ECE 183DA & **MAE 162D**

Joint Lab Assignment 1

Team Speech
Jennifer Bui, Ryan Nemiroff, Andrew Lucchesi,
Cheryl Lee, Laurens Hasell, & Megan Williams

01/22/2021

1 System Overview

The objectives of this lab was to characterize and simulate a simple instrumented two-wheeled robot as shown in Figure 1 where this simple robot could first be extended to a Paperbot, and then to a more complex segway.

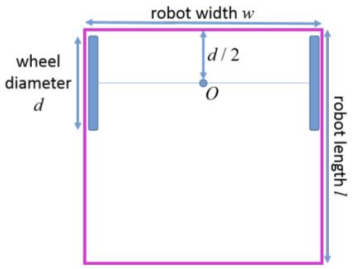


Figure 1: Illustration of the simple robot

Here the robot is equipped with two wheels independently powered by a continuous rotation servo as well as two laser range sensors and an inertial measurement unit (IMU).

2 The Mathematical Formulation

2.1 Defining the Variables

We will consider the origin of our system to be in the North-East corner of the room. From this, we define X as the distance East from the origin to the center of the robot axle and we define Y as the distance North from the origin to the center of the robot axle (Therefore, X and Y will always be negative in this coordinate system).

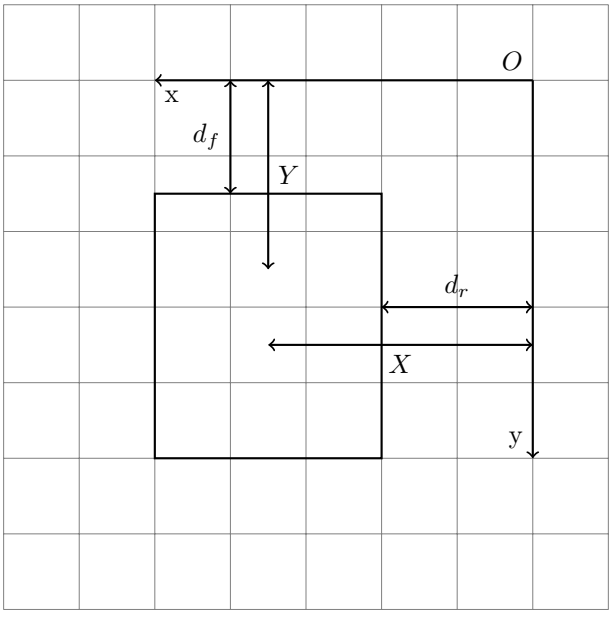


Figure 2: Schematic of the reference coordinate axes with respect to the robot's state.

2.1.1 The Input

The input is defined to be the PWM inputs to the left and right motors, where PWM values are normalized to the range $[-1, 1]$:

$$u_t = \begin{bmatrix} \text{PWM}_l \\ \text{PWM}_r \end{bmatrix}$$

Then, the angular speed of each motor is defined as a function of the PWM input:

$$\omega_l = g(\text{PWM}_l)$$
$$\omega_r = g(\text{PWM}_r)$$

2.1.2 The Output

The sensor noise is defined as an arbitrary length-5 vector, where each value specifies the noise for each sensor measurement:

$$v_t = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix}$$

The following "true" values are defined to help in defining the output:

1. d_r : distance to a wall to the right of the robot
2. d_f : distance to a wall in front of the robot
3. $\dot{\theta}$: in-plane angular velocity
4. b_x : x-axis (lateral) magnetic field strength
5. b_y : y-axis (longitudinal) magnetic field strength

Finally, the output is defined, where the tilde denotes the sensor measurement with noise included:

$$z_t = \begin{bmatrix} \tilde{d}_r \\ \tilde{d}_f \\ \tilde{\dot{\theta}} \\ \tilde{b}_x \\ \tilde{b}_y \end{bmatrix} = \begin{bmatrix} d_r \\ d_f \\ \dot{\theta} \\ b_x \\ b_y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix}$$

2.1.3 The State

The state is defined as such:

1. x : x position of the robot
2. y : y position of the robot
3. θ : in-plane angle

$$x_t = \begin{bmatrix} \text{x} \\ \text{y} \\ \theta \end{bmatrix}$$

There will also be noise η for each state transition to account for actuator errors and real-world imperfections. This noise is *not* pure additive noise, as seen later.

$$\eta_t = \begin{bmatrix} \eta_{\text{x}} \\ \eta_{\text{y}} \\ \eta_\theta \end{bmatrix}$$

In order to determine the state transition equations, the following scenario of robot motion is considered.

