Designing Robot Behaviors

Week #6

Prof. Ryan Kastner

Designing Robot Behaviors

- Be Creative
- Recognize the Strengths and Limitations of the Robot
- Know the environment the robot will be working in
- Know available paradigms for programming robots behaviors

Braitenberg Vehicles

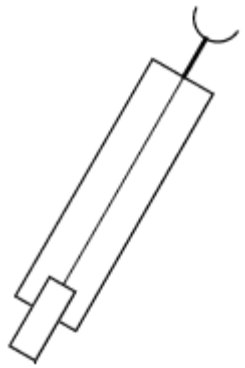
- Valentino Braitenberg wrote a book titled: *Vehicles: Experiments in Synthetic Psychology* (MIT Press)
- Describes several thought experiments that are centered around the **creation of simple vehicles** with **very simple control mechanisms** that exhibit seemingly **complex behavior**
- Illustrates insights into animal (human) brain
- Shows that vehicles are capable of complex behavior like Fear, Love, Logic etc.

Law of Uphill Analysis and Downhill Invention

- One central theme underlying Braitenberg's experiments
- Trying to postulate the internal structure purely by observing certain behavior is an uphill (harder) task
- Trying to create an entity that exhibits a certain behavior is a downhill (easier) task
- A Scribbler can do most of what the robots Braitenberg described can do

Vehicle 1

- Alive
  - One Sensor – One Motor
- Scribbler
  - One Sensor – Two motors



- You can use the center light sensor and connect what it reports directly to both motors of the robot
- Try this:
  - *motors(c,c) where 'c' is the value from light sensor*

Movement Using Light Sensors

- Light sensors report values in range of 0 – 5000 while motors take values from -1 to 1
- Normalizing (mapping light sensor values to the range of motor values)

```
def normalize(v):    # normalizes v to range 0.0 to 1.0


def main():
    while True:
        L = getLight("center")
        forward(normalize(L))
```

Normalizing Sensor Values

- Normalizing is transforming a value proportional to a required range (here 0.0 to 1.0, 1.0 being brighter light)
- Light Sensor is actually a Darkness sensor
  - Small values for bright light e.g. 50
  - Larger values for dark light e.g. 3000

```
def normalize(v):
# Normalize v (in the range 0..5000) to 0..1.0, inversely
        return 1.0 - v/5000.0
```

  - For brightness v = 35 what is the normalized value

Light Sensing

- You can set the ambient light value as the darkest condition so that you can identify a brighter light, say, a Flash Light
- How:

```
def normalize(v):
    if v > Ambient:
        v = Ambient
    return 1.0 - v/Ambient
```

- Either you can set this value or let the Robot sense it
- You can also find the values sensed by left and right sensors and take an average

Coward and Aggressive

- Speed of individual motor depends on the sensor on one side
- Coward mode:
  - Left sensor controls Left motor
  - Right sensor controls Right motor

```
Ambient = sum(getLight())/3.0  # values from 3 sensors/3
def normalize(v):
    if v > Ambient:
    v = Ambient
    return 1.0 - v/Ambient
```

Coward and Aggressive

L = getLight("left")

R = getLight("right")

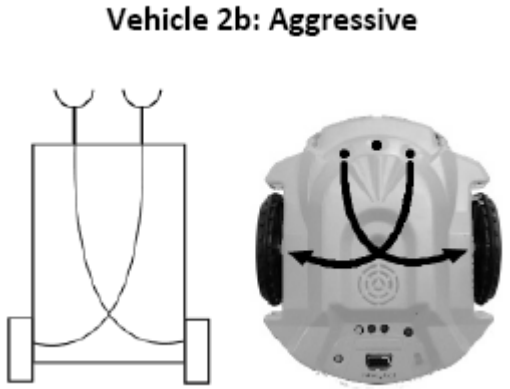
motors(normalize(L), normalize(R))

- Aggressive
  - Left sensor controls Right motor
  - Right sensor controls Left motor
- Repeat as you did before now

L = getLight(“right")

R = getLight(“left")

motors(normalize(L), normalize(R))



