EEE4320-01

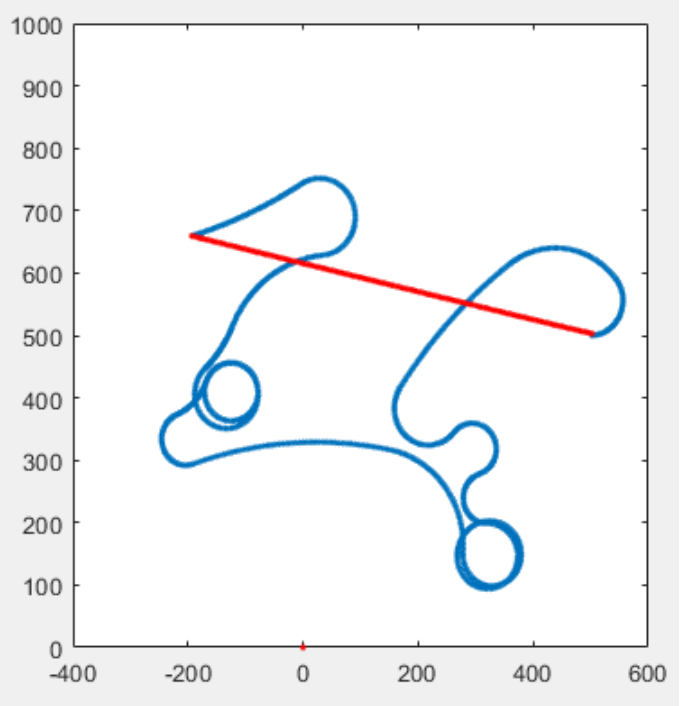
Digital Control Engineering

Project #1

2018142059

김서영

1. Robot moves randomly for each step in the map without any obstacle.



로봇이 [500,500]에서 출발하여 움직인 trajectory는 위와 같이 나타났다. 초반의 1000 step은 random하게 움직이고, 그 이후에는 path integration한 좌표를 통해 시작점과의 각도 dhead를 계산하여 head가 dhead와 같아지도록 회전하고 직진하여 path integration 값이 약 [0,0] (오차: x, y 각각 0.7)가 될 때까지, 즉 시작점에 다시 도달할 때까지 이동한다. random하게 움직인 부분의 trajectory는 푸른색으로, path integration하여 시작점으로 돌아오는 과정의 trajectory는 붉은색으로 나타내었다.

Path integration mode가 시작되자 로봇이 제자리에서 방향을 바꾸어 최단경로인 직선으로 돌아오는 것을 알 수 있다.

<Path integration>

$$pi_{x_{t+1}} = pi_{x_t} + V * \cos(head) * \Delta t$$

$$pi_{y_{t+1}} = pi_{y_t} + V * \sin(head) * \Delta t$$

$$(\text{desired head}) = \text{atan}\left(\frac{pi_{y_t}}{pi_{x_t}}\right)$$

2. 10 obstacles. Robot moves randomly and avoid obstacles

주어진 코드 mapenv.m을 통해 벽과 장애물이 있는 환경을 만들고, 로봇이 장애물을 피하며 랜덤하게 움직이도록 하였다. 장애물을 피하기 위해 고려한 특징은 다음과 같다.

1) 정면의 두 센서에서 감지되는 장애물은 이동 경로를 더 크게 바꿔야 하기 때문에 낮은 센서 값에서도 obstacle avoiding이 수행되도록 한다.

   A. 이 때 head angle을 바꿀 방향은 로봇의 좌, 우 센서의 값에 따라 정한다.

   B. Obstacle_location 함수를 통해 센서가 감지한 장애물의 위치를 센서의 좌표축, 로봇의 좌표축, 전체 map의 좌표축으로 나타내었다.

   C. 로봇이 후진하지 않으므로, 후면의 센서는 전면과는 반대로 방향을 바꾸어야 한다.

   D. 처음에는 로봇과 센서에 인식된 장애물간의 벡터합을 이용하여 desired head를 구하려 하였으나, 값이 좌우 대칭으로 나올 경우 로봇이 직진하여 장애물과 충돌하였다.

   E. 좌우 대칭인 센서들의 값을 각각 비교하여 경우를 나누는 방법 또한 시도하였으나 경우가 많고, 센서들 간의 우선순위가 생겨 장애물과 충돌하는 경우가 많았다. 이 방법은 충돌을 방지하기 위해서는 줄이려면 후진을 필요로 하였으며 한계가 있었다.