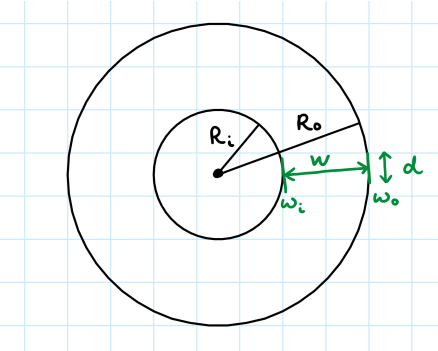


Figure 3: Schematic of the robot moving in a circle

Figure 3 depicts the relationship between angular velocities and turning radius as shown in Equations 1 and 2. R_i and R_o are the respective inside and outside radii, whereas ω_i and ω_o are the respective angular velocities. Here w, is the width between the two wheels as defined in Figure 1.

$$\frac{R_o}{R_i} = \frac{\omega_o}{\omega_i} \tag{1}$$

$$R_o - R_i = \mathrm{w} \tag{2}$$

We can write R_o as:

$$R_o = R_i \frac{\omega_o}{\omega_i}$$

Then derive an expression for R_i by substituting into Equation 1.

$$R_i \frac{\omega_o}{\omega_i} - R_i = \mathrm{w}$$

$$R_i \left(\frac{\omega_o}{\omega_i} - 1 \right) = \mathrm{w}$$

$$R_i = \frac{\mathrm{w}}{\left(\frac{\omega_o}{\omega_i} - 1 \right)} \tag{3}$$

Then, the in-plane angular velocity of the robot (in radians) is:

$$\dot{\theta} = \frac{\frac{d}{2}\omega_i}{R_i} = \frac{\frac{d}{2}\omega_i \left(\frac{\omega_o}{\omega_i} - 1 \right)}{\mathrm{w}}$$

$$\dot{\theta} = \frac{\frac{d}{2}\left(\omega_o - \omega_i\right)}{\mathrm{w}} \tag{4}$$

Finally, this equation can be generalized to any situation; if the outside wheel is the right wheel and the inside wheel is the left, we find that $\dot{\theta}$ is positive (the robot turns counterclockwise) when $\omega_r > \omega_l$. If follows that Equation 5 will work in all situations and produce the correct sign:

$$\dot{\theta} = \frac{\frac{d}{2}\left(\omega_r - \omega_l\right)}{\mathrm{w}} \tag{5}$$

In order to write translation equations for the position of the robot, it is useful to define the velocity of the center of the axle of the robot (where the position coordinates are defined). This is simply the average of the wheel velocities:

$$v_{\text{center}} = \frac{\left(\frac{d}{2}\omega_l + \frac{d}{2}\omega_r\right)}{2} \tag{6}$$

At this point, Δt is defined as the sample period of this discrete system. This allows the definition of θ in terms of $\dot{\theta}$, with the effect of the noise also included. The reasoning for this noise model is explained in Section 6.

$$\theta_{t+1} = \theta_t + (\dot{\theta}_t \Delta t)(1 + \eta_{\theta,t}) \tag{7}$$

If the discrete-time steps are sufficiently short, the robot motion can be linearly approximated for each time step:

$$\mathrm{x}_{t+1} = \mathrm{x}_t + (v_{\text{center}} \cos(\theta_t)\Delta t)(1 + \eta_{\mathrm{x},t}) \tag{8}$$

$$\mathrm{y}_{t+1} = \mathrm{y}_t + (v_{\text{center}} \sin(\theta_t)\Delta t)(1 + \eta_{\mathrm{y},t}) \tag{9}$$

2.2 State Transition and Output

From combining all these equations, the following relationships are derived:

$$\mathrm{x}_{t+1} = \mathrm{x}_t + \left(\frac{\frac{d}{2}\left[g(\mathrm{PWM}_l) + g(\mathrm{PWM}_r)\right]}{2} \cos(\theta_t)\Delta t \right)(1 + \eta_{\mathrm{x},t}) \tag{10}$$

$$\mathrm{y}_{t+1} = \mathrm{y}_t + \left(\frac{\frac{d}{2}\left[g(\mathrm{PWM}_l) + g(\mathrm{PWM}_r)\right]}{2} \sin(\theta_t)\Delta t \right)(1 + \eta_{\mathrm{y},t}) \tag{11}$$

$$\theta_{t+1} = \theta_t + \left(\frac{\frac{d}{2}\left(g(\mathrm{PWM}_r) - g(\mathrm{PWM}_l)\right)}{\mathrm{w}} \Delta t \right)(1 + \eta_{\theta,t}) \tag{12}$$

