3D Off-road Terrain Mapping for Autonomous Ground Vehicle Energy-Optimal Path Planning

Matt Nguyen*
University of North Carolina at Charlotte, Charlotte, NC, 28223

Navigating unknown terrain may be difficult due to unstructured paths, obstacles, and inconsistencies in ground properties. This paper examines the challenge of creating a local terrain elevation map from GPS measurements collected by an autonomous ground vehicle (AGV) to enhance navigation efficiency. A methodology is developed that enables the Clearpath Jackal AGV to gather onboard sensor data to be visualized and interpolated into a digital elevation map. In this way, prior data obtained while traversing the terrain of interest is used to inform the planning of future paths. The platform was configured to collect IMU, gyroscope, and GPS data using the Robot Operating System (ROS). An automated data processing pipeline was developed to convert raw data into readable formats (e.g., CSV tables), filter out irrelevant information, and organize the data efficiently to reduce computational overhead. The path traversed by the AGV was then reconstructed, and a linear regression model was used to interpolate between GPS points to create a 3D map of the terrain. An energy-optimal path planner based on A* was then developed by defining a cost function between nodes related to changes in terrain height and a planar distance heuristic to improve efficiency. Our approach demonstrated that the AGV could successfully avoid steep terrain and find paths around difficult areas rather than attempting to traverse them directly. Two methods for sampling the terrain to form a roadmap for planning were evaluated and compared – gridded sampling and random sampling. Simulated energy optimal paths over the experimentally determined elevation model are used to illustrate the approach.

I. Nomenclature

AGV	=	autonomous ground vehicle
GPS	=	global positioning system
GNSS	=	global navigation satellite system
IMU	=	inertial measurement unit
RRT	=	rapidly exploring Random trees
LiDAR	=	light detection and ranging
ROS	=	robot operating system
Y	=	northing distance between nodes
X	=	easting distance between nodes
α	=	parameter reflects energy's importance in our optimal path
θ	=	latitude
λ	=	longitude
h	=	altitude
G	=	graph model
w_j	=	barycentric weight of a set of nodes
y_q	=	interpolated result
$f(\cdot)$	=	cost function
$g(\cdot)$	=	cost from node to goal
$h(\cdot)$	=	heuristic estimation
$D(\cdot)$	=	distance between vertices
$C(\cdot)$	=	actual cost associated with the edges between vertices

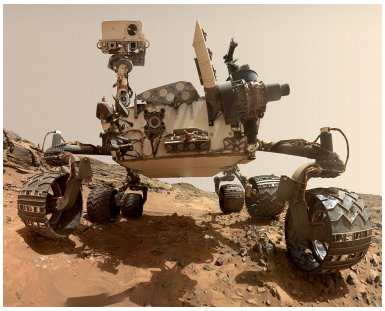
*Undergraduate Student, Department of Mechanical Engineering and Engineering Science, AIAA Student Member, nnguye90@charlotte.edu

II. Introduction

Autonomous ground vehicles (AGVs) are widely used to explore areas that are too dangerous or difficult for humans to reach, like the surface of other planets [1] and underground mines [2]. AGVs also have applications in working alongside humans in outdoor rugged terrain for military, scientific, and commercial applications [3]. Standard ground vehicle missions include helping carry gear and instrumentation, retrieving samples, and recording footage to gain more insight into areas of interest. To improve the efficiency and robustness of AGVs performing such tasks, it is essential to consider factors such as terrain elevation and slope and their impact on vehicle stability, performance, and power consumption. Path planning for AGVs typically considers 2D maps of the environment where the straight path is the shortest. A short deployment of the robot in an unknown environment can provide a prediction of the environment's geometry properties through data collected with onboard sensors. This estimation formulates a 3D path planning problem for the robot for future navigation to a random location within the range mentioned earlier. The optimal path-finding problem can be solved using the A-star algorithm [4]. Performance can be justified by calculating the properties of simulated terrain, such as distance and elevation.

This paper is organized as follows. Section III describes the data-collecting process and formulates the path-planning problem. Section IV outlines the equipment utilized, the procedures for processing raw data, the methods for estimation and sampling, as well as the strategies for optimal pathfinding. Finally, the paper will conclude with the results and perspective for future work in SectionV.

(a) An AGV detects landmines, reducing injury risks and enhancing safety in affected areas. [5]



(b) NASA Curiosity rover used for planetary exploration and collecting samples on Mars.[6]

Fig. 1 Example applications of AGVs.