2) 정면의 센서에서는 장애물이 감지되지 않지만 Head angle로부터 45도인 두 센서에 장애물이 감지되는 경우를 고려한다. 정면 센서보다는 장애물이 더 가까이 있을 때 obstacle avoiding이 수행되도록 한다.

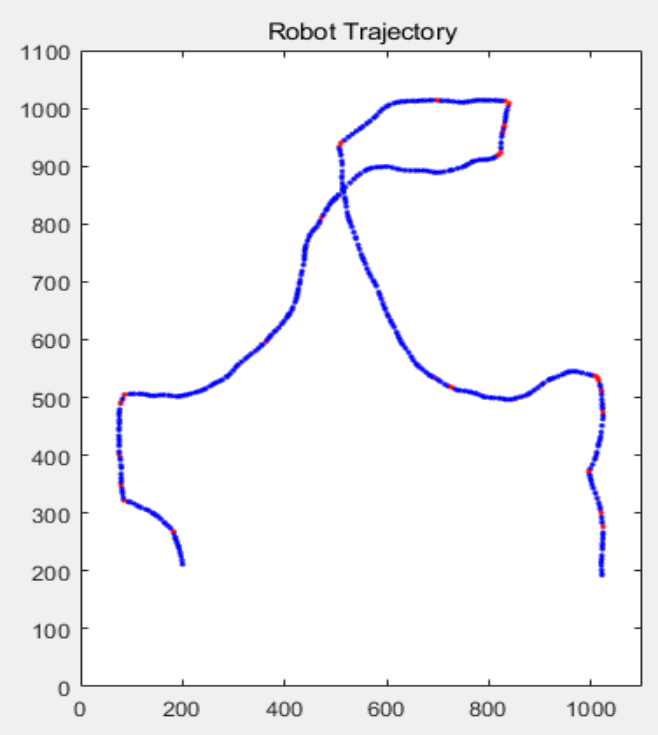
   A. 이 때, 바꾼 경로는 두 센서 중 장애물이 더 멀리서 감지된 (혹은 감지되지 않은) 센서의 방향으로 움직이도록 한다.

3) 방향 전환 시 제자리에서 회전하도록 하였다.

A. 회전할 때 움직이는 거리가 0이 되어 방향전환 시에 이를 고려하지 않아도 된다.

B.

위와 같이 코드를 작성하였을 때, 세 면이 모두 막히고 그 면들이 모든 센서에 감지될 때 로봇은 진동만 하고 그 자리를 벗어나지 못한다는 한계가 있었다. 따라서 mapenv.m를 수정하여 이러한 상황이 나타나지 않도록 하였다. 또한 각 센서가 감지하는 distance line을 3개로 늘려 그 평균 값을 센서의 출력값으로 하였는데, 이를 통해 센서의 개수보다 더 많은 각도에서 장애물을 감지할 수 있었다. 전면부의 센서는 45도마다 있으므로, 센서 하나가 사이각이 약 15도 정도인 3개의 distance line을 갖도록 하여 사각지대를 줄일 수 있도록 하였다.



궤도에서 파란색으로 나타난 것이 random movement일 때고, 빨간색으로 나타난 것이 obstacle avoiding을 수행할 때이다. Random movement로 자유롭게 움직이는 가운데 빨간색으로 나타난 지점에서의 방향 전환이 눈에 뜨는 것을 확인할 수 있다. F score (F의 누적합)은 500 time step동안 1.8590e+04 로 나타났다.

3. Avoid obstacles and do path integration after wandering

1과 2를 합쳐 나타낸 것으로, 초반의 time steps 동안은 장애물을 피하며 랜덤하게 이동한다. 그 후 desired head, dhead를 계산하여 시작점으로 돌아오게 된다. Obstacle avoidance를 위한 이동을 path integration에서 구한 dhead와 로봇의 이동방향인 head를 같게 하는 것보다 우선하여 장애물을 만나면 기존 알고리즘대로 장애물을 먼저 피하고, 그 후 다시 path integration 하여 시작점으로 돌아온다. 코드는 다음과 같은 순서로 작성하였다.

1) 로봇이 움직이는 mode를 random movement, obstacle avoiding, path integration(go to the start point)의 3가지 모드로 구성하였다.

2) 마지막으로 정해진 값이 로봇이 구동하는 데 사용된 실제 값이므로 obstacle avoiding에 대한 코드가 가장 마지막 부분에 와야 한다.

   A. Random movement mode와 path integration mode는 time step에 따른 것이므로 if와 else문으로 묶어서 나타냈다.

3) Random mode에서 각 바퀴의 각속도는 (미리 지정한 각속도의 최댓값)*rand으로 하였다.