The general form of the output equations is as follows:

$$\tilde{d}_f = h_1(x_t, u_t, v_{1,t}) \tag{13}$$

$$\tilde{d}_r = h_2(x_t, u_t, v_{2,t}) \tag{14}$$

$$\tilde{\dot{\theta}} = h_3(x_t, u_t, v_{3,t}) \tag{15}$$

$$\tilde{b}_x = h_4(x_t, u_t, v_{4,t}) \tag{16}$$

$$\tilde{b}_y = h_5(x_t, u_t, v_{5,t}) \tag{17}$$

Of these outputs, one that is easy to define explicitly is $\tilde{\dot{\theta}}$:

$$\tilde{\dot{\theta}} = \frac{\frac{d}{2}\left(g(\mathrm{PWM}_r) - g(\mathrm{PWM}_l)\right)}{\mathrm{w}} + v_{3,t} \tag{18}$$

It is also clear that the range sensor and magnetometer measurements will not depend on the current inputs, since they depend only on the current position and orientation which is included in the state:

$$\tilde{d}_f = h_1(x_t, v_{1,t}) \tag{19}$$

$$\tilde{d}_r = h_2(x_t, v_{2,t}) \tag{20}$$

$$\tilde{b}_x = h_4(x_t, v_{4,t}) \tag{21}$$

$$\tilde{b}_y = h_5(x_t, v_{5,t}) \tag{22}$$

3 Range Sensor Model

To model the range sensors, we assume they have an infinitesimally thin line of sight directly in front of them, and that the robot exists in a rectangular room with 4 walls. The eastern and western walls have fixed x positions x_E and x_W, while the northern and southern walls have fixed y positions y_N and y_S.

Based on these assumptions, the angle θ of the robot determines which walls the front and right sensors may

see. For example, if $\frac{\pi}{2} \leq \theta < \pi$, then the right sensor will either see the eastern wall or the northern wall, but not the western wall or the southern wall.

This idea can be extended to all angles $0 \leq \theta < 2\pi$ and both range sensors. For each scenario, the following locations can be determined:

- $(\mathrm{x}_{r,i1}, \mathrm{y}_{r,i1})$: point where the right sensor's line of sight intersects the eastern or western wall.
- $(\mathrm{x}_{r,i2}, \mathrm{y}_{r,i2})$: point where the right sensor's line of sight intersects the northern or southern wall.
- $(\mathrm{x}_{f,i1}, \mathrm{y}_{f,i1})$: point where the front sensor's line of sight intersects the eastern or western wall.
- $(\mathrm{x}_{f,i2}, \mathrm{y}_{f,i2})$: point where the front sensor's line of sight intersects the northern or southern wall.

Obviously, the sensors do not actually see two walls at the same time. The wall that the sensor sees is the one such that the intersection point of it's line of sight is closer to the center of the robot.

To calculate these locations, the geometry of the line of sight of each sensor is expressed as a simple equation for a line, based on the robot's state. For the right sensor:

$$\mathrm{y} - \mathrm{y}_t = (\tan\theta)(\mathrm{x} - \mathrm{x}_t) \tag{23}$$

And likewise for the front sensor:

$$\mathrm{y} - \mathrm{y}_t = -(\cot\theta)(\mathrm{x} - \mathrm{x}_t) \tag{24}$$

Here, x_t and y_t describe the robot's position, while x and y encompass all possible solutions for the line.

By manipulating these equations, the locations where the sensor lines of sight intersect the walls can be calculated. The calculations are different for each 90 degree quadrant that θ can reside in, due to the walls that the sensors may see at those angles.

