**CSB310: Artificial Intelligence**

**Date Assigned :27/9/2023 Date Submitted: 24/10/2023**

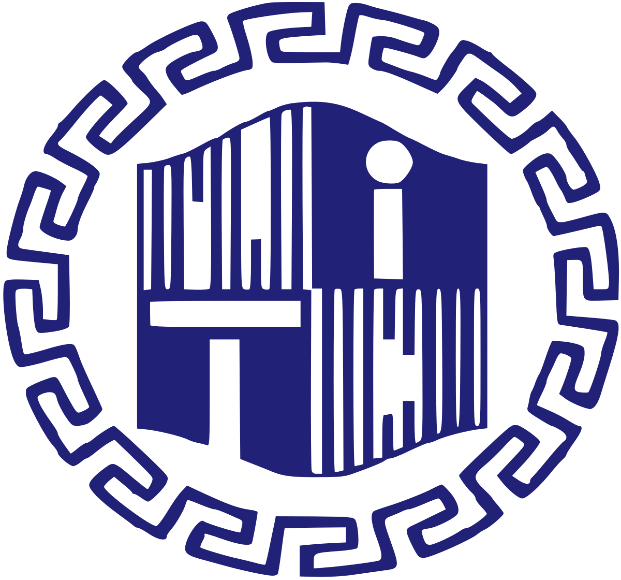
**Submitted To: Dr. Chandra Prakash (Asst. Professor)**

**Submitted By: Rohit Kumar (211210053)**

**CSE (5th Sem)**

**Group : 2**

**Department of Computer Science and Engineering**



**NATIONAL INSTITUTE OF TECHNOLOGY DELHI**

2023

LAB 8

PART A: Explore Motion Planning with MATLAB.

1. MATLAB Navigation Toolbox. [LINK]

-Navigation Toolbox provides algorithms and analysis tools for motion planning, simultaneous local-

ization and mapping (SLAM), and inertial navigation.

-The toolbox includes customizable search and sampling-based path planners, as well as metrics for

validating and comparing paths.

-You can create 2D and 3D map representations, generate maps using SLAM algorithms, and interac-

tively visualize and debug map generation with the SLAM map builder app.

2. Task: Apply any two motion planning algorithms available in MATLAB Navigation Toolbox:

Observation:

I have successfully implemented RRT and SLAM using the toolbox

Create a state space.

ss = stateSpaceSE2;

Create an occupancyMap-based state validator using the created state space.

sv = validatorOccupancyMap(ss);

Create an occupancy map from an example map and set map resolution as 10 cells/meter.

load exampleMaps.mat

map = occupancyMap(simpleMap,10);

sv.Map = map;

Set validation distance for the validator.

sv.ValidationDistance = 0.01;

Update state space bounds to be the same as map limits.

ss.StateBounds = [map.XWorldLimits; map.YWorldLimits; [-pi pi]];

Create RRT\* path planner and allow further optimization after goal is reached. Reduce the maximum iterations and increase the maximum connection distance.

planner = plannerRRTStar(ss,sv, ...

ContinueAfterGoalReached=true, ...

MaxIterations=2500, ...

MaxConnectionDistance=0.3);

Set the start and goal states.

start = [0.5 0.5 0];

goal = [2.5 0.2 0];

Plan a path with default settings.

rng(100,'twister') % repeatable result

[pthObj,solnInfo] = plan(planner,start,goal);

Visualize the results.

