# CS 081 - Robotics - Dynamic Mapping

Carter Kruse, Alex Fick, Kevin Cao, Aneesh Patnaik

## Introduction

In recent years, there have been rapid advancements in the fields of autonomous navigation and the proliferation of delivery services, especially expedited by the COVID-19 pandemic, as everything from groceries to car deliveries to retail purchases were being made online and brought straight to consumers' doorsteps. In order to increase the safety and efficiency of these operations, some have begun to focus on combining these two technologies. However, for an autonomous vehicle to be able to navigate through a real-world environment, it must be able to adapt to moving obstacles within its path.

The ability of robots to navigate in dynamic environments is a complex task, requiring intelligent decision-making and robust algorithms. By addressing the challenges posed by moving obstacles, we can ensure the safe and seamless integration of robots in our society, fostering trust and maximizing their potential for enhancing efficiency and convenience.

In this paper, we approach this broad problem from a narrower perspective with a specific scenario: an autonomous robot in a gated community needs to be able to map the community, then navigate to specific targets within the community (to deliver food, groceries, etc.), without colliding with any obstacles, whether they be humans, other vehicles, wild animals, etc. As such, this problem is tackled as follows: first, the robot performs frontier exploration to iteratively map the closed environment. Second, the robot is sent a list of target destinations, and it then plans the most efficient path to reach all of the said destinations once using the map that it had just mapped. While the robot is moving to map the environment or to reach each target destination, it has to navigate itself around any moving obstacles that are in its path or too close to its path. We consider the robot as having completed its job once it reaches all of the points in its list of target destinations, which can only occur after the robot has not only utilized frontier exploration to map its environment, but also determined the optimal path to reach all of its customers. In particular, this paper makes four contributions:

- We introduce an algorithm to identify central points along the mapping frontier, which then moves the robot to the nearest such point, where the frontier is re-evaluated, and the process happens again. This essentially creates a BFS region growing algorithm, as the closest point along the frontier is always the one to be explored first.
- We establish a procedure to distinguish a moving obstacle from a static one, so that the moving obstacles do not get mapped, and thus do not influence the path planning that is done later.
- We implement a way for the robot to avoid any moving obstacles within the robot's path, which prevents the robot from crashing and allows the the robot to dynamically update its path so that it can both move around and past the obstacle on its way to its destination.
- We approach a TSP-esque problem of visiting a specific list of targets, using an A* path planning approach combined with a modified Held-Karp algorithm that doesn't consider the distance for the robot to travel from the last target back to the start (since we don't care where the robot ends up)

The remainder of this paper is structured as follows: the next section provides an overview of related work. Section III describes the problem in detail. Section IV discusses the modules employed to successfully map and navigate through the dynamic environment. The following section presents experimental results from numerous simulations with multiple robots. The paper concludes with lessons learned and a discussion of future directions, as well as potential ethical concerns.

## Related Work

Google announced its self-driving car project back in 2009. During the announcement of this project, Google had promised to have an autonomous vehicle model in the market by the year 2020. Evidently, this has not been the case, and the creation of reliable and safe autonomous vehicles that could be trusted for mass consumption remains a problem that has yet to be solved. While Google missed its deadline and the end of this project remains unforeseeable, there has been a considerable amount of reputable work done attempting to address this issue. When it comes to obstacle avoidance and path planning within the context of nonstatic environments, in particular, research has been done by both academia and industry to try solving this issue. Vehicles need to be able to not only map quickly and precisely real-time changes in its environment, but also plan a safe and efficient path to its destination. Therefore, improvements in the ability for autonomous vehicles to map their environment are extremely important towards their performance.