First, the coordinate values which are equal to the fixed x or y position of a wall are determined. These differ depending on θ:

For $0 \leq \theta < \frac{\pi}{2}$:

- $\mathrm{x}_{r,i1} = \mathrm{x}_E$
- $\mathrm{y}_{r,i2} = \mathrm{y}_S$
- $\mathrm{x}_{f,i1} = \mathrm{x}_E$
- $\mathrm{y}_{f,i2} = \mathrm{y}_N$

For $\frac{\pi}{2} \leq \theta < \pi$:

- $\mathrm{x}_{r,i1} = \mathrm{x}_E$
- $\mathrm{y}_{r,i2} = \mathrm{y}_N$
- $\mathrm{x}_{f,i1} = \mathrm{x}_W$
- $\mathrm{y}_{f,i2} = \mathrm{y}_N$

For $\pi \leq \theta < \frac{3\pi}{2}$:

- $\mathrm{x}_{r,i1} = \mathrm{x}_W$
- $\mathrm{y}_{r,i2} = \mathrm{y}_N$
- $\mathrm{x}_{f,i1} = \mathrm{x}_W$
- $\mathrm{y}_{f,i2} = \mathrm{y}_S$

For $\frac{3\pi}{2} \leq \theta < 2\pi$:

- $\mathrm{x}_{r,i1} = \mathrm{x}_W$
- $\mathrm{y}_{r,i2} = \mathrm{y}_S$
- $\mathrm{x}_{f,i1} = \mathrm{x}_E$
- $\mathrm{y}_{f,i2} = \mathrm{y}_S$

Next, the remaining coordinate values can be calculated by utilizing Equations 23 and 24.

$$\mathrm{y}_{r,i1} = (\tan\theta)(\mathrm{x}_{r,i1} - \mathrm{x}_t) + \mathrm{y}_t \tag{25}$$
$$\mathrm{x}_{r,i2} = (\cot\theta)(\mathrm{y}_{r,i2} - \mathrm{y}_t) + \mathrm{x}_t \tag{26}$$
$$\mathrm{y}_{f,i1} = -(\cot\theta)(\mathrm{x}_{f,i1} - \mathrm{x}_t) + \mathrm{y}_t \tag{27}$$
$$\mathrm{x}_{f,i2} = -(\tan\theta)(\mathrm{y}_{f,i2} - \mathrm{y}_t) + \mathrm{x}_t \tag{28}$$

Finally, the expected measurement of each distance sensor is the shortest Euclidean distance to a line-of-sight intersection point from the robot center, minus the distance the sensor is offset from the robot center (denoted D_r and D_f):

$$d_r = \min\left(\sqrt{(\mathrm{x}_{r,i1} - \mathrm{x}_t)^2 + (\mathrm{y}_{r,i1} - \mathrm{y}_t)^2}, \sqrt{(\mathrm{x}_{r,i2} - \mathrm{x}_t)^2 + (\mathrm{y}_{r,i2} - \mathrm{y}_t)^2}\right) - D_r \tag{29}$$

$$d_f = \min\left(\sqrt{(\mathrm{x}_{f,i1} - \mathrm{x}_t)^2 + (\mathrm{y}_{f,i1} - \mathrm{y}_t)^2}, \sqrt{(\mathrm{x}_{f,i2} - \mathrm{x}_t)^2 + (\mathrm{y}_{f,i2} - \mathrm{y}_t)^2}\right) - D_f \tag{30}$$

4 Magnetometer Model

To model the magnetometer, we will assume that the magnetic field B_E is constant and uniform throughout the room. In this case, the expected measurement of the magnetometer will depend on θ and $||\mathrm{B}_E||$.

$$\mathrm{b}_y = \mathrm{proj}_x \mathrm{B}_E \tag{31}$$
$$\mathrm{b}_x = \mathrm{proj}_y \mathrm{B}_E \tag{32}$$

Note that here, x and y refer to axes of the magnetometer, not the room's axes. We can write this in terms of the θ. We set $\mathrm{b}_y = 0$ for $\theta = 0, \pi$.

$$\mathrm{b}_y = ||\mathrm{B}_E|| \sin(-\theta) \tag{33}$$
$$\mathrm{b}_x = ||\mathrm{B}_E|| \cos(-\theta) \tag{34}$$

5 Servo Model

The FS90R servo data sheet provides a few measures that can be used to develop a non-linear model. First, the maximum speed at no load is 110 RPM (11.5 rad/s) if operating at 4.8V and 130 RPM (13.6 rad/s) if operating at 6V. Additionally, the stall torque is 1.3 kg·cm if operating at 4.8V and 1.5 kg·cm if operating at 6V. Finally, the PWM pulse width range is [700, 2300] µsec, and the motor is designed to stop for pulse widths in the range 1500±45 µsec. A constant voltage is assumed at all times, and 6V is used for the simulation.