map.show

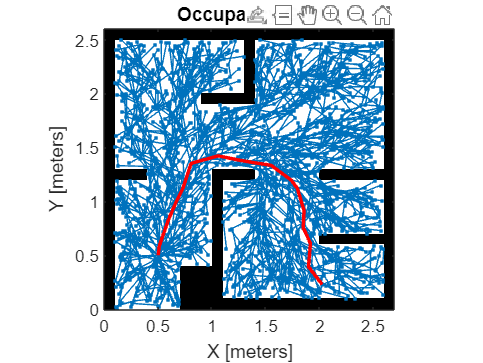
hold on

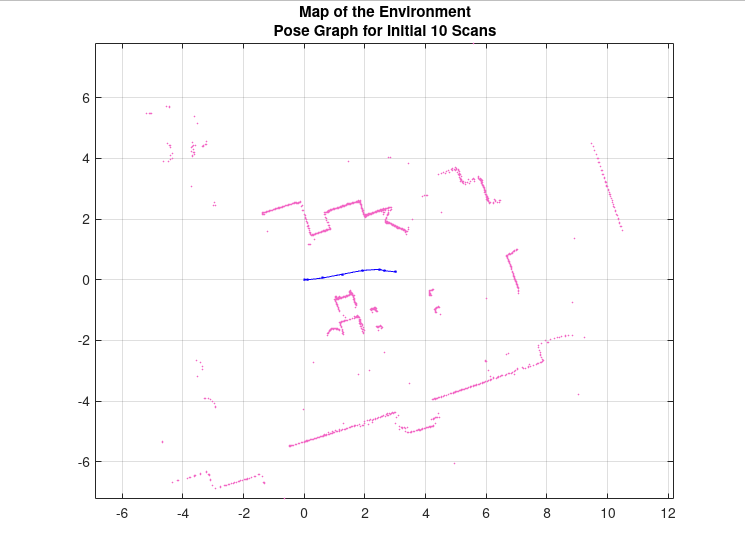
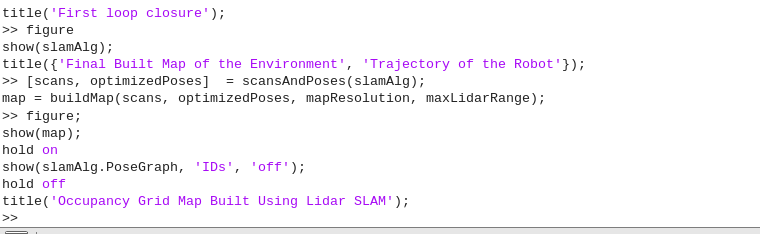
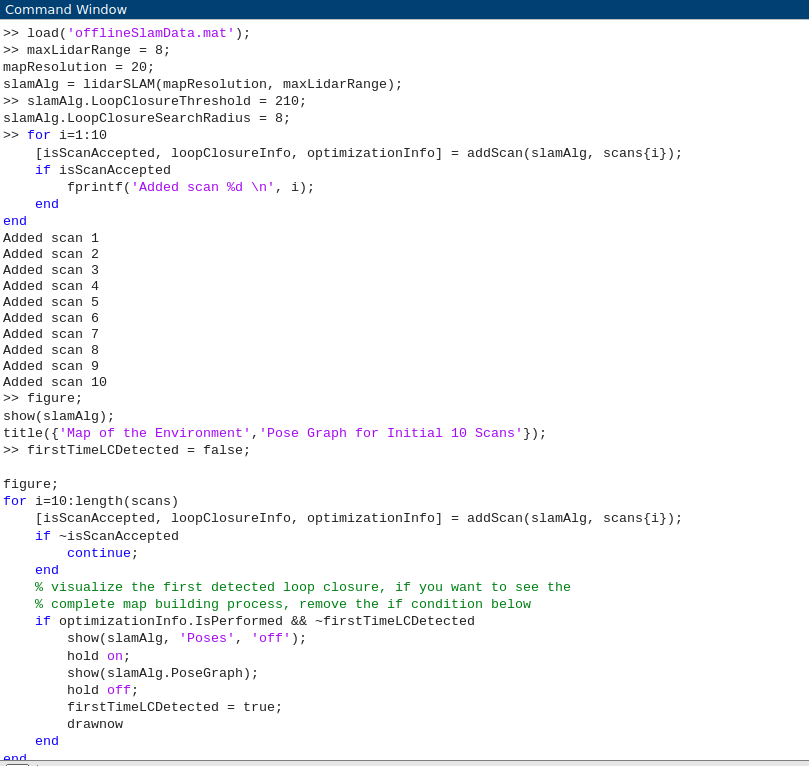
% Tree expansion

