Project 1 - Deliverable 6

Brian Huynh
BME 503
October 10th, 2014

I have adhered to the Duke Community Standard in completing this assignment.

Contents

List of Figures

1 Deliverable 6

A differential steering system was implemented into the bug to drive movement towards the food. The sensory inputs were based on the magnitude of the distance between the bug and the food. The sensory inputs were fed into `brain_avoid.m` where they were converted into spiking outputs. These spiking outputs were then applied to the left and right motor velocities given that

$$\frac{dv_L}{dt} = -\frac{v_L + v_{max} * motor_L}{\tau_{motor_L}} \tag{1}$$

$$\frac{dv_R}{dt} = -\frac{v_R + v_{max} * motor_R}{\tau_{motor_R}} \tag{2}$$

where v_{max} was a scalar for the maximum velocity, $motor_L$ and $motor_R$ were a binary output of whether a spike had occurred or not, and τ was the decay rate of the increase in velocity due to a spike.

The x- and y-direction as well as the heading angle were also modified using the following differential equations:

$$v = \frac{1}{2}(v_L + v_R) \tag{3}$$

$$\frac{dx}{dt} = -v * \cos(\theta) * dt \tag{4}$$

$$\frac{dy}{dt} = -v * \sin(\theta) * dt \tag{5}$$

2 Code and Outputs

2.1 bugnetworkmodel.m

```
clear; clc; clf

%% Initialize Global and Local Variables
global new_verts fverts xR yR xL yL
x=0; y=0;

arena_size=75; %bigger arena makes things more interesting.
draw_robot() % show robot for the first time (no arguments)
axis([-arena_size arena_size -arena_size arena_size]);  % create dummy arena
set(gca,'color','k');
axis square
set(gca,'Visible','on');

t       = 0;
dt      = 0.1;
tstop   = 50;
tvec    = t:dt:tstop;

% food target location
allowedLocations = 'all';
switch allowedLocations
    case 'all'
        maketgt = @() [150*rand(1,2)-[75 75]];
    case 'NE'
        maketgt = @() [75*rand(1,2)];
end

tgt = maketgt();

%Make Food
f_center    = tgt;
f_width     = 5;
food        = rectangle('Position',[f_center(1)-f_width(1)/2,...
    f_center(2)-f_width(1)/2,f_width(1),f_width(1)],'Curvature',[0.5,0.5],'EdgeColor','w');
set(food,'Position',[f_center(1)-f_width(1)/2,f_center(2)-f_width(1)/2,f_width(1),f_width(1)],...
    'Curvature',[0.5,0.5]);

heading_angle=0;

% Identify the X and Y coordinate of the R and L sensor from the robot description
xR = 10;
yR = -5;
xL = 10;
yL = 5;

% Compute the initial magnitude of the sensor info
sensorMag = @(tgt,loc) sqrt((tgt(1)-loc(1))^2+(tgt(2)-loc(2))^2);
sensorL = 1/sensorMag(tgt,[xL yL]);
sensorR = 1/sensorMag(tgt,[xR yR]);

% Compute the angle of the bug
sensorAng = @(tgt,loc1,loc2) ...
```

```
    atan( (tgt(2)-(loc1(2)+loc2(2))/2) / (tgt(1)-(loc1(1)+loc2(1))/2) );

heading_angle = sensorAng(tgt,[xL yL],[xR yR]);

if (tgt(1)<((xL+xR)/2) && tgt(2)>((yL+yR)/2)) || ... % quadrant II
        (tgt(1)<((xL+xR)/2) && tgt(2)<((yL+yR)/2))   % quadrant III
    heading_angle = pi+heading_angle;
end

draw_robot(x,y,heading_angle);

personalityType = 'coward';
Vm = zeros(1,2); % initial membrane voltage

vel_left = 0;
vel_right = 0;
v = 0;

for k=1:length(tvec)
    [motorL,motorR,Vm] = brain_avoid(sensorL,sensorR,dt,Vm);
    fprintf(sprintf('Vm=[%2.2g %2.2g] \n',Vm(1),Vm(2)))
    if any([motorL motorR])
        fprintf(sprintf('motorL=%2.2g motorR=%2.2g \n',motorL,motorR))
    end

    switch personalityType
        case 'coward'
            tau_motor = 100; % decay factor of motor inputs
            vmax = 100; % scale factor
            % change the left motor velocity
            vel_left = vel_left + (-vel_left + vmax*motorL)/tau_motor;

            % change the right motor velocity
            vel_right = vel_right + (-vel_right + vmax*motorR)/tau_motor;

            v = (vel_left+vel_right)/2;

            % change x direction
            x = x - v * cos(heading_angle)*dt;

            % change y direction
            y = y - v * sin(heading_angle)*dt;

            % update heading angle
            heading_angle = heading_angle - (vel_left-vel_right)/2*dt;

            fprintf(sprintf('Location: (%2.2g,%2.2g)\n',x,y))
            fprintf(sprintf('Velocity: L=%1.2g, R=%1.2g \n',vel_left,vel_right))
    end


    % Identify the X and Y coordinate of the R and L sensor from the
    %   robot description
    xR = x+5*sqrt(5)*cos(atan(-5/10)+heading_angle);
    yR = y+5*sqrt(5)*sin(atan(-5/10)+heading_angle);
```

```
    xL = x+5*sqrt(5)*cos(atan(5/10)+heading_angle);
    yL = y+5*sqrt(5)*sin(atan(5/10)+heading_angle);

    % Compute the magnitude of the sensor info
    scalar = 1;
    sensorL = sensorMag(tgt,[xL yL])*scalar;
    sensorR = sensorMag(tgt,[xR yR])*scalar;

    m = (max(sensorL,sensorR)-min(sensorL,sensorR))/max(sensorL,sensorR);
    if sensorL > sensorR
        sensorL = (1+m) * sensorL;
        sensorR = (1-m) * sensorR;
    elseif sensorR > sensorL
        sensorL = (1-m) * sensorL;
        sensorR = (1+m) * sensorR;
    end

    fprintf(sprintf('Sensor: L=%2.2g, R=%2.2g \n',sensorL,sensorR))

    % have arena wrap on itself
    if x>arena_size; x=-arena_size; end
    if y>arena_size; y=-arena_size; end
    if x<-arena_size; x=arena_size; end
    if y<-arena_size; y=arena_size; end

    % heading_angle = pi/2;
    draw_robot(x,y, heading_angle);
    drawnow

    DL = sqrt((x-tgt(1))^2+(y-tgt(2))^2);
    if DL < 10
        tgt = maketgt();
        f_center = tgt;
        set(food,'Position',[f_center(1)-f_width(1)/2,f_center(2)-f_width(1)/2,...
            f_width(1),f_width(1)],'Curvature',[0.5,0.5]);
    end
    fprintf('\n')
%     if mod(k,50) == 0; keyboard; end
end

%% spiking
% figure(2); hold on
% plot(tvec,sumexpR,'k-')
% plot(tvec,sumexpL,'k--')
% legend('R','L')
```