The variable PWM is defined to be $\frac{(\text{pulse width in µsec}) - 1500}{2300 - 1500}$, which puts it in the range $[-1, 1]$ such that PWM $= 1$ corresponds to a pulse width of 2300, PWM $= 0$ corresponds to a pulse width of 1500, and PWM $= -1$ corresponds to a pulse width of 700. As a result, the dead zone maps to the range PWM $\in [-0.05625, 0.05625]$.

Some assumptions are made in order to keep the model reasonably simple. First, it is assumed that the no-load motor speed is proportional to PWM when it is not in the dead zone. Next, it is assumed that the stall torque is the torque required to make the motor motionless when it is receiving any PWM value. This is a reasonable assumption because this is what stall torque is intended to specify. Finally, it is assumed that the torque exerted on the motor is directly proportional to the reduction in speed compared to no-load. This behavior is more specific to DC motors, as seen in Figure 4 so it is likely not a perfect representation of the servo motor. The way the servo responds to torque depends on its electromechanical properties, which are not investigated in detail.

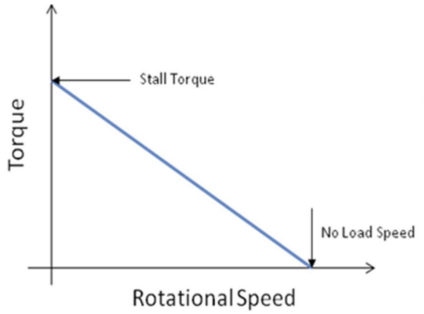


Figure 4: Simple relationship between torque and speed for DC motors. The same relationship is used to approximate the servo model.

The assumptions made lead to the following equation for motor angular speed where, $\omega_{\max}$ is the maximum angular speed under no load, τ is the applied torque, and τ_{stall} is the stall torque.

$$\omega = g(\text{PWM}) = \begin{cases} (\omega_{\max} \cdot \text{PWM})(1 - \frac{\tau}{\tau_{\text{stall}}}) & \text{PWM} \in [-1, -0.05625) \cup (0.05625, 1] \\ 0 \text{ rad/s} & \text{PWM} \in [-0.05625, 0.05625] \end{cases} \tag{35}$$

Some consideration is required for determining what the applied torque would be. Torque is required both when accelerating and when moving at constant speed to fight friction. The model used here is simplified by neglecting acceleration and assuming that the torque required to overcome friction is proportional to speed. To simplify further, the torque is approximated as being proportional to |PWM|. Arbitrarily, the torque required for overcoming friction when $|\text{PWM}| = 1$ is chosen to be $\tau = (0.4 \cdot \tau_{\text{stall}})$; this is simply chosen to demonstrate the model, and a physical estimation may be done at a later time. The final expression for torque is: $\tau = (0.4 \cdot |\text{PWM}| \cdot \tau_{\text{stall}})$

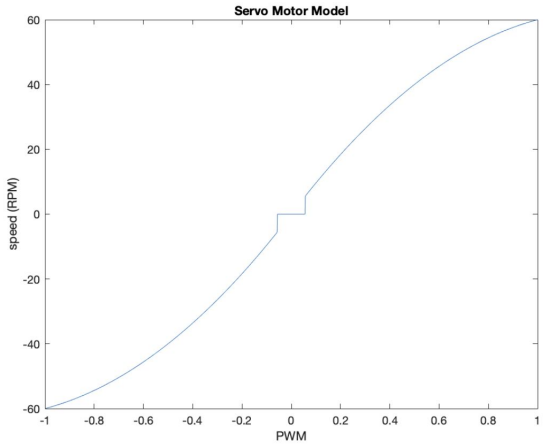


Figure 5: Rotational speed for the full PWM range based on the developed servo motor model.

Figure 5 shows the rotational speed for the full PWM range using defined specification for τ and a maximum no-load speed of 100 RPM. This curve results from the fact that friction increases with speed in the chosen model. The horizontal section in the middle of the graph is due to the dead zone as detailed in the servo specifications. The speed maxes out at 60 RPM since it was chosen that τ maxes out at $0.4 \cdot \tau_{\text{stall}}$.

6 Actuator Noises

As evidenced above, one of the largest sources of uncertainty in actuation is the relationship between PWM and wheel speed. Another source of noise might be wheel slippage, but this probably has a much smaller effect.

