is the angle $\angle_{f_2,wall}$. Note that this angle must form a right angle between the point where the second laser projects on the wall, the robot, and the point of least distance, such that the ideal angle

$$\hat{\theta} = |90 - \beta| \tag{69}$$

The sign of $\hat{\theta}$ determines whether the robot has to travel back or forward to find this angle. We measure $\hat{f}_1$ and $\hat{r_1}$ at $\hat{\theta}$, and $\hat{f}_2$ and $\hat{r_2}$ at $\theta \hat{+} \pi$, such that the width of the box is $\hat{f}_1 + \hat{f}_2 + 2d$, where d is the distance between the front sensor and the center of the robot axle. The complementary angles are the same, except with the right sensor.

11 Motion Planning

With a decent state representation, we send the robot a state and target value, and the robot performs a *rapidly exploring random tree* algorithm to find the target. An x-y plot of the robot converging on the target is shown

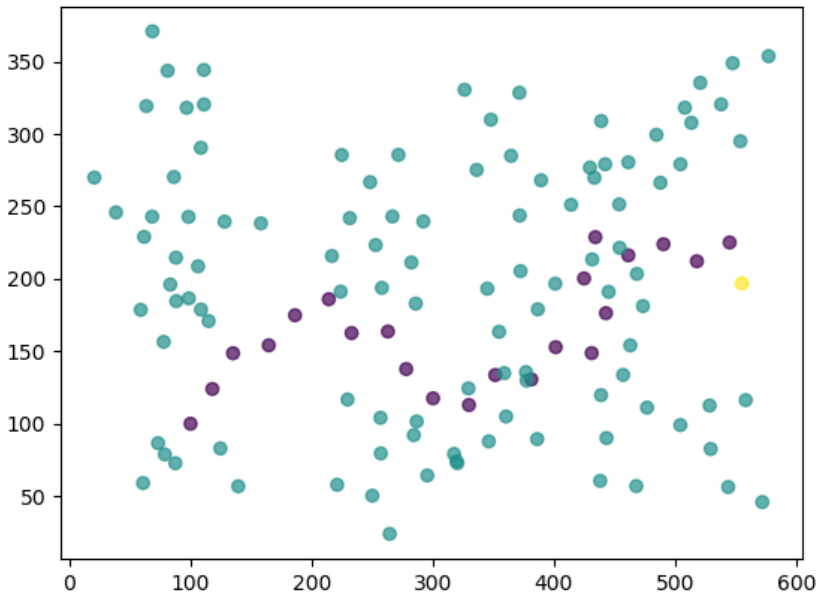

Figure 3: Rapidly-exploring random tree convergence.

The algorithm works by generating a random configuration tree, then selecting vertices that are incrementally closer to the target.

12 Code

For complete code, see

https://github.com/jachang820

In particular, note these directories:

Model, simulation

Code on the robot