Introduction to Mobile Robot Path Planning

Unit 1: Course Introduction

- Summary -

Estimated time to completion: **30 min**

This unit is a description of the contents of the course **Introduction to Mobile Robot Path Planning**.

You'll have a quick preview of topics that you are going to learn, and you will also see some of the code that you will develop in action.

- End of Summary -

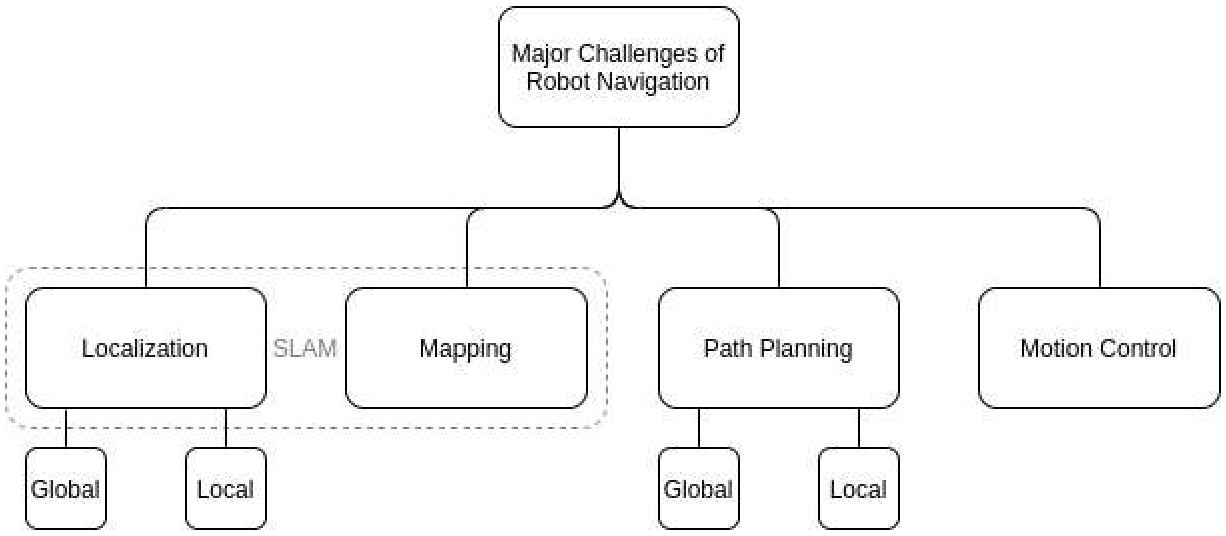
Surely you have asked yourself many times about what's the best way to get to a new place where you haven't been before. It's no surprise, then, that navigation apps are popular and widely used in our life. To solve the problem of finding the best route to a destination, all navigation apps rely on a **path planning algorithm** under the hood. Furthermore, path planning algorithms have a broad range of applications in many other significant fields, for example, computer networking, business analytics, video games, gen sequencing, and, no surprise: robotics!

In robotics, path planning is a key component required to solve the larger problem of “autonomous robot navigation”. Let's quickly see how path planning integrates into autonomous navigation and what the other related sub-problems are.

Challenges of Autonomous Mobile Robot Navigation

The main feature of autonomous mobile robots is the unique ability to move independently from a starting point to any point of choice, through a changing environment, without having any collisions. To achieve this goal, several engineering challenges must be addressed. Thus, in itself, the topic of Robot Navigation covers a very wide field of subjects!

From a higher point of view, Robot Navigation can be broken down into the following interrelated subproblems:



- **Localization**: The robot needs to know where it is
- **Mapping**: The robot should be able to build a virtual representation of its environment
- **Path Planning**: The robot needs to be able to plan a route
- **Motion Control**: The robot has to able to follow a planned route correctly

Aditionally, as roboticists, we have to deal with sensing, perception, managing uncertainty, and the tight integration of all of the above.

In the following units, we will cover the fundamentals of **path planning for mobile robot navigation**. Note that it's quite common to come across other terms for the same concept, such as *path finding*, *motion planning* and also *routing*, but let us stick to the term **path planning** for this course.

So what is path planning anyway?

