Traversing through unknown vegetated environment using proprioceptive sensors

David Seong, Weichen Qiu, Benny Zhu
A 16350 Final Project
April 29, 2025
Code: github.com/w3ichen/vegmap

1 Abstract

Robots' navigation of unstructured environments, especially vegetation, poses a significant challenge, as traditional perception-based methods are excessively conservative when it comes to determining traversable vegetation. This paper discusses a novel planning approach that assumes that all obstacles can be traversed at an estimated cost. Upon collision with obstacles, the true cost of the obstacle is determined, which is then used to update the estimated cost of other obstacles. Then a new path is generated based on the updated estimated costs for a more cost-efficient navigation. An end-to-end simulation was developed using ROS2 Humble and Gazebo, showing the successful planning of the algorithm to prove the validity of the method.

2 Introduction

Many approaches for robots navigation have been developed; however, these attempts have been mostly tested in structured and urban environments. Robot's navigation in unstructured, vegetated areas is still a challenge to robot navigation. The many types and properties of vegetation such as bushes, grass, twigs, trees add complexity to the problem - some obstacles are easily traversable, some are harder, and some are untraversable. Therefore, traditional methods that classify obstacles' traversability using exteroceptive (i.e. LiDAR and cameras) often fail to generate a reliable path as shown in Figure 1, where the LiDAR identifies all neighboring vegetation as untraversable, even though they are traversable. This poses the question of how a robot can plan an efficient path through a heavily vegetated environment.

We propose a novel approach that initially assumes that all obstacles are traversable with an estimated cost, based on prior knowledge or intuition, then updates the cost of obstacles with which the robot collides along its original path. Once the true cost is determined using its proprioceptive sensors, the cost database is updated and a new path is planned for more efficient navigation.

We implemented an end-to-end simulation to demonstrate our planner using ROS2 Humble, Gazebo, and Nav2 stack. Figure 2 is an example from our simulation, where the baseline (navfn planner with obstacle layer) could not find a path, but our planner could.

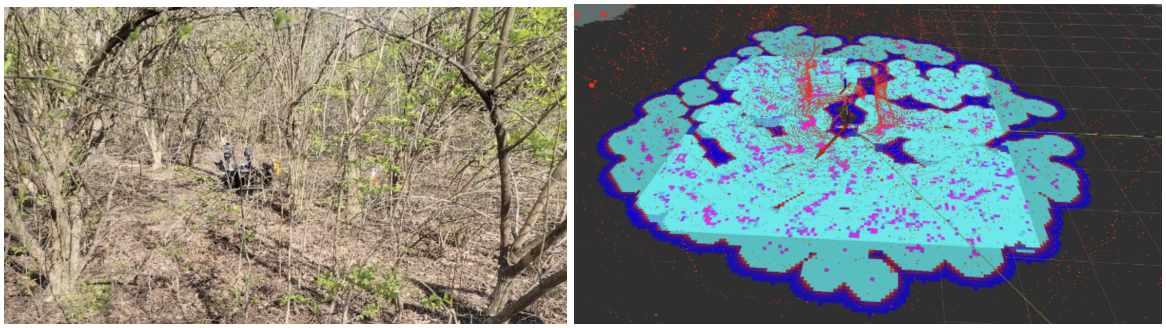

Figure 1: LiDAR data in vegetated area

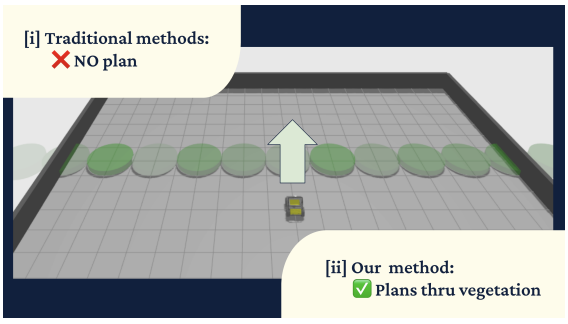

Figure 2: Baseline scenario with no plan

3 Approach

3.1 The math

In our simulated environment, the state of the robot is $(x, y, \theta, costmap)$, where x, y, θ is the pose of the robot with (x, y) representing the robot's location in the 2D space and θ is the orientation. The cost map is included as part of its state, as it is continuously updated as the robot collects new information about the environment.

The cost function is $f(state) = g(state) + \epsilon * h(state)$, where g is the cost to reach the current state from the starting state, ϵ is the weight applied to the heuristic. As the robot traverses the map, the contact model determines the cost of the traversal of the obstacles.

