Welcome to Module 3: Introduction to Mobile Robotics Sensory Motor Loops, Motion of Rigid Bodies

Quantitative Engineering Analysis

Spring 2019

1 Schedule

- 0900-1030: Sensory Motor Loops and Braitenberg Vehicles
- 1030-1045: Coffee
- 1045-1200: Motion of Rigid Bodies
- 1200-1230: Overnight preview and Mathematica refresher

2 Overview

Welcome to Module 3! In this module you'll be learning some of the fundamental ideas, concepts, and algorithms that lie at the heart of robotics. Along the way we'll be revisiting some mathematical and analytical concepts we touched upon earlier in the semester. Not only will we be applying these concepts in new contexts and to new purposes, but also extending them in important ways. The module is structured around a series of challenges in which you will be programming your robot to perform various tasks autonomously. As you and your robot face tougher and tougher challenges, you will need to carefully integrate a wider range of techniques in order to successfully complete the task at hand.

3 What is a Robot?

Before diving into the challenges, let's take a step back and look at some definitions of the word "robot". Merriam-Webster provides three definitions of the word.

- a machine that looks like a human being and performs various complex acts (such as walking or talking) of a human being
- a device that automatically performs complicated often repetitive tasks
- a mechanism guided by automatic controls

Exercise (1) Jot down a list of devices. For each device, determine which, if any, of the three definitions of the word "robot" apply.

Figure 1: C-3PO from the Star Wars franchise.

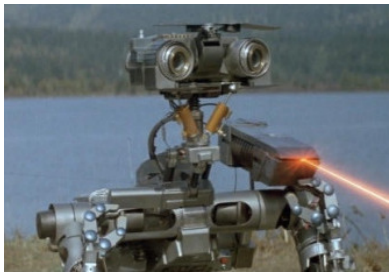

Figure 2: Johnny Five from the Short Circuit movies.

These disparate definitions highlight the fact that depending on who you ask, the answer to the question "what is a robot?" will likely be very different. In a sense these three definitions proceed along a continuum of more restrictive to looser definitions, with the definition "a mechanism guided by automatic controls" being the loosest. Under this definition many things that you probably wouldn't intuitively call "robots" are just that. Take for example a thermostat. A thermostat is a mechanism that automatically regulates the heat in a building by comparing the measured temperature with a "desired" temperature. By the third definition, a thermostat is certainly a robot. At this point you may be thinking that if something as simple as a thermostat is a robot, then definition 3 must be completely bogus. After all, robots are supposed to be complicated and hard! However, today we will see that robots can in fact be quite simple. Further, simple robots can do some pretty complicated things.

4 Sensory-Motor Loops

We ended our last section on a somewhat cryptic note. If we seek to design simple robots, how on earth can they do complex things? The answer to this question lies in the fact that robots are not isolated machines, but instead interact with a complex and ever-changing world. A simple model that captures this idea is the sensory-motor loop (see Figure **??**). The model situates the robot within an environment that it interacts with through two pathways.

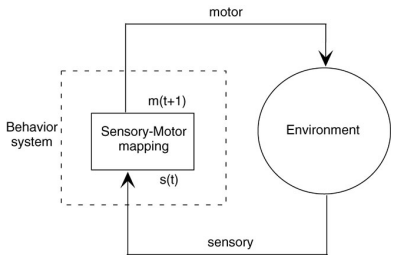

Figure 3: A schematic of a robot controlled by a sensory-motor loop.

The first is a **motor pathway** by which the robot executes actions which affect its environment. In our thermostat example these actions would be turning the heating system on or off. In a more conventional example of a robotic arm in a manufacturing plant, this could be operating the motors that control the joints of the arm.

The second is a **sensory pathway** by which the robot perceives its environment. In our thermostat example this could be a temperature sensor such as a thermocouple. In the case of a robotic arm in a manufacturing plant, this could be a potentiometer that measures the angle of each of the arm's joints or pressure sensors that measure contacts between the robot arm and other objects.

The "brain" of the robot, if you will, is defined by the box labeled "behavior system". In the general case you could imagine that the robot's brain might integrate multiple pieces of sensory information over time to form representations of the world around it. Take for example a robot mapping a building. The robot could build a progressively more detailed map by moving around in the building and collecting sonar readings (which provide an estimate of distance to objects in the world) over time. Putting aside this more complex

form, let's restrict ourselves to robots with fairly simple behavior systems. **What about a robot that has no memory at all?** Such a robot would have to make all of its decisions based on its current sensory information.

Exercise (2) Design robots using the model in Figure **??**. Restrict yourselves to robots that have no memory (i.e. ones that act at any moment in time directly based on their sensory input). To help get your creative juices flowing, it may help to make lists of sensors and actuators. You can then create interesting ideas by seeing what would happen if you paired a particular sensor with a particular actuator in a particular context. Don't worry too much about trying to design useful robots (whimsical is good), the goal here is to be creative and to think through the mental simulations necessary to understand how your robot would behave. Here are some suggestions for sensors and actuators.