Path planning is one of the most basic and vitally important abilities of any autonomous robot, which enables it to carry out assigned tasks. It consists in finding an adequate route that it can follow to move from a starting point to a target point without hitting obstacles.

For example, consider an autonomous mobile robot inside a factory, moving objects between distant areas. Let's say you want the robot to proceed to a certain point where it will be loaded and then transport its load to a different location. It should execute this task while navigating safely around people and factory equipment. A path planning algorithm would take the start and goal location as input, and produce a sequence of valid waypoints according to a map of the factory floor.

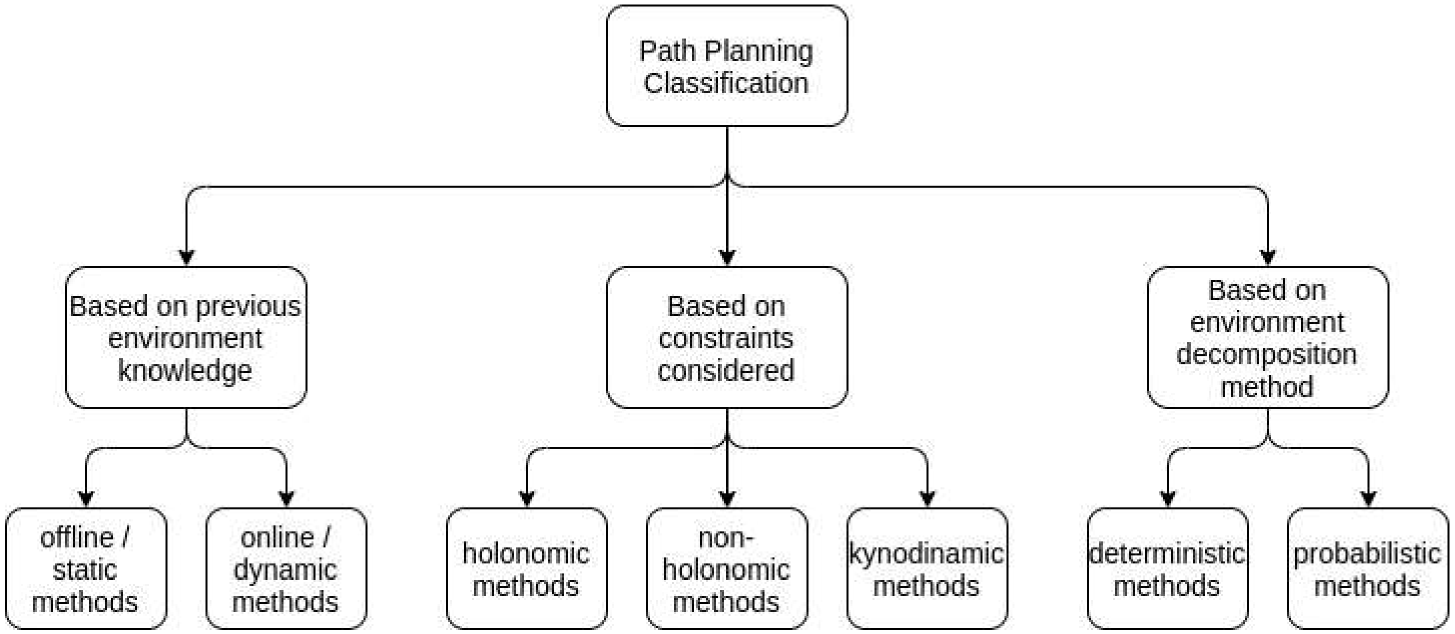
Lots of algorithms!

Because of the many environments and situations in which path planning can be applied, it also comes in many flavors!

For instance, depending on if the environment is previously known or not, path planning can be either **online** or **offline**, although sometimes these methods are called **static** or **dynamic** planning. In any case, the distinction being made refers to whether the entire path is calculated before the motion begins, with a previously existing map of the environment, or incrementally, during motion using recent sensor information.

Path planning can also be classified into **holonomic path planning** and **nonholonomic path planning**, depending on if kinemetic constraints are considered or not. If the generated path also considers constraints on velocity and acceleration, the term **kinodynamic path planning** is used.

Also, based on the decomposition method of the environment used, path planning algorithms can be classified into **deterministic planners** and **probabilistic planners**.


