ME 597 Lab #3:
Path Planning

Introduction:

In this lab you will be creating your own ROS package which will perform path planning and tracking using the turtlebot robots. This lab will require more involved implementation and data manipulation, and if done poorly, will run quite slowly on the netbooks or even on laptops. These types of limitations are common in robotics applications, so this should be more good experience for future robotics projects.

Like the second lab, **this lab requires a demonstration to be performed for the TAs during lab hours, and a short lab report to be submitted**. Section 5 details what is required in the report. It is highly recommended that you complete the simulated portion of the lab prior to coming to your lab time as you will want to use your lab time to work with the live robot.

Description:

The main component of this lab is the implementation of the Probabilistic Roadmap (PRM) algorithm as described in class. Lecture notes are available which discuss the implementation of this method for an arbitrary map. The lab will have tape on the ground indicating obstacle locations for a standard map, which can be filled in with boxes if desired. The map will be static and predetermined and will be given to lab groups. Instructions on reading the map and outputting visualizations are discussed in the lab setup section below. Note that the packages used in the first lab, gmapping and amcl, will not be used in this lab. Position information will be given from the indoor positioning system (IPS) introduced in the previous labs. All software running on the robot should be of your own creation. No open source packages may be used in the completion of the PRM or shortest path algorithms, but you may use libraries to provide random samples etc. as in the second lab.

Lab Setup:

This section describes various setup procedures for the lab. Please be sure to read this section carefully as it will make the lab run significantly more smoothly for you.

Indoor Positioning System Setup (Live Robot Only)

The same IPS system used in the previous lab will again be used for this lab. The position and heading of turtlebots are published on `/indoor_pos` topic at approximately 1Hz. See Lab 1 for instructions on connecting to the lab's wireless network and for use the multiple machine configurations to control the turtlebot from your personal machine.

```
$rostopic echo /indoor_pos
```

Simulation Position and Environment (Simulation Only)

As there is no IPS system in the simulation you will use the actual position provided by the simulation environment. This is in some respects considered "cheating" as the simulation position will be perfectly accurate since it is the position Gazebo is using to generate the simulation results. As such you should remember that when switching to the actual robot your position information may contain noise not present in the simulation. The topic used to subscribe to the position in the simulation is */gazebo/model_states*; however this is a wrapper of all the model positions so it is slightly non-intuitive. An example of subscribing to this topic is included in the new example package on learn. This topic will be available once Gazebo is running, no other nodes are needed.

The environment used to run the simulation has changed for this lab. The new environment is included with the updated *turtlebot_example* package. However, if you wish to update your old package, you must first move the file *mapped_world.world* to the launch folder in your *turtlebot_example* package. To change the environment setup of the simulator, open the file *turtlebot_gazebo.launch* in the launch folder of the *turtlebot_example* package and change *"world_name"* argument to:

```
value="$(find turtlebot_example)/launch/mapped_world.world"
```

This will change the simulated world to the new version which will align with the map provided.

Reading Map Data

The maps for the simulation as well as the real world are provided on learn. Each map consists of a *yaml* and a *pgm* file. You will need to ensure that both files are present in the same directory on your computer. To start the map, run the command:

```
$rosrun map_server map_server <map_location>.yaml
```

The map server then publishes the map information to the */map* topic (remember this will be affected by your namespace setting). The map topic is an OccupancyGrid message which contains the information that represents the map.

Visualizing the PRM

The PRM can be difficult to visualize due to the expanding and large nature of the tree. Rviz also has limited visualization primitives, for example only straight lines. Example code is provided on learn which demonstrates how to visualize arbitrary curves as a Marker Array. The code is only a basic implementation and will need to be adapted to suit your own purposes.

Lab Instructions:

1. Once again you will first implement your solution in simulation. Given the simulated map implement the PRM algorithm to plan a path between the waypoints listed below. The ordering of the waypoints is arbitrary and can be traversed in any order. The starting location of the robot is also arbitrary. A tolerance threshold around the waypoints should be used and set to a radius of 0.25m.

