Low-Cost Quadruped Robot with Rough Terrain Traversal, Obstacle Avoidance, and Autonomous Navigation

Henry Zhao

Collingwood School, West Vancouver
B.C, Canada

Unless otherwise noted, all images, data tables, and graphs were taken or created by finalist

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. One such example is Boston Dynamics' "Spot", which is able to perform extremely lifelike and complex movements. Currently, Spot is being employed to help people in many places, from helping doctors communicate with COVID patients, to running fully automated and remote routine checkups in factories and dams. It is not hard to see that in the future, robots such as Spot could be able to improve society as a whole. However, currently, Spot is very expensive, at around $75,000, making the cost a barrier for many non-profit or public organizations.

The main goal of this project was to build a cost effective legged robot that would be able to achieve autonomous movement and navigation through a variety of different terrain. The ultimate use of such a robot would be to explore human-unreachable or hazardous areas, as well as provide domestic help to people in need.

2 Previous work

Previously, the physical prototype of the robot was created from scratch, from 3D designing, to manufacturing, to basic control system programming. It could move in a straight line using some simple walk cycles, and could adaptively walk over slopes, stairs, and other rough terrain. However, the robot still was missing one major component: the vision system. Without it, the robot would not be able to detect and avoid obstacles, plan its own paths, or follow objects.

3 Hypothesis

It is hypothesized that with only a stereo camera and no prior knowledge of the area, a walking robot should be able to autonomously pick an optimal path, and traverse rough terrain. Together with the built-in motor feedback system developed in the previous project and a sophisticated control system, a robot costing less than $1,000 should be able to be created, and still achieve autonomous navigation as well as a variety of other functions.

4 Design

4.1 Hardware and Electronics

In addition to the previous hardware, the following items have been added :

- Oak-D stereo camera for depth perception
  - Detection range: 35cm to 38.4m
  - Outputs depth image
- VL53L0X Time-Of-Flight (LIDAR) sensor
  - Detection range: 5cm to 1.2m
  - Detects objects closer than camera's range
- MG90S servo motor to turn camera and LIDAR
  - Allows for up to 180° view angle
- Rechargeable battery module for untethered and unrestricted autonomous movement
  - 2200mAh 30c 11.1v LiPo battery with 12v to 5v voltage converter
  - Allows for a running time of approx. 45 minutes.

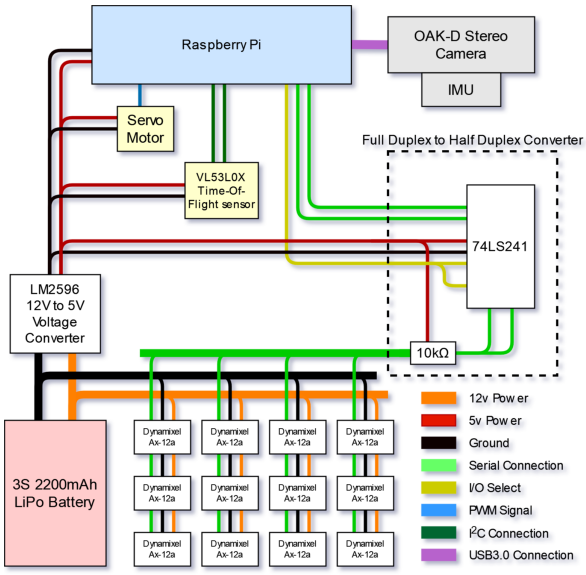


Figure 1: Electronic diagram of the robot

4.2 Overall Control System

To achieve autonomous navigation, including obstacle detection, avoidance, and rough terrain traversal, there are several major components which have been researched and developed. This includes vision-based environment, terrain, and obstacle detection, mapping and path planning, and procedural gait generation. Several novel algorithms have been developed and implemented, along with a full control system that integrates all the different components.