Path Planning Classification
Based on previous environment knowledge
Based on constraints considered
Based on environment decomposition method
offline / static methods
online / dynamic methods
holonomic methods
non-holonomic methods
kynodinamic methods
deterministic methods
probabilistic methods

Separating concerns

Path planning is, as you might have already guessed, not a trivial problem. With the many different types of algorithms that exist, there are also many pros and cons that each solution has.

But why restrict yourself to using just one path planning method? For instance, both offline and online planning capabilities are very important. Using an existing map to find the shortest path to a goal is just as valuable as being able to react to unexpected obstacles in that path.

A common way of satisfying the requirements of a robust autonomous navigation system is to use a two-level planning architecture.

In such systems, a **global path planner** is paired with a **local path planner** and both work in a complementary manner.

The *global path planner* is concerned with long range planning and uses the available map information, which can be slow, but is key to finding the most efficient path to a distant goal. It is not concerned with the robot's dynamics or how to avoid unexpected obstacles, which are left to the *local path planner*.

In this way, each planner deals with only one set of concerns: finding a traversable path to a distant goal, and following that path while reacting to unforeseen situations like the appearance of obstacles.

Therefore, global path planning and local path planning can be considered complementary solutions and are commonly built into path planning systems in real world applications.

In this course, we're going to focus on **global path planning** because it's one of the main components of a standard ROS navigation system and because it actually allows us to better introduce the basics of path planning without having to consider complex constraints typically handled by local path planners.

Do you want to have a taste?

With the proper introductions made, it is time to actually start. And... as we always do in the Robot Ignite Academy, let's start with practice! In the following example, you will be launching Dijkstra's path planning algorithm to see how it guides a robot to your goal position of choice. So...let's go!

- Demo 1.1 -

For this exercise, you will run **Dijkstra's** shortest path algorithm (which you will learn in the next unit) in order to guide a robot in its environment while avoiding any collision with the map's obstacles. To start, execute the following command:

▶ Execute in WebShell #1

In [ ]:
```
roslaunch unit1 dijkstra_demo.launch
```

After a few seconds the graphical tools window with **Rviz** should pop up. This tool will allow us to graphically visualize a robot model on an occupancy grid map. If the graphical tools window does not automatically pop up, you can manually open it by clicking on the icon with a screen located at the bottom bar.

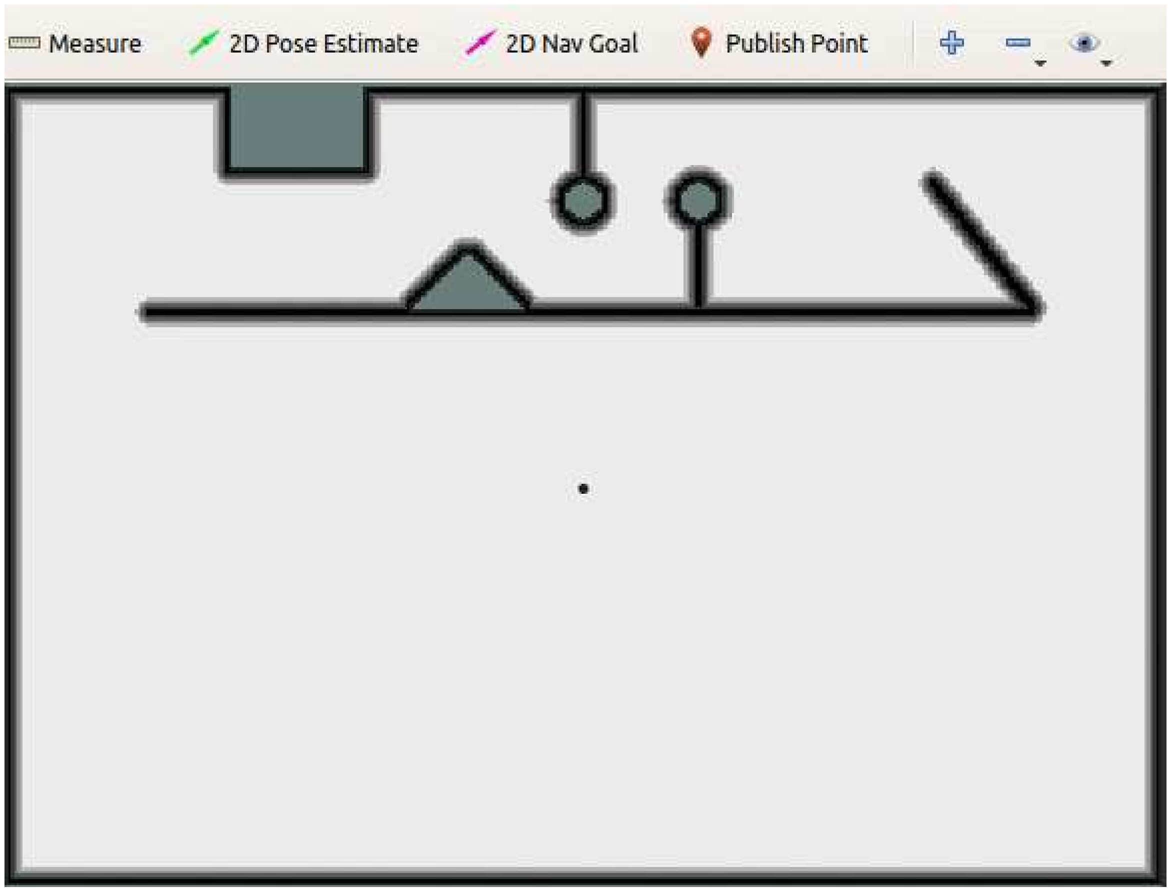
Note: You can minimize this windows to the taskbar any time you want and open it again by clicking on the icon with a screen in the lower right side of the bottom taskbar.