Therefore, it would be useful to apply noise to the wheel speeds in the model. This would likely be multiplicative noise, noting that a stopped robot has practically zero chance of moving unintentionally, and faster speeds are likely to result in more uncertainty. If one wanted to incorporate this noise in the dynamics equations, one could simply multiply the angular velocities by the the random noises when they are calculated. Although this is fairly trivial, estimating the characteristics of these noises is hard, as is propagating the covariance.

Instead, since it has not been argued that a complex noise model would benefit the simulation, a very simple one is used instead. Noise is assumed to be a Gaussian random vector with zero mean that is proportional to the expected change in state at each time step. It is also independent at each time step. In other

words, the state-transition noise at each time step, $\eta = \begin{bmatrix}\eta_{\mathrm{x}} & \eta_{\mathrm{y}} & \eta_{\theta}\end{bmatrix}^T$, is Gaussian with covariance matrix H.

Equations 7, 8, and 9 are repeated below to clarify how this noise affects the state. The practical effect is that there is additive noise which is proportional to the expected change in each element of the state.

$$\theta_{t+1} = \theta_t + (\dot{\theta}_t \Delta t)(1 + \eta_{\theta,t})$$

$$\mathrm{x}_{t+1} = \mathrm{x}_t + (v_{\text{center}} \cos(\theta_t)\Delta t)(1 + \eta_{\mathrm{x},t})$$
$$\mathrm{y}_{t+1} = \mathrm{y}_t + (v_{\text{center}} \sin(\theta_t)\Delta t)(1 + \eta_{\mathrm{y},t})$$

Regarding the covariance H, the most accurate covariance might involve interdependent statistics based on the dynamics. For now, however, each element of η_t is assumed to be independent, meaning that H takes the form:

$$H = \begin{bmatrix} \mathrm{var}(\eta_{\mathrm{x}}) & 0 & 0 \\ 0 & \mathrm{var}(\eta_{\mathrm{y}}) & 0 \\ 0 & 0 & \mathrm{var}(\eta_{\theta}) \end{bmatrix}$$

Analysis has not been performed to choose reasonable variances, so for now they are arbitrarily set such that the standard deviation is 10%:

$$\mathrm{var}(\eta_{\mathrm{x}}) = (0.1)^2$$
$$\mathrm{var}(\eta_{\mathrm{y}}) = (0.1)^2$$
$$\mathrm{var}(\eta_{\theta}) = (0.1)^2$$

This noise model is convenient because the next state mean and covariance becomes:

$$\bar{\mathrm{x}}_{t+1} = \mathrm{x}_t + v_{\text{center}} \cos(\theta_t)\Delta t$$

$$\bar{\mathrm{y}}_{t+1} = \mathrm{y}_t + v_{\text{center}} \sin(\theta_t)\Delta t$$
$$\bar{\theta}_{t+1} = \theta_t + \dot{\theta}_t \Delta t$$

$$\mathrm{var}(\mathrm{x}_{t+1}) = (v_{\text{center}} \cos(\theta_t)\Delta t)^2 \cdot \mathrm{var}(\eta_{\mathrm{x},t})$$
$$\mathrm{var}(\mathrm{y}_{t+1}) = (v_{\text{center}} \sin(\theta_t)\Delta t)^2 \cdot \mathrm{var}(\eta_{\mathrm{y},t})$$
$$\mathrm{var}(\theta_{t+1}) = (\dot{\theta}_t \Delta t)^2 \cdot \mathrm{var}(\eta_{\theta,t})$$

7 Sensor Noise

The Lidar Lite v3 laser range sensor data sheet for the segway indicates probable noise due to the nature of the operation, acquiring a measurement when the laser signal is reflected back to the device after hitting an object. There is additive conditional noise due to wall distance, device temperature, etc. that may also affect the laser range performance. The data sheet does indicate a recommended receiver bias correction every 100 measurements due to changing light levels. Because we assumed our environment to be a rectangular room with 4 walls and minimal obstructions, we can set the peak detection threshold bypass to be 0x60, reducing erroneous measurements. All of this taken into consideration, the Lidar Lite v3 estimates accuracy for distances less than 5m to be within ±2.5cm of the true value. For distances greater than 5m, the laser range sensor indicates to be within ±10cm of the true value.

The MicroStrain Sensing 3DM-GX5 IMU data sheet for the segway indicates a noise density of $0.005\,^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}}$ with a bandwidth of 250 Hz for the gyroscope and a noise density of $400\,\mu\mathrm{Gauss}/\sqrt{\mathrm{Hz}}$ for the magnetometer. There is additive noise due to the alignment error on the gyroscope of ±0.08° and the magnetometer of ±0.05°. The gyroscope also has a vibration induced noise $0.072\,^\circ/\mathrm{s}$ RMS/g RMS. There is an initial bias error to consider, ±0.04°/s for the gyroscope and ±0.003Gauss for the magnetometer.

