# ME 425 LAB 7 Report
# Path Planning for a Mobile Robot

Arda Gencer
34124
**Instructor:** Mustafa Ünel

Fall 2025-2026
**Due Date:** 06/01/2026

# 1 Introduction

This lab focuses on implementing path planning algorithms for a mobile robot using potential field methods. The objective is to navigate the robot from a start position to a goal position while avoiding obstacles by calculating attractive and repulsive forces. Two tasks were performed: the first task involved path planning without repulsive force calculations (repulsive forces were commented out in the code), and the second task attempted to include repulsive forces but failed due to errors in the code and incorrect gain values for the attractive and repulsive forces.

# 2 Procedure

The MATLAB script provided implements the potential field path planning algorithm. The robot's position is updated iteratively by calculating the attractive force towards the goal and the repulsive forces from obstacles. In the first task, the repulsive force calculations were commented out, effectively disabling obstacle avoidance, and the robot was guided solely by the attractive force towards the goal. This allowed the robot to reach the goal but without obstacle avoidance.

For the second task, the repulsive force calculations were enabled to allow the robot to avoid obstacles. However, the task failed due to errors in the code and incorrect tuning of the gains for the attractive and repulsive forces. These incorrect gains caused the robot to behave unexpectedly, preventing successful navigation to the goal.

The MATLAB code uses parameters such as gain values for attractive and repulsive forces, threshold distances for obstacle influence, and step sizes for position updates. The robot's trajectory is computed until it reaches the goal or a maximum number of iterations is exceeded.

