ARW – Autonomous Robot Workshop

Lab 3

Path Planning

1. INTRODUCTION

The purpose of this lab is to introduce participants to the Probabilistic Road Map (PRM) and Rapidly exploring Random Tree (RRT) motion planning algorithms. By the end of the lab, students should have implemented a single-query PRM (aka an RRT) planner that constructs collision free paths for the robot from any initial robot state to any feasible robot goal state within the workspace. This lab can be done in pairs.

2. CODEBASE

For this lab, you will implement your planner in the file called constructPath.m which will be called from the main control loop located in arw_simulate.m. See Fig.1.

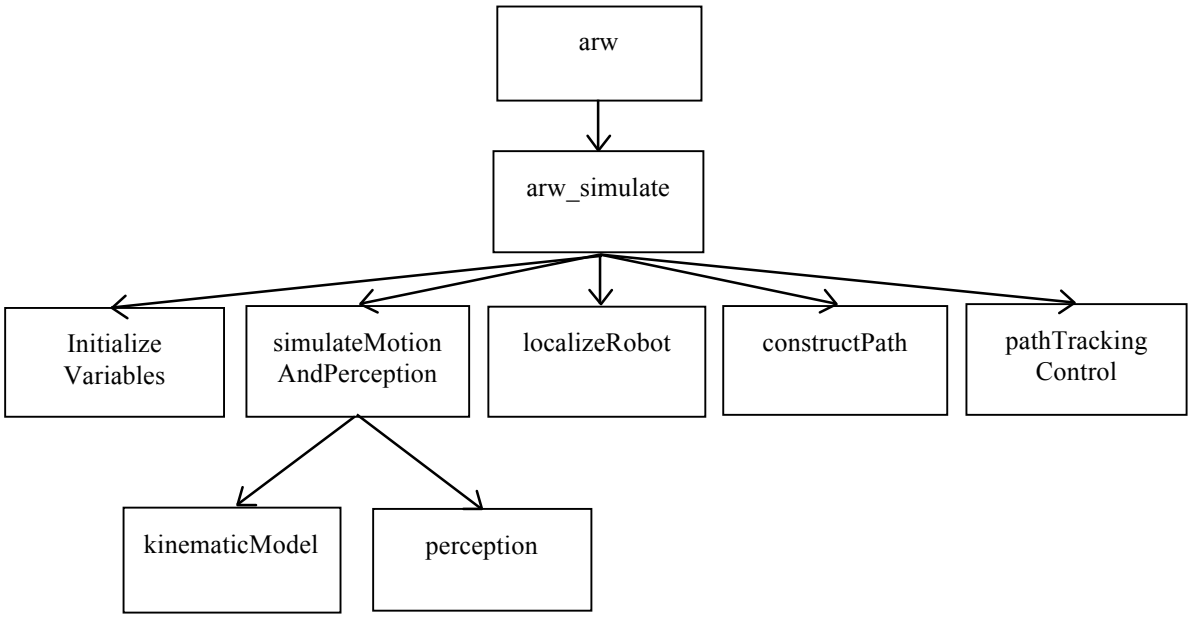

Figure 1: Codebase Architecture

3. CONTROL LOOP MODIFICATIONS

Open the file named arw_simulator.m. It should look similar to (but not exactly like) the code shown in Fig. 2. As in Lab 2, some of the other function calls in the control loop are commented. The call to constructPath() will no longer be commented so that your modifications to constructPath will be invoked. The control vector U will still be hardcoded for the time being. Modify your arw_simulator() function as shown below in Fig. 2.

```
function [X] = arw_simulator(X_0, X_des)

    % Initialize variables
    [ X_real, X_est, U, pathTracked, timeStep, M, P, T, e, nodes, deltaT ]
                                            = initializeVariables( X_0 );

    % Loop over time, until pathtracking is complete
    maxTimeStep = 100;
    while pathTracked == false && timeStep <= maxTimeStep

        % Simulate the actual robot to get the real state, measured
        % odometry O, and range measurements Z
        [ X_real, O, Z ] = simulateMotionAndPerception( X_real, U, M, deltaT );

        % Localize the robot, i.e. estimate the current state X
        [X_est, P] = localizeRobot(X_est, O, Z, M, P);

        % Construct a new path if it is the first iteration
        [T, nodes] = constructPath(X_0, X_des, M, T, nodes);

        % Determine the control inputs U to track the path T
        U=[1200 50*timeStep]; %[U, T, pathTracked] = pathTrackingControl(X_est, T);

        % Save states and calculate the error e
        X(timeStep,1:3) = X_real;
        e(timeStep) = sqrt((X_real(1)-X_est(1))^2 + (X_real(2)-X_est(2))^2);

        % Plot the output
        plotState(X_real, e, X, Z, M, P, nodes, T);
        timeStep = timeStep + 1;
    end
end
```