In the simulated environment, the ground truth cost of the obstacles is dependent on how much speed reduction it applies to the robot. However, the initial estimated cost of the obstacles is sampled from a Gaussian distribution, with an estimated mean corresponding to a type of vegetation. For example, our grass obstacle cost was sampled from a Gaussian distribution with a mean of 20 and a standard deviation of 15. So different patches will have different costs depending on the sampled cost.

The cost is calculated by $velocity_{desired} - velocity_{actual}$, which returns the difference between the robot's commanded velocity and its actual velocity. When the robot encounters a traversable obstacle, the obstacle will either reduce the robot's speed by a factor, resulting in an output velocity of 0.01 0.99 m/s or make the robot slip, resulting in an output velocity of 0 m/s (i.e., if the desired velocity is 1m/s and the resistance causes the robot to slow down, resulting in a velocity of 0.4 m/s, the cost would be 0.6).

Once the cost is calculated, it will update the cost of similar obstacles in the data base, updating the Gaussian distributions with their mean and standard deviation. Gaussian distributions are used to represent the uncertainty in the measurements, noise, and variability of each vegetation instance. In order to take the existing Gaussian distribution that exists in the database and incorporate the new measurement, Bayesian updating is used. The formula is:

$$\mu_{\text{post}} = \frac{\mu_{\text{prior}} \cdot P_{\text{prior}} + \mu_{\text{obs}} \cdot P_{\text{obs}}}{P_{\text{prior}} + P_{\text{obs}}} \tag{1}$$

Where post = posterior, obs = observation

$$P_{\text{prior}} = \frac{1}{\sigma^2_{\text{prior}}} \tag{2}$$

$$P_{\text{obs}} = \frac{1}{\sigma^2_{\text{obs}}} \tag{3}$$

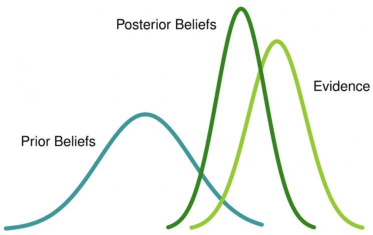

Figure 3: Bayesian Updating

This is illustrated in Figure 3, where prior beliefs are the Gaussian distribution stored in the cost database and represent the current estimated cost of an obstacle; for example, trees could have a mean cost of 250 with a standard deviation of 6. The evidence is the newly determined cost with its standard deviation set to 1/2 of the prior beliefs to indicate its higher certainty. Furthermore, if the evidence is 254 then its standard deviation is 6/2=3, causing the Bayesian update to return a new posterior belief of 253.

3.2 The algorithm

The algorithm is illustrated in Figure 4.

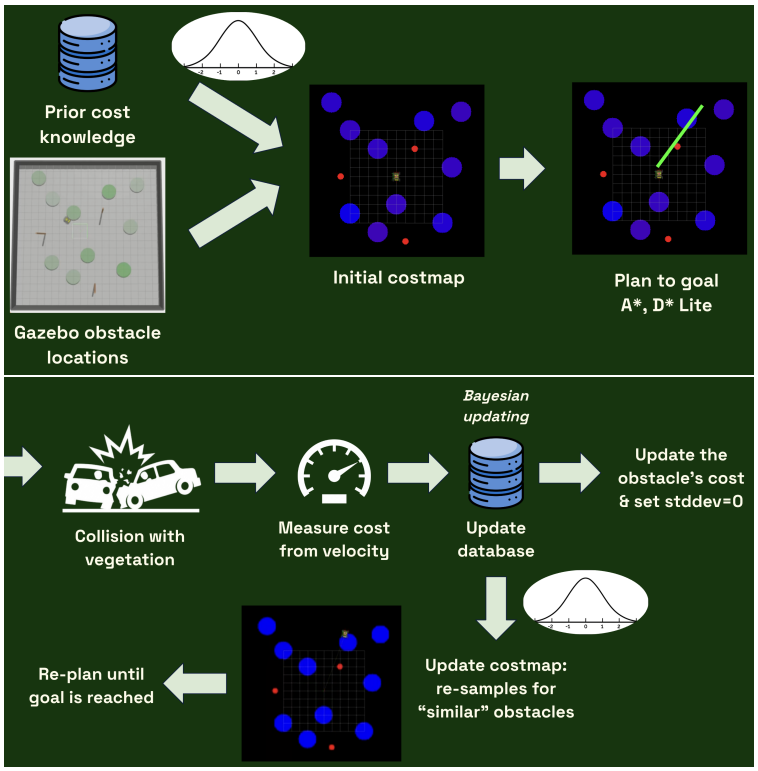

Figure 4: The algorithm

3.2.1 Initial Cost Map

Firstly, the initial cost map is constructed from previous cost knowledge and obstacle locations. The prior cost knowledge database is essentially an estimated-cost Gaussian distribution for each obstacle type (i.e. tree, grass, etc.) and is derived from intuition: trees have higher cost, and grass has lower cost, or from saved data from prior experiments.