2.2 brain_avoid.m

```
function [motorL,motorR,v] = brain_avoid(sensorL,sensorR,DT,v)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Two neuron controller for BraitenBug
%
% Avoids the light source
%
% Neuron 1 receives input from the LEFT sensor and drives the LEFT motor
% Neuron 2 receives input from the RIGHT sensor and drives the RIGHT motor
%
% Neurons are modeled as leaky integrate-and-fire units
%
% Mark Nelson, Feb 2004
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% persistent v % membrane voltage

% fixed parameters
nNeurons = 2;
GAIN = 0.05;
VTHR = 5;
VSPK = 20;
TAU = 0.3;


% get sensor input
% (sensor input range: 0.0 - 1.0)
input = [sensorL sensorR];

% add some steady-state input current, otherwise
% the bug won't move when it's far from the light,
% resulting in a very boring simulation!

% loop over neurons
for n = 1:nNeurons

    % update membrane voltage
    if(v(n)==VSPK)
        v(n) = 0;
    else
        dvdt = (v(n) + GAIN*input(n))/TAU;
        v(n) = v(n) + dvdt*DT;
    end

    % check spike threshold
    if(v(n) >= VTHR)
        v(n)= VSPK;
    end
end

% drive the motors with spike output
motorL = (v(1)==VSPK);
motorR = (v(2)==VSPK);
```