Table 1 - Waypoints for simulated path planning implementation

Waypoint #	Position (x [m], y [m], θ [rad])
1	(4.0, 0.0, 0.0)
2	(8.0, -4.0, 3.14)
3	(8.0, 0.0, -1.57)

Note that the waypoints have not only position but orientation information as well. The minimum requirement for the lab only requires you to use the position information of the waypoints; however those who wish an additional challenge may also include orientation.

2. Beyond simply planning the path, your robot must also execute the path to traverse between the waypoints. Therefore a scheme for tracking the desired path in necessary. The PRM algorithm gives only straight line paths, you must generate a tracking controller that navigates to the path and follows it through the environment. There are many methods of performing this task and it is up to you which method you pick.

3. Once you have your implementation working in simulation, you may proceed to testing it on the live robot. As noted in the lab setup instructions there may be minor changes necessary to you code when changing to the live robot. Be sure to read the lab setup instructions carefully. A new map is used for the live robot and new waypoints are given below.

Table 2- Waypoints for live robot path planning

Waypoint #	Position (x [m], y [m], θ [rad])
1	(5.0, 5,0, 0.0)
2	(1.0, 0.0, 3.14)
3	(1.5, 4.5, -1.57)
4	(3.0, 0.5, 1.57)

Note again that all the caveats described in part 1 also apply in this case.

4. Before you leave the lab ensure that you have collected all the necessary data, plots, and information you will need in order to complete the required lab report. This should include at least: representation of the PRM graph on the given map, selected path through the graph, and IPS position as the robot executes the path.

Lab Report Instructions:

The write up should be approximately 2-3 pages long not including appendices, with a 4 page maximum. The report should have the following information, organized into sections however best fits your methods. There is no need to describe the PRM method in detail, but instead focus on the implementation details and decisions

that you made in the lab, the reasoning behind why those decisions were made and the results as described below.

1. Theory: The goal of this section is to summarize how you did everything in the lab, so we can assess if you understood the underlying principles. Briefly present the main ideas of the lab, and present motion models and methods used.

2. Implementation: Describe any implementation decisions that your group made that are not a part of the PRM algorithm. This should include but is not limited to: decisions on re-planning vs generating a global plan, determining waypoint order, executing and tracking the desired plan, generation of random or pseudo-random configurations. Include any other details which you used to improve your implementation.

3. Results and Discussion: For each section of the lab, present the simulation and experimental results, and discuss any of the major discrepancies between the two. Describe whether these results bore out in the implementation of the algorithms. Also describe any limitations that inhibited performance of the algorithms. What would you change about the current system, algorithm, or implementation to improve the performance?

4. Source Code: The source code for you implementation should be included as an appendix. Only source files and headers (.cpp, .c, .hpp, .h) are require. Please do not include any makefiles or configuration files.

This is perhaps a more vague description of the lab report and a different way of writing reports than you may be used to. The idea is to make you think like an academic, to present your ideas in a way that convinces everyone of their validity and does so quickly, without a lot of excessive detail. Since the work you are doing is coming so close to the cutting edge, it only seems natural to have you write this way. Feedback and questions on are welcome! Please submit a single PDF to the dropbox on Learn.

Lab Report Marking Scheme:

Please note that the difficulties with being the first class through these labs with the new IPS, and any issues with getting the robots functioning will be taken into account in terms of the marking of the lab reports, and we will be as fair as possible.

- Content & Results: 60% total (~20% each)
    - Background and Theory
    - Implementation Details
    - Planning Results
- Discussion of simulation and experimental results, pros and cons of methods, issues, limitations, possible improvements : 20%
- Presentation: 20%

- Nicely formatted, either single or double column, no need to double space
- Page limit observed
- Figures, tables labeled, no superfluous figures, no excessive displays of raw data
- Appropriate references provided, if necessary
- No table of contents needed