A. Related Work

Traversability is crucial in determining whether ground vehicles can deploy on a mission successfully. Each terrain type has its unique characteristics, which can be unpredictable and unstructured, making the approximation process more challenging. Previous research has addressed this issue by categorizing different terrains into familiar classifications, such as gravel, dust, mud, and others [7]. This classification allows robots to adapt to their environments while humans can adjust the vehicles to enhance their performance. The classification process can be carried out using various data collection methods. In [8], the authors use accurate locomotion data for deployment to capture the robot's surrounding environment and analyze different robot performances based on the collected data. By emitting light beams around the robot, LiDAR technology captures the three-dimensional shape of the surrounding area. This data can create a map and assess traversability, as demonstrated in [9] and [10]. Data collection is followed by estimation using several techniques. Standard methods include interpolation, Monte Carlo methods [11], and Gaussian estimation [12]. More recent approaches involving machine learning require substantial processing power and data storage to determine whether a path is traversable or non-traversable [13].

Pathfinding involves exploring practical ways to navigate terrain optimally, allowing robots to move through environments similarly to how humans use logic or imagination before executing actions. Robots can replicate this process using computing and algorithms. Optimal path planning has been around since the early days of robotics, with established methods such as Dijkstra's algorithm [14], the A* algorithm [4], and the more recent Rapidly Exploring Random Trees (RRT) algorithm [15]. These methods have built a foundation for controlling and optimizing robots, leading to much more involved research in much specialized applications or areas of robotics. For instance, Truong and Hong utilized the A* algorithm to find the optimal path in a 3D environment based on a known map [16]. Dijkstra's

algorithm has been employed to consider distance, traceability, and power consumption factors when determining the path from a start to a goal position through challenging terrain [17]. In addition, new methods for pathfinding are being explored, indicating promising advancements in the field. Another study [18] implemented an Ant Optimization algorithm combined with stereo cameras to identify the most efficient route. Furthermore, an approach using actuator space involves adjusting the path by utilizing various steering angles instead of relying solely on kinematics [19].

III. Problem Formulation

This section outlines the problem of optimizing pathfinding by first introducing related concepts such as graph theory and optimal path planning. It also describes some of the challenges in making the pipeline capable of finding the optimal path from real-world data and setting a goal for our solutions.

A. Graph Theory

A graph G is defined as a set of nodes (or vertices), represented by n_i, and a set of edges represented by e_{ij} [4]. The nodes can be considered locations or points within a structure, while the edges represent the relationship between them. In a three-dimensional (3D) graph, each node corresponds to a point in space and is described by Cartesian coordinates. Specifically, each node n_i has a location represented as $n_i = (x_i, y_i, z_i)$ where $n_i \in \mathbb{R}^3$, uniquely defined its position in 3D space. The edges $e_{ij} = (n_i, n_j)$ represent the connection between node n_i and node n_j,and can be associated with a cost c_{ij}, such as the Euclidean distance between them. This representation of nodes and edges allows us to apply our cost function to quantify the relationships between nodes. This information is crucial for running the pathfinding algorithm, whose accuracy largely depends on the output of the cost function.

B. 3D Path Planning

A path is defined as a sequence of nodes from n_s to n_k that are connected. The smallest quantitative unit of a path is the edge. When planning, there will be many paths between the two points of interest, the start node n_s and the goal node n_g. The optimal path is the path with the minimum cost that fits all the predetermined constraints. While 2D path planning only involves the cost of the distance between two points on a flat surface, we argue that it becomes more complex when analyzing the cost between points in a 3D graph, where efficiency and traversability play crucial roles in determining the optimal path. It might be trivial to measure the robot's power consumption directly from the internal electronic system as cost. Although it is possible, this might not yield an accurate result since power consumption involves both computing power and mechanical power that helps move the robot. In this paper, we are interested in finding a path that is not only short but also minimizes the mechanical power consumption required for the robot to move from n_S to n_G on our map.

C. Data Collection

Our data is collected from recordings captured through the robot data collection protocol in ROS, known as rosbag. These recordings contain a dataset of interest, taken at discrete time intervals while the robot operates in an unknown environment. We aim to analyze and formulate this data into a format that can be further processed and optimized before running it through an optimization path planning algorithm. Currently, this approach is done manually in post-processing, but our long term objective is to automate this process, allowing the robot to handle it in real time. Looking at the data, the original GPS data from the robot should be in the form of

$$R = (\theta, \lambda, h) \ , \tag{1}$$

where θ is the latitude, λ is the longitude, and h is the altitude.

In our problem, graph G is not readily available for path planning. Instead, we have a set of data points representing the robot's path, referred to as the base path, at the location of interest. Our objective is to construct a virtual map using data collected from the Jackal as it randomly traverses the terrain.

D. Problem Statement

The problem of finding the optimal path can be divided into two distinct challenges. First, we must find a way to convert raw data into a 3D model terrain, which is essential for applying any path-finding algorithm. Our priority is to minimize deployment time and resources, which results in the constraints in length and time of the robot base path. Our

estimation process should guarantee interpolating the original path into an area map while conveying enough detail about the physical characteristics of the terrain. The next problem is finding fast and accurate solutions to this map. This process requires developing an efficient sampling approach to generate nodes along the map. The goal is to select the pathfinding algorithm capable of quickly and accurately solving the optimal route from the start to the goal nodes.