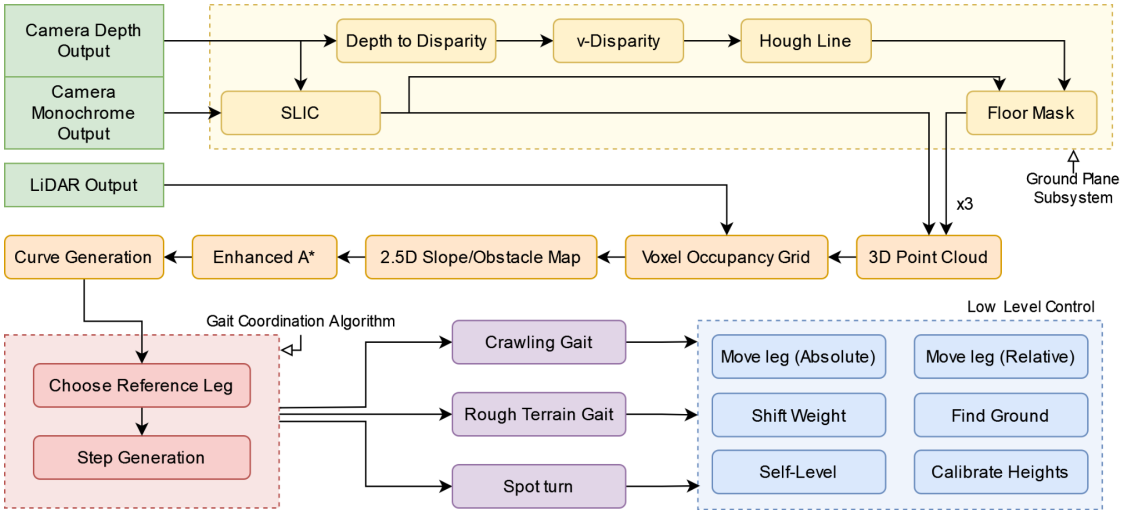
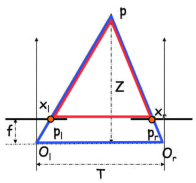


Figure 2: Flowchart of robot control system.

4.3 Depth and Disparity Processing

Stereo vision is one of the key components in this vision-based environment detection system. It allows the robot to see its surroundings in 3d, while being fast and relatively inexpensive. Through the OAK-D stereo camera, a depth image can be created. Additional processing to convert the depth image into a disparity image is required for later v-disparity processing.



(1.0): $Disparity = \frac{focal\ length\ in\ pixels \bullet baseline}{depth}$

Using equation 1.0, the disparity can be calculated by dividing the focal length times the baseline by the depth of each pixel.

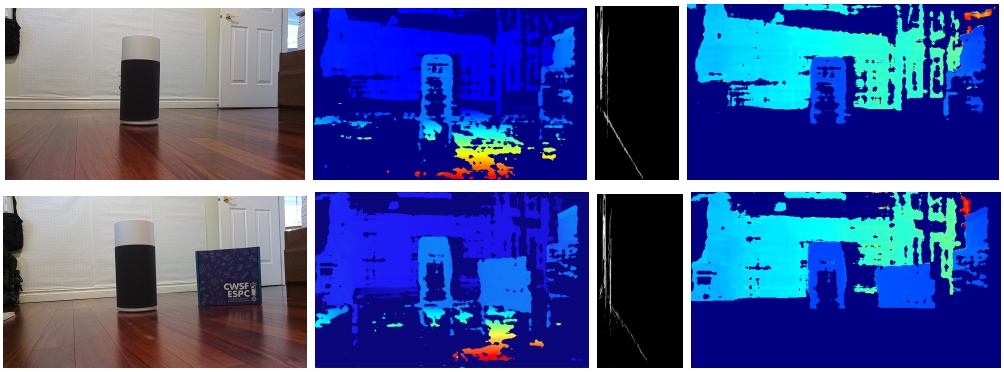
Image credit: cs.toronto.edu

4.3.1 Augmented Ground Plane Detection

In order to set a reference point for what areas the robot can easily navigate, the robot needs to know what objects can be walked over or under. As such, the floor height must be determined. An augmented ground plane detection algorithm based on v-Disparity and superpixel segmentation was researched and developed. V-Disparity is a matrix which stores the frequencies of disparity values for every row of the disparity image.