4) Path integration mode는 2에서와 같이 path integration하여 구한 출발점을 기준으로 한 예상 좌표 pi_x와 pi_y의 값을 통해 desired head값 dhead를 구하여 진행하였다.

   A. 시작점과의 예상 거리와 θ=dhead-head값에 $(-\pi < \theta < \pi)$ 따라 각속도를 변하게 하여 좀 더 세밀하게 로봇의 head angle과 속도를 조절하였다.

$$\theta = r_{wheel} * \frac{\omega_r - \omega_l}{B}$$

$$\omega_r = -\omega_l = \theta * \frac{B}{r_{wheel} * 2}$$

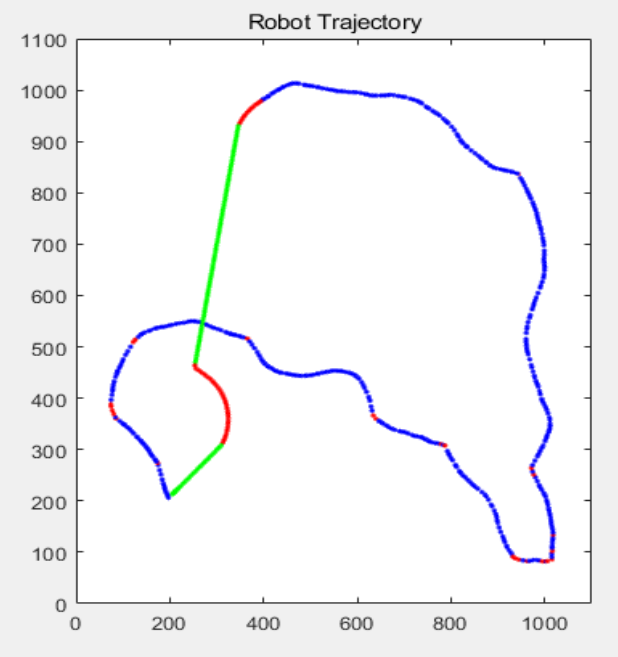
   B. Pi_x와 pi_y값은 정확하지 않으므로 threshold value를 지정하여 그 이내에만 들어오면 되도록 하였다.

   C. Obstacle avoidance mode와 path integration mode가 번갈아가면서 일

어날 때 로봇이 제자리에서 방향만 바꾸고 움직이지 못하는 상태가 되기도 하였다. 이를 방지하기 위해 path integration mode 직전에 obstacle avoidance mode였다면 직선으로 약간 운행하여 이러한 상태를 벗어나고자 하였다.

5) Obstacle Avoidance mode는 2번과 같은 방법으로 진행되었다.

이와 같이 작성한 코드를 이용하여 simulation하였을 때 나타난 trajectory는 다음과 같다.



파란색으로 나타난 것이 random movement mode일 때, 붉은색으로 나타난 것이 obstacle avoidance mode일 때, 초록색으로 나타난 것이 path integration mode일 때이다. Obstacle avoidance mode와 path integration mode가 충돌할 때 로봇이 장애물의 표면을 따라 방향을 자주 바꾸면서 천천히 이동하여 (붉은색이 점이 아닌 곡선처럼 나타난 부분) 시간이 오래 걸리나 시작점으로 다시 돌아간다.

F score는 random mode까지의 500 time step동안 1.3978e+04, path integration이 끝나는 simulation 전체 동안엔 1066 time step동안 2.0099e+04 로 나타났다.

<참고문헌>

Control of Mobile Robots – Georgia Tech (https://youtu.be/7_eJ_L-NYTg)