Within the past decade, the use of machine learning, particularly deep learning, has revolutionized the way autonomous vehicles understand their environment. Convolutional neural networks (CNNs) have been used for image recognition and object detection tasks, which can help autonomous vehicles detect and identify different kinds of obstacles within its environment such as a pedestrian, a road sign, or another vehicle. CNNs can even distinguish between different kinds of vehicles, different kinds of traffic signs, and even the direction a pedestrian is facing. This level of detailed understanding is critical for the safe and efficient operation of an autonomous vehicle, empowering it to predict the short-term future behaviors of other road-users and react accordingly. In a similar vein, reinforcement learning algorithms have started being implemented to help the decision making process of a vehicle, determining the optimal way to navigate an environment to reach a target destination. Reinforcement learning algorithms can be employed to help the vehicle figure out the type of path it should take to successfully map its surroundings, in addition to being applied to an existing mapped out environment. While our final project does not utilize any deep learning in order to differentiate between different types of obstacles within the robot's surroundings, our robot's ability to discriminate between static and nonstatic obstacles within its environment was conceptually derived from these more sophisticated methods. We also attempted to use an ML classifier in order to classify cells as occupied or nonoccupied by the history of their laser sensor readings. This worked relatively well, but we figured we could have implemented a simpler method that also has high degrees of success in classification without needing invest time in creating testing data.

This emergence of the applications of machine learning in robotics have helped streamline frontier exploration. For example, Belvadi et al. (2017) developed a frontier exploration technique for 3D autonomous SLAM utilizing K-means based hierarchical clustering algorithms. Hierarchical clustering algorithms can divide the environment into different regions that the robot can prioritize for exploration in order to optimize the path required for the robot to map out its environment. Though this technique is used in a 3D space, it uses the same principles of dividing the environment into cells of 3 states: unoccupied, occupied, and unknown. For our own final project, while we do not implement SLAM for our final project, we also implemented a frontier exploration technique which is similar to the one described in the paper. We did, in fact, originally test the K-means based hierarchical clustering algorithm, but we ultimately settled with a frontier exploration algorithm that utilized something more akin to a modified BFS algorithm.

Researchers at Waymo and Google who were responsible for developing Google's self-driving car have found ways and methods to process LIDAR data into information that can discriminate between the different types of 3D objects that are sensed by the robot. Much of current autonomous driving sensor detection methods support only 2D vehicle detection but actual autonomous driving involves the detection of 3D perception of vehicles. There will also be a decreased risk of failure in autonomous driving if vehicles could support 3D vehicle detection and also the ability to classify the kinds of obstacles that are in its environment, which would in turn help predict nonstatic environment behavior. Again, similar to the deep learning methods for the classification of environment objects, these methods of processing LIDAR data into information that can discriminate between different types of objects helped inspire how our final project's robot is able to discriminate between static and nonstatic obstacles.

Researchers at Carnegie Mellon University have also proposed an approach to path planning in dynamic environment that takes into account a 4th dimension of time. In particular, this 4th dimension is only considered in locations where there is a high probability of a collision taking place; however, in other locations, a usual low-dimensional (i.e. something comparable to standard A* search) is employed. This method has significant time advantages over other approaches that avoid the dimension of time, since those approaches almost always require significant re-planning, due to moving obstacles in the search space, improving how autonomous vehicles can predict future arrangements of its environment. We also attempted to modify our path planning algorithm within our final project's code to avoid locations where there is a high probability collisions (e.g. where nonstatic obstacles were more frequently found). However, because we ultimately decided to make the nonstatic obstacles behave exactly like the random walking robot from PA0 instead of patrolling a specified area of the map as we originally intended, we felt that it would be more optimal to have the robot focus on avoiding the obstacle and then recalculate the new path after avoiding the obstacle.

**Sources:**


- Belavadi, S.; Beri, R.; and Malik, V. 2017. Frontier Exploration Technique for 3D Autonomous SLAM Using K-Means Based Divisive Clustering.
- Bahraini, M.S.; Rad, A.; and Bozorg, M. 2019. SLAM in dynamic environments: A deep learning approach for moving object tracking using ML-RANSAC algorithm.
- Chen, Y.; Liu, M.; Schumann, T.; and Vallespi-Gonzalez, C. 2019. A general pipeline for 3D detection of vehicles.
- Vemula, A.; Muelling, K.; & Oh, J. 2016. Path Planning in Dynamic Environments with Adaptive Dimensionality.