$$(2.0): v_{dj} = \sum_{i=0}^{cols} h_{ij},\ h_{ij} = 1\ if\ disp_{ij} = d,\ else\ 0$$

where v_{dj} represents the amount of pixels with a disparity of d in row j of the disparity image. As in figures 5 and 9, the ground surface is indicated with a slanted line of bright pixels. A line detection algorithm, called the hough line transform, is run on the v-disparity image to find the slope of the ground plane. During testing, it was observed that under certain conditions the camera could not detect the depth of certain areas, or would detect it wrongly, resulting in noise. To reduce these errors and result in better masking between ground and non-ground pixels, the monocular image was processed into groups of similarly coloured and grouped pixels, called superpixels. Specifically, the SLIC (Simple Linear Iterative Clustering) algorithm was implemented and tested for the generation of superpixels. Each pixel in a superpixel is then set to the average depth of all the pixels within the superpixel, excluding the unknown values. This algorithm will be run if the ground detection does not provide an accurate result, as an attempt to better segment the floor from obstacles. The result is then used to create a mask of the floor, to differentiate between walkable and unwalkable areas.



Figures 3-10: From left to right: color image, depth image, v-disparity, floorless depth image.

4.4 Object Detection

The next step of the process is to determine the size and position of obstacles, so that the robot would know what areas to avoid.

4.4.1 Depth to 3D

$$(3.0):\ X_{point} = depth_{image}$$

$$(4.0):\ Y_{point} = x_{image} \bullet \frac{depth_{image}}{focal\ length}$$

$$(5.0):\ Z_{point} = y_{image} \bullet \frac{depth_{image}}{focal\ length}$$

With equations 3.0 to 5.0, each pixel in the depth image can be mapped to a 3D coordinate, where X_{point}, Y_{point}, and Z_{point} are relative to the camera, with the camera pointing towards the positive X direction, positive Y being right of the camera, and positive Z being vertically upwards of the camera. This results in a point cloud (Figure 11), which can be further processed.

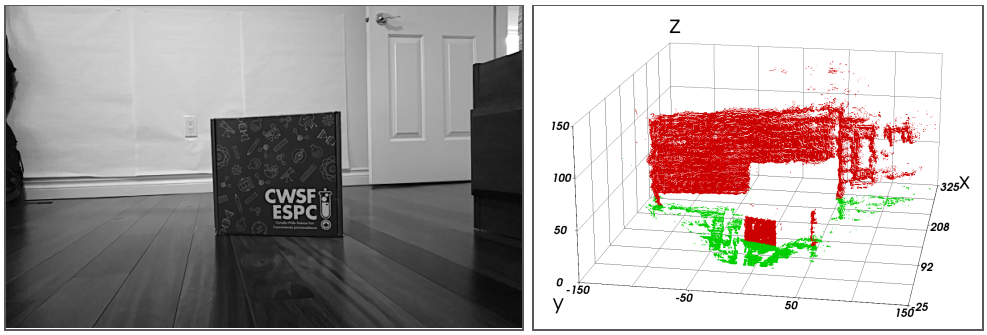


Figure 11: Depth image converted into a point cloud.

4.4.2 Panoramic View of 3D Point Cloud

To gain a wider view of the environment, the camera and LiDAR can be turned up to 80° in either direction. Thus, three depth images, and therefore 3 point clouds, can be captured. Each point cloud is rotated with the rotation matrix in equation 3.0, where θ is the angle at which each point cloud was captured. These point clouds can be merged together, to result in a wide panoramic view of the surrounding areas (Figure 12). This provides a wider field of view for finding potential paths and detecting obstacles.

$$(3.0)\ \begin{bmatrix} X_{\text{rotated}} \\ Y_{\text{rotated}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} X_{\text{original}} \\ Y_{\text{original}} \end{bmatrix}$$