<코드>

y2018142059_m1.m

```
head = 0; x=500; y=500; % starting position and heading angle
objectNumber = 10;
diameter = 55; radius = diameter /2; % robot diameter
wdiameter = 5; wradius = wdiameter /2; % wheel diameter
B = 50; % wheel base
t = 0:0.1:2*pi+0.2; % to draw robot body
dt=0.1;

Numberofloop = 3000;
pi_x = 0.0; pi_y = 0.0;

savex=zeros(1,Numberofloop);
savey=zeros(1,Numberofloop);

for N=1:1:Numberofloop
hold off;

% (x,y, head) is the current position and head angle
rx = x + radius * cos(t); ry = y + radius * sin(t); % robot drawing
subplot(1,2,1);
p1 = 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
line([x               x+sqrt(pi_x^2+pi_y^2)*cos(dhead)],               [y
y+sqrt(pi_x^2+pi_y^2)*sin(dhead)],'Color','r')
hold off;

%. . .
% robot motor control part,
% map from sensor values to motor commands (L,R)
% it depends on what kind of behaviours you wish to test

%torque to robot system
 if (rem(N,50)==1 && N<Numberofloop/3)
    wL = rand * 10;
    wR = rand * 10;
 elseif (N >= Numberofloop/3)
    distance = sqrt(pi_x^2+pi_y^2);
   % start point에 도달했을때
   if (pi_x<=0.7 && pi_y<=0.7)
      wL=0;
      wR=0;
      break;
   %현재 head와 desired head가 같을 때 (직진)
   elseif (abs(dhead - head)<0.003)
      wL = min(maxL/2,max(distance*0.9,4));
      wR = min(maxR/2,max(distance*0.9,4));
   %현재 head와 desired head가 다를 때 (회전)
   else
      if (dhead-head>pi)
         theta=dhead-head-2*pi;
```

```
       else
          theta=dhead-head;
       end
       if (theta>0)
          wL = -abs(theta)*B/wradius/2;
          wR = +abs(theta)*B/wradius/2;
       else
          wL = +abs(theta)*B/wradius/2;
          wR = -abs(theta)*B/wradius/2;
       end
   end
 end

 %%%%
 VL=wL*wradius;%wheel speed
 VR=wR*wradius;%wheel speed
 V = (VL + VR) / 2; %robot speed

% kinematic simulation - next position (x,y) and next head angle
% which uses transient motor actions [ VL and VR ]
%. . .

 if(VL==VR)
 x=x+VL*dt*cos(head);
 y=y+VL*dt*sin(head);
 else
 w=(VR-VL)/B;
 R=(B*(VR+VL))/((VR-VL)*2);
 x=R*sin(w*dt+head)+x-R*sin(head);
 y=-R*cos(w*dt+head)+y+R*cos(head);
 head=mod(head+w*dt,2*pi);
 end

 pi_x = pi_x + V*cos(head)*dt;
 pi_y = pi_y + V*sin(head)*dt;

 %현재 위치에서 시작점을 바라보는 desired head 계산
 if (pi_y == 0)
    dhead = mod(pi_x/abs(pi_x) * -1,2*pi);
 elseif (pi_x > 0)
    dhead = mod(atan(pi_y/pi_x) + pi,2*pi);
 elseif (pi_x < 0)
    dhead = mod(atan(pi_y/pi_x),2*pi);
 else
    dhead = mod(pi_y/abs(pi_y) * -0.5,2*pi);
 end


 savex(1,N)=x;
 savey(1,N)=y;
 subplot(1,2,2);
 plot(savex(1,1:Numberofloop/3),savey(1,1:Numberofloop/3),'.'); hold on;

plot(savex(1,Numberofloop/3:Numberofloop),savey(1,Numberofloop/3:Numberof
loop),'.','Color','r');
 axis([0 1100 0 1100])

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

y2018142059_m2.m

```
%% main program for robotic behaviours

clc; clear; close all;
%
[objx, objy, oradius] = mapenv();

head = pi/2; x=200; y=200;            % starting position and heading
angle
objectNumber = 10;
diameter = 55; radius = diameter /2;    % robot diameter
wdiameter = 5; wradius = wdiameter /2;  % wheel diameter
B = 50; % wheel base
t = 0:0.1:2*pi+0.2; % to draw robot body
dt=0.1;

Numberofloop = 500; % Numberofloop

pi_x = 0.0; pi_y = 0.0;           % path integral initialize
obstacle_avoidance=false;          % obstacle avoidance mode flag
savex=zeros(1,Numberofloop);       % to plot x
savey=zeros(1,Numberofloop);       % to plot y
savef=zeros(1,Numberofloop);       % to plot f
collide = false;                  % collide flag (if 1, the loop ends)
wL =0.0; wR = 0.0;                % angular velocity initialize
dhead = head;          % desired head initialize
N=0;                  % loop number

for N=1:1:Numberofloop
% (x,y, head) is the current position and head angle
rx = x + radius * cos(t); ry = y + radius * sin(t); % robot drawing

%draw robot body
subplot(1,3,1); hold off;
if(obstacle_avoidance)    p1 = plot(rx, ry,'Color','r'); % red when
obstacle avoiding mode
else                    p1 = plot(rx, ry,'Color','b'); % blue when random
movement mode
end
set(gcf,'units','pixels','pos',[500 500 1500 500])
axis([0 1100 0 1100])
hold on;
line([x x+radius*cos(head)], [y y+radius*sin(head)],'LineWidth',1.5) %
mark head


mapenv(); % draw the environment with obstacles

% get sensor value
[IR] = IRsensor_reading (head, x, y, objx, objy, oradius, objectNumber);