Fig. 2 Google satellite image of the terrain for which experimental tests were conducted.

IV. Methodology

This section first describes the equipment used in the proposed method for data collection (Section IV.A). The collected data is analyzed in Section IV.B and interpolated in Section IV.C to create an estimated digital elevation map. Sampling methods from Section IV.E are then applied to acquire the terrain graph. Finally, The A* algorithm with the developed cost function in IV.D is applied to the sampling graph G, delivering the optimal path to reach the goal node.

A. Equipments

Clear Path Robotics developed Jackal as an autonomous robot platform designed for research applications. It is a compact, durable, versatile, and customizable AGV with a robust 4x4 chassis and is made from high-grade aluminum. Jackal is rated IP62 for weather resistance and has insulated connections to ensure optimal performance in various environments. The wide platform, measuring 508 mm in length and 430 mm in width, provides plenty of space for mounting accessories. It offers 5 V, 1 V, and 24 V power options and features an onboard computer connection port suitable for various types of sensors [20]. The robot's high-performance Ubuntu computer can handle complex

Fig. 3 Jackal AGV from Clear Path Robotics used in this research.

processing tasks. It also supports synchronization with ROS, the RViz control interface, and Gazebo simulation. Additionally, the Jackal offers various connectivity options, including Bluetooth and Wi-Fi, providing multiple ways to access and control the robot. The primary sensor used in this project is the Swift Navigation Duro GPS sensor, which is built into the robot. This sensor is a robust and durable GNSS device designed to operate across multiple frequencies and constellations., utilizing cutting-edge technology to deliver accurate positioning. This sensor is a perfect fit for ground robot application, which requires reliable and precise location data in a challenging environment [21].

The Robot Operating System (ROS) is a well-developed framework wildly used across various robotic platforms today. ROS is a Linux-based system that supports numerous robotic tasks, including the development and control of robots. Our Jackal robot operates on ROS Noetic, the latest version of ROS 1. This system facilitates robot control and

real-time data collection from various sensors, as detailed in the hardware section above. Such capabilities provide a convenient means of extracting data for analysis.

B. Data Analysis and Preparation

Data are taken from the robot in the form of a ROS bag and built-in extraction functions to generate CSV files containing data stored at discrete time intervals, ensuring the capture of all the information of every activity on the robot system with high precision. The geographic coordinates R are extracted from these files for further analysis. These bag files are generally very disk space-intensive. Processing them can significantly slow down the analysis, although it is necessary to track the robot's traversal time precisely. Upon reviewing the generated CSV file, we noticed repeated data over specific periods, as if the robot were stationary. This indicates a mismatch between the high frequency of the collected data and the movement of the robot. The cause of this issue is that the data sampling rate T was faster than the sampling rate ΔS of the GPS data. This issue is resolved by removing duplicated data points from the original file, leaving all unique data points continuous through the traversing of the robot through the terrain. The result was impressive, our test CSV file was reduced from 7243 items to only 716 items. This could potentially pose a significant improvement in the data processing time and computing resources of our robot. Having optimized data with meaningful points, it would be necessary to convert geographic coordinates (latitude, longitude, altitude) to Universal Transverse Mercator (UTM) [22] coordinates for precise spatial analysis. This process involves transforming the spherical coordinates into a Cartesian coordinate system, accounting for the Earth's curvature through a suitable map projection. UTM coordinates are a well-known coordinate system that includes three main components: easting (x), northing (y), and altitude (z). This conversion maps the data into a planar coordinate system, which is highly versatile for modeling the terrain. This model can then be used to visualize the process and perform calculations based on the terrain representation.

C. Graph Estimation

Running the robot extensively along the terrain can help cover many details that can be effectively used to build maps. However, this process is generally not feasible since it is work-intensive and consumes endless deploying time, computing resources, and storage, which is not available. Estimation can be a good solution to this problem with the data of the based path. The general estimation idea involves generating additional data points by leveraging the known relationship between two or more existing points while adhering to predetermined constraints. This paper uses linear barycentric interpolation for its simplicity and wide practical applications. Linear barycentric interpolation is a widely used method for curve fitting that estimates unknown points from a set of known points. The technique is grounded in the principle of linearity, which assumes that the change in the known point of addiction is a straight line. The algorithm first performs Delaunay triangulation from the base path data to prepare for the next step and reduce interpolation error [23]. From the point q located inside each triangular node n_1, n_2, n_{d+1}, using the inverse to a power to compute the weighted average to determine the barycentric weight of them w_i [24]

$$w_j = \frac{1}{\sum_{i=1}^{d+1}(x - x_i)} \quad \text{given} \quad \sum_{i=1}^{d+1} w_i = 1 \ , \tag{2}$$

From there, y_q can be interpolated using linear interpolation formulas

$$y_q = \sum_{i=1}^{d+1} w_i y_i \ , \tag{3}$$

where y_i represents the known values at the vertices x_i of the simplex surrounding the point x_q and d is the dimension of the space under consideration. In this case d is set to two because 3D data is being considered. Method linearly connects all the known points to create a mesh representing the behavior and trends of the data set. When new points are added to the existing mesh, a line segment is formed between each pair of adjacent data points to estimate the intermediate values. This process makes the new points valuable additions to the mesh and updates it accordingly.