# 3 Results

**First Task Video:** https://drive.google.com/file/d/1t5p1JJqSSnu-TMk1yQxPqDrIqIZc7uAl/view?usp=sharing

**Second Task Video:** https://drive.google.com/file/d/1RKfdU_ddEMUoQPQ5Ujm3sOIHWbHLrYbd/view?usp=sharing

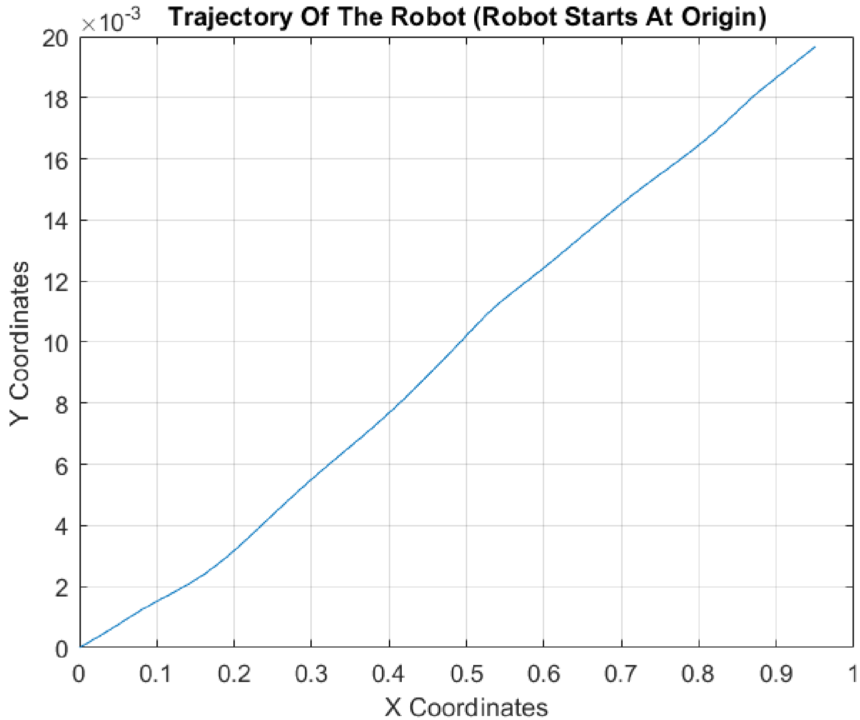


Figure 1: Task 1 With Goal Set To X = 1, Y = 0

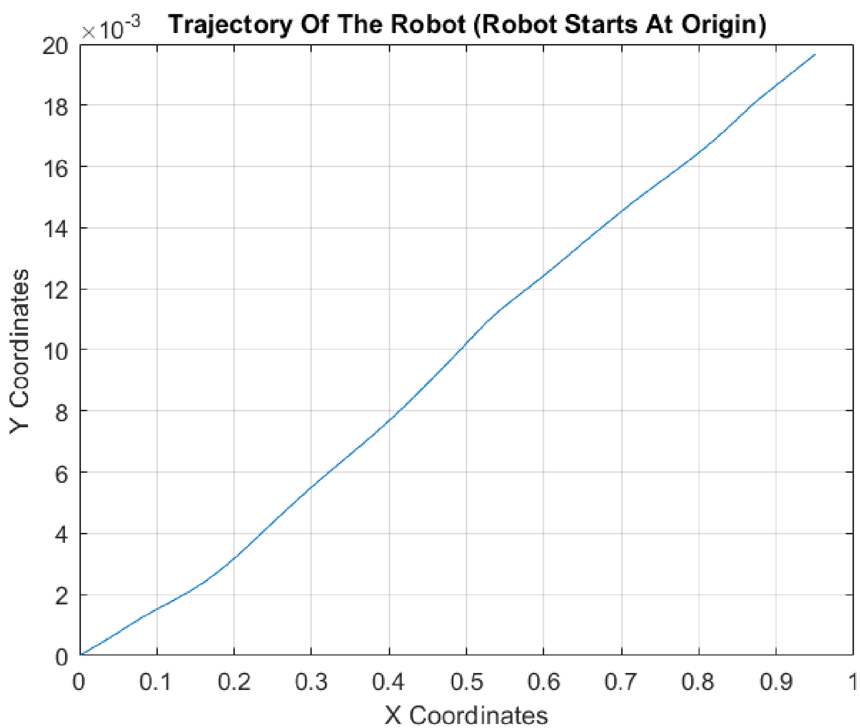


Figure 2: Task 1 With Goal Set To X = 1, Y = -0.5

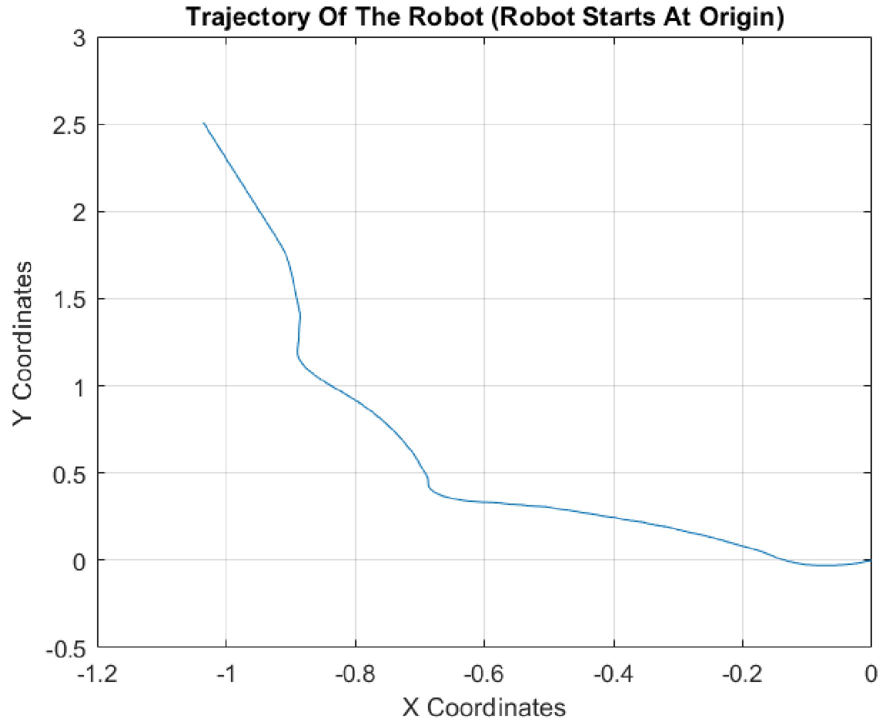


Figure 3: Task 2 With Goal Set To X = 1.5, Y = 0

The first task successfully guided the robot to the goal using only the attractive force. The path was direct but did not account for obstacles, which could lead to collisions in a real environment.

The second task did not produce a valid path due to errors in the repulsive force implementation and inappropriate gain values. The robot's movement was erratic, and it failed to reach the goal while avoiding obstacles.

# 4 Conclusion

The lab demonstrated the implementation of potential field path planning for a mobile robot. The first task showed that attractive forces alone can guide the robot to the goal but without obstacle avoidance. The second task highlighted the importance of correct implementation and tuning of repulsive forces to achieve effective obstacle avoidance. Future work should focus on debugging the repulsive force calculations and properly tuning the gains to ensure safe and efficient navigation.

# 5 Discussion

The failure of the second task underscores the challenges in tuning potential field parameters. Incorrect gains can cause oscillations, local minima, or failure to converge to the goal. Additionally, errors in the code logic for repulsive force calculations can lead to unexpected robot behavior. Careful debugging and parameter tuning are essential for successful path planning using potential fields. Alternative methods or hybrid approaches may also be considered to overcome limitations of pure potential field methods.

