Height Inference and Enhanced Path Planning for Low-Risk rough Terrain Traversal

Henry Zhao

Collingwood School, Burnaby, Canada

henry@zlww.com

Keywords: Navigation, Path planning, A-star, Rough terrain, Quadruped robot

Abstract: The problem of path planning through rough and uneven terrain has been studied extensively in recent decades. However, many such methods are designed to optimize path length, and are not fit for potentially lower-cost legged robots that may drift or suffer from external influence. This paper proposes a method for inferring the presence and heights of objects unseen by camera angles as well as a method to navigate through rough terrain while minimizing risk taken by a robot. The proposed method utilizes and combines multiple camera viewpoints to fill in areas blocked by an object, while allowing for additional viewpoints to be appended. The method also marks out areas that have not yet been seen, allowing a robot to explore said areas, should no other paths be apparent. The path planning section of this method takes into account a variety of costs, notably the total elevation gained and lost, as well as the distance from the path to the closest obstacle. Thus, it prioritizes flat terrain over rough as well as attempting to stay in the middle of two objects, greatly reducing the risk of collision or falling. The proposed methods have been tested both in simulated environments as well as integrated into and tested on a small quadruped robot. The algorithm results in a lowered average distance to obstacles, as well as a lowered total elevation gained or lost, in return for a longer path length. Even with a 5% lateral drift on the robot, this algorithm successfully navigates around obstacles without contact.

1. Introduction

Over the past several decades, many engineers and scientists have used animals for clues into effective robot design, and have developed a diverse set of legged robots. These robots excel at traversing rough terrain, and are able to adapt to many differing scenarios and environments. For many real-world applications of legged robots, they must have the capability to traverse and explore rough and unknown terrain autonomously, using onboard sensors to create a map of the environment. However, due to limitations such as the robot's specifications, sensor noise, and sensor range, the problem of autonomous mapping and path planning remains a difficult and open-ended challenge.

This paper presents a stereo camera based mapping system, combined with a navigational path planner designed to minimize risks taken by a four-legged robot traversing over rough and uneven terrain. The method proposed builds on existing methods of mapping and path planning, and uses a graph- based search method to compute the path with the least cost between the start and end points, including a cost for the slope risk of the path. The main contribution is the classification of sensor-unseen areas as unknown, as well as a combination of costs in the path planning algorithm to reduce the risk of the chosen path, keeping the robot away from obstacles and cliffs, and choosing less sloped paths.

The proposed approach first generates a 2.5 dimensional height map from a point cloud, and identifies obstacles. The obstacle map is expanded to compensate for the robot's width. The method performs path planning using an enhanced A* search over the 2.5D height map, which keeps in account the total vertical distance traveled as well as the distances to obstacles along the path. This path is then simplified and smoothed. The A* was chosen as a base search algorithm due to the grid-based representation of the environment, as well its efficiency and simplicity. The system has been tested both on real sensor data and in simulations on a variety of terrain types and environments.

Integrated with depth camera and motion control subsystems, this has also been tested on a quadruped robot autonomously navigating through rough terrain.

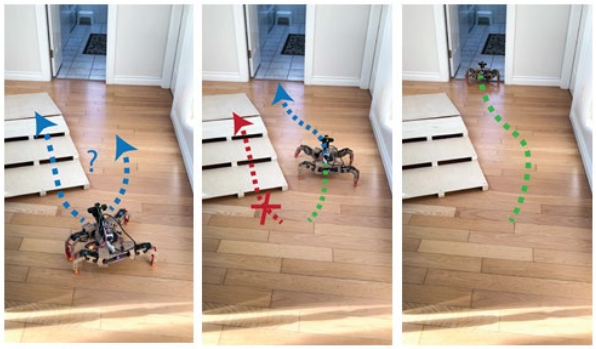


Fig.1 Robot Choosing Flat Path over Stairs

The contents of this paper are as follows: section II describes related work. Section III-A describes the system architecture of the legged robot. The 2.5D height map and obstacle map construction is described in section III-B. The path planning and smoothing methods are described in Section III-C. The experimental results are described in section IV, and the conclusions are presented in section V.