The result interpolation from our data set successfully creates a 3D map of the terrain, restricted by the closest point on each axis, as seen in Figure 4. The color represents the attitude of the terrain, with darker color meaning a lower attitude.

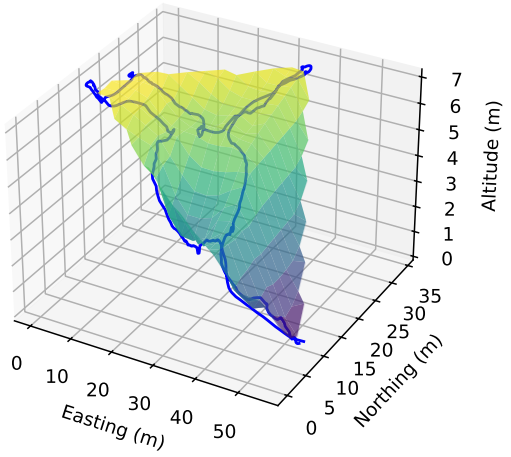


Fig. 4 3D terrain map interpolated from collected data of the based path (blue).

D. Cost Function and Weight Estimation

Cost is the critical factor that justifies the performance of a path-finding algorithm, especially considering the constraints of robotic systems in real-world environments. Our cost function is designed to account for the distance traveled and the energy the robot consumes as it moves through varying terrain. Given that the robot's battery is limited to 2 hours (according to Jackal's specifications), this constraint plays a crucial role in shaping the cost function to prioritize energy-efficient pathfinding. To calculate the distance traveled, we employ the 2D Pythagorean theorem to determine the distance between nodes n_i and n_j, based on their x and y coordinates. The distance function is given by:

$$D(n_i, n_j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \ . \tag{4}$$

We also account for terrain inclination, an essential factor in estimating power consumption. Steep terrain requires more energy to overcome gravitational forces and slippage. However, we must ensure that the cost function handles situations where nodes at lower elevations could generate negative power consumption if the goal node n_G is higher than the start node n_S. To address this issue, we designed an energy cost function based on an exponential model, where energy consumption increases non-linearly with distance. This approach better reflects the increasing difficulty of movement over steeper terrain and ensures the cost accurately represents the robot's energy usage without violating physical principles.

$$C(n_i, n_j) = \left(1 + \exp(\alpha \cdot \Delta_{\text{Altitude}}(n_i, n_j))\right) \cdot D(n_i, n_j) \ , \tag{5}$$

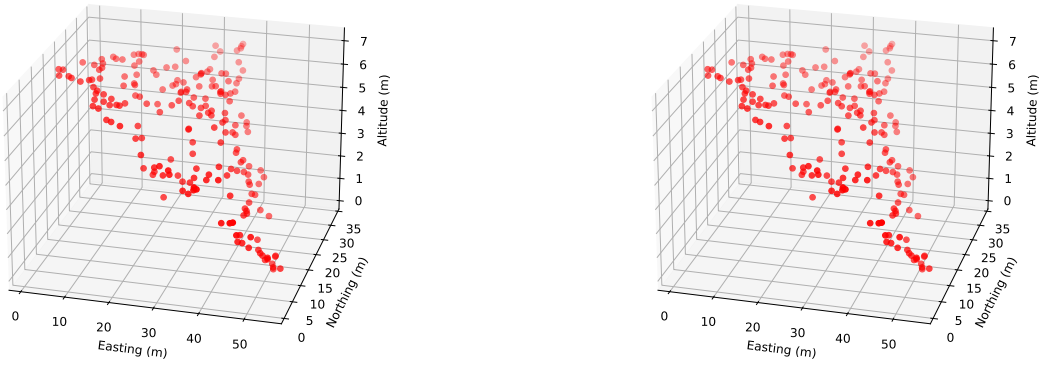
where α is a constant representing the significance of terrain elevation to the cost, Δ_{Altitude} is the change in altitude between two nodes, and D is the 2D distance between nodes.

E. Scatter Sampling Approaches

Constrained random point sampling: The first proposal allows us to determine nodes on G and randomly generate a certain number of nodes distributed across the surface of the generated terrain model. The total number of nodes is determined by a preset maximum number of sampling points required for our analysis. Note that the larger the sample size, the more accurate and defined the path we can get. Each three-dimensional node from the sample will occupy a unique position within the graph, representing a traversable point. The constraints for these points are defined as $x_i \in (x_{\min}, x_{\max})$, $y_i \in (y_{\min}, y_{\max})$, and z_i is derived as an interpolation of x and y, constrained within the range $(z_{\min}, z_{\max})$. The random points are generated using a uniform distribution across the graph, ensuring an even spread of nodes that reflect the traversable areas of the environment. This method allows us to create a robust representation of the terrain in a graph form, which can be further refined in subsequent steps of the path process.