% sensored obstacle position
[X,Y] = Obstacle_location(head,x,y,IR);



% IR sensor value plot
```

```
subplot(2,6,9);
bar(IR);
axis([0 9 0 1024])
hold off;


%. . .
% robot motor control part,
% map from sensor values to motor commands (L,R)
% it depends on what kind of behaviours you wish to test

% Random Movement mode
obstacle_avoidance=false;
maxL=50;
maxR=50;
wL = maxL*rand;
wR = maxR*rand;

% Obstacle Avoidance
if(max(IR(1,3:4)) > 1024/8)                  % obstacle in front angle
   obstacle_avoidance=true;
   if(sum(IR(1,1:3))+IR(1,7)>sum(IR(1,4:6))+IR(1,8))   % change head
angle to right
      wL = -maxL/2;
      wR = maxR/2;
   else                                    % change head angle to left
       wL = maxL/2;
       wR = -maxR/2;
   end
elseif(max(IR(1,2),IR(1,5)) > 1024/4)         % obstacle in side angle
   obstacle_avoidance=true;
   if(IR(1,2)>IR(1,5))                      % change head angle to right
      wL = -maxL/2;
      wR = maxR/2;
   else                                    % change head angle to left
       wL = maxL/2;
       wR = -maxR/2;
   end
end

%%%
VL=wL*wradius;%wheel speed
VR=wR*wradius;%wheel speed
V = (VL + VR) / 2; %robot speed
%  %fprintf('%d %f %f \n', N,VL, VR);


%% kinematic simulation - next position (x,y) and next head angle
% which uses transient motor actions [ VL and VR ]
%. . .
if(VL==VR)
x=x+VL*dt*cos(head);
y=y+VL*dt*sin(head);
else
w=(VR-VL)/B;
R=(B*(VR+VL))/((VR-VL)*2);
x=R*sin(w*dt+head)+x-R*sin(head);
y=-R*cos(w*dt+head)+y+R*cos(head);
head=mod(head+w*dt,2*pi);
end
```

```
%% F score
 F=(VL+VR)/2*(1-abs(VL-VR)/(maxL*wradius+maxR*wradius))*(1-max(IR)/1023);
 savef(1,N)=F;

 subplot(2,3,2);
 plot(savef(1:N));

%% Trajectory
 savex(1,N)=x;
 savey(1,N)=y;

 subplot(1,3,3);
 if(obstacle_avoidance)    plot(x, y,'.','Color','r');    % obstacle
avoiding mode
 else                    plot(x, y,'.','Color','b');    % random movement
mode
 end
 hold on;
 axis([0 1100 0 1100])

%% path integration
pi_x = pi_x + V*cos(head)*dt;
pi_y = pi_y + V*sin(head)*dt;

% 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.
for obj = 1:1:objectNumber
    objdistance=sqrt((x-objx(obj))^2 + (y-objy(obj))^2);
    if (objdistance <= radius + oradius(obj))
        fprintf('collide_obj\n');
        collide = true;
    end
end
% (2) distance(robot center, nearest wall) < radius of robot
if ((x>=30)&&(x<=1070)&&(y>=30)&&(y<=1070))
    if (min([abs(x-30),abs(x-1070),abs(y-30),abs(y-1070)]) < radius)
        fprintf('collide_wall\n');
        collide = true;
    end
else
    fprintf('collide_map\n');
    collide = true;
end

if collide
    break;
end

subplot(2,3,2); title('F Value');
subplot(2,6,9); title('IR Value');
subplot(2,6,10); title('Sensored distance in Robot Coord');
subplot(1,3,3); title('Robot Trajectory');

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

```
% %fprintf(sum(savef));
```

