Eight Degrees of Difficulty for Autonomous Navigation

Getting from Here to There Isn't Always a Walk in the Park

Last Updated: November 2020
Author: Dr. David Lu!!

It has been over fifty years since Shakey the robot started autonomously navigating around its environment to the tune of Take Five. Yet to this day, getting machines to move around without hitting obstacles remains a major area of research and development, from the major tech companies developing autonomous vehicles to hobbyists trying to drive their creations around their homes. Through various client projects, PickNik has identified eight key areas that can affect the overall difficulty of autonomous navigation.

- Mode of Mobility
- Agility
- Robot Shape
- Planning Space
- Obstacles
- Localization
- Number of Agents
- Speed

Mode of Mobility: Walk and/or Roll

Source: ANYMal Robot, ETH Zurich

Our cars don't fall over when we turn them off, but I fall over if I fall asleep standing up. This is one of many reasons why wheels are generally preferred over legs as the mode of locomotion in robotics. It is much simpler to command two motors to rotate at a particular velocity than it is to do the complex kinematics required to figure out where the best place to step is to move forward while also maintaining balance. That's not to say that there haven't been incredible advancements in legged systems in the past few years, but it does make the problem harder. On the flip side, it's much easier for me to go up stairs than a Roomba.

Agility: Go Where You Wanna Go

Source: More Parkour Atlas by Boston Dynamics

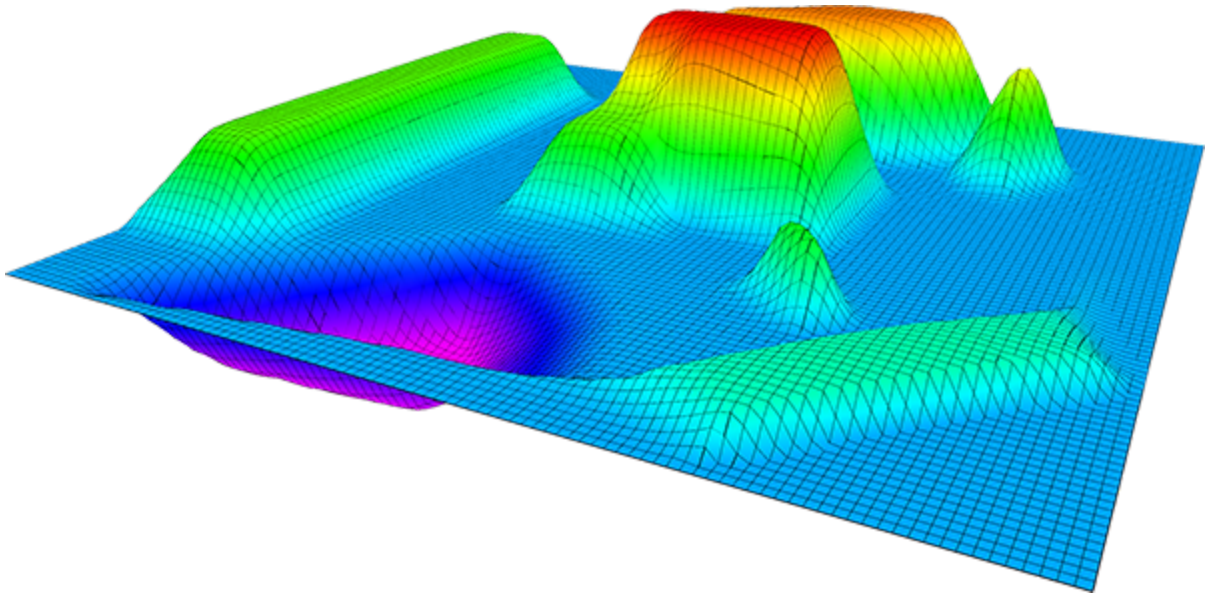
The best athletes can move in any direction almost instantaneously. They can dodge an opponent, do a quick side step and turn around nearly effortlessly. For robots, it's easiest to navigate when there aren't restrictions on which direction they can move at any particular time (aka holonomic robots). Willow Garage's PR2 was nearly holonomic in that it had to rotate its casters internally first before moving in any direction. More common, however, are differential drive robots, with two powered wheels, allowing the robot to move forward, backward and turn in place. Such a robot is not holonomic because it cannot move directly sideways. The restricted movement model slows down some maneuvers, but still gives enough flexibility to handle many situations. Car-like steering systems introduce an additional layer of complexity because cars can't turn in place, which is why parallel parking is such a pain and why three point turns were invented. Legs on the other hand can provide a lot of additional agility, although that freedom to move also comes with a lot more ways to fall over.