Sensors: vibration sensor, microphone, camera, thermal camera, wheel rotation sensor, pressure sensor, light intensity sensor, laser range sensor, bump detector, temperature sensor, breathalyzer, etc. (Wikipedia has a good list).

Actuators: DC motors, combustion engines, stepper motors, solenoids, speakers, laser beams, LEDs, etc.

5 *Grey Walter's Tortoises*

Two very early examples of electric robots that worked using the principle of sensory-motor mappings were Grey Walter's robotic "Tortoises" Elmer and Elsie (see Figure **??**). This YouTube video probably tells the story better than we possibly could.

As Grey Walter said himself, the robots behave as if they had a very simple two-cell nervous system that specifies the sensory-motor mapping (or behavior system). Despite this striking simplicity, the robots are capable of complex behavior such as obstacle avoidance and phototaxis (navigating towards the light that marks the charging kennel). This is an example of what we've been alluding to several times in this document: simple sensory-motor mappings can lead to complex behavior when put into a complex environment.

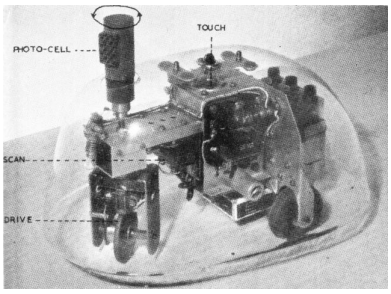

Figure 4: Gray Walter's Tortoise Elsie

6 *Braitenberg Vehicles*

The pioneering work of Grey Walter was followed up by a number of others. One particularly interesting work was by Valentino Braitenberg. Valentino Braitenberg was interested in how vehicles controlled by very simple sensory-motor loops could execute behaviors that

when viewed by humans would cause them to ascribe emotion and feelings of intelligence and intentionality to these vehicles. The name typically used to refer to these hypothetical robots is "Braitenberg Vehicles". While Braitenberg never actually built these vehicles (he was more interested in how these simple vehicles might inform various philosophical issues, particularly in the area of philosophy of mind), others have followed up and actually built these vehicles. Here is a video from a group at MIT that built several of Braitenberg's vehicles.

A schematic of the vehicle (or robot) in the video is shown in Figure **??**. The robot has two motors that each control one of the robot's wheels. We can use the symbols V_L and V_R to refer to the velocities of each of the wheels (positive indicating a forward velocity). Using the framework from the previous section, these are our actuators. The robot also has a number of sensors. A light sensor outputs a continuous value which reads out larger values when in the presence of bright light and smaller values in the presence of low light (this is basically a one-pixel camera!). Additionally the robot has two bump sensors that have binary outputs. That is, they output 1 when they strike an object and 0 otherwise. Finally, the robot has a whisker sensor that is also binary and outputs a 1 when it contacts something and 0 otherwise.

Before getting on to the task of figuring out how one might program a robot like this, we need to understand a bit about how the drive systems of these robots work. The configuration shown in Figure **??** is known as *differential drive*. We will be thinking much more systematically about differential drive in the first robot challenge, but for now let's try to understand it from a qualitative perspective.

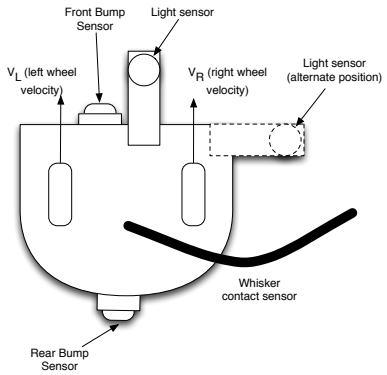

Figure 5: A schematic of the vehicle from the real-life Braitenberg vehicles video. The robot is a differential drive vehicle with two wheels. We use V_L and V_R to refer to the velocities of the left and right wheels (positive is forward by convention). Also labeled are the other sensors on the robot, including the light sensor that reads higher values when exposed to more light, a whisker sensor that reads 0 when it is not in contact with something and 1 if it is, and two bump sensors that read 1 when they hit something and 0 otherwise.

Exercise (3) To build a qualitative understanding of differential drive it helps to understand a few limiting cases. Good ones to start with are ones that involve the wheels moving at equal speeds in either the forward (positive) or reverse (negative) direction. In these cases the robot will either move forward in a straight line or backwards in a straight line. In these cases the speed of the robot is directly proportional to the speed of its wheels. Now, let's consider cases where the velocities of the two wheels are unequal. To help you with your intuition it might help to imagine the right wheel pulling either forwards or backwards on the right side of the robot and the left wheel pulling either forwards or backwards on the left side of the robot. Here is a potential list of limiting cases to consider. Make predictions about what would happen in these cases. It may help to sketch a couple of key frames (poses of the robot) over time.