# 6 Appendix

Listing 1: The Code That Was Used For Both Of THe Tasks

```
clc; clear; close all;
%% ROS Setup
clear node;

setenv('ROS_DOMAIN_ID','47');
setenv('ROS_LOCALHOST_ONLY', '0');
setenv('RMW_IMPLEMENTATION','rmw_fastrtps_cpp');

node = ros2node('PathFinding');

%% GET INITIAL POSE & HEADING
odomSub = ros2subscriber(node, "/odom", "nav_msgs/Odometry");
startMsg = receive(odomSub, 10);
pose_start = startMsg.pose.pose.position;
quat_start = startMsg.pose.pose.orientation;
x_start = pose_start.x;
y_start = pose_start.y;
angles_start = quat2eul([quat_start.w, quat_start.x, quat_start.y,
    quat_start.z]);
yaw_start = angles_start(1);

%% INITIALIZATIONS
lidarSub = ros2subscriber(node, "/scan", "sensor_msgs/LaserScan", ...
    "Reliability", "besteffort", "Durability", "volatile", "Depth", 10);

velPub = ros2publisher(node, "/cmd_vel", "geometry_msgs/Twist", ...
    "Reliability", "reliable", "Durability", "volatile", "Depth", 10);
velMsg = ros2message(velPub);

goalReached = 0;
qGoal = [1.5 0];   %Goal Location
kAtt = 0.15;
kRep = 6;
p0 = 0.8;
alpha = 0.08;
k_theta = 2.5;
threshold = 0.05;

%For Plotting
x_poses = [];
y_poses = [];

%% MAIN LOOP
while goalReached == 0
```

```
    %% --- 1. ROBOT LOCALIZATION ---
    odomMsg = receive(odomSub, 10);
    pose = odomMsg.pose.pose.position;
    quat = odomMsg.pose.pose.orientation;
    angles = quat2eul([quat.w, quat.x, quat.y, quat.z]);
    yaw = angles(1);
    x_pos = pose.x;
    y_pos = pose.y;

    % Normalize Odometry Pose and Heading
    dx = x_pos - x_start;
    dy = y_pos - y_start;
    x_calibrated = dx * cos(yaw_start) + dy * sin(yaw_start);
    y_calibrated = -dx * sin(yaw_start) + dy * cos(yaw_start);
    q_robot = [x_calibrated, y_calibrated];
    theta = yaw - yaw_start;

    %% --- 2. CALCULATE ATTRACTIVE FORCE ---
    Fatt = -kAtt * (q_robot - qGoal);

    %% --- 3. CALCULATE REPULSIVE FORCE ---
    scanMsg = receive(lidarSub, 10);
    ranges = double(scanMsg.ranges(:));
    n = length(ranges);
    angles_scan = double(scanMsg.angle_min) + (0:n-1)' * double(scanMsg.angle_increment);

    % Filter valid obstacles
    valid_range = (ranges > double(scanMsg.range_min)) & ...
                  (ranges < double(scanMsg.range_max)) & ...
                  isfinite(ranges);
    front_angle = (angles_scan > -pi/2) & (angles_scan < pi/2);
    valid_obstacles = valid_range & front_angle;

    filtered_ranges = ranges(valid_obstacles);
    filtered_angles = angles_scan(valid_obstacles);

    % Calculate Total Repulsive Force
    Frep = [0, 0];
    for i = 1:length(filtered_ranges)
        range = filtered_ranges(i);
        angle = filtered_angles(i);
        qObstacle = [range * cos(angle), range * sin(angle)];

        pq = range;

        fprintf('  Obstacle %d: range=%.2f m, p0=%.2f, condition=%d\n', ...
            i, pq, p0, (pq <= p0 && pq > 0.01));
        if pq <= p0 && pq > 0.01
            repForceIncrement = kRep * (1/pq - 1/p0) * (-qObstacle) / (pq^2);
            Frep = Frep + repForceIncrement;

```

```
        end
    end
    %Frep =0;   %Activate For Task 1
    %% --- 4. MOTION CONTROL USING TOTAL FORCE ---
    Ftot = Frep + Fatt;
    V = alpha * Ftot;
    v = sqrt(V(1)^2 + V(2)^2);
    thetaDesired = atan2(V(2), V(1));

    thetaError = thetaDesired - theta;
    thetaError = atan2(sin(thetaError), cos(thetaError));
    w = k_theta * thetaError;

    %Safety checks
    if ~isfinite(v) || ~isfinite(w)
        v = 0;
        w = 0;
        fprintf('Warning: Invalid velocity computed\n');
    end

    %Limit velocities
    v = max(0, min(v, 0.1));
    w = max(-1, min(w, 1));

    %Ensure double type
    velMsg.linear.x = double(v);
    velMsg.angular.z = double(w);
    send(velPub, velMsg);

    %% Check How Close
    distanceToGoal = sqrt((q_robot(1) - qGoal(1))^2 + (q_robot(2) - qGoal(2))^2);
    if distanceToGoal < threshold
        goalReached = 1;
        fprintf('Goal reached! Final position: [%.2f, %.2f]\n', q_robot(1), q_robot(2));
    end

    % Display status
    fprintf('Pos: [%.2f, %.2f], Dist: %.2f m, v: %.2f, w: %.2f\n', ...
            q_robot(1), q_robot(2), distanceToGoal, v, w);

    x_poses = [x_poses x_pos-x_start];
    y_poses = [y_poses y_pos-y_start];
    pause(0.1);
end

%% Stop Robot
velMsg.linear.x = 0.0;
velMsg.angular.z = 0.0;
send(velPub, velMsg);
fprintf('Robot stopped.\n');

%% Plotting
```

```
figure;
plot(x_poses,y_poses);
title("Trajectory Of The Robot (Robot Starts At Origin)");
xlabel("X Coordinates");
ylabel("Y Coordinates");
grid on;
```