# Problem Statement

A robot that will explore and map (with path planning) an unknown environment with possible dynamic obstacles. After mapping out the environment, the robot will visit all target destinations in the most efficient path. While the robot is traveling, the robot will have to avoid moving obstacles while ignoring the presence of these moving obstacles for the purposes of mapping.

**Inputs:**

- LIDAR sensor data
- A list of "customer" locations that need to be visited

**Outputs:**

- A map of the environment that ignores nonstatic obstacles

- Efficient robot behavior when traveling to customer locations

## Description of Approach and Implementation

**Frontier-Based Exploration**
In implementing a frontier-based exploration, the aim is to identify central points corresponding to the boundaries of the map. The boundaries are determined by an occupancy grid, which is responsible for storing the information associated with the mapping of the environment.

To implement such a frontier-based approach to exploration, we first need to identify where the boundaries are, which involves taking differences between the various cell values, allowing us to compute the boundary between "free space" and the "unknown". After these cells have been identified, a region growing algorithm is implemented (using BFS) to determine the different "regions" (groups of points).

This allows us to disregard regions with relatively few points, which tend to be those that develop due to noise in the readings of the LIDAR sensor, resulting in an occupancy grid that does not represent the real world with 100% accuracy. After the regions are appropriately identified, we may determine the geometric mean of these groups of points, which allows us to determine the center of the regions identified.

This works relatively well given the set-up, as the robot's mapping depends on a ray-tracing algorithm that produces straight lines on the boundaries.

Initially, a $k$-means clustering algorithm was introduced to identify the areas, yet this was determined to be relatively unsuccessful, as it is difficult to determine the appropriate value for $k$ which accurately identifies the different boundary regions. Thus, the frontier-based exploration using a region growing BFS algorithm seems appropriate.

**Mapping with Static & Dynamic Obstacles**
The mapping of the environment involves a ray-tracing algorithm that reads in the LIDAR sensor data and updates the appropriate occupancy grid based on the distances to obstacles. This requires keeping track of the odometry of the robot, specifically the position and orientation.

The ray-tracing algorithm is a specific implementation of Bresenham's Algorithm, which is used to mark "occupied" and "free space" cells in an efficient manner. The algorithm involves interpolation of points/cells, which may lead to slight inaccuracies, which we aim to correct in the implementation of the mapping.

The mapping algorithm needs to be designed to be more sophisticated due to the presence of nonstatic obstacles within the environment, which we do not want the robot to map. We approached this by keeping tracking of the history of the readings of each cell in the map. With each iteration of laser scanner data, we do not directly fill in the cells in the grid. Instead we update an array that stores the history of each cells' occupied/unoccupied readings, and then from the history of each cell, we make the determination of whether to classify the cell as a free space or occupied space. The determination we ultimately went with involved calculating the probability/percentage of overall occupied space readings of a cell and make the determination based on whether this percentage was over or under a certain threshold that we set, but we did consider other methods which included feeding cell histories to an ML classifier. This threshold that we set had to be optimized such that it didn't erase or miss any static walls within the environment, but it also had to ignore moving obstacles.

**Exploring Areas**

In "exploring" the unknown areas of the map, the easiest algorithm to implement is simply determining the closest boundary point, based on the map of the environment encoded in the occupancy grid, and visiting that point. This algorithm is iterative, as after each point is explored, the boundary points are re-calculated, and the new closest point is explored.

This models a DFS approach to mapping, where a single area of the map is fully explored before transitioning to the next boundary point. In this way, the algorithm rigorously covers the entire environment, yet may not finish timely if the map is relatively large.

