Lecture 11: Final Robot Tag prep

Professor Carr Everbach

Course web page: http://www.swarthmore.edu/NatSci/ceverba1/Class/e5/E5Index.html
Remember...

- Tuesday (today) and Wednesday: Wizards available (Hicks 213) from 7:00-10:00 p.m. for robot tag prep.

- Robot tag in class one from today in class. If you don’t do any work outside of class, there is only one lab period Thursday to get everything working.

- There is a final Moodle Dropbox for handing in your group’s final autonomous tag programs, but you must also post these on your group’s E5 WordPress site along with links to the videos of the final tag sessions.
function Q = getcoords
% usage: Q = getcoords where Q = [robot; xposition; yposition]
% returns scaled coordinates of all robots from “eye in the sky”
% Uses judp.m to read position of ARToolKitPlus tracking of robots in E5
msg = judp('receive',4951,10000000);
smsg = [char(msg),':']  % append terminator
rindex = find(smsg==''); % find all the colons
pindex = find(smsg=='>'); % find all the >
cindex = find(smsg==','); % find all the commas
for index = 1:length(rindex)-1
    robot(index) = str2num(smsg(rindex(index)+1:pindex(index)-1));
xposition = str2num(smsg(pindex(index)+1:cindex(index)-1));
yposition = str2num(smsg(cindex(index)+1:rindex(index+1)-1));
Q = [robot’; xposition’; yposition’];
end

You’ll need to call “Q = getcoords” each time around your program’s “event loop” to find out where all the robots are currently located. If you save the previous positions, you can subtract them vectorially from the current positions and obtain \( v_x \) and \( v_y \) for each ‘bot.
function Q = getcoords
% usage: Q = getcoords where Q = [robot; xposition; yposition]
% returns scaled coordinates of all robots from "eye in the sky"
% Uses judp.m to read position of ARToolKitPlus tracking of robots in E5
msg = judp('receive',4951,10000000);
smsg = [char(msg),':'] % append terminator
rindex = find(smsg==':'); % find all the colons
pindex = find(smsg=='>'); % find all the >
cindex = find(smsg==','); % find all the commas
for index = 1:length(rindex)-1
    robot(index) = str2num(smsg(rindex(index)+1:pindex(index)-1));
    xposition = str2num(smsg(pindex(index)+1:cindex(index)-1));
    yposition = str2num(smsg(cindex(index)+1:rindex(index+1)-1));
    G = gscale(xposition,yposition); % scale coordinates to 0-1 in x&y
    x(index) = G(1);
    y(index) = G(2);
    fprintf('robot %d(%6.1f,%6.1f)
',robot(index),x(index),y(index))
end
if length(smsg) <= 1
    Q = [0,0,0]; % no robots
else
    Q = [robot', x', y']; % construct output variable
end
function G = gscale(xposition,yposition)
% usage: G = gscale(xposition,yposition); where x = G(1); y = G(2)
% rescales camera coordinates on 0-1 scale in x,y
% x = (xposition + 95)/210; % based on calibration 11/15/11
% y = 1 - (yposition + 100)/161;
x = xposition + 95;
y = 95 - yposition;
G(1) = x;
G(2) = y;
end
% program tag_template2.m
% Main program for implementing the game of Tag using E5 robots
% updated with velocity calculation and boundary checking
clear
xmin = 0; ymin = 0; xmax = 250; ymin = 225; % bounds of playing area in pixels
mybot = 7; % **insert number of your robot here**
phi = 0; % left-right angle, in degrees
time = 500; % speed parameter, in milliseconds
figure
% initialize serial port and put robot in starting pose
s = instrfind; % Find any serial links (we can have only 1)?
delete(s); %... and delete.
s = serial('COM1', 'Baudrate', 38400, 'Terminator', 'CR'); % s is serial port object.
fopen(s); %... and open it
gotoAnglesPhi(s, phi, time); % center arm
% put commands for other two motors here, if you wish

fprintf('Hit space to run Tag program\n');
pause
w=1; % flag to keep track of when to stop data acquisition
previousposition = [0 0]; % empty starting value
previoustime = tic; % start timer
Q = [0 0 0]; % dummy start values
% main event loop
while w
    Q = getcoords;
    while (Q(1,2) == 0 & Q(1,3) == 0) % avoid case of no robots visible
        Q = getcoords; % get updated robot positions
        disp('no robots found')
    end
    robot = Q(:,1); % get list of robots currently visible by camera
    x = Q(:,2); % get list of x-positions that correspond with robotlist
    y = Q(:,3); % get list of y-positions that correspond with robotlist
end

Note: the list of robots is not guaranteed to be sorted by robot number. The list may re-arrange depending upon if robot tracking drops out and pops back into the video.
itindex = find(robot < 0); % look for "it" robot
  itbot = robot(itindex); % number of "it" robot

  if isempty(itbot)
    itbot = 0; % robot0 is it
    itindex = find(robot == 0);
  elseif abs(itbot) == mybot
    itbot = mybot; % I'm "IT" so chase
  else
    itbot = abs(itbot); % another robot is it (flee)
  end
myindex = find(robot == mybot); % find our robot in list
    if ~isempty(myindex)
        currentposition = [x(myindex), y(myindex)]; % current position
        velocity = (currentposition - previousposition)/toc(previoustime);
        n = velocity/norm(velocity); % unit vectors in current direction
        nx = n(1); ny = n(2);
        fprintf('position: [%4.0f, %4.0f], velocity: [%4.2f, %4.2f]\n', ...
            currentposition, velocity);
        plot(x,y,'ko'); % plot robots
        hold on
        quiver(x(myindex),y(myindex),nx*10, ny*10,'go','filled');
        quiver(x(itindex),y(itindex),0,0,'ro','filled') % "it" robot in red
        axis([xmin xmax ymin ymax]); % fix axes
        hold off
        pause(0.1)
% test to see if we are out of bounds
if x(myindex) <= 0 & ny >= 0  % check bounds on x
    phi = swerve_right(s, phi);
elseif x(myindex) <= 0 & ny <= 0
    phi = swerve_left(s, phi);
elseif x(myindex) >= 250 & ny >= 0
    phi = swerve_left(s, phi);
elseif x(myindex) >= 250 & ny <= 0
    phi = swerve_right(s, phi);
end
if y(myindex) <= 0 & nx >= 0  % check bounds on y
    phi = swerve_left(s, phi);
elseif y(myindex) <= 0 & nx <= 0
    phi = swerve_right(s, phi);
elseif y(myindex) >= 225 & nx >= 0
    phi = swerve_right(s, phi);
elseif y(myindex) >= 225 & ny <= 0
    phi = swerve_left(s, phi);
end
if itbot == mybot  % I'm "IT"
    % put chasing code in here
    disp('chasing closest robot');
else % I'm not it
    % put fleeing code in here
    disp('fleeing "it" robot');
end
previousposition = currentposition;
previoustime = tic;
end

% event loop complete, close down robot and program
phi = 0
gotoAnglesPhi(s,phi,time);
% put commands for other two motors here, if you wish
fclose(s); % close the serial port cleanly
swerve routines:

function output = swerve_left(s,phi)
% swerves galumphing robot to the left
output = -45
end

function output = swerve_right(s,phi)
% swerves galumphing robot to the right
output = 45
end