Let's take a quick look at how you will send a navigation goal to the path planning algorithms that you will develop.

Once you see a map like the one shown below. press the **2D Nav Goal** button from the toolbar, then click anywhere on the map to set the the robot's target position.

Expected result

After certain processing time, it should now look like the image below.



You will see a small red arrow, which indicates the target position and orientation. You will also see a solid green line, which is the global path generated by **Dijkstra**, starting at the position of the robot and finishing at the goal location. Next, you will see the robot follow the planned path.

That's really cool! But how is this actually working?

Note: Please shut down the program now by pressing **Ctrl+C** on WebShell #1.

- End of demo 1.1 -

What will you learn?

We will start the course by learning how to develop what is allegedly one of the most famous algorithms in Computer Science: **Dijkstra's shortest path algorithm**. Along the way, you will get to know important **terminology** in the field before moving to more elaborate path planning methods.

I continue by introducing **Greedy Best-First Search**, which evolves the fundamental principles set by Dijkastra to include a **heuristic** function, which in some cases can speed up the search process significantly. As your understanding progesses, you will expand your path planning skills, evolving the properties of the algorithm to convert it into the implementation of **A*** (A -Star).

Then you will turn to learn a method that takes a completely different approach to path planning, namely **RRT**. Topics in this unit include an introduction to sampling-based algorithms and a step-by-step guide to create your own RRT implementation in Python, which will greatly facilitate your understanding of this unique technique.

At the end of this course, you will be well aware of various different approaches that have been developed and applied to successfully solve the global path planning problem. Furthermore, you will be able to understand and explain the differences between them as well as the advantages and drawbacks of each. Last but not least, you will have gained solid practical experience by implementing these methods yourself.

How will you learn it?

You will learn through hands-on experience from day one! Gazebo is the robot simulator the we will be using and the TurtleBot is the robot that we will use.

Minimum requirements for the course

In order to be able to fully understand the contents transmitted, we highly recommend taking the following three course as a prerequisite:

- Basic Python: Python 3 for Robotics (https://app.theconstructsim.com/#/Course/58)
- Linux for Robotics (https://app.theconstructsim.com/#/Course/40)
- Basic ROS. You can get this knowledge by following the ROS Basics in 5 Days (Python) (https://app.theconstructsim.com/#/Course/55) Course.

Special Thanks

- To Yujin Robotics (Korea) and Willow Garage (USA), who developed the awesome TurtleBot2 (Kobuki base) robot.
- This course wouldn't have been possible without the knowledge and work of the ROS Community, OSRF, and Gazebo Team.

So, what do you say? Want to really learn how path planning works? Want to become a robotics master? Then let the real fun begin!

English proofread