2. Related Works

Path planning methods for finding suitable and optimal paths can be grouped into two main categories: graph-based approaches, like Djikstra's algorithm [1], and sampling based algorithms, like rapidly-exploring random tree (RRT) [2]. Of the graph based approaches, there are classical algorithms like breadth first search (BFS), depth first search (DFS) and Djiktra's [1]. There are also heuristic based algorithms, like the A star (A*) [3] and D star (D*) [4]. In pathfinding, the two common and well-known algorithms are the A* and D* algorithms. Both algorithms repeatedly expand from the cell with the lowest cost that has not yet been visited, with the cost typically designated as the sum of the path length from the start to the cell and the distance from the cell to the goal. The D* algorithm includes a method for rapid and efficient updating of obstacles and costs, making it suitable for when new data is being added [4]. A large amount of robotic path planning systems use either of these two algorithms due to their simplicity and effectiveness, with modifications to improve efficiency, the quality of the path found, or adapted to higher dimensions [5], [6].

In order for these algorithms to be used on rough terrain, a method is needed to not only identify obstacles, but terrain that may be too risky or impossible for a robot to traverse. Dudzik [7] uses a traversability map method detailed by Kim [8], and uses a gaussian blur to make a map of cells reachable by the robot's center of mass, ensuring the robot can walk past obstacles without collisions. Wang [9] uses a similar method, and expands the obstacles to achieve the same result. These methods, although effective on flat ground, do not consider the risk a robot takes when attempting to traverse over sloped or bumpy terrain. Ideally, should a robot have the choice between a sloped path and a flat path, it should pick the flat path, even if it is longer, to a degree. Gu [10] has determined a method to create a map of "fuzzy traversability", marking every cell as either low, medium, or high risk, or untraversable. This method allows for quick evaluation and relatively fast path planning. However, it may not capture enough detail. For example, a robot may be unable to decide between two medium risk paths, even when one is slightly less sloped or bumpy than the other. Gu [11] uses the slope between two grid cells, described as the distance between their heights, as an additional cost, allowing the algorithm to trade off between a longer path length and a less sloped path. Görner [12] also factors in the roughness of the terrain, and sets cells with a slope larger than a threshold to be untraversable, to prevent the robot from attempting to walk over impossibly steep terrain. Wermelinger [13] uses slope and roughness to calculate the cost of a foothold, and McCrory [14] uses a similar method, though combines the foothold cost of positions on the right and left of a cell

to create a map of risk levels for the robot's center of mass. These methods provide an effective path planning algorithm that chooses lower-risk paths rather than higher-risk but shorter paths. However, even with the expansion or gaussian blur, the A* or D* algorithm still aims for the shortest path, and thus makes the path stick as close to walls and corners as possible, thereby ignoring risks from disturbances or drift. In the case that the robot is not entirely accurate and tends to drift, or is pushed by an external disturbance, a robot may crash into a wall and lose accurate track of its position, or even fall down a ledge or hill.

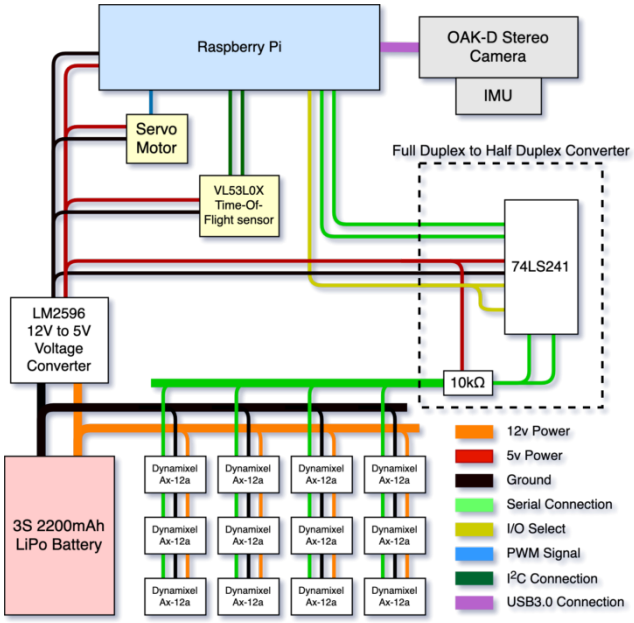


Fig.2 Schematic of the Quadruped Robot

3. Inferences and Path Planning

3.1 System Overview

The robot used in this study is a custom-made quadruped robot designed to be low-cost. The robot has 12 Dynamixel AX-a12 servos with 3 for each leg. An OAK-D stereo depth camera, in addition to a VL53L0X time-of-flight sensor are mounted on an MG90S servo, and are used for three dimensional environmental reconstruction. A schematic of the robot can be seen in figure 2. The OAK-D has a horizontal field of view of 72 degrees, and an effective range of 0.35 to 38.5 meters. The VL53L0X has a range of 5 to 120 cm. An IMU, contained in the OAK-D, consists of gyroscopes, accelerometers, and digital compasses, and is used in conjunction with the depth camera for localization and stabilizing the robot when on rough terrain. The robot's control system runs on a Raspberry Pi 4 through Python 3.7.