Uniform Grid Point Distribution: Another strategy for generating nodes in our graph is to divide the surface into a uniform grid of small rectangular segments, creating a set of grid cells Q that cover the entire graph area. Each segment Q_{i} is designed to have approximately equal area, ensuring a balanced distribution of potential path points. Since each rectangular segment has four corners, we designate each corner as a candidate for optimal path points. This

method allows us to systematically sample the environment, ensuring no significant area is left unrepresented in our node generation. The uniform grid point distribution approach offers several advantages, including improved coverage of the graph and the ability to maintain consistent spacing between potential nodes. This strategy can facilitate more efficient path-finding algorithms, as it reduces the likelihood of missing critical traversal points. Additionally, the structured nature of the grid simplifies the analysis of connectivity and distance between nodes, further enhancing our path-planning capabilities.



(a) Randomly sample points along the terrain curvature. **(b) Sampling points along the grid lines.**

Fig. 5 Two sampling approach visualization.

F. A* Path Planning Algorithm

A* (A-star) is a path-finding algorithm renowned for its efficiency and ease of use. It is an enhanced version of Dijkstra's algorithm, designed to reduce the time and computational resources required to find the shortest path. Unlike Dijkstra's algorithm, which exhaustively explores all nodes in the search space to determine the optimal path, A* uses heuristics to estimate the direction towards the goal. This heuristic approach allows A* to focus on the most promising nodes and avoid irrelevant portions of the search space, thereby improving efficiency 6.

In this paper, A* employs a cost function to prioritize the nodes that are most likely to lead to the optimal path.

$$f(n) = g(n) + h(n) \tag{6}$$

where $f(n)$ represents the total estimated cost, $g(n)$ is the actual cost from node n to the goal, and $h(n)$ is the heuristic estimation of the cost from node n to the goal.

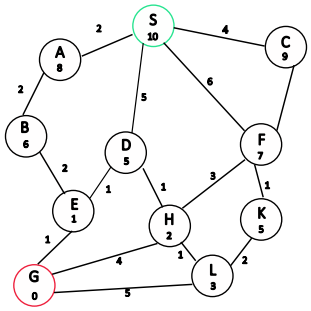


Fig. 6 A* algorithm graph visualization.

The version of the A* algorithm developed in this paper is described in Algorithm 1. This algorithm takes a set of nodes Q, sampled along the terrain surface, a start node n_s, and a goal node n_g, and returns the optimal path, which is a set of nodes that connect the start and the goal. Initially, the cost of n_s is set to 0, while n_g is marked with infinite cost (lines 1-2). The set $\mathcal{N}$ of neighbors for each node in Q is determined based on the Euclidean distance constraint $D_{\max}$ (lines 4-13), where the algorithm checks the distance between each node and adds it to the neighbor set if the distance is

Algorithm 1 Terrain-Aware A*

function: `TerrainA wareA*`(Q, n_s, n_g)
input: sampled vertices Q, start node n_s, goal node n_g
output: Optimal path (a set of nodes)

1: $Q \leftarrow Q_i \cup \{n_s, \text{cost} = 0\}$
2: $Q \leftarrow Q_i \cup \{n_g, \text{cost} = \infty\}$
3: $\mathcal{N} \leftarrow \emptyset$
4: **for** $n_j \in Q$ **do**
5: $\quad \mathcal{N}_j \leftarrow \emptyset$
6: $\quad$ **for** $n_i \in Q$ **do**
7: $\quad\quad D \leftarrow \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$
8: $\quad\quad$ **if** $D \leq D_{\max}$ **then**
9: $\quad\quad\quad \mathcal{N}_j \leftarrow \mathcal{N}_j \cup \{n_i\}$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad \mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}_j$
13: **end for**
14: `open` $\leftarrow$ `heuristic`(n_s, n_g)
15: `Parents` $\leftarrow \emptyset$
16: **while** `open` $\neq \emptyset$ **do**
17: $\quad$ `sort(open)`
18: $\quad$ `current` $\leftarrow$ `first(open)`
19: $\quad$ **if** `current` $= n_g$ **then**
20: $\quad\quad$ `final` $\leftarrow$ `parents`
21: $\quad$ **end if**
22: $\quad$ **for** `child` $\in \mathcal{N}_{\text{current}}$ **do**
23: $\quad\quad$ `totalCost` $\leftarrow$ `cost(current, child)` + `cost(start, current)` + `heuristic(current, goal)`
24: $\quad\quad$ **if** `totalCost` $<$ `cost(start, child)` **then**
25: $\quad\quad\quad$ `child(cost)` $\leftarrow$ `totalCost`
26: $\quad\quad\quad$ `parents(child)` $\leftarrow$ `current`
27: $\quad\quad\quad$ **if** `child` $\notin$ `open` **then**
28: $\quad\quad\quad\quad$ `open` $\leftarrow$ `open` $\cup$ `child`
29: $\quad\quad\quad$ **end if**
30: $\quad\quad$ **end if**
31: $\quad$ **end for**
32: **end while**
33: $n_c \leftarrow goal$
34: `path` $\leftarrow \emptyset$
35: **while** $n_c \in$ `parents` **do**
36: $\quad$ `path` $\leftarrow$ `path` $\cup\, n_c$
37: $\quad n_c \leftarrow parents(n_c)$
38: **end while**
39: `optimal Path` $\leftarrow$ `reverse(path)`

