School of Electrical and Electronic Engineering, Yonsei University

Course: Digital Control Engineering and Robotics

Project 1 due date: before class on 3th May

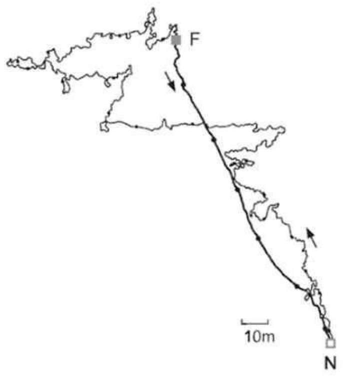

Figure 1. The desert ant (left: *Cataglyphis fortis*) and the ant's winding path outward (foraging) and straightforward homeward path recorded in a featureless salt pan.

Many insects and animals can return home accurately by using their own navigation system, after exploring to find the food. Basically, ants use pheromones to mark their moving path and they can get back to their nest by pheromones. But in the desert, as the surface temperature is high, pheromones can be easily evaporated. It is well known that in the Saharan desert, ants (*Cataglyphis fortis*) travel thousands of times their own body length and finally get back to their home directly as represented in Figure 1. Many researches show that desert ants, fiddler crabs, and honeybees use the path integration algorithm, also called dead reckoning. For path integration, the current moving speed and the angular information with the reference, for example, light compass are measured and integrated all through traveling, and the conclusive direction is selected for returning home. Without any help of a specific external landmark, the animals and insects can implement the path integration for their navigation.

Basically, **path integration** (PI) can be represented by following equations:

$$\frac{dx}{dt} = v\cos\theta, \quad \frac{dy}{dt} = v\sin\theta$$

Where (x, y) is the home vector, *v* is the animal's speed and θ is compass heading. **By accumulating these values every step, animals can get the summation vector finally, which enables to get them back home directly.**

We will simulate path integration by using a two wheeled robot. The robot controls each of the two wheel motors with an appropriate speed and thus the robot is exploring a given arena. A two-wheeled mobile robot model is given as shown below:

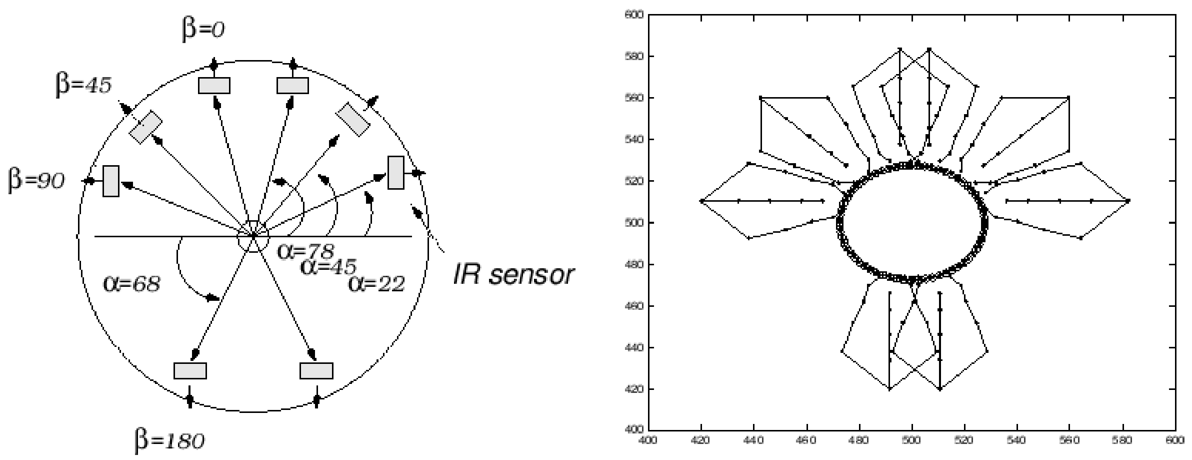

Figure 2. Robot system in simulation

From the homework #2, we simulated random movement of this robot. Suppose that this robot has two wheels to ensure the mobility and 8 infra-red sensors to detect the obstacles. Eight infra-red sensors are distributed along the rim as shown in Figure 2. The robot has 55 mm diameter and the length of wheel base is 50 mm.

The robot controller will issue the motor commands *(L, R)* of motor speeds to the motor system. The motor commands will vary in time course, depending on sensor information. This mapping from sensor information to motor commands should be coded as the robot control.

There are 8 infra-red sensors mounted on the robot. The sensor model follows a point-scanning method with 1 distance line (you may add more lines, if you wish). Each sensor has a projection line to detect any obstacle near the robot. The sensor model first calculates the distance to an obstacle and returns the corresponding sensor value. Apply +/-5% random noise to each sensor value. The sensor readings of infrared sensors range from 0 to 1023 (integer values with 10 bit AD converter). You can use the attached MATLAB code, *IRsensor_reading.m,* which returns 8 infra-red sensor readings at an instantaneous time. With the sensor values, it is hard to identify what kind of object is near the mobile robot, but we can determine how much the mobile robot is close to any obstacle.