Logically, the robot would first be initialized and given a short period of time to map its visible surroundings. It also helps to move the robot just slightly during this initial phase in order to get a better rendering of the visible surroundings around its starting location. Then, using this beginning map, the frontier-based exploration algorithm would generate points for the robot to visit, and the robot would simply pick the closest one to move to (closest not in terms of euclidean distance, but rather closest in terms of shortest possible path). Then, once it reaches this new point – and in turn having updated the grid while doing so – the robot will recalculate the new frontier points and then move to the closest one again, which will be done in repetition until the frontier-based exploration algorithm can no longer find any frontier points to visit.

**Traveling Salesperson Problem (TSP)/Path Planning**
Once the environment is appropriately mapped, we are ready to visit the "customer" points, which were given as an input. We plan a path to visit each one of these customer locations by writing a path planning algorithm that closely matches that of the "traveling salesperson problem," though it does not involve visiting the starting point after the target destinations have been visited. The algorithm, which is based off of the Held-Karp Algorithm, determines the order of customer locations to traverse such that it produces the shortest possible path, given the occupancy grid that was just mapped through our frontier exploration algorithm. Because of this module's dependency on the occupancy grid that is explored, it is essential that our frontier exploration and mapping algorithm works to high precision. It should be noted that we "padded" the occupied cells of the final occupancy grid in order to ensure that the path the robot takes won't bring the robot too close to the wall.

# Experiments

**Experimental Setup**
In testing the implementation of the modules, as both individual units and one integrated final product, the virtual simulation was primarily used due to the increased accuracy of the LIDAR sensor. In the real world, the LIDAR sensor seems to produce slightly inaccurate results, likely due to errors in both measurement and odometry, alongside other complications. To account for these issues would be taking away from the objective of our project, and thus, the simulation serves to demonstrate the appropriate implementation and experimentation conducted.

Before we even considered testing the integrated final product, we decided to conduct unit tests in order to ensure that each of our individual modules worked as desired at the very basic level (see our slide deck for FP4). Once our unit tests worked, it was time for us to integrate the unit tests. The environment used in our integration test is `newmaze.png`, which we feel provides enough complexity to test the robustness of our approach to this project, but also not too much complexity so each individual test trial wouldn't take too long.

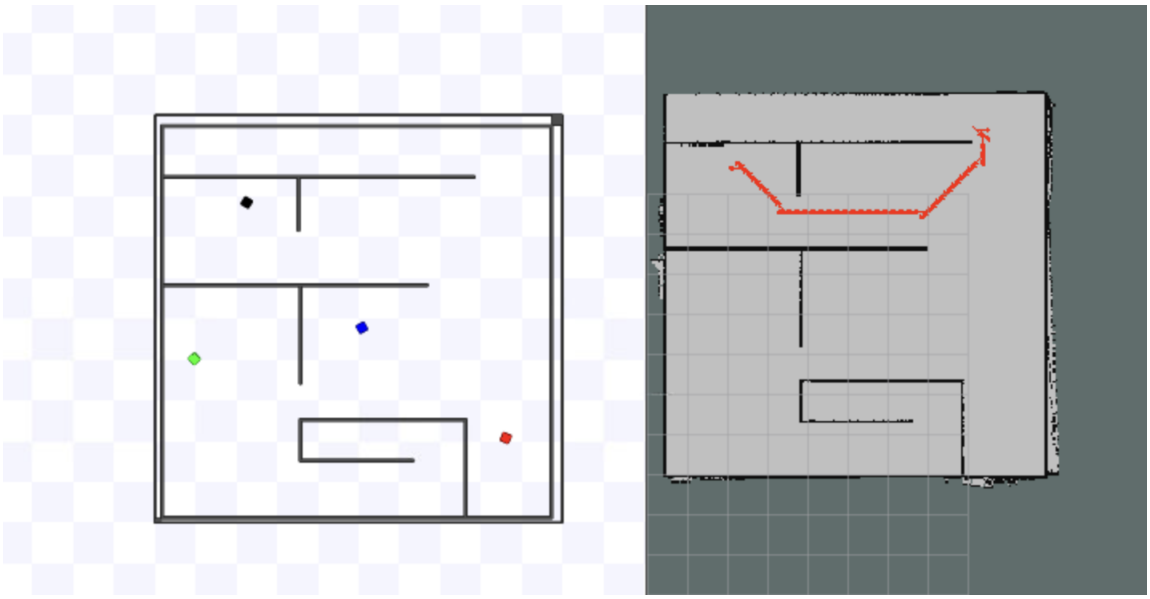
To create the dynamic obstacles in the environment, we opted to simply introduce colorful robots that move in a random walk fashion. These "random walk" robots have the same specifications as the controlled robot, with the exception of the fact that they make no effort to avoid hitting the controlled robot. In this sense, the algorithm implemented for the controlled robot is entirely responsible for avoiding the dynamic obstacles, as it would be in a real world situation (for example,