For simplicity, the exact location of the obstacles is obtained from the Gazebo simulator. In reality, the locations and types of obstacles at each location would be gathered from satellite imaging and/or exteroceptive sensors.

Then, for each obstacle, a cost is assigned by sampling from its corresponding Gaussian distribution. The sampling of the distribution represents the variability that exists in vegetation, where no two obstacles are exactly the same.

3.2.2 Planning and Replanning

Next, a goal pose is defined and the planner finds a path through the vegetation to reach the goal. We experimented with A* and D* Lite planners. Upon collision with vegetation, the cost of traversal is measured from its change in velocity using the formula described in the above math section. The cost of the collided obstacle is updated to the new cost, and its standard deviation is set to 0, indicating the certainty. This new cost then updates the cost database using Bayesian updating. With this new cost, similar obstacles (e.g. all obstacles of type tree) that have not already been interacted with are resampled from this new distribution and will have their costs updated in the cost map.

After the cost map has been reconstructed from the new observations, the path is replanned, and this process is repeated until the goal is reached.

3.3 The simulation

The simulation was implemented on the ROS2 Humble and Gazebo simulator. Nav2 stack was used to coordinate planning with our custom costmap plugin and custom planner plugin. The Clearpath Husky 4-wheeled robot was used.

3.3.1 Contact model

We experimented with various contact models in Gazebo to simulate a traversable yet resistive obstacle. We tried variable friction patches on the ground, spring damper joint obstacles, online vegetation models, which were not enough to simulate a realistic robot to vegetation obstacle contact.

The solution we ended up using was using "resistance zones", that apply a speed reduction to different areas of the map.

Future work would be to develop a more sophisticated contact model that uses the robot's proprioceptive sensors, such as wheel encoders and IMU.

4 Experimental Results

In the Gazebo simulation, we compared the behaviors of the custom A* planner and our D* Lite planner under two scenarios.

When the goal was far from the obstacle, both planners generated similar paths. However, when the goal was close to the obstacle, the custom A* planner generated a path that detoured around the obstacle, while the D* Lite planner allowed the robot to encounter the obstacle, update the costmap using proprioceptive sensor and then generate a more optimal path. The figure below shows the two resulting paths (red lines). As can be seen, the path from the custom A* planner is suboptimal compared to the updated D* Lite path.

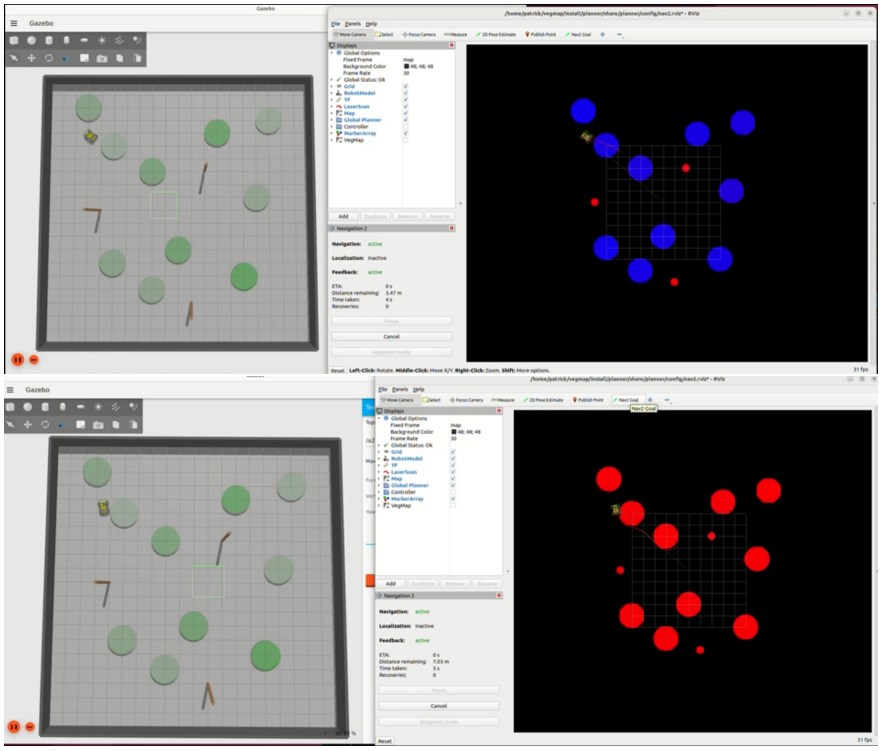

Figure 5: Difference between the path generated by custom A* planner(up) and D* Lite(down)