(a) What if V_L is positive and $V_R = -V_L$?

(b) What if V_R is positive and $V_L = -V_R$?
(c) What if $V_L = 0$ and V_R is positive?
(d) What if $V_R = 0$ and V_L is positive?
(e) What if $V_L = 0$ and V_R is negative?
(f) What if $V_R = 0$ and V_L is negative?
(g) What if V_R is positive and $V_L = \frac{1}{2}V_R$?
(h) What if V_L is positive and $V_R = \frac{1}{2}V_L$?

7 *Programming a Robot on a Whiteboard*

Next, you will be programming this vehicle to perform the behaviors you saw in the video. However, instead of programming the robot using a computer, you will be programming it at the whiteboard. How can you program using a whiteboard (we imagine you conveniently might ask)?!? Remember, we are thinking of our robot's brain as a sensory-motor mapping. Translated into the language of mathematics this simply means that our robot program is specified by a function from sensors to motors!

There are many ways to represent a function. One way is as an equation. For instance, a robot that moves faster and faster as it see more and more light could have $V_L(\ell) = \ell^2$ and $V_R(\ell) = \ell^2$ (where ℓ is the intensity of the light measured by the light sensor). Another way to represent a function is to define it graphically. You could draw a function that has ℓ on the x-axis and V_L on the y-axis. You could then sketch the relationship between those two quantities. Doing this graphically you could either have a quantitatively accurate sketch or a sketch that simply characterizes the qualitative behavior of the function. For sensors that have binary values, like the bump sensors, you can exhaustively enumerate all conditions. For instance, here is the program of a robot that drives forward at a $0.5\frac{m}{s}$ until it rams into something

$$V_L(bump_F) = \begin{cases} 0.5\frac{m}{s}, & bump_F = 0 \\ 0\frac{m}{s}, & bump_F = 1 \end{cases}$$

$$V_R(bump_F) = \begin{cases} 0.5\frac{m}{s}, & bump_F = 0 \\ 0\frac{m}{s}, & bump_F = 1 \end{cases}$$

where $bump_F$ is the value of the forward bump sensor (1 when in contact with something, 0 otherwise).

Exercise (4) Work through generating robot programs to realize the behaviors in the video. Questions and considerations to keep in mind while doing this activity are:

(a) In order to validate your proposed program, it helps to do a quick whiteboard simulation. Sketch out a few key instants in time, what the robot's sensors would read, and what the wheel velocities would be.

(b) At least one of the behaviors cannot be reproduced without some primitive form of memory (although perhaps if you are very creative it can work). Which behaviors are these? How can you tell?

Next, Imagine your robot can remember a small amount of information. Specifically, your robot has access to a single flag that starts out with the value 0. Its value can be toggled from 0 to 1 or from 1 to 0 when a particular event occurs. For instance, if the light sensor reads a certain value, the robot might toggle its flag. The value of the flag can then inform the behavior of the robot. Given this new capability, implement any behaviors in the video that you couldn't before.

If you get done earlier than other groups, consider adding an additional light sensor to your robot. Now that you have two light sensors, you can get the robot to do a richer set of behaviors. Sketch the configuration of sensors on your new Braitenberg vehicle (equipped with two light sensors). Try to reproduce behaviors such as light seeking and light avoiding. For more ideas see the Wikipedia page on Braitenberg vehicles.

8 Coffee [15 minutes]

9 The Motion of Rigid Bodies [75 mins]

We are going to explore the concept of angular velocity using an "app" for your phone. Please download and install "SensorKinetics" to your phone, and open the "Gyroscope sensor".

Exercise (5) **Qualitative**: For each of the questions below, you should "plot" the data on the board, and interpret the data qualitatively in terms of the coordinate system of the phone.

(a) Place your phone on the table and spin it, in place, counterclockwise. Note that you should be spinning it about an axis that is orthogonal to the face of the phone. Which axis is this on the data graph? What happens if you spin it clockwise?

(b) Now spin the phone about the two other axes. Which axis is which? Which direction is clockwise and which is counterclockwise? Clearly draw the coordinate system the phone is using - use the unit vectors $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$.

(c) What happens if you move the phone along a straight line (on the table) without turning the phone?

(d) What happens if you move the phone uniformly in a circle **without turning the phone**? i.e. the orientation of the phone does not change.

(e) What happens if you move the phone uniformly in a circle **while turning the phone at the same time**? i.e. imagine the phone is a car and you are driving in a circle.