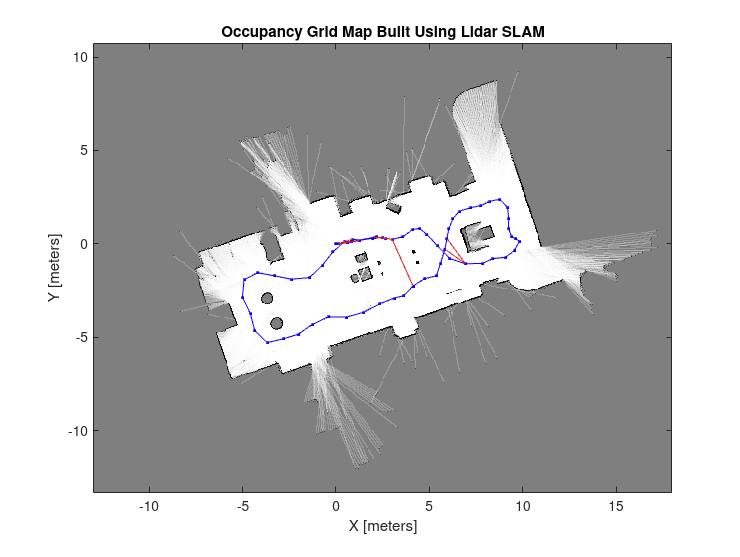
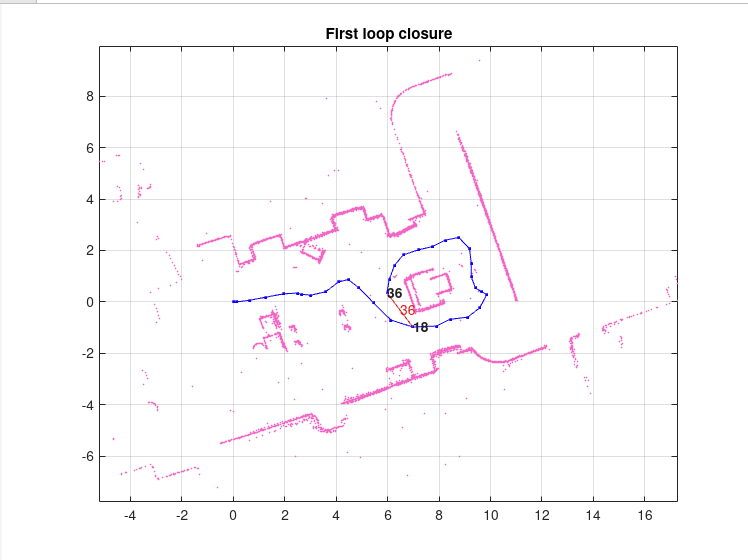
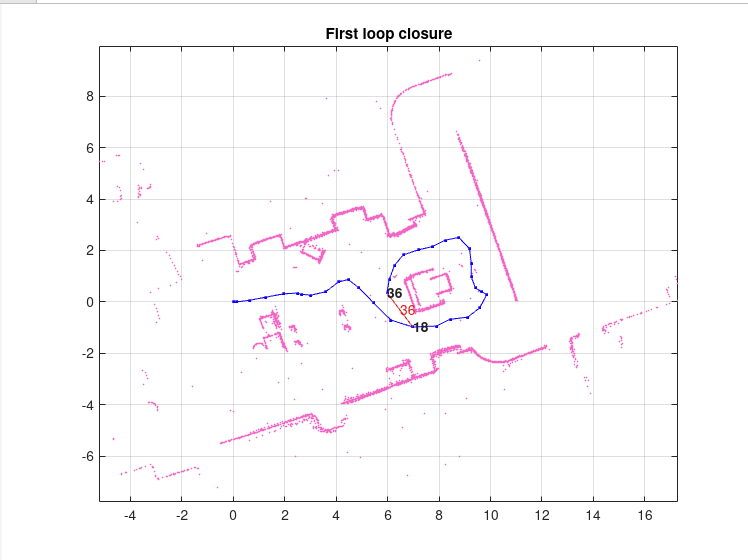
plot(solnInfo.TreeData(:,1),solnInfo.TreeData(:,2),'.-')

% Draw path

plot(pthObj.States(:,1),pthObj.States(:,2),'r-','LineWidth',2)







PART B: Explore MATLAB Mobile Robotics Simulation Toolbox. [LINK]

Task: Run the example code of Differential Drive Path Planning and Navigation, it uses a path found

in an occupancy grid using a probabilistic roadmap (PRM). Draw a flow chart of the workflow and

functional dependencies for the sample code.