The robot takes images with the stereo camera, and creates a point cloud from all images using localization and mapping, while adding in data from the time-of-flight sensor and IMU. The point cloud is voxelized to reduce complexity and allow it to be projected onto a 2.5D height map. The height map is the primary input into the method detailed in this study. The method outputs a path made of bezier curves, which the robot interprets and follows. A flowchart of this system is shown in figure 3.

3.2 2.5D and Obstacle Grid Construction

1)Height Inference and Unknowns: To assist the path planning and exploration systems, cells which have not yet been seen by images from the differing camera positions due to obscurance by an object or wall must be marked out. This allows the path planning algorithm to treat unknown areas differently compared to normal objects. This allows the path planning algorithm to treat unknown areas as potential for further exploration should no path to the goal be apparent.

Furthermore, this algorithm infers the height of the unknown areas, repairing missing data from imperfect point clouds.

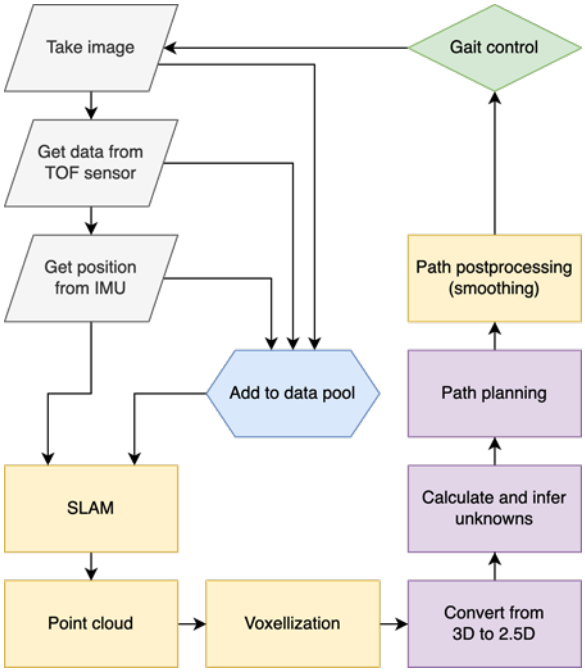


Fig.3 Flowchart of the Robotic Control System

The algorithm effectively assumes each object as infinitely deep, unless proven otherwise by another camera angle. Each cell in the 2.5D height map can be in the field of view of 0…*N* camera viewpoints. For each viewpoint, the cell is either directly seen by the camera, or has been obscured by a cell taller than it between it and the camera position. The algorithm works from every camera viewpoint, inferring the height of cells within its field of view. Each cell's camera-wise inferred height is calculated as either relative to the height of the highest cell collinear to and in between the current cell and the camera's position, or the height of the current cell, should no collinear cell be higher than the current cell's height. If the height of the highest collinear cell is lower than the camera's height, the camera-wise inferred height is linearly relative to the height of the blocking cell and the camera. Else, the camera-wise inferred height is equal to the blocking cell's height. This may underestimate the height of obscured cells, and prevents large holes not stemming from camera errors from being filled in.

2) Obstacle Boundary Detection: Obstacle boundary detection is used to mark what places a robot cannot traverse. If a method does not detect obstacle boundaries, and only utilizes the slope in between two cells as a cost, it is very likely that a path planning algorithm would attempt to traverse over a very steep slope in order to reach a goal or for a significant reduction in distance.

For obstacle boundary detection, a maximum slope thresh- old must be specified. This threshold should be specific to a robot's design, limitations, and payload. This method takes into account the slope between a cell and its 8 neighboring cells, as in figure 4. If the slope between a cell and its neighbor, designated as the difference in heights of the cells divided by the distance between the centers of the two cells, is greater than the threshold, the boundary between the two cells is considered an obstacle. This method also considers that if a slope is greater than the threshold, and the neighboring cell is "unknown", the boundary is marked as an inferred boundary. This means that the path planning algorithm may explore this boundary to ensure it is present should no other paths to a goal be apparent. This results in 8 obstacle boundary grids, one for each direction a cell has a neighbor.