there is realistically no expectation for autonomous vehicles to be able to avoid collisions when other roadside obstacles or vehicles initiate them).

Finally, it should be noted that while our code has been compiled into one script and written successfully, for the sake of experimentation, we have two separate modules for the exploration and TSP portions of the project. The exploration module will thus run until there are no more frontier points to be discovered and then export an occupancy grid in the form of a .csv file. Then, we can simply have our TSP module read in the saved .csv file (without needing to rerun the exploration module every time, which takes time), generate the occupancy grid, and then have the robot conduct the TSP on it.

**Observed Effectiveness and Limitations**

Overall, our robot successfully accomplishes its task of fully exploring its surrounding environment and then being able to visit multiple points-of-interest within the environment in the most efficient manner using the occupancy grid it just mapped.



Link to drive with images and videos of testing: https://drive.google.com/drive/folders/1imArk8sLQ3Gl4jWsRKF-OoytvJj8cwiO?usp=sharing

There are however some limitations within our code, however these do not really take away from the robot accomplishing the task at hand for the most part.

First, our frontier-exploration algorithm does not necessarily produce the most efficient path of points to visit in order to fully explore the entire map. This limitation is very reasonable as the robot has no way of knowing beforehand how the rest of the map would look like without exploring it in the first place.

Another limitation comes from noise. While the simulation significantly limits the amount of noise introduced in the LIDAR measurements, mapping, and odometry, there is still a relative amount of noise that creates difficulty in implementing the algorithms. A very apparent result of this noise are a few random smudges of filled occupancy grid points outside the confines of the map, but fortunately these do not take away from the robot its ability to completely map its environment nor its ability to traverse the customer locations in the most efficient way possible. To account for this, various "work-arounds" were implemented in the modules described previously, which accounts for standardization of most environments. Regardless, we anticipate there may still be issues surrounding noise, particularly in the real world, which may be a further area of research.

In addition, there are some slight limitations when it comes to our occupancy grid mapping. Due to the presence of nonstatic obstacles that we do not want to map, we had to modify our occupancy grid mapping algorithm to take this into account when analyzing laser sensor data. Our method of using cell histories and the percentage/probability that a cell is actually occupied means that we must choose some sort of threshold value that is both high enough such that cells temporarily occupied by nonstatic obstacles are not permanently marked as occupied and low enough such that parts of static walls are not missing in the final map. We found the threshold value that produces the best results with this respect, but there are certain moments within the simulation such that a wall visible to the laser scanner of the robot is temporarily not marked as occupied (since cell histories could require some time to update such that the threshold is crossed and also noise). However, by the end of the exploration, all walls are marked present.

# Conclusion

Ultimately, we find that we were able to successfully program a robot such that it is able to completely map an unknown, bounded environment which contains dynamic obstacles, and then utilize the map it had just generated in order to visit a list of target destinations in the most efficient manner. While we find that there exist some limitations in our result, these do not really take away from our robot's fundamental ability to perform both the exploration and the TSP tasks. Some future work that could spring off of this final project include using a live robot and a camera in order to have a more detailed map of the environment and its contained obstacles.