less than or equal to $D_{\max}$ (lines 7-9). The pathfinding algorithm then proceeds by examining the lowest-cost node in the open set, which has the potential to be the next node in the optimal path (lines 14-18). For each node currently being considered, referred to as *current*, all its neighbors (children) are evaluated by calculating the total cost, which is the sum of the cost to reach the neighbor from the start node, the cost to reach *current*, and the heuristic cost from the neighbor to the goal (line 23). If the total cost to reach a neighbor is lower than its current known cost, the neighbor's cost is updated to the lower value (lines 24-26). The *current* node is then assigned as the parent of the neighbor node, and the neighbor node is added to the open set if it is not already present (lines 27-29). The algorithm continues by examining the neighbors of the next lowest-cost node in the open set until the current node equals the goal node (line

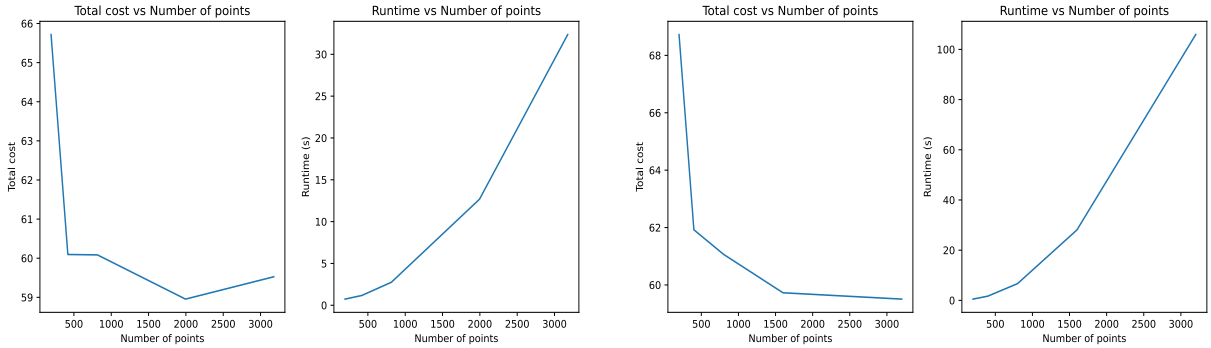
20). Once the goal is reached, the algorithm reconstructs the optimal path by tracing back from the goal to the start node using the *parents* set (lines 33-39).

V. Results

The algorithm successfully searched for the optimal path to traverse through our terrain from the curve-fitted graph with data points along its surface. The distance is treated as equal weight with the energy, which sets the value of α in the (5) to have the value of 1. The method yields similar results in different runs, which poses the consistency in our path-finding approach. However, when considering the two sampling methods, there are some variations in the output and performances.

A. Comparing The Two Sampling Approaches

Figure 7 illustrated the total cost and runtime at different sample sizes for the gridded points sampling and randomized points sampling, respectively. Data was collected at a predetermined number of sampling points. Through experiment, the shortest path can only be found with a sampling size larger than 100 points. To minimize redundancy, our minimum sampling size started at 200 points and doubled until it reached 3200 points, which is estimated to be the convergence threshold. A larger sample would not result in a much smaller cost for the shortest path rather than consuming more computing power and taking longer. Overall, both methods showed the same trends in the price of the shortest path as the number of points increased. As the larger sample size was above 1500, there was a slight increase in the total cost of the fastest route in the gridded sample case. This increment trend results from over-sampling where points overlapped, causing less precise results. The run time in both methods increased linearly with sample size. However, while the randomized approach tends to be faster at smaller sample sizes, the gridded strategy proved to be almost three times faster than the randomized when more points are introduced to the map. From observation, with the interest of minimizing computing time and maximizing accuracy, a sample size of 1000, used for later analysis, is considered to set our scale to approximately balanced.