Robot Shape: The Shape of You

Source: Disney's Big Hero 6

The size and shape of the robot also affect how hard it is to navigate. Circular robots are common because they can rotate in place and be guaranteed not to hit anything. The same cannot be said for other shapes. A square robot directly next to an obstacle can't turn. There can also be situations where robots can fit through doorways when at one orientation, but not fit through when turned another way. One helpful rule of thumb for operating in environments designed for humans is that most doors in the United States are designed to be ADA Compliant and thus designing a robot greater than 32 inches/81 cm wide can cause problems. (ADA Compliance also means that wheeled robots won't be stymied by stairs blocking their navigation at every turn.)

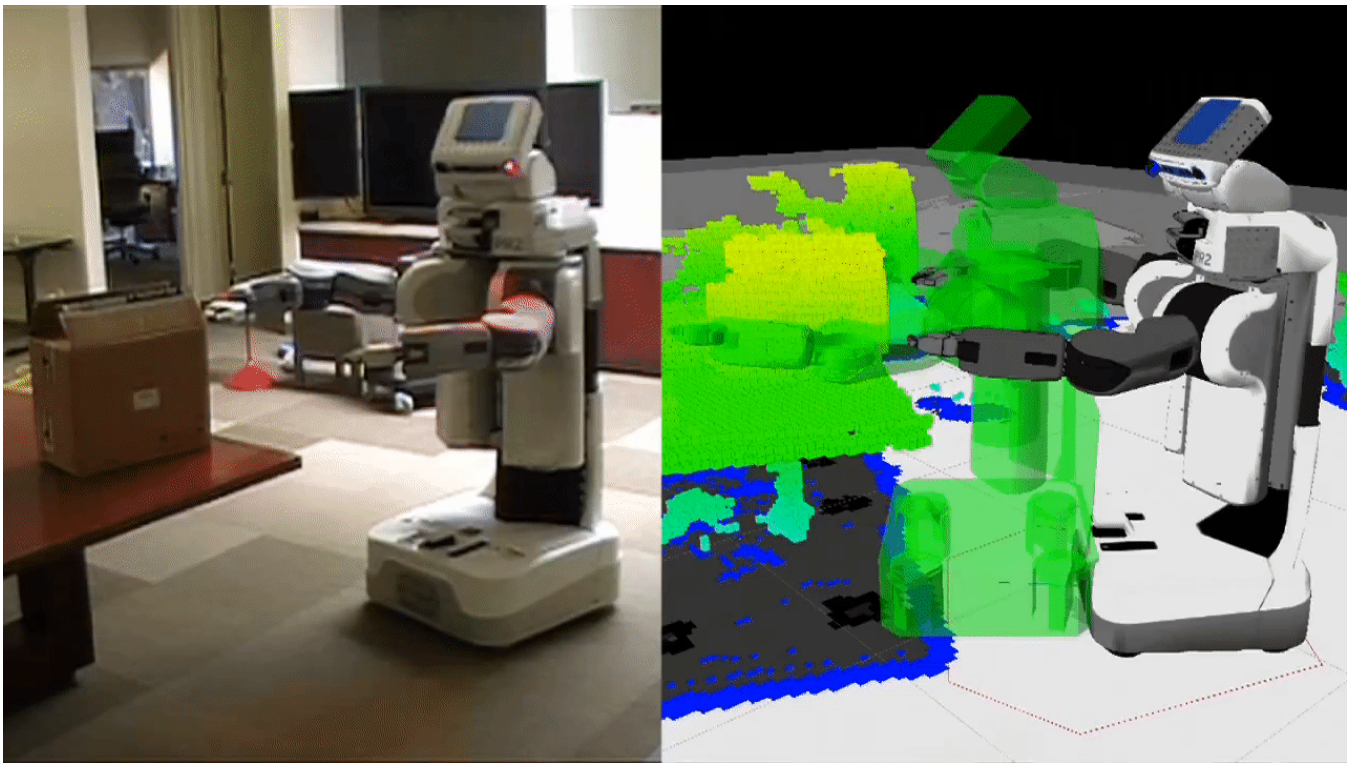
Planning Space: Flatland



Source: 3D Elevation Map by ANYbotics

Most robots operate on the two dimensional plane, where the position can be described using an x and a y coordinate and an orientation. This is sufficient for many interior places like warehouses or schools, and lends itself to very efficient representations of the environment. However, the world is not flat. A flat representation cannot represent all the places wheeled robots can drive, like a multistory parking garage. Furthermore, not all robots can operate indoors on the floor. Operating outside often requires knowing elevation as well; the shortest path may go over a mountain and the easiest path goes around the mountain. Certain applications, namely flying robots, also need a third dimension for a z coordinate, and need to track the robot's orientation with a more complex representation, like roll, pitch, and yaw.