1) Enhanced A star: The A* algorithm is a common path planning algorithm, utilized to find the shortest path between two nodes. The algorithm works by expanding the unfinished path with the least cost – determined by the cost from start to the current node, as well as a heuristic distance function from the current node to the goal. This results in a fast searching algorithm. However, this

algorithm only seeks to minimize the path length, and does not work well when slopes and risk need to be taken into account.

The path, and by extension the robot, stays close to obstacles for a shorter path, increasing the risk of collision and uncompensatable external influence. As well, this standard algorithm ignores the risk from traversing over rough and sloped terrain.

Thus, an enhanced version of the A* algorithm was developed to better prioritize risk over path length. At every iteration of the loop, this algorithm determines the most effective path to extend, or more specifically, the path which minimizes:

$$cost(n) = a \times g(n) + b \times h(n) - c \times f(n) + d \times k(n) \quad (1)$$

where n is the last node on the path being checked. A, b, c, and d are weights that should be adjusted to balance between path length, distance to obstacles, and path roughness. $G(n)$ is the cost from the start to n, designated as the euclidean path length. Specifically, it is calculated as:

$$g(n) = \sum_{i=1}^{path\ length\ -\ 1} dist(P_i, P_{i+1}) \quad (2)$$

where P_i is the ith node on the current path to node n. $h(n)$ is the heuristic function for the cost from n to the goal, set as the euclidean distance from n to the goal. $F(n)$ is the euclidean distance between the current path to n and the closest obstacle. This is represented with:

$$f(n) = \min\{D_1 \dots D_k\},\ D_i = \min_{j=1 \to obs} \left(dist\left(P_i, O_j\right)\right) \quad (3)$$

Algorithm 1: Detection and computation of unknown cells

Input:
w by h height map H
Amount of cameras N
Camera viewpoints $C^1 \dots C^N$

Output:
w by h total inferred height map I
w by h "unknowns" map U

Initialization:
N w by h camera-wise inferred height maps $D^1 \dots D^N$
w by h total inferred height map I filled with infinity
w by h "unknowns" map U

for camera viewpoint C^i from 1 to N **do**
  for cell (x, y) in field of view of C^i **do**
    $D^i_{x,y} \leftarrow H_{x,y}$
    $\theta_1 \leftarrow \tan^{-1}(\frac{y - C^i_y}{x - C^i_x})$
    $\theta_2 \leftarrow \tan^{-1}(\frac{y+1 - C^i_y}{x - C^i_x})$
    $\theta_3 \leftarrow \tan^{-1}(\frac{y - C^i_y}{x+1 - C^i_x})$
    $\theta_4 \leftarrow \tan^{-1}(\frac{y+1 - C^i_y}{x+1 - C^i_x})$
    $\theta_L \leftarrow \max(\theta_1, \theta_2, \theta_3, \theta_4)$
    $\theta_R \leftarrow \min(\theta_1, \theta_2, \theta_3, \theta_4)$
    for cell (a, b) where
      $\theta_R < \tan^{-1}(\frac{b - C^i_y}{a - C^i_x}) < \theta_L$ or
      $\theta_R < \tan^{-1}(\frac{b+1 - C^i_y}{a - C^i_x}) < \theta_L$ or
      $\theta_R < \tan^{-1}(\frac{b - C^i_y}{a+1 - C^i_x}) < \theta_L$ or
      $\theta_R < \tan^{-1}(\frac{b+1 - C^i_y}{a+1 - C^i_x}) < \theta_L$ **do**
        if $H_{a,b} > C^i_x$ **then**
          $D^i_{x,y} \leftarrow \max(D^i_{x,y}, H_{a,b})$
        else
          $D^i_{x,y} \leftarrow \max(D^i_{x,y}, -\frac{C^i_x - H_{a,b}}{2\text{Ddist}(C^i,(a,b))} + C^i_z)$
    $I_{x,y} \leftarrow \min(I_{x,y}, D^i_{x,y})$
for cell (x, y) in H **do**
  if $I_{x,y} = H_{x,y}$ **then**
    set $U_{x,y}$ to "known"
  else
    set $U_{x,y}$ to "unknown"