## Ethics

In the real world, various similar technologies exist and are becoming more prevalent, as autonomous vehicles become more advanced. There are number of sensors available for exploration and mapping, including LIDAR and visual cameras, which continue to improve to this day.

Through a significant amount of research and development has furthered the field of robotics, specifically autonomous vehicles, many system have yet to be perfected to the point where they are marketable to customers.

Autonomous vehicles need to avoid obstacles, particularly in situations where the vehicle is fully autonomous (with no driver control/component whatsoever). In further research and development of autonomous vehicles and other forms of mapping with dynamic elements, there are various ethical considerations to be taken into account, as follows.

**Privacy** - As with various mapping technologies, privacy is a significance concern for researchers in the field. There are laws and regulations on exploring private areas, and such must be taken into consideration when creating an autonomous robot. There is also risk that autonomous vehicles could be utilized for spying (in the context of both civilian and military applications). In the implementation discussed in this paper, privacy concerns are not considered, as we would need to adjust the range of motion of the robot in order to account for potential privacy invasions.

**Deontology View** - Autonomous vehicles must be programmed to follow the law, specifically traffic regulations, as they are not "above the law" in any way different from regular drivers. In this sense, further development to identify street signs, stop lights, and other traffic signals would be necessary in developing a robot that is entirely autonomous as a vehicle to provide transportation. As mentioned earlier in our discussion of related work, researchers have created and implemented convolutional neural networks to process camera images in order to identify such traffic signals to high degrees of success.

**Utilitarian View** - The algorithms implemented in autonomous vehicles must not only follow the law, but should maximize safety, by ensure that the occupants of the vehicle are not placed

in harm's way as a result of faulty operation of the vehicle. In this sense, "efficiency" may have to be disregarded to allow for safety considerations, such as a limit on the maximum speed and acceleration of autonomous vehicles. Public policy may prove effective in implementing various control mechanisms.

**Job Security** - As with most technologies, autonomous vehicles threaten the jobs of many "blue collar" workers; those that are responsible for the transportation of goods from place to place. Incidentally, trucking is most common job within the United States, so there will be an out-sized effect on the employment of blue collar workers. When developing a robotic system that is able to operate in a dynamic environment similar to the way a human might, the moral dilemma naturally arises as to whether it is appropriate to replace human jobs with technology. Economists argue that the productivity gains from self-driving vehicles will outweigh this threat to job security; however, throwing millions of Americans out of work represents a fundamental threat of the dignity of these people.

**Environment** - Though autonomous vehicles may actually serve to benefit the environment through lower emissions, there are naturally still considerations surrounding the impact of such systems on the environment.

Overall, with these ethical considerations in mind, we find that autonomous vehicles do present certain ethical dilemmas which can be mitigated with proper technological advancement and public policy. More research and development can be made such that there are less "accidents" with these vehicles such that privacy and safety issues are minimized. Moreover, adequate government regulation can play a role in ensuring that these vehicles do not have too large of an impact on job security, public safety and privacy, or the environment. The convenience, productivity, and quality-of-life benefits autonomous vehicles could potentially bring to humanity are limitless. As researchers work to develop better models that can deliver more safety and privacy, we believe that it is justified to introduce autonomous vehicles given adequate government regulation that goes with it. This way humanity can harness the numerous benefits of autonomous vehicles without needing to worry about the potential ethical dilemmas they carry.

# Allocation Of Effort

**Kevin**

- Mapping with the discrimination between static/nonstatic obstacles
- Integration of the separate modules
- Creation of testing environment
- Final paper writing

**Carter**

- Frontier-based exploration
- Robot's path planning
- Integration of the separate modules
- Final paper writing

**Alex**

- TSP module
- Robot's path planning

- Integration of the separate modules
- Final paper writing

**Aneesh**

- Obstacle avoidance
- Final paper writing