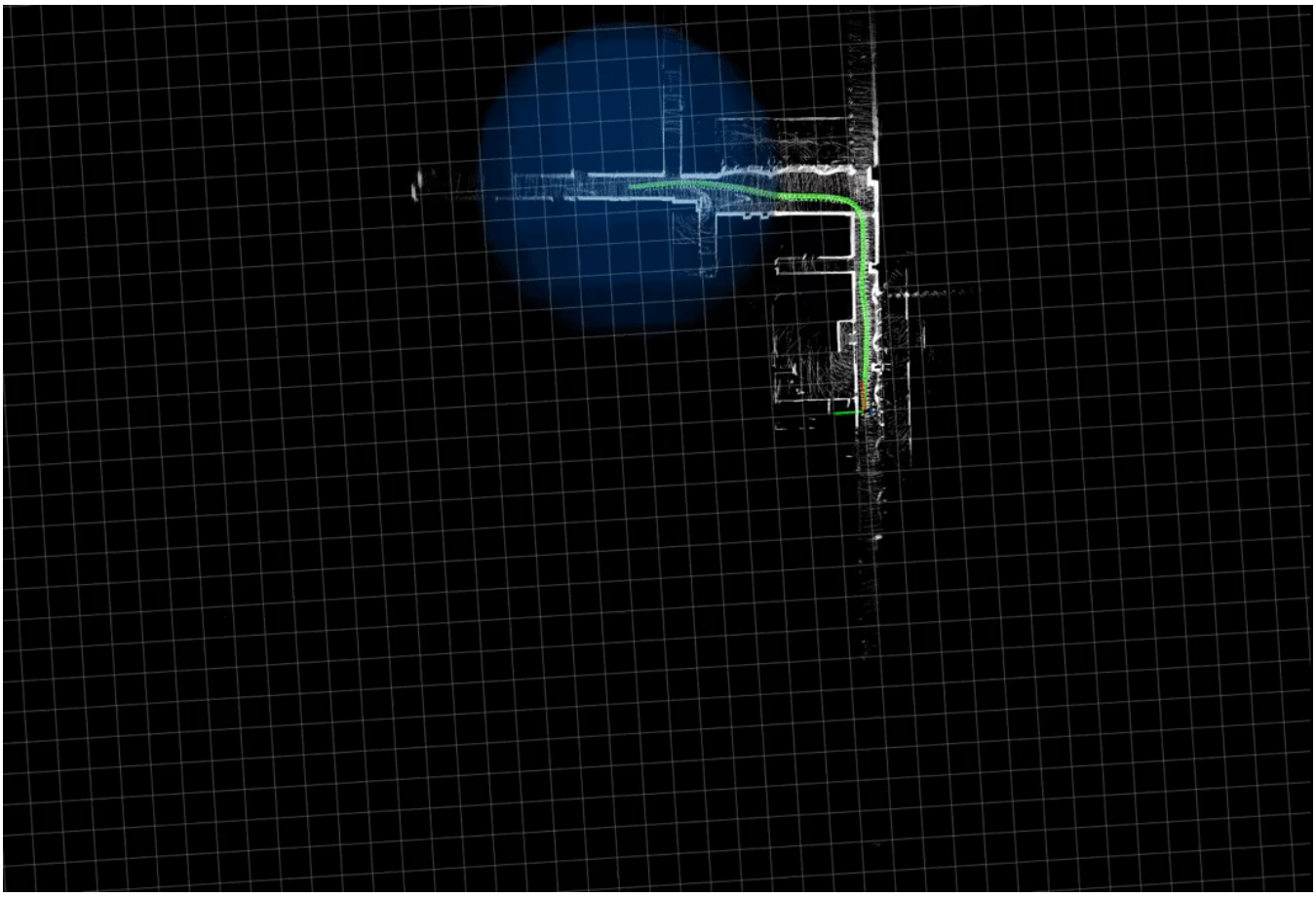
Obstacles: Avoidance Mechanisms



Source: 3D Collision Avoidance for Navigation in Unstructured Environments using OctoMap