Figure 2: Modified arw_simulator.m file for use with Lab 3.

4. BUILDRRT MODIFICATIONS

Open the file named constructPath.m and find the buildRRT() function. It should look like the code shown in Fig. 3. In this case much of the code is missing. Given the input values of the initial robot state X_start (which is supposed to represent $X_{start} = [x_{start}\ y_{start}\ \theta_{start}\ ]$ in units of meters and radians), the goal state X_goal and the map M, the algorithm will output the path T, which is an n x 4 matrix that stores the n path waypoints. Each waypoint consists of 3 values defining the waypoint state $[x\ y\ \theta\ ]$, as well as an extra parameter that is only set to 0 or 1. It is set to 1 only after the robot has successfully visited the waypoint, or 0 otherwise. Don't worry about setting these values for now. An example Path may look like:

```
T=[0 0 0 1;
   1 0 0 0;
   1 0 pi/2 0;
   1 2 pi/2 0];
```

Note that in this path, all waypoints have the final digit set to 0 except the first.

Another data structure that is important is the node list called nodes that stores all nodes in the PRM. This data structure is an m x 3 array of m nodes, where each node's first 2 values correspond to the nodes x, y position, and the 3rd parameter corresponds to the index of the node's parent node. This last entry will be useful when building the final path T after the PRM tree grows all the way to the goal state.

The first thing to do in the buildRRT() function is to implement code that selects a node in the roadmap for expansion. Don't randomly pick a node from the PRM giving all nodes equal likelihood of being selected. Implement the RRT node selection method:

Randomly pick a location in the workspace (use the mapRanges vector to help determine workspace boundaries), then loop through all nodes in the PRM and find the node closest to the randomly selected location. Set nodeToExpand to be this node.

```
function [T, nodes] = buildRRT(X_start, X_goal, M)

    % Create NodeList
    nodes = [X_start(1) X_start(2) 0];
    pathFound = false;
    maxIterations = 10000;
    iterations = 0;
    mapRanges = getMapRanges(M);


    % Expand the tree until a path is found OR too many iterations have
    % passed.
    while iterations < maxIterations && pathFound == false

        % Choose node for expansion, set nodeToExpand = ...



        % Expand Node, set newNode = .....


        % Check for collision, add newNode and check if connected to goal



        % Increment number of iterations
        iterations = iterations + 1;
    end

    % Create the trajectory to follow
    T = BuildOptimalPath(nodes, mapRanges, M);

end
```

Figure 3: The BuildRRT function to be modified.

Once nodeToExpand has been set, a new node will be created by expanding from it to a new node location. To do this, randomly pick a direction randAng and distance randDist. Construct a newNode that is in the direction of randAng from nodeToExpand, and a distance of randDist from nodeToExpand.

Before adding the node to the PRM, check to see if there is collision on the edge connecting nodeToExpand to newNode. If there is no collision, add the newNode to nodes. Then see if there is a collision on the edge connecting the newNode to Xgoal. If

there is no collision, create a goalNode located at Xgoal with parent node index set to be the index of newNode , (i.e. where it is stored in nodes). Add the goalNode to nodes, and set the pathFound to be true.

At the end of this function, the BuildOptimalPath() function will construct T for you, using your nodes. By running arw, your new path T should appear as red lines on the screen.

4. PARAMETER MODIFICATIONS

Once your PRM is up and running, you can modify a few parameters to keep the lab exciting. First, try modifying the plotState.m file and uncomment the plotNodes() function call. This will let you visualize how your PRM expanded. Unfortunately, this will also dramatically slow down your code.

You can also try different goal destinations by modifying the arw.m file.