Based on this information, and assuming tolerances represent a 95% confidence interval (2 standard deviations), the output noise can be derived. For simulation purposes, all noises are assumed to be independent at each time step. Even though this is not true (bias and alignment error change very slowly or are constant), it simplifies the model and potential methods for state estimation.

Recall that the output noise v contains one element corresponding to each element of the output, and the additive noise is expressed as such:

$$z_t = \begin{bmatrix} \tilde{d}_r \\ \tilde{d}_f \\ \tilde{\dot{\theta}} \\ \tilde{b}_x \\ \tilde{b}_y \end{bmatrix} = \begin{bmatrix} d_r \\ d_f \\ \dot{\theta} \\ b_x \\ b_y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix}$$

Each element of v is assumed to be a Gaussian random variable with variance determined by the sensor specifications. Taken as a whole, the covariance of the vector v is denoted V:

$$V = \begin{bmatrix} \text{var}(v_1) & 0 & 0 & 0 & 0 \\ 0 & \text{var}(v_2) & 0 & 0 & 0 \\ 0 & 0 & \text{var}(v_3) & 0 & 0 \\ 0 & 0 & 0 & \text{var}(v_4) & 0 \\ 0 & 0 & 0 & 0 & \text{var}(v_5) \end{bmatrix}$$

Calculating the variances from the sensor specs yields:

$$\text{var}(v_1) = \begin{cases} (2.5\text{cm}/2)^2 & d_r < 5\text{m} \\ (10\text{cm}/2)^2 & d_r \geq 5\text{m} \end{cases}$$

$$\text{var}(v_2) = \begin{cases} (2.5\text{cm}/2)^2 & d_f < 5\text{m} \\ (10\text{cm}/2)^2 & d_f \geq 5\text{m} \end{cases}$$

$$\text{var}(v_3) = ((0.005\,^\circ/\text{s}/\sqrt{\text{Hz}})/\sqrt{\Delta t} + 0.04\,^\circ/\text{s})^2$$

$$\text{var}(v_4) = (400\,\mu\text{Gauss}/\sqrt{\text{Hz}}/\sqrt{\Delta\text{t}} + 3000\,\mu\text{Gauss})^2$$

$$\text{var}(v_5) = (400\,\mu\text{Gauss}/\sqrt{\text{Hz}}/\sqrt{\Delta\text{t}} + 3000\,\mu\text{Gauss})^2$$

8 Simulation

From this mathematical model, we created a computational simulation of the two-wheeled robot, represented by a simple blue rectangle. This computational simulation was made to simulate either the Paperbot or a segway with 4 walls and boundaries specified by x_W, x_E, y_N, y_S, representing the west, east, north, and south wall limits respectively.

In the simulation shown in Figure 6 (and in the video linked in the Appendix), the walls are located at the edges of the window. Additionally, the input to the robot is determined by the arrow keys on the keyboard as such:

- No key pressed: Zero input
- Up arrow key: Both motors drive forward with PWM $= 0.5$.
- Down arrow key: Both motors drive backward with PWM $= -0.5$.

- Left arrow key: Left motor drives backward (PWM $= -0.5$) and right motor drives forward (PWM $= 0.5$), causing the robot to spin counterclockwise.
- Right arrow key: Left motor drives forward (PWM $= 0.5$) and right motor drives backward (PWM $= -0.5$), causing the robot to spin clockwise.

The simulation is initialized with the user-specified values and is scaled such that one pixel is 1mm:

- West x_W, East x_E, North y_N, South y_S or the fixed x and y positions of the walls, respectively
- Segway dimensions $d = 502$mm, $w = 530$mm; or Paperbot dimensions $d = 50$mm, $w = 90$mm
- D_r, the right distance sensor offset
- D_f, the front distance sensor offset
- H, the actuator/state transition covariance
- V, the sensor/output covariance
- Δt, the sample period