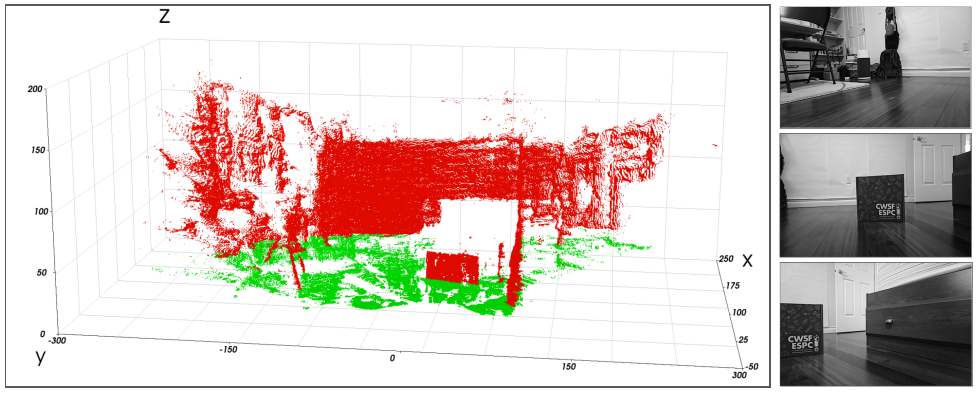


Figure 12: 3 point clouds merged together, with θ = -45, 0, and 45 respectively.

4.4.3 Fast Voxel Occupancy Grid Generation

Algorithm: Fast voxel occupancy grid and 2.5D height map construction using 3D point cloud

Input:
Point cloud P
Voxel size w, in mm
Total grid size g, in mm
Density threshold slope T_{slope}, threshold minimum T_{min}, and threshold maximum T_{max}

Initialization:
Initialize g/w by g/w by g/w density grid D , max height grid G , and occupancy grid O
Initialize g/w by g/w height map H
Create list of floor heights F

for points p in P **do**
  $v_x \leftarrow \text{floor}(\frac{p_x+g/2}{w})$
  $v_y \leftarrow \text{floor}(\frac{p_y+g/2}{w})$
  $v_z \leftarrow \text{floor}(\frac{p_z+g/2}{w})$
  if $0 < v_x < g/w$ and $0 < v_y < g/w$ and $0 < v_z < g/w$ **then**
    $D_{v_x,v_y,v_z} \leftarrow D_{v_x,v_y,v_z} + 1$
    $G_{v_x,v_y,v_z} \leftarrow \max(G_{v_x,v_y,v_z}, p_z)$

for coordinate (i,j,k) in D **do**
  if $D_{i,j,k} > T_{max}$ or $D_{i,j,k} > T_{slope} * \frac{1}{\sqrt{i^2+j^2+k^2}} + T_{min}$ **then**
    set $O_{i,j,k}$ to "object"
    if (i,j,k) is part of floor **then**
      append k to F

for objects o in O **do**
  $H_{o_x,o_y} \leftarrow \max(H_{o_x,o_y}, G_{o_x,o_y,o_z} - \text{average}(F))$

for coordinate (i,j) in H **do**
  $u_x \leftarrow i * w - g/2$
  $u_y \leftarrow j * w - g/2$
  $\Theta \leftarrow [\tan^{-1}(\frac{u_x}{u_y}), \tan^{-1}(\frac{u_x+w}{u_y}), \tan^{-1}(\frac{u_x}{u_y+w}), \tan^{-1}(\frac{u_x+w}{u_y+w})]$
  $a_l \leftarrow ((g * \cos(\Theta_{max}) + g/2)/w \ , \ (g * \sin(\Theta_{max}) + g/2)/w)$
  $a_r \leftarrow ((g * \cos(\Theta_{min}) + g/2)/w \ , \ (g * \sin(\Theta_{min}) + g/2)/w)$

  for coordinate (x,y) in H **do**
    if $(x,y$ within the polygon formed by $(g/2, g/2)$, a_r, and a_l **then**
      $H_{x,y} \leftarrow \max(H_{x,y}, H_{i,j})$

A 3D point cloud provides a lot of useful information, in the form of a non-discrete view of the surrounding environment. However, it suffers from computational complexity and maintains noise stemming from the original depth images. To solve these problems, a fast voxelization and occupancy grid generation algorithm has been developed to reduce both computational complexity and the majority of present noise. The algorithm accumulates all the points in the point cloud into discrete cubes, or voxels. This results in a 3D density grid of voxels (Figure 13). A threshold proportional to the distance to the camera is applied by filtering out voxels with less density than the threshold. This greatly reduces the amount of noise, and gives the program a solid set of objects. To facilitate the path planning algorithm, this occupancy voxel grid is then projected onto a 2.5d plane, with each cell containing the highest filled voxel in the vertical column (Figure 14). Cells which are collinear to an obstacle and the camera are considered as "unknown", as they could be obscured by an object, and are set to the maximum of the current height of the cell and the height of the obscuring cell (Figure 15). This ensures that taller features only half-obscured are preserved. In order to detect unwalkable areas, if the difference between a cell and a neighboring cell is greater than a set threshold, the boundary is considered an obstacle, and unwalkable (Figure 17). Finally, all areas outside of the robot's view are considered as unwalkable, as the robot does not know what could be in those areas.

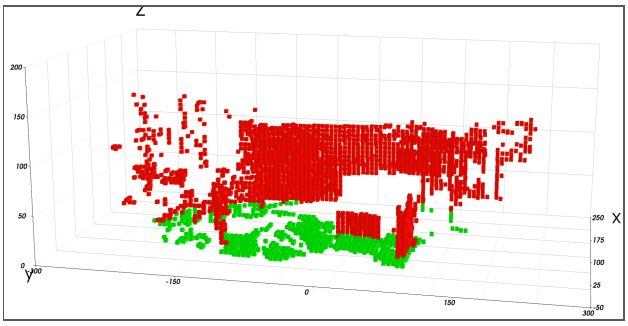
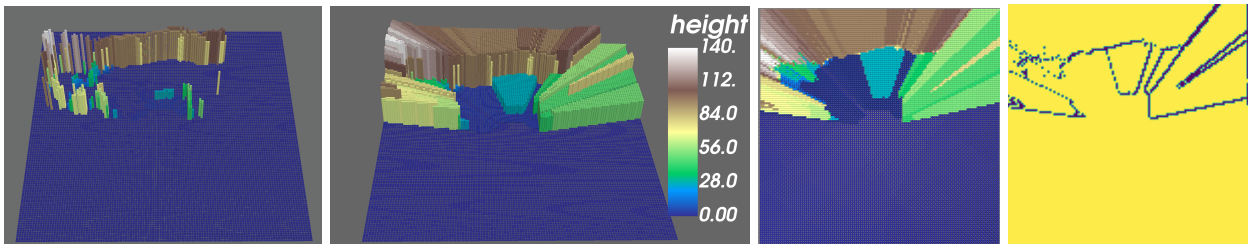


Figure 13: Generated voxel occupancy grid.



Figures 14-17: From left to right: height map, height map with inferred cells, birds-eye view of height map, map of walls.

4.5 Path Planning

The 2d grid map is preprocessed, by expanding the unwalkable areas by half of the robot's walking width. This allows the path planning algorithm to ignore the robot's width, while ensuring that the robot does not collide with obstacles.

*4.5.1 Enhanced A**

Path planning has been a deeply-studied field in robotics. There have been many algorithms developed, each built for different limits and use cases. Some examples include ant colony algorithms, particle swarm optimization, artificial potential fields, and the A* algorithm. 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 is unfit for the purpose of a walking robot. As can be seen in Figure 19, the path stays close to obstacles for a shorter path. This causes the robot, following the path, to stay close to obstacles as well, 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

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

where *a*, *b*, *c*, and *d* are tuned weights, and *n* is the end node on the path being checked. *g(n)* is the cost from the start to *n*, designated as the path length. Specifically, it is calculated as

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

where P_i is the *i*th node on the current path to node *n*. *h(n)* is the heuristic function for the cost from *n* to the goal. It is set as the distance (Euclidean or Manhattan) from *n* to the goal. *f(n)* is the distance between the current path to *n* and the closest obstacle. This is represented with

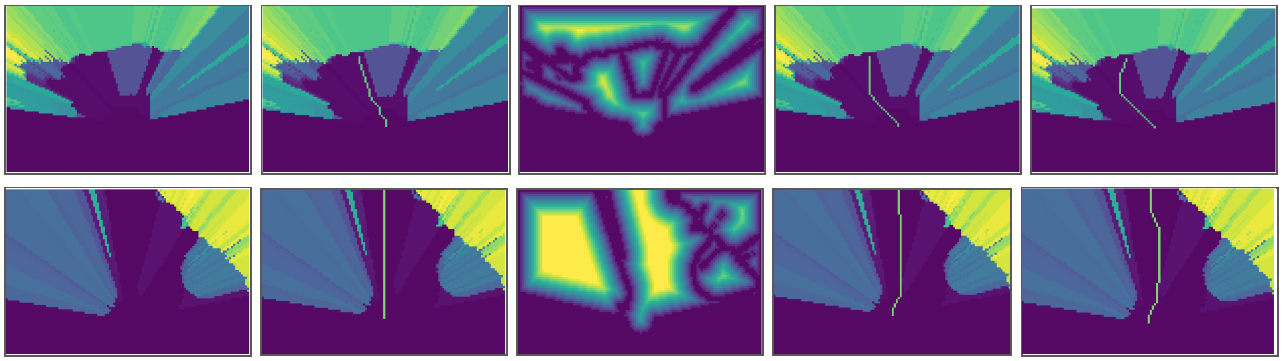
$$f(n) = \min\{D_1, ..., D_k\}\ ,\ D_i = \min_{j=1 \to \text{obs}} (dist(P_i, O_j))$$

where P_i is the *i*th node on the current path to node *n*, *obs* is the amount of cells marked as "obstacle", and *O* is the set of all cells marked as "obstacle". This causes the algorithm to prefer paths which are further from obstacles, smoothing the path and preventing the algorithm from

attempting to cut corners. *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

$$k(n) = \sum_{i=1}^{\text{path length}-1} \left| P^{i}_{\text{height}} - P^{i+1}_{\text{height}} \right|$$

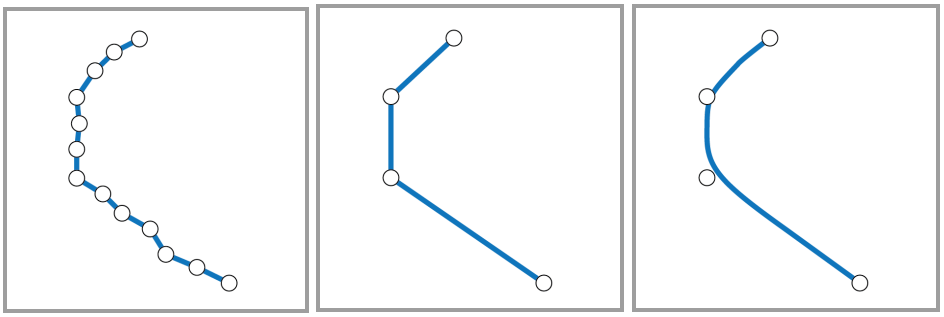
where P^i is the *i*th node on the current path to node *n*. This makes it so the algorithm would prefer a flat path over a bumpy or sloped path. For *g*, *h*, and *f*, multiple distance functions have been tested, including Manhattan, Euclidian, and other Minkowski distance metrics. For the robot to autonomously navigate, the goal was set to the outer edges of the map, incurring the robot to walk as far from its origin as possible.



Figures 18-27: From left to right: 2d occupancy grid, A* with Manhattan distance, A* with Euclidean distance, map of distances to closest object, enhanced A* with Euclidean distance.

4.5.2 Curved Path Generation

The enhanced A* algorithm provides a list of points on the path. However, for the robot to properly walk along this path accurately and stably, the amount of points, as well as high-frequency noise, must be removed, while keeping intact the shape of the path. The Ramer-Douglas-Peucker (RDP) algorithm was chosen. The algorithm recursively divides the path, removing points that deviate less than a set threshold, resulting in a simplified path (Figure 29). For more smoothness, the path is converted into a set of quadratic bezier curves, using the midpoints of the line segments as ending and starting points (Figure 30).



Figures 28-30: From left to right: raw path, simplified path, path as bezier curve.

4.6 Procedural Gait Generation

To make the robot walk smoothly, a novel gait generation algorithm was created to generate leg movements from a set of bezier curves. The robot simulates its body following the curve, while keeping the legs in the same global position. Every time a leg is projected to be outside of its reachable range, the leg is stepped forwards both in simulation and in real life. The relative positions of the legs are calculated, and the motor angles required to move the leg to its destination are calculated using inverse kinematics. However, during testing, it was found that the legs were not properly staggered, resulting in the robot becoming unbalanced and falling over in certain conditions. This was fixed by adding a reference leg system