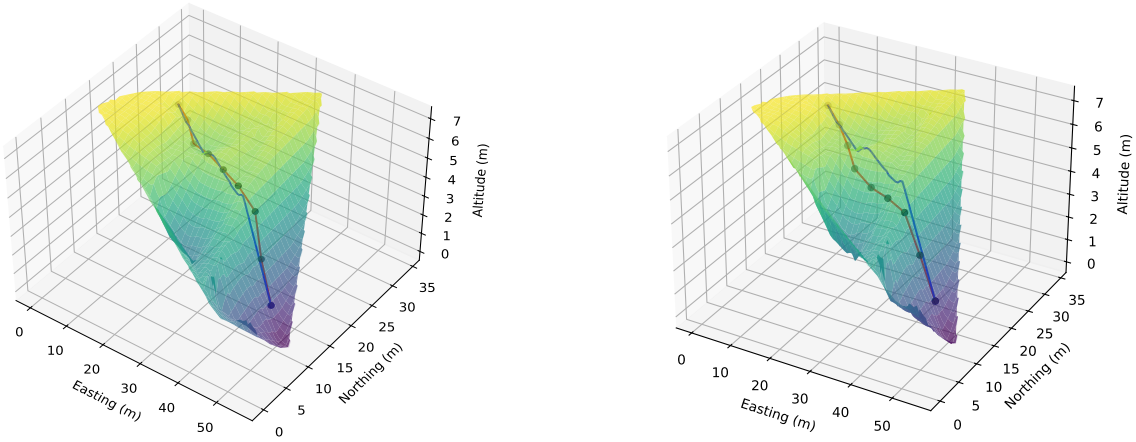
(a) Gridded points sampling results. **(b) Randomized points sampling results.**

Fig. 7 Performance of the algorithm with different sampling approaches.

B. Path Found Discussion

Distance-wise, the shortest path is the straight path from start to destination note, assuming no obstacle exists. To ensure this condition is also applied to our method as expected when energy is not considered, the exponential term in (5) will be removed before running the algorithm, leaving only the value D to be considered when finding the best path. Figure 8a represents our check; the red line with nodes along it is the shortest path found, while the black line is the reference path of a straight line from the start to the goal node when fitted into the terrain. When comparing the two paths, the path found by the algorithm is very close to the straight line, which aligns with our assumption. The variation is due to variation in node location, which is approximately close but does not exist in the combination of nodes that lay precisely on top of the straight line. Rerunning the algorithm on the same sample with energy is considered by reintroducing the exponential term in our cost function and choosing $\alpha = 1$ to represent the importance of altitude in the

optimal path found. The result is shown in Figure 8b. Regarding distance, the algorithm did not pick the shortest path,



(a) Path-finding without considering energy in the cost function.

(b) The path found from the gridded sampling approach with consideration of energy.

Fig. 8 Path finding results.

the straight line from start to end. This was expected because, as mentioned earlier, our algorithm does not always prioritize the shortest route, but rather the most efficient way to reach the goal while conserving as much battery as possible. The solution path followed a similar trend to the straight line, but with more curvature. This result reflects the characteristic of the optimal route, which avoids going straight uphill to minimize the robot's climbing effort. Instead, the algorithm finds it more efficient to traverse steep areas of the terrain.

VI. Conclusion and Future Work

The graph image of the curve-fitted data defines the characteristics of the real-world terrain that the robot has autonomously traversed. Our optimized data reduction approach makes the computation time from raw data to a 3D terrain map instantaneous. The method presents a promising solution for on-board computer and mapping of robots, allowing path planning directly on the robot at any time. As a result, it can enhance the robot's efficiency, increasing battery life and operation time for each mission.

Follow-up work may involve fusing additional data sources into the terrain mapping algorithm to enhance accuracy. Sensors such as IMU and LiDAR create data layers that can be more precise than GPS alone. The map may also encode battery energy consumption, terrain type, and obstacles. The ultimate goal is to enable the robot to perform the mapping and path-planning tasks autonomously in real-time.

Funding and Acknowledgments

This work was funded by the University of North Carolina at Charlotte's Office of Undergraduate Research (OUR) program. We would like to thank Dr. Artur Wolek for his guidance and support throughout the research, and Collin Hague for his mentorship and assistance with coding.

References

[1] Hebert, M. H., Thorpe, C. E., and Stentz, A., *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon*, Vol. 388, Springer Science & Business Media, 2012.

[2] Ge, S., Wang, F.-Y., Yang, J., Ding, Z., Wang, X., Li, Y., Teng, S., Liu, Z., Ai, Y., and Chen, L., "Making Standards for Smart Mining Operations: Intelligent Vehicles for Autonomous Mining Transportation," *IEEE Transactions on Intelligent Vehicles*, Vol. 7, No. 3, 2022, pp. 413–416.

[3] Olejarz, D. A., "An Assessment of the Use of Autonomous Ground Vehicles for Last-Mile Parcel Delivery," Ph.D. thesis, University of Toronto, 2020.

