Distributed Intelligent Systems

Home Work 2

Afroze Ibrahim Baqapuri

Q.1/

10Hz means that the time interval between two updates is about 100ms. Since the robot's speed is 3.5cm/s , it travels 0.35cm between two updates.

In order to maintain the same performance of the obstacle avoidance we have to make sure that the robot travels same amount of distance in the update time interval. Since there is a delay of 155ms one and since update is a two-way process (signal from robot to computer and back again) so it takes 155x2 = 310ms for the update to complete. And assuming that the next update is initiated by the robot the very moment it receives the results of the previous update, the time interval bwteen updates come out to be in fact 310ms.

So we want 0.35cm distance to be travelled in 310ms, which makes the mean speed of the robot equal to **1.12cm/s.**

Q.2/

When a high motor speed is set, the interval for interrupt generated by the timer becomes smaller, meaning that now the motor rotates more often (since the motor rotates only when it receives an interrupt signal). While when the motor speed is set low, interrupt interval becomes larger and motor rotates less often.

We also note (as mentioned above) that the motor speed only rotates at specific points (when it receives an interrupt signal) and not continuously (like DC motor for example). This means that the speed of the motor is granular and not continuous.

If function to change speed is called so often that the time between two calls to set (i.e. change) speed is lesser than the time interval required by the previous speed for generating interval, then basically the motor would not rotate at all and robot would stop moving. This is because the motor would be waiting for the interrupt signal before it can rotate, but it would never receive the interrupt signal because every time a new speed is set the timer is reset and if this happens before the timer could generate even the first interrupt, the timer would infact not be able to generate any interrupt at all.

Q.3/

First we calculate the angular speed of the robot (which is same for both wheels because the change in angle for the both wheels is same since they are rotating around the circle at the same rate).

w = 2.pi/30 where 30s is the time required to complete a revolution of 2.pi radians

v(r) = 2.pi/30 * (0.5 – axle-length/2) where 0.5m is radius of circle of revolution
v(r) = 0.0992 m/s

v(l) = 2.pi/30 * (0.5 + axle-length/2) plus sign (and minus above) because robot turning clockwise
v(l) = 0.1103 m/s

Now we know the linear speeds of the right and left wheel and need to calculate the angular speeds of the wheels:

w(r) = v(r)/(wheel-diameter/2) note that all distance readings are in meters (also above)
w(r) = 4.839 rad/s
w(r) = 46.2 rpm calculate this by multiplying with 60 and dividing with 2.pi

w(l) = v(l)/(wheel-diameter/2)
w(l) = 5.379 rad/s
w(l) = 51.4 rpm

Now since 60 rpm corresponds to wheel command of 1000, so according to this ratio:

command(right wheel) = 770 multiplying speed in rpm by 1000 and dividing by 60

command(left wheel) = 857 approximately, rounded to nearest integer

I.1/

(a)

Since distance = velocity * time, and time interval = 1s;

ΔS(l) = 0.1103 m

Δs(r) = 0.0992 m

(b)

Position matrix at t=30 => [-0.006; 4.36X10^-5; 6.28]

This is very close to the ideal result of => [0; 0; 2*pi]

(c)

Position uncertainty co-variance matrix at t=30 =>

$$\begin{matrix} 17.64 & 3.84 & -1.49 \\ 3.84 & 1.86 & -0.508 \\ -1.49 & -0.508 & 0.211 \end{matrix}$$

Q.4/

In a normal obstacle free space the performance was very good since the other robots merely replicated the wheel speeds of the leader with a factor so as to maintain the formation near-perfectly.

However when there were obstacle the performance became very bad because if the follower robots encountered an obstacle they would still try to replicate the motion of the leader (even if the leader did not encounter the obstacle) and this would destroy the formation. Once the formation was destroyed it would be impossible to come back to it since the follower robots have no information about the position of the leader (and information about the leader's motion/speed) is lost while they were hitting obstacle and not being able to replicate the motion)

Q.5 /

If the leader robot rotated and the follower robots tried to replicate the rotation, the robot further away from the center of circle of rotation would start to have bad performance. They wouldn't be able to follow the path they should to maintain the formation and start deviating from it (this behavior increases in intensity as you move towards robots farther away from the center of circle).

This is because the algorithm tells the wheels of the follower robots to simply replicate the speed of the leader with a constant factor. This factor is directly proportional to the distance of the wheel from the radius of the circle's center. So if a robot is farther from the center as compared to the leader, the speed of its wheels would in fact be greater than the speed of leader's wheels. And since there is a fixed maximum speed the wheels can attain, if the algorithms tells it to go to a speed greater than this amount, it would just stop at the maximum speed. So the follower robots will not be able to rotate with the same angular speed and this will cause deviation in path and break-down of formation.