The easiest way to navigate is in a featureless plane, which is why people often learn to drive in empty parking lots. However, most mobile robots need to avoid the obstacles around them, if for no other reason than roboticists hate getting their shins bumped all the time. This is relatively simple if the robot knows its entire environment beforehand, but the introduction of dynamic obstacles means that the robot will need a sensor suite to detect whatever obstacles are thrown at it. The standard for many years was to use a planar laser scanner, but that only worked to avoid obstacles that were at the exact height of the laser. The real world, as it turns out, has more than two dimensions. Thus, robots that relied on laser scanners could avoid table legs, but would often hit the table top, or run over small obstacles like toes. Some applications use standard RGB cameras, but since 2010 and the introduction of the Microsoft Kinect, three dimensional sensors have become fairly standard for robots. They recognize many more obstacles with high accuracy, but come with their own calibration complexities and limitations. Sensing an obstacle is also only the first step, after which the robot may need to recognize it, predict where it's going and plan around it.

Localization: A Maze of Twisty Little Passages, All Alike



Source: Modified from 3D Lidar-SLAM With Loop Closure

While having no obstacles around the robot makes it easier to not hit anything, it makes it harder to know precisely where you are. Even adding GPS will only give your position within a few meters. Instead, most robotic systems rely on a system of localization based on existing static obstacles like walls. This can be easier when a map of the environment is known beforehand, otherwise you have to do simultaneous localization and mapping, which is a complex field unto itself. However, even when you know the map fully, the problem of localization can still be difficult, depending on the accuracy of the sensors and how unique the environment is, for example, if the robot must navigate through an environment full of visually similar aisles.

Number of Agents: Multiplayer Navigation

Source: Optimal Reciprocal Collision Avoidance / UNC

Navigating one robot can be hard enough. You could also make a system of multiple robots navigating around where each robot pretends it is the only robot. However, there are many benefits to getting a fleet of robots to communicate with each other and coordinate their motion. Where it gets really fun is when the other agents that the robot must navigate around are people. The difficulty of that can vary, depending on whether the people are trained to be around robots, or they're unsuspecting customers who've never seen a robot before.

Speed: Of the Essence

Source: Unitree Robotics

The last component to touch on that can make navigation significantly harder is speed. The difficulty is not only a function of robot speed (Shakey goes a bit slower than Tesla's "autopilot"), but also the computation availability. It's one thing if you're trying to drive an autonomous vehicle with the full force of a Fortune 500 cloud infrastructure behind you; it's another if you're trying to run the whole thing on an Arduino board. Moving at greater speeds without having adequate processing means can lead to one of the least desirable outcomes in navigation: collisions. In the name of safety, it is often better to execute slower moves that you can guarantee will not collide with obstacles than to move faster on a trajectory that might collide with something.

The eight degrees listed above represent some broad ways in which robot navigation can be difficult. As always, the devil is in the details. The specific context in which *your* robot operates will present unique challenges that will require custom domain-specific solutions. At PickNik, we've expanded our navigation capabilities to provide services that fit your needs.

Previous Draft and Outline

While robotic arms represent a large swath of innovative robotics work today, and contain many interesting subproblems including perception, controller tuning and planning, in many cases there's one thing they don't have: mobility. Getting a machine to autonomously navigate around an environment has been a fundamental part of robotics for at least the last half century, and now, it's one of the fundamental competencies that PickNik Robotics offers.

In this essay, …

Use Cases

The concept of mobile robots covers a wide range of robot types, from the simplest differential drive robot, to walking robots, cars, and beyond. Some of these use cases are easy, and some represent grand challenges at the forefront of technological research.

Use Case 1:

1. Use Cases
    a. Mostly Fit
        i. Laundry Basket demo
            1. https://www.cs.cmu.edu/~maxim/files/planfor3Dnav_icra12.pdf
        ii. https://www.youtube.com/watch?v=sot6gjj3SzU
        iii. Tuck the arms
        iv. Other Shapes
    b. Bad Fit
        i. Other Motion Models
        ii. Cars - Autoware et al
2. Components
    a. Decomposing Navigation
        i. Global Plan
        ii. Local Plan
    b. Costmaps
    c. Coordination
3. Adaptability
    a. Three Robots One Node
        i. https://www.youtube.com/watch?v=mKmqgVUbQQM
        ii. https://www.ros.org/news/2010/08/three-robots-one-ros-node.html
4. Best Practices
    a. Parameter Files
5. Code Bases
    a. OG Navigation
    b. Robot Navigation
    c. Navigation2
    d. Plugins

i. SBPL
ii. Hector