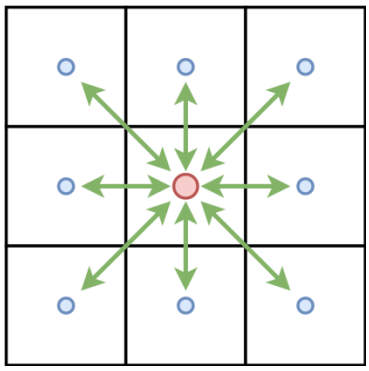


Fig.4 Neighboring Cells to Check for Obstacle Boundaries

where P_i is the ith node on the current path to node n, obs is the amount of boundaries marked as "obstacle", and O is the set of all boundaries marked as "obstacle". This makes the algorithm prefer paths which are further from obstacle boundaries, preventing cut corners and reducing the amount of turns. $K(n)$ is the cost to traverse over the terrain from the start to n, calculated as the sum of slopes on the path, or:

$$k(n) = \sum_{i=1}^{path\ length\ -\ 1} \left| P_{height}^{i} - P_{height}^{i+1} \right| \quad (4)$$

where P_i is the ith node on the current path to node n. This makes it so the algorithm would prefer a flat path over a bumpy or sloped path.

3)Path Postprocessing: The output of the A* algorithm is a list of nodes, denoting the ideal path found. However, due to the grid-constrained search of the A* algorithm, it is impossible to represent straight lines in angles not aligned with the grid's cardinal or diagonal directions. As well, every grid cell that is passed through is represented as a point on the path, leading to many redundant points and high frequency, low amplitude noise in lines not aligned with the grid's cardinals or diagonals. As such, before such a path can be utilized by a robotic system, some post processing operations must be applied to reduce or remove the noise and redundant points.

The first post processing operation used is the Ramer- Douglas-Peucker algorithm. It recursively divides the path by the node with the largest distance away from the line connecting the start and end nodes of a section. If the node is further than a threshold, it is kept, and the path is divided. If not, then the node is deleted. This algorithm removes most of the redundant points in the path, and with it removes a large portion of the noise.

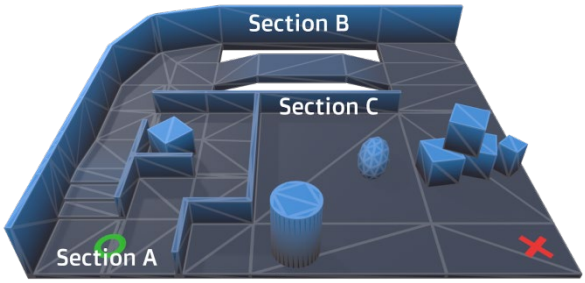


Fig.5 An Obstacle Course Containing rough and Flat Paths, a Bridge, and Open Area with Obstacles Used to Test the Unknowns Inference and Path Planning Algorithms.

The second post processing operation is used both to further smoothen out the path as well as to adapt the path into a format that can be used by the robot's gait control algorithm. The operation converts the path into a set of quadratic bezier curves, by designating the midpoints of two consecutive path segments as the endpoints of the curve, and the junction between the path segments as the control point.

4. Results

The presented method of detecting and inferring unknowns, as well as the enhanced method for

rough-terrain path planning were tested on a variety of simulated and real-life terrain. Figure 5 shows the construction of the simulated course, including paths through rough and flat terrain (section A), a bridge over a gap (section B), and an open area with obstacles (section C), totaling 7m by 6m. The start of the course is denoted with the green circle, and the goal the red cross. The goal of this course is to travel from the starting point to the goal, while minimizing the risks taken by a hypothetical robot.