%% EXAMPLE: Differential Drive Path Planning and Navigation

% In this example, a path is found in an occupancy grid using a

% probabilistic roadmap (PRM) and followed using Pure Pursuit

%

% Copyright 2018-2019 The MathWorks, Inc.

%% Define Vehicle

R = 0.1; % Wheel radius [m]

L = 0.5; % Wheelbase [m]

dd = DifferentialDrive(R,L);

%% Simulation parameters

sampleTime = 0.1; % Sample time [s]

tVec = 0:sampleTime:25; % Time array

initPose = [2;2;0]; % Initial pose (x y theta)

pose = zeros(3,numel(tVec)); % Pose matrix

pose(:,1) = initPose;

%% Path planning

% Load map and inflate it by a safety distance

close all

load exampleMap

inflate(map,R);

% Create a Probabilistic Road Map (PRM)

planner = mobileRobotPRM(map);

planner.NumNodes = 75;

planner.ConnectionDistance = 5;

% Find a path from the start point to a specified goal point

startPoint = initPose(1:2)';

goalPoint = [11, 11];

waypoints = findpath(planner,startPoint,goalPoint);

show(planner)

%% Pure Pursuit Controller

controller = controllerPurePursuit;

controller.Waypoints = waypoints;

controller.LookaheadDistance = 0.35;

controller.DesiredLinearVelocity = 0.75;

controller.MaxAngularVelocity = 1.5;

%% Create visualizer

load exampleMap % Reload original (uninflated) map for visualization

viz = Visualizer2D;

viz.hasWaypoints = true;

viz.mapName = 'map';

%% Simulation loop

r = rateControl(1/sampleTime);

for idx = 2:numel(tVec)

% Run the Pure Pursuit controller and convert output to wheel speeds

[vRef,wRef] = controller(pose(:,idx-1));

[wL,wR] = inverseKinematics(dd,vRef,wRef);

% Compute the velocities

[v,w] = forwardKinematics(dd,wL,wR);

velB = [v;0;w]; % Body velocities [vx;vy;w]

vel = bodyToWorld(velB,pose(:,idx-1)); % Convert from body to world

% Perform forward discrete integration step

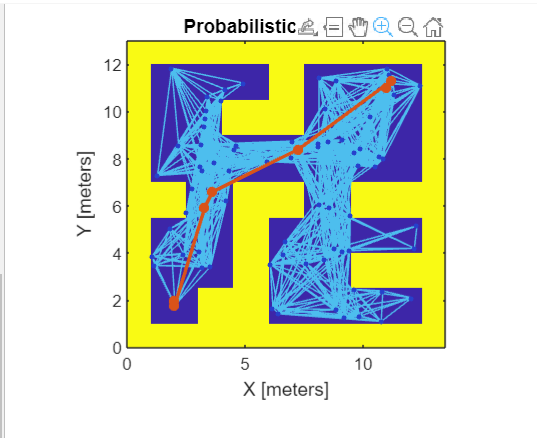
pose(:,idx) = pose(:,idx-1) + vel\*sampleTime;

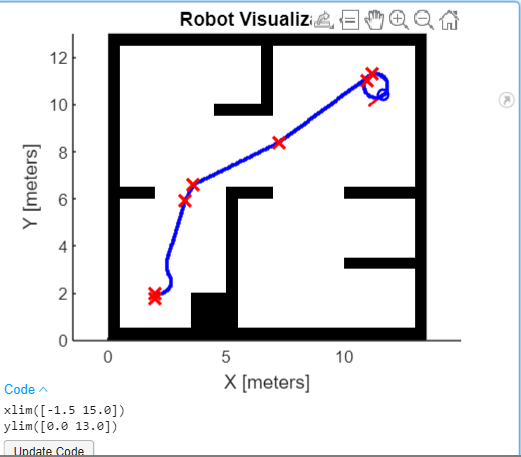
% Update visualization

viz(pose(:,idx),waypoints)

waitfor(r);

end





PART C: Exploratory Problem

5. Using Tinkercad, design a circuit for controlling a DC motor using a potentiometer. Program the

Arduino board for the operation and stimulate the scenario over Tinkercad.