y2018142059_m3.m

```
%% robotic behaviours main program for m3

clc; clear; close all;
[objx, objy, oradius] = mapenv();

head = pi*2/3; x=200; y=200; % starting position and heading angle
objectNumber = 10;
diameter = 55; radius = diameter /2; % robot diameter
wdiameter = 5; wradius = wdiameter /2; % wheel diameter
B = 50; % wheel base
t = 0:0.1:2*pi+0.2; % to draw robot body
dt=0.1;

Numberofloop = 2500; %Numberofloop

pi_x = 0.0; pi_y = 0.0; % path integral initialize
mode=0;               % control mode (0: random movement,1: obstacle
avoidance, 2: path integral(go to start point))
collide = false;       % collide flag (if 1, the loop ends)
wL =0.0; wR = 0.0;     % angular velocity initialize
dhead = head;          % desired head initialize
N=0;                  % loop number

savef=zeros(1,Numberofloop);    % to save f
savex=zeros(1,Numberofloop);    % to save x
savey=zeros(1,Numberofloop);    % to save y

rmove = 5;
for N=1:1:Numberofloop
%    N=N+1;
% (x,y, head) is the current position and head angle
rx = x + radius * cos(t); ry = y + radius * sin(t); % robot drawing

subplot(1,3,1); hold off;
% draw robot body
if(mode==1)        p1 = plot(rx, ry,'Color','r'); % red when obstacle
avoiding mode
elseif(mode==2)    p1 = plot(rx, ry,'Color','g'); % green when path
integral (go to start point) mode
else              p1 = plot(rx, ry,'Color','b'); % blue when random
movement mode
end
set(gcf,'units','pixels','pos',[500 500 1500 500])
axis([0 1100 0 1100])
hold on;
line([x x+radius*cos(head)], [y y+radius*sin(head)],'LineWidth',1.5) %
mark head
if(N > Numberofloop/rmove)                          % line between start
point and the robot in path integral mode
   line([x x+sqrt(pi_x^2+pi_y^2)*cos(dhead)], [y
y+sqrt(pi_x^2+pi_y^2)*sin(dhead)],'LineStyle','--
','Color','g','LineWidth',0.3)
end

mapenv(); % draw the environment with obstacles
```

```
if (N==Numberofloop/rmove)         % path integral mode
   fprintf('go to home\n');
end

% get sensor value
[IR] = IRsensor_reading (head, x, y, objx, objy, oradius, objectNumber);

% sensored obstacle position
[X,Y] = Obstacle_location(head,x,y,IR);

% IR sensor value plot
subplot(2,6,9);
bar(IR);
axis([0 9 0 1024])
hold off;


%. . .
% robot motor control part,
% map from sensor values to motor commands (L,R)
% it depends on what kind of behaviours you wish to test
%. . .
%. . .

%random movement mode
if (N<Numberofloop/rmove)
   mode=0;
   maxL=50;
   maxR=50;
   wL = maxL*rand;
   wR = maxR*rand;

% go to start mode
else
   distance = sqrt(pi_x^2+pi_y^2);
   % start point에 도달했을때
   if (pi_x<=0.7 && pi_y<=0.7)
      wL=0;
      wR=0;
      break;
   %현재 head와 desired head가 같을 때 (직진)
   elseif (abs(dhead - head)<0.003)
      wL = min(maxL/2,max(distance*0.9,4));
      wR = min(maxR/2,max(distance*0.9,4));
   %현재 head와 desired head가 다를 때 (회전)
   else
      if (dhead-head>pi)
         theta=dhead-head-2*pi;
      else
         theta=dhead-head;
      end
      if (theta>0)
         wL = -abs(theta)*B/wradius/2;
         wR = +abs(theta)*B/wradius/2;
      else
         wL = +abs(theta)*B/wradius/2;
         wR = -abs(theta)*B/wradius/2;
```

```
       end
    end
    if (mode==1)
       wL=maxL/2;
       wR=maxR/2;
    end
    mode=2;
end

% obstacle avoidance
if(max(IR(1,3:4)) > 1024/8)           % obstacle in front angle
    mode=1;
    if(sum(IR(1,1:3))+IR(1,7)>sum(IR(1,4:6))+IR(1,8))   % change head
angle to right
      wL = -maxL/2;
      wR = maxR/2;
    else                                          % change head angle to
left
       wL = maxL/2;
       wR = -maxR/2;
    end
elseif(max(IR(1,2),IR(1,5)) > 1024/3)   % obstacle in side angle
    mode=1;
    if(IR(1,2)>IR(1,5))     % change head angle to right
       wL = -maxL/2;
       wR = maxR/2;
    else                  % change head angle to left
       wL = maxL/2;
       wR = -maxR/2;
    end
% elseif( min(IR(1:6)) ~= 0 && N >= Numberofloop) % to avoid collision
between obstacle avoiding mode and path integration mode
%     mode=1;
%     % go straight
%     wL = maxL*rand;
%     wR = maxR*rand;
% %     wL = maxL/2;
% %     wR = maxR/2;
end

if mode==1
    %fprintf('avoid\n');
end
 %%%%
 VL=wL*wradius;%wheel speed
 VR=wR*wradius;%wheel speed
 V = (VL + VR) / 2; %robot speed


%% kinematic simulation - next position (x,y) and next head angle
% which uses transient motor actions [ VL and VR ]
%. . .
 if(VL==VR)
 x=x+VL*dt*cos(head);
 y=y+VL*dt*sin(head);
 else
 w=(VR-VL)/B;
 R=(B*(VR+VL))/((VR-VL)*2);
 x=R*sin(w*dt+head)+x-R*sin(head);
 y=-R*cos(w*dt+head)+y+R*cos(head);
```

```
 head=mod(head+w*dt,2*pi);
 end

 %% F score
 F=(VL+VR)/2*(1-abs(VL-VR)/(20*wradius+20*wradius))*(1-max(IR)/1023);
 savef(1,N)=F;

 subplot(2,3,2);
 plot(savef(1:N));

 %% trajectory
 savex(1,N)=x;
 savey(1,N)=y;

 % plot trajectory
 subplot(1,3,3);
 if(mode==1)       plot(x, y,'.','Color','r');     % obstacle avoiding
mode
 elseif(mode==2)    plot(x, y,'.','Color','g');     % path integration
mode
 else             plot(x, y,'.','Color','b');     % random movement mode
 end
 hold on;
 axis([0 1100 0 1100])

%% path integration
 pi_x = pi_x + V*cos(head)*dt;
 pi_y = pi_y + V*sin(head)*dt;

 % 현재 위치에서 시작점을 바라보는 desired head 계산
 if (pi_y == 0)
    dhead = mod(pi_x/abs(pi_x) * -1,2*pi);
 elseif (pi_x > 0)
    dhead = mod(atan(pi_y/pi_x) + pi,2*pi);
 elseif (pi_x < 0)
    dhead = mod(atan(pi_y/pi_x),2*pi);
 else
    dhead = mod(pi_y/abs(pi_y) * -0.5,2*pi);
 end

% 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.
for obj = 1:1:objectNumber
    objdistance=sqrt((x-objx(obj))^2 + (y-objy(obj))^2);
    if (objdistance <= radius + oradius(obj))
        fprintf('collide_obj\n');
        collide = true;
    end
end
% (2) distance(robot center, nearest wall) < radius of robot
if ((x>=30)&&(x<=1070)&&(y>=30)&&(y<=1070))
    if (min([abs(x-30),abs(x-1070),abs(y-30),abs(y-1070)]) < radius)
        fprintf('collide_wall\n');
        collide = true;
    end
else
```

```
    fprintf('collide_outofmap\n');
    collide = true;
end

if collide
    break;
end

subplot(2,3,2); title('F Value');
subplot(2,6,9); title('IR Value');
subplot(2,6,10); title('Sensored distance in Robot Coord');
subplot(1,3,3); title('Robot Trajectory');

pause(0.0001); % this may be unnecessary
end

% fprintf(sum(savef(1:Numberofloop/rmove));
```