4.1 Height Inference and Unknowns

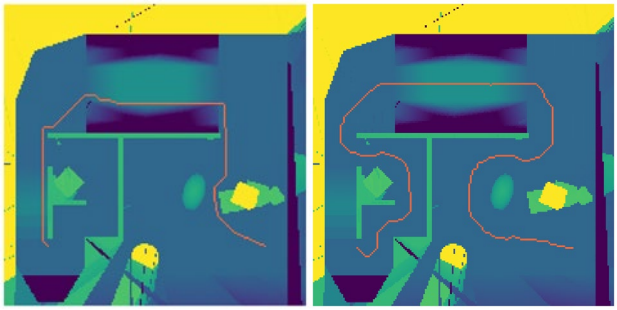
As can be seen in figure 6b, areas not explicitly seen by the cameras, marked in yellow, have had their heights inferred, creating boundaries. As well, holes due to imperfections in data, as can be seen in the top left corner of figures 6a and 6c, have been filled. However, as in the case of section B, the width of the bridge has been overstated by the algorithm due to the prospective nature of cameras and LIDARs. Thus, unless an edge can be explicitly seen by a camera, the unknowns algorithm will likely attempt to extend the edge both horizontally and depth-wise relative to the camera, causing the overstatement of the width and length of objects. As such extended areas are marked as unknown, additional data can be gathered closer to the edge in order to resolve this inference issue.

4.2 Path Planning

Figure 7 shows a comparison between a standard A* path planner and the proposed enhanced path planner traversing over and around various obstacles and terrain. As evident in figure 7a, the standard A* always aims for the shortest path, regardless of risk. In section A, the standard A* chooses to traverse over a sloped section of terrain in return for a shorter path. In section B, the path stays extremely close to the edge of the bridge, and in section C, the path attempts to travel through a small gap, again in return for a shorter path. On the contrary, the enhanced path planner aims for a less risky path, as shown in 7b. It chooses a flat path, though it is longer, and attempts to maintain in the center of the bridge, reducing the risk of falling off. It also avoids the small passageway in section C, instead taking a longer path. Quantitative data can be seen in table 1, where the standard A* path has a shorter path length than the enhanced path planning, though has a higher total elevation change and lower average distance to obstacles. This means that throughout the path, the standard A* gains and loses more elevation on average than the enhanced path, as well as staying closer to obstacles and non-traversable areas.

a) Raw height map b) Inferred areas c) Inferred heights

Fig.6 The Original Raw Height Map, Areas Which Have Had Their Heights Inferred, and the Complete Height Map with Inferred Heights.



a)Standard A* b) Enhanced A*

Fig.7 Standard a* Path Versus Enhanced a* Path

Table 1: Quantatative Performance of Standard Versus Enhanced a*.

TABLE 1: Quantatative Performance of Standard Versus Enhanced A*.			
Path planning algorithm	*Path length (m)*	*Total elevation change (m)*	*Average distance to obstacle (m)*
Standard A*	14.78	1.865	0.180
Enhanced A*	20.18	0.557	0.535

4.3 Real World Testing

Real-world testing has been conducted by integrating the proposed methods into the quadruped robot described in the system overview. Due to the low-cost nature of the robot, the motor precision causes the robot to drift approximately 5%. A variety of situations have been tested to determine the reliability of the path planning algorithm when applied to an imperfect system. As shown in figure 8, the robot is able to completely avoid obstacles on the path to the goal even with the aforementioned drift. In figure 1, the robot chooses a flat path over a sloped path, while also maintaining distance from the wall of the sloped path, despite the drift.

5. Conclusion

This paper presents a novel method for inferring obstacles from incomplete data, as well as an enhanced A* algorithm designed to reduce risk taken by a robot, thus making it specialized for rough terrain traversal. The height of a cell in the height map is inferred based on the tallest object which obscures it from a camera viewpoint, and inferred heights from multiple viewpoints are combined to create a holistic estimate of the heights of unseen cells. This assists in reducing holes left by incomplete data, as well as marking out what areas have not yet been investigated and allowing for the introduction of additional data. In addition, slopes too steep to be traversed are marked out. As for the enhanced A* algorithm, a slope cost to reduce the favorability of rough terrain, and an obstacle distance cost to prevent a path from taking more risk by attempting to stay near obstacles were added. Finally, the path is simplified using the Ramer-Douglas-Peucker algorithm and smoothed into a set of bezier curves. The proposed method has been tested and has shown its effectiveness both in simulation as well as on a low-cost quadruped robot. Even with 5% lateral drift, the robot was able to autonomously navigate around obstacles without contact.