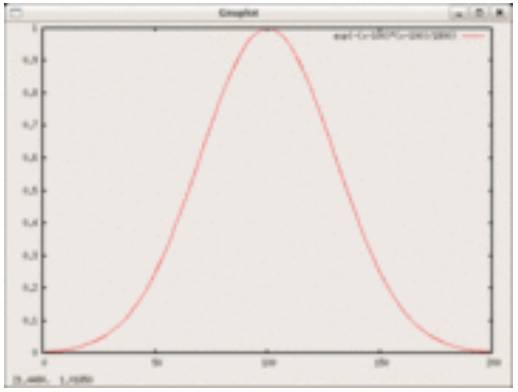
Try flashing the light on either side of the robot

Mathematical Functions

- Braitenberg suggests using non-monotonic mathematical functions
  - Monotonic: More light, faster motor speed; or more light, slower motor speed
  - Non-monotonic: the relationship is increasing in proportion to sensory input but only up to a certain point and after that it decreases

$$f(x) = e^{-x2}$$

Bell Curve or Gaussian Curve



- You can use exp() for e^x

Bell Curve Behavior

```
def normalize(v):
    mean = Ambient/2.0
    stddev = Ambient/6.0
    if v >= Ambient:
        v = Ambient
    return exp(-(v - mean)**2 / 2*(stddev**2))
```

- Other mathematical functions using math library (*from math import* *):
  - Step
  - Threshold

Multiple Sensing

- Adding more sensors makes the robot behavior more interesting
- Scribbler in addition to light sensors has
  - Stall Sensors (detect if the robot has stopped)
  - IR sensors and more
- Digital Sensors: Give either On or Off (e.g Stall)
- Try this:
  - *Replace the light sensors with obstacle sensors and try moving the robot*

Exercise

- How would you design paranoid, shadow fearing, behavior?

Paranoid should be capable of turning.

This can be accomplished by moving the right motor forward and moving the left motor in reverse direction at the same time.

If – else

```
if <condition>:
        <this>
else:
        <that>
```

Else if : *elif*

```
if <condition-1>:
<this>
elif <condition-2>:
<that>
...
else:
<other>
```

How do you use the if, else and elif

```
while timeRemaining(30):
if    #left light is brighter that right light:
        turnLeft(1.0)
else:
        turnRight(1.0)
```

Try this:
If the left side is brighter go right, vice versa

Light Follower

```
# Light follower
from myro import *
initialize(ask("What port?"))

# program settings...
thresh = 50
fwdSpeed = 0.8
cruiseSpeed = 0.5
turnSpeed = 0.7 # left turn, -0.7 will be right turn

def main():
    while True:
        # get light sensor values for left, center, and right
        L, C, R = getLight()
         # decide how to act based on sensor values
        if C < thresh:
              # bright light from straight ahead, go forward
              move(fwdSpeed, 0)
        elif L < thresh:
             # bright light at left, turn left
              move(cruiseSpeed, turnSpeed)
         elif R < thresh:
            # bright light on right side, turn right
            move(cruiseSpeed, -turnSpeed)
         else:
            # no bright light, move forward slowly (or stop?)
            move(cruiseSpeed/2, 0)

main()
```

Avoiding Obstacles

```
# Avoiding Obstacles
from myro import *
initialize(ask("What port?"))

# program settings...
cruiseSpeed = 0.6
turnSpeed = 0.5 # this is a left turn, -0.5 will be right
turn

def main():
    while True:
        # get sensor values for left and right IR sensors
        L, R = getIR()
        L = 1 - L
        R = 1 - R
        # decide how to act based on sensors values
        if L and R:
            # obstacle straight ahead, turn (randomly)
            move(0, turnSpeed)
        elif L:
            # obstacle on left, turn right
            move(cruiseSpeed, -turnSpeed)
        elif R:
            # obstacle on right, turn left
            move(cruiseSpeed, turnSpeed)
        else:
            # no obstacles
            move(cruiseSpeed, 0)
main()
```

Summary

- You now have an idea of Psychology
- Programming internal structures in a robot thus building brains for robot
- if – else ; elseif (*elif)* statements