You will test both obstacle avoidance behavior and path integration behavior. Obstacle avoidance behavior is to avoid any obstacle near the robot and move forward if there is no sense of obstacle. Path integration behavior is to accumulate the movement for each step, and the robot can go back home by using the accumulated data during random movement. Currently the robotic system has 8 infra-red sensors and two wheel motors. You need to develop your own style of sensor-motor mapping and show the trace of robotic movements.

The environment includes four side walls and 10 circular objects as displayed in Figure 3. The robot needs to read sensor information every time step and determine the appropriate motor commands for specific behaviors. The whole procedure may be described in a MATLAB program (see the program in the last page).

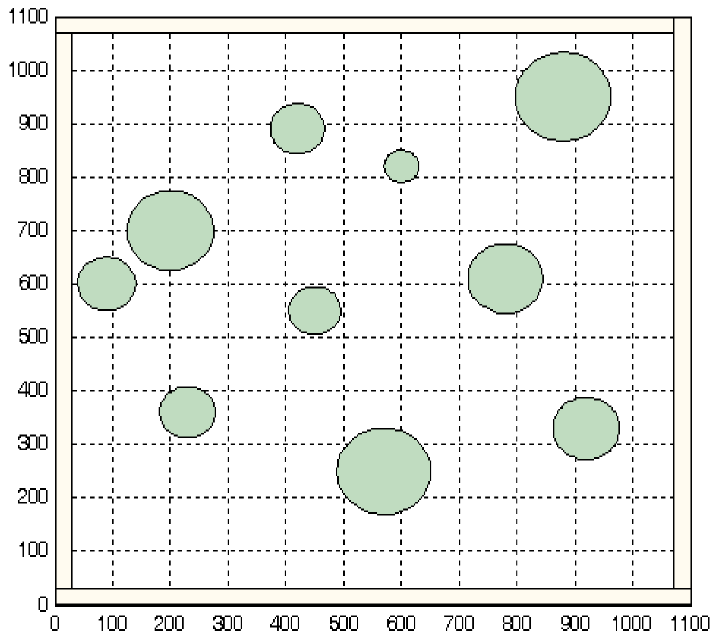

Figure 3. An example of environment with 10 objects

For obstacle avoidance behavior, you may calculate the following measure *F* for each time step.

$$F = (L+R)/2\ (1 - |L-R| / [\max(L)+\max(R)])\ (1 - \max(IR) / 1023)$$

Where *L* and *R* are motor commands for left and right wheels, and *IR* is a vector of infrared sensor values. *Max(IR)* indicates the maximum infrared sensor value. For the robotic behavior, you need to calculate the sum of the error measure *F* over the fixed whole time steps.

For path integration behavior, you have to check how much the angle of the robot changed, and how much the robot movement during each step.

You need to design robotic behaviors carefully to achieve the goal. Simulate robotic movements following the motor commands with the time step $\Delta = 0.1$ sec. Determine appropriate time span of the simulation, initial position, initial head angle, location of the obstacles, etc. Show the output as M-file.

Submissions

(1) Make the simulation program that robot moves randomly for each step in the map without any obstacle. After random move is done, perform path integration so that the robot can back to the start point. Plot the trajectory of the robot.

(2) Make the simulation with 10 obstacles using the prepared codes, *mapenv.m*. Basically, robot should move randomly. If the robot encounters some obstacles, it has to perform the obstacle avoidance behavior.

(3) Using the environment in problem (2), make the simulation program for the robot that performing both path integration and avoiding obstacles after some random movements. In this part, the robot does the path integration after wandering but if the robot encounters some obstacles in returning to the starting point, the robot should show obstacle avoidance motion first.

You need to write a report for the above results and your program code (12pt font is recommended) and add the source codes you have modified and the result

```
% main program for robotic behaviours
%

[objx, objy, oradius] = mapenv();


head = 0; x=200; y=200;  % starting position and heading angle

objectNumber = 10;

diameter = 55;   radius = diameter /2;  % robot diameter
B = 50;  % wheel base
t = 0:0.1:2*pi+0.2; % to draw robot body

while(1)
   hold off;
   %  (x,y, head) is the current position and head angle
   rx = x + r * cos(t); ry = y + r * sin(t); % robot drawing
   plot(rx, ry) % draw robot body
   axis([0 1100 0 1100])
   hold on;
   line([x x+radius*cos(head)], [y y+radius*sin(head)]) % mark head
   mapenv();  % draw the environment with obstacles


   [IR] = IRsensor_reading (head, x, y, objx, objy, oradius,
objectNumber);
      . . .
   % robot motor control part,
   %    map from sensor values to motor commands (L,R)
   %    it depends on what kind of behaviours you wish to test
      . . .
      . . .
   % kinematic simulation - next position (x,y) and next head angle
   %    which uses transient motor actions [ V_L and V_R  ]

      . . .
   % check if the robot collides with any obstacle or with the wall
   %    this can be checked by measuring the distance between the
   %    robot center position and a center position of an obstacle
   %    (1) distance(robot center, center of an object)
   %            < radius of robot + radius of an object.
   %    (2) distance(robot center, nearest wall)  < radius of robot
   %
      . . .


 pause(0.0001);   % this may be unnecessary
end
```