Obstacle_location.m

```
function [X, Y] = Obstacle_location(head,x,y,IR)
%로봇 관련 좌표 계산
%head는 로봇 방향
%(x,y)는 로봇 위치
%IR은 IRsensor에서 받은 값

r = 55/2;
%% IRsensor value to distance (digital to analog)
D = zeros(1,length(IR));        % 센서와 감지 장애물 사이 거리
for i=1:1:length(IR)
    tmp = IR(i);
    if (tmp == 1023)
        D(i) = 15;
    elseif (tmp > 756)
        D(i) = 5 * (1023 - tmp) / (1023 - 756) + 20;
    elseif (tmp > 400)
        D(i) = 5 * (756 - tmp) / (756 - 400) + 25;
    elseif (tmp > 260)
        D(i) = 5 * (400 - tmp) / (400 - 260) + 30;
    elseif (tmp > 145)
        D(i) = 5 * (260 - tmp) / (260 - 145) + 35;
    elseif (tmp > 92)
        D(i) = 5 * (145 - tmp) / (145 - 92) + 40;
    elseif (tmp > 74)
        D(i) = 5 * (92 - tmp) / (92 - 74) + 45;
    elseif (tmp > 60)
        D(i) = 5 * (74 - tmp) / (74 - 60) + 50;
    else
        D(i) = 55;
    end
end

%% obstacle location in robot coordination
theta_a = [22 45 78 180-78 180-45 180-22 180+68 360-68] / 180 * pi;     %
센서 각도 알파
theta_b = [-90 -45 0 0 45 90 180 180] / 180 *
```

```
pi;                                % 센서 각도 베타
S = zeros(4, length(IR));
distance = zeros(1, length(IR));
for i=1:1:length(D)
    %(D,0,0)의 각 beta를 a에 맞추고, -r만큼 이동 후 다시 pi/2-a만큼 회전
    d = [D(i); 0; 0; 1];
    trans_r = transfer(-r,0,0);
    rot_r = rotation(pi/2-theta_a(i));
    rot_s = rotation(theta_a(i)-theta_b(i)-pi/2);
    S(:,i) = rot_r*trans_r*rot_s*d;
    distance(i) = sqrt(S(1,i)^2+S(2,i)^2);
end

%% obstacle location in world coordination
X = zeros(1,length(IR));       % world coord 상의 감지 물체 x좌표
Y = zeros(1,length(IR));       % world coord 상의 감지 물체 y좌표

for j=1:1:length(D)
    % robot coord에서 -head만큼 회전, (-x,-y) 이동
    trans = transfer(-x,-y,0);
    rot = rotation(-head);

    pos = trans *rot * S(:,j);
    X(j)= pos(1);
    Y(j)= pos(2);
end

%sensor plot on world coord
subplot(1,3,1);
scatter(X,Y);

%sensor plot in robot coord
subplot(2,6,10);
scatter (S(1,:),S(2,:));
hold on;
for k = 1:1:length(S(1,:))
    line([0 S(1,k)],[0 S(2,k)])
end
axis([-100 100 -100 100]);
hold off;
subplot(1,3,1);
end

function rot = rotation(theta)
    rot = [cos(theta) sin(theta) 0 0; -sin(theta) cos(theta) 0 0; 0 0 1
0; 0 0 0 1;];
end

function trans = transfer(x,y,z)
    trans = [1 0 0 -x; 0 1 0 -y; 0 0 1 -z; 0 0 0 1;];
end
```