[4] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *ACM SIGART Bulletin*, 1972, pp. 28–29. https://doi.org/10.1145/1056777.1056779.

[5] Clearpath Robotics, "Foxy's Husky Guide," , 2015. URL https://clearpathrobotics.com/assets/guides/foxy/husky/index.html, accessed: 2025-01-12.

[6] ISS National Lab, "Rover Parts Lesson Plan," , 2024. URL https://issnationallab.org/stem/lesson-plans/rover-parts/, accessed: 2025-01-12.

[7] Ojeda, L. V., Borenstein, J., Witus, G., and Karlsen, R. E., "Terrain Characterization and Classification with A Mobile Robot," *Journal of Field Robotics*, Vol. 23, 2006, pp. 103–122. URL https://api.semanticscholar.org/CorpusID:11341013.

[8] Eder, M., Prinz, R., Schöggl, F., and Steinbauer-Wagner, G., "Traversability Analysis for Off-road Environments Using Locomotion Experiments and Earth Observation Data," *Robotics and Autonomous Systems*, Vol. 168, 2023, p. 104494. https://doi.org/https://doi.org/10.1016/j.robot.2023.104494.

[9] Fukuda, Y., Mii, Y., Yano, Y., Iwai, H., Inoue, S., and Tamukoh, H., "Dense Traversability Estimation System for Extreme Environments," *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–6. https://doi.org/10.1109/IV55152.2023.10186556.

[10] Gao, B., Xu, A., Pan, Y., Zhao, X., Yao, W., and Zhao, H., "Off-Road Drivable Area Extraction Using 3D LiDAR Data," *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1505–1511. https://doi.org/10.1109/IVS.2019.8814143.

[11] Dam, T., Chalvatzaki, G., Peters, J., and Pajarinen, J., "Monte-Carlo Robot Path Planning," *IEEE Robotics and Automation Letters*, Vol. 7, No. 4, 2022, pp. 11213–11220. https://doi.org/10.1109/LRA.2022.3199674.

[12] Dragiev, S., Toussaint, M., and Gienger, M., "Gaussian Process Implicit Surfaces for Shape Estimation and Grasping," *Proc. 2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2845–2850. https://doi.org/10.1109/ICRA.2011.5980395.

[13] Sevastopoulos, C., and Konstantopoulos, S., "A Survey of Traversability Estimation for Mobile Robots," *IEEE Access*, Vol. 10, 2022, pp. 96331–96347. https://doi.org/10.1109/ACCESS.2022.3202545.

[14] Dijkstra, E. W., "A note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, Vol. 1, No. 1, 1959, pp. 269–271.

[15] LaValle, S. M., and Kuffner, J. J., "Rapidly-exploring Random Trees: Progress and Prospects," *Algorithmic and Computational Robotics*, 2001, pp. 303–307.

[16] Truong, X.-P., and Hong, S. H., "Rough Terrain Path Planning for Autonomous Ground Robot," *Proceedings of the AIAA SciTech 2024 Forum*, AIAA, 2024. https://doi.org/10.2514/6.2024-2764.

[17] Santos, A. S., Ignacio Perez Azpúrua, H., Pessin, G., and Freitas, G. M., "Path Planning for Mobile Robots on Rough Terrain," *Proc. 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, 2018, pp. 265–270. https://doi.org/10.1109/LARS/SBR/WRE.2018.00056.

[18] Cappalunga, A., Cattani, S., Broggi, A., McDaniel, M. S., and Dutta, S., "Real Time 3D Terrain Elevation Mapping Using Ants Optimization Algorithm and Stereo Vision," *Proc. 2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 902–909. https://doi.org/10.1109/IVS.2010.5548025.

[19] Overbye, T., and Saripalli, S., "Path Optimization for Ground Vehicles in Off-Road Terrain," *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7708–7714. https://doi.org/10.1109/ICRA48506.2021.9561291.

[20] Clear Path robotics, "Overview," Website, 2024. URL https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/, accessed: 2024-12-12.

[21] Clear Path Robotics, "Duro Product Summary," , 2024. URL https://docs.clearpathrobotics.com/assets/files/clearpath_robotics_015822-TDS1-4b708bb6aad4c7f36f53a913365b36a7.pdf, accessed: 6 Jan. 2025.

[22] Langley, R. B., "The UTM grid system," *GPS world*, Vol. 9, No. 2, 1998, pp. 46–50.

[23] Chen, L., and chao Xu, J., "Optimal Delaunay Triangulartions," *Journal of Computational Mathematics*, Vol. 22, No. 2, 2004, pp. 299–308. URL http://www.jstor.org/stable/43693155.

[24] Berrut, J.-P., and Trefethen, L. N., "Barycentric Lagrange Interpolation," *SIAM Review*, Vol. 46, No. 3, 2004, pp. 501–517. https://doi.org/10.1137/S0036144502417715.