Now that we've explored angular velocity using the phone, let's make it quantitative. We will use the following notation for the angular velocity

$$\boldsymbol{\omega} = \omega_x \hat{\mathbf{x}} + \omega_y \hat{\mathbf{y}} + \omega_z \hat{\mathbf{z}}$$

so that ω_x is the component of angular velocity corresponding to rotation about the x-axis and so on. The units of angular velocity are in radians per second, e.g., rotating in a complete circle in one second would be a 2π radian/second rotation.

Exercise (6) **Quantitative**: For each of the questions below, you should "plot" the data on the board, and interpret the data quantitatively in terms of the coordinate system of the phone.

(a) Predict the angular velocity if you place the phone down on the table and then spin the phone in place so that it spins once in 5 seconds. Confirm your prediction using the data.

(b) Predict the angular velocity if you uniformly move the phone in a circle in 5 seconds, and confirm your prediction using the data. Does it matter how large the circle is?

(c) What type of motion would give rise to a constant ω_z of 2 radians per second for 5 seconds, with both $\omega_x = 0$ and $\omega_y = 0$? Confirm your prediction with the phone.

(d) What type of motion would give rise to a sinusoidal ω_z with amplitude of 2 radians per second, and a period of 10 seconds, with both $\omega_x = 0$ and $\omega_y = 0$.

(e) What type of motion would give rise to a constant ω_z and ω_x of 2 radians per second for 5 seconds, with $\omega_y = 0$. Confirm your prediction with your phone.

(f) Sketch a graph of your own choosing of ω_x, ω_y, and ω_z and challenge yourself to produce it using the phone!

Hopefully we have a qualitative and quantitative sense for angular velocity now. The angular velocity is a vector with magnitude and direction. We've seen that the direction is the axis of rotation (and

counterclockwise is positive), and the magnitude is defined as the rate of change of the angle in rad/s. For example, if we use θ_x to represent the angle of rotation about the x-axis, then the x-component of the angular velocity is

$$\omega_x = \frac{d\theta_x}{dt}$$

Although the literature tends to use different Greek letters to represent the different angles of rotation in 3D, we will use θ_x, θ_y, and θ_z for simplicity.

Since the angular velocity is the time derivative of the angle, the angle must therefore be determined by the integral of the angular velocity

$$\theta_x(t) = \theta_x(0) + \int_0^t \omega_x(t)\, dt$$

where $\theta_x(0)$ is the initial angle at $t = 0$. In the case where the angular velocity is constant the integral gives

$$\theta_x(t) = \theta_x(0) + \omega_x t$$

which means that the angle increases linearity in time as expected - that rate of increase is just the angular velocity. **If the angular velocity is not constant, we need to integrate it in time to determine the angle.**

Exercise (7) **Mathematical:** For each of the following questions you should carry out the relevant integral, and think about the motion of the phone.

(a) Determine $\theta_z(t)$ if $\omega_z(t) = 2$ radians per second. Describe the motion of the phone in this case.

(b) Determine $\theta_z(t)$ if $\omega_z(t) = \alpha t$ radians per second. Describe the motion of the phone in this case.

(c) Determine $\theta_z(t)$ if $\omega_z(t) = \sin(\alpha t)$ radians per second. Describe the motion of the phone in this case.

We will finish this exploration by connecting back to rotation matrices from earlier in this module. We will now use MATLAB to define a set of unit vectors corresponding to the coordinate system of the phone, and we will use rotation matrices to rotate these unit vectors.

Exercise (8) Look up and write down the rotation matrices for 3D rotations - use θ_x, θ_y, and θ_z for the angles.

Exercise (9) In MATLAB, define a set of unit vectors $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ in 3D. For example,

```
>> xhat = [1;0;0]
```

Use "quiver3" to plot these unit vectors in 3D with the origin at $(0,0,0)$. Now choose a specific rotation angle about each of the axes in turn, transform the vectors by multiplying with the rotation matrix, and use "quiver3" to visualize the new vectors to confirm the rotation matrix does what you expected.

Exercise (10) Now write a "for" loop that rotates the unit vectors in 3D for the following scenarios so that you can produce an animation - you will need to issue the "drawnow" command so that the graphic updates every time you issue the quiver3 command.

(a) $\omega_z = 2$ radians per second, and $\omega_x = \omega_y = 0$.

(b) $\omega_x = 2$ radians per second, and $\omega_y = \omega_z = 0$.

(c) $\omega_y = 2$ radians per second, and $\omega_x = \omega_z = 0$.

(d) $\omega_z = \omega_x = 2$ radians per second, and $\omega_y = 0$.

(e) $\omega_z = 2t$ radians per second, and $\omega_x = \omega_y = 0$.

(f) $\omega_z = \sin(2t)$ radians per second, and $\omega_x = \omega_y = 0$.

Exercise (11) **Challenge (if you have time)** Record some data with the phone, upload it to MATLAB, and try to produce an animation of the unit vectors identical to the motion of the phone.

10 *Overnight preview and Mathematica refresher [30 mins]*