IRsensor_reading.m [수정한 부분만]

```
function [D] = IRsensor_reading(head,x,y,a,b,rr,onum)
```

```
%head는 로봇의 방향
%(x,y)는 로봇의 위치
%(a,b)는 장애물의 위치
% rr 은 장애물 반지름
% onum 은 장애물 수
% D는 센서 값.(8개의 배열로 ouput이 나옴),우측 전면부터 반시계 방향
% M은 발자국 수(그래프)

XY=zeros(2,length(x));
XY(1,:)=x;
XY(2,:)=y;
r=55/2;  % robot radius

…

%put three distance lines for a one sensor
dangle=pi/12;
D(1) = mean([Psensor(T0,head0,a,b,rr,onum),Psensor(T0,head0 -
dangle,a,b,rr,onum),Psensor(T0,head0 + dangle ,a,b,rr,onum)]);
D(2) = mean([Psensor(T1,head1,a,b,rr,onum),Psensor(T1,head1 -
dangle,a,b,rr,onum),Psensor(T1,head0 +
dangle ,a,b,rr,onum)]);    %Psensor ÇÔ¼ö¸¦ ¸¸µé¾î »ç¿ë.
D(3) = mean([Psensor(T2,head2,a,b,rr,onum),Psensor(T2,head2 -
dangle,a,b,rr,onum),Psensor(T2,head0 + dangle ,a,b,rr,onum)]);
D(4) = mean([Psensor(T3,head3,a,b,rr,onum),Psensor(T3,head3 -
dangle,a,b,rr,onum),Psensor(T3,head0 + dangle ,a,b,rr,onum)]);
D(5) = mean([Psensor(T4,head4,a,b,rr,onum),Psensor(T4,head4 -
dangle,a,b,rr,onum),Psensor(T4,head0 + dangle ,a,b,rr,onum)]);
D(6) = mean([Psensor(T5,head5,a,b,rr,onum),Psensor(T5,head5 -
dangle,a,b,rr,onum),Psensor(T5,head0 + dangle ,a,b,rr,onum)]);
D(7) = mean([Psensor(T6,head6,a,b,rr,onum),Psensor(T6,head6 -
dangle,a,b,rr,onum),Psensor(T6,head0 + dangle ,a,b,rr,onum)]);
D(8) = mean([Psensor(T7,head7,a,b,rr,onum),Psensor(T7,head7 -
dangle,a,b,rr,onum),Psensor(T7,head0 + dangle ,a,b,rr,onum)]);

…
```

mapenv.m [장애물 위치 수정]

```
% 장애물 크기와 위치
rr=[ 50  75  48  65  30  58  45  81  47  83];
xx=[ 90 130 230 780 600 890 450 570 420 960];
yy=[600 730 360 610 820 330 550 250 890 970];
```