function [boolean\_collision, sample\_point] = collision\_check (sample\_point,closest\_point,obstacles,r)

if sample\_point(1,2:3) == closest\_point(1,2:3)

boolean\_collision = 2;

return;

end

boolean\_collision = 0;

%y = ax + b

a = (sample\_point(1,3)-closest\_point(1,3))/(sample\_point(1,2)-closest\_point(1,2));

b = sample\_point(1,3) - (a \* sample\_point(1,2));

rand\_segment = sort(rand(20,1),'descend');

rand\_segment(1,1) = 1;

%choose 20 random points along the line and check if they are inside any of

%the obstacles boundary

for j = 1:20

if boolean\_collision == 0

x = ((sample\_point(1,2) - closest\_point(1,2))\*rand\_segment(j,1))+(closest\_point(1,2));

y = (a\*x) + b;

for i=1:length(obstacles)

x\_obst\_min = obstacles(i,1) - (obstacles(i,3)\*0.5+r); %center of circle minus both radii of circle and robot

x\_obst\_max = obstacles(i,1) + (obstacles(i,3)\*0.5+r);

y\_obst\_min = obstacles(i,2) - (obstacles(i,3)\*0.5+r);

y\_obst\_max = obstacles(i,2) + (obstacles(i,3)\*0.5+r);

if (x\_obst\_min < x) && (x < x\_obst\_max) && (y\_obst\_min < y) && (y < y\_obst\_max)

boolean\_collision = 1;

break;

else

sample\_point(1,2:3) = sample\_point(1,2:3);

boolean\_collision = 0;

end

end

end

end

end