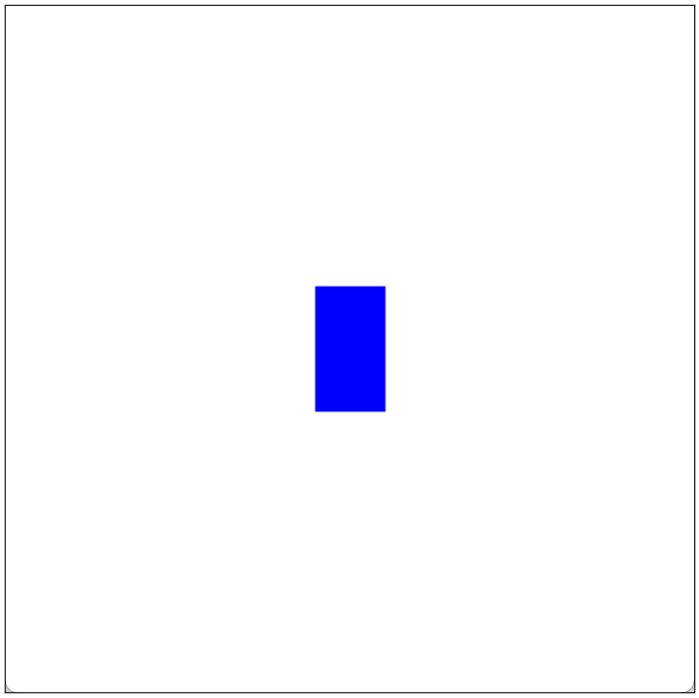
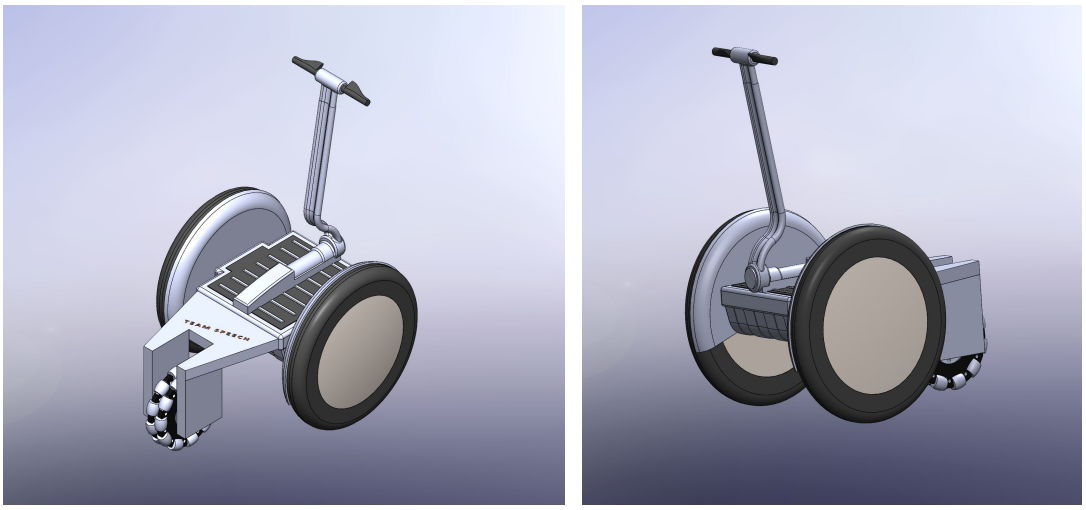


Figure 6: Screenshot of the running simulation of a moving robot

A link to a video of the running simulation is included in the Appendix.

9 CAD Model

This mathematical model can be applied to various two-wheeled robotic systems, such as a segway. By editing a simple, open-source segway assembly in SolidWorks, we created a CAD model of the segway robot, as shown in Figure 7. We will later use SolidWorks to perform physical simulations on this segway robot design, and compare these physical simulation results to the computational simulation results.



(a) (b)

Figure 7: Isometric views of the segway CAD model

10 Conclusions

All in all, this developed parameterized framework can be broadened and built upon in order to apply it to larger scopes. It has demonstrated the repeatable engineering process for prototyping a system and will later be validated by further simulation and analysis.

A Appendix

A.1 Running Simulation

- Video of Running Simulation

A.2 Code Documentation

- Git Repository

A.3 Datasheets for Sensors

- Segway
  - Lidar Lite v3 Proximity Sensor
  - MicroStrain Sensing 3DM-GX5 IMU
- Paperbot
  - FS90R Continuous Rotation Servo
  - GYVL53L0X Proximity Sensor
  - MPU-9250 Gyro

A.4 References

- The following link was used to model the servo as it was treated as a simple DC motor. https://www.motioncontroltips.com/torque-equation/