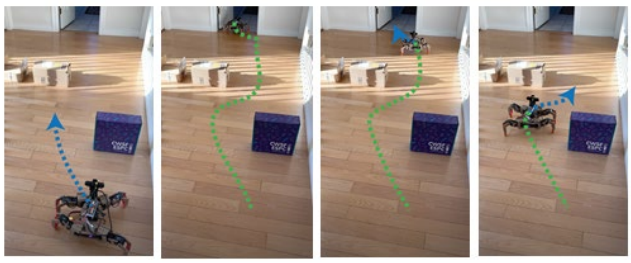


Fig.8 Robot Avoiding Boxes by Walking in s-Curve. the Blue Line Shows the Detected Path, the Green Line Shows the Already-Walked Path.

For further improvement, a roughness cost to quantify the risk of traversing over terrain with high-frequency variations in height, such that the variations cannot be captured within one grid cell, would assist a robot in deciding between a flat path and a roughly flat but rocky path. As well, a slope cost that takes into account the width of the robot may assist in further reducing risk taken by a robot in the case that the ground roughness greatly differs throughout the width of the robot. Both these improvements would continue to reduce risk taken by a robot.

References

[1] W. Dijkstra, “A note on two problems in connexion with graphs,” NUMER MATH, vol. 1, pp. 269-271, 1959.

[2] S. M. LaValle, “Rapidly-exploring random trees: a new tool for path planning,” The annual research report, 1998

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968, doi: 10.1109/TSSC.1968.300136.

[4] A. Stentz, “Optimal and efficient path planning for partially-known environments,” in Proceedings of the 1994 IEEE International Con- ference on Robotics and Automation, 1994, pp. 3310–3317 vol.4. doi: 10.1109/ROBOT.1994.351061.

[5] J. Carsten, D. Ferguson, and A. Stentz, “3D Field D: Improved Path Planning and Replanning in Three Dimensions,” 2006 IEEE/RSJ In- ternational Conference on Intelligent Robots and Systems. IEEE, Oct. 2006. doi: 10.1109/iros.2006.282516

[6] F. Duchonˇ et al., “Path Planning with Modified a Star Algorithm for a Mobile Robot,” Procedia Engineering, vol. 96. Elsevier BV, pp. 59–69, 2014. doi: 10.1016/j.proeng.2014.12.098.

[7] T. Dudzik et al., “Robust Autonomous Navigation of a Small-Scale Quadruped Robot in Real-World Environments,” 2020 IEEE/RSJ Inter- national Conference on Intelligent Robots and Systems (IROS). IEEE, Oct. 24, 2020. doi: 10.1109/iros45743.2020.9340701.

[8] D. Kim et al., “Vision Aided Dynamic Exploration of Unstructured Terrain with a Small-Scale Quadruped Robot,” 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, May 2020. doi: 10.1109/icra40945.2020.9196777.

[9] H. Wang, S. Lou, J. Jing, Y. Wang, W. Liu, and T. Liu, “The EBS-A* algorithm: An improved A* algorithm for path planning,” PLOS ONE, vol. 17, no. 2. Public Library of Science (PLoS), p. e0263841, Feb. 17, 2022. doi: 10.1371/journal.pone.0263841.

[10] J. Gu, Q. Cao, and Y. Huang, “Rapid Traversability Assessment in 2.5D Grid-based Map on Rough Terrain,” International Journal of Advanced Robotic Systems, vol. 5, no. 4. SAGE Publications, p. 40, Nov. 01, 2008. doi: 10.5772/6233.

[11] J. Gu and Q. Cao, “Path planning for mobile robot in a 2.5- dimensional grid-based map,” Industrial Robot: An International Jour- nal, vol. 38, no. 3. Emerald, pp. 315–321, May 03, 2011. doi: 10.1108/01439911111122815.

[12] M. Go¨rner, A. Chilian, and H. Hirschmu¨ller, “Towards an Autonomous Walking Robot for Planetary Surfaces,” Aug. 2010. [Online]. Available: https://elib.dlr.de/67912/

[13] Wermelinger,Martin,Fankhauser,Pe´ter,Diethelm,Remo,Kru¨si,Philipp Andreas, Siegwart, Roland, and Hutter, Marco, “Navigation Planning for Legged Robots in Challenging Terrain,” ETH Zurich, 2016, doi: 10.3929/ETHZ-A-010686519.

[14] S. McCrory, B. Mishra, J. An, R. Griffin, J. Pratt, and H. E. Sevil, “Humanoid Path Planning over Rough Terrain using Traversability Assessment.” arXiv, 2022. doi: 10.48550/ARXIV.2203.00602.