We also compared the performance of the A* and D* Lite planners in terms of planning time, number of states expanded, and path length. Different start and goal states were used to evaluate performance in

various scenarios. Table 1 and Table 2 show the result of the difference between different planners with different goals.

Categories	D* Lite	A*
Planning Time	6000 ms	6000 ms
States Expanded	23101	25462
Path Length	52	50

Table 1: Difference between A* and D* Lite Algorithms for Goal 1

Categories	D* Lite	A*
Planning Time	10123 ms	11421 ms
States Expanded	148523	176854
Path Length	82	79

Table 2: Difference between A* and D* Lite Algorithms for Goal 2

We found that the planning time for both planners is similar. A* expands more states than D* Lite, but the path length of D* Lite is longer. The probable reason for this is that A* is a one-shot optimal search from start to finish with a consistent heuristic, so it aggressively explores all necessary nodes to guarantee the shortest possible path. D* Lite is designed for replanning efficiently in dynamic environments; it prioritizes minimizing re-expansion of nodes rather than always ensuring a globally optimal path after updates, leading to slightly longer paths but fewer expanded states.

Meanwhile, we experimented with collision with solid obstacles such as trees and friction patches(Figure 6) to help develop different obstacles in the vegetated zones. For solid obstacles, we assign a cost of one, indicating that they are completely untraversable. For traversable obstacles, we apply a speed reduction when the robot attempts to move through the vegetation.

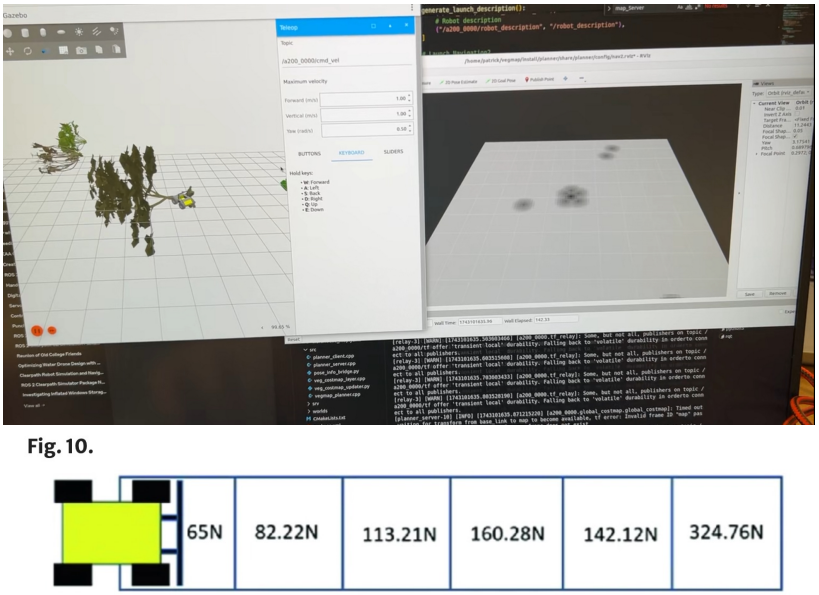

Figure 6: Experiments with solid obstacle collision [1]

5 Conclusions

We implemented an end-to-end simulation in ROS2 and Gazebo using a Husky robot, successfully demonstrating planning and traversal through vegetation. In addition, we develop a resistance zone contact model based on proprioceptive sensing, which updates the cost map for A* and D* Lite planners.

The experiments show that the planner works as intended, generating a cost-effective path through and around the traversable obstacles, effectively updating the cost of the obstacle data base, and adaptively improving its planning through the vegetated environment. We also compared the performance of A* and D* Lite under identical conditions, providing information to guide the selection of the appropriate planner.

6 References

[1] C. Ordonez et al., "Characterization and traversal of pliable vegetation for robot navigation," Springer Proceedings in Advanced Robotics, pp. 293–304, 2020. doi:10.1007/978-3-030-33950-$0_2$6