Yes this problem would exist on real robots too. Because there would be a maximum speed set on real robots too and if you try to assign a speed greater than that, the robot would just translate it to the maximum speed.

Q.6/

(a)

Maintain a safe range of IR sensor readings. When a robot comes too close, one (or more) of the IR sensor values will go higher than the safe range. Once this happens try to move slightly away from the direction in which the sensor reading is high while still following the overall movement of the flock. Do this until sensor readings are in safe range.

(b)

Each robot in the flock should transmit its wheel speeds (using wheel encoders) and orientation (using compass) to its closest neighbors as were known at the start when the flock was formed (using radio communication). So a robot would in fact receive a number of wheel speed and orientation reading s (number equaling to its closet neighborsMean speed of robot should be equal to the average of all mean speeds received and difference in wheel speeds should be such that it tries to match the average orientation.

(c)

Maintain a safe range of IR sensor readings. When a robot goes too far, one (or more) of the IR sensor values will go lower than the safe range. Once this happens try to move slightly towards the direction in which the sensor reading is low while still following the overall movement of the flock. Do this until sensor readings are in safe range.

I.2/

The strategy for implementing this is as follows: Initially (at start of experiment) robots should get the exact range and bearing readings of the leader from the module. After this they should use odometry to try to predict the leaders range and bearing from the information they get about his relative orientation and wheel speeds. However after specified intervals (say of about 1 minute) they would get the exact values from the module so they can update their predictions (if they were faulty).

Average performance achieved = **0.78**

Q.7/

$L = I * W * I^T$

$$I = \begin{bmatrix} 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Q.8/

$$\begin{bmatrix} d(x1)/dt \\ d(x2)/dt \\ d(x3)/dt \\ d(x4)/dt \\ d(x5)/dt \\ d(x6)/dt \end{bmatrix} = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 \\ 1 & -3 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & -3 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \end{bmatrix}$$

$$\begin{bmatrix} d(y1)/dt \\ d(y2)/dt \\ d(y3)/dt \\ d(y4)/dt \\ d(y5)/dt \\ d(y6)/dt \end{bmatrix} = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 \\ 1 & -3 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & -3 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \end{bmatrix}$$

I.3/

Approach, x =2

Approach, y = 3

I.4/

```
void calc_fitness(double weights[SWARMSIZE][DATASIZE], double fit[SWARMSIZE], int its,
int numRobs) {
  double buffer[255];
  double *rbuffer;
  int i,j,iter;
  int repeat = SWARMSIZE/MAX_ROB;

  /* Send data to robots */
  for (iter=1;iter<=repeat;iter++) {
    for (i=0;i<numRobs;i++) {
      random_pos(i);
      for (j=0;j<DATASIZE;j++) {
        buffer[j] = weights[((iter-1)*10)+i][j];
      }
      buffer[DATASIZE] = its;
      wb_emitter_send(emitter[i],(void *)buffer,(DATASIZE+1)*sizeof(double));
    }

  /* Wait for response */
    while (wb_receiver_get_queue_length(rec[0]) == 0)
      wb_robot_step(64);

  /* Get fitness values */
    for (i=0;i<numRobs;i++) {
      rbuffer = (double *)wb_receiver_get_data(rec[i]);
      fit[((iter-1)*10)+i] = rbuffer[0];
      wb_receiver_next_packet(rec[i]);
    }
  }
}
```

Q.9/

Swarmsize	Average Performance
10	0.505
20	0.518
30	0.560

It is clearly obvious that increasing the number of particles improves the performance. However there is a drawback of increased time required to receive the results from the larger number of particles.

Q.10/

In heterogeneous approach all the robots come up with a different solution, while in homogeneous they come up with the same solution. To illustrate it more consider the example of PSO, if there were 10 robots and swarmsize of 10, in homogeneous each robot will at first get one of the 10 particle (same for each) and try to compute solution, then in next iteration each robot will get another of the remaining 9 particles. While in heterogeneous each of the ten robots would have gotten a different particle and computed solution for a different particle, thus each particle would be assigned only to one robot. The advantage of heterogeneous is that is faster.

Q.11/

In private policy the different robots do not share their solution with other robots, while in public solution they do share their solution with other robots in the experiment. For example a homogeneous approach is automatically public because at each iteration each robot computes result for the same particle and to come up with the overall performance of that particle they need to combine results (and say take an average or pick best), for which they need to know the others results. So a private homogeneous approach would not be able to work because the individual robots would not be willing to tell the population manager of the performance they achieved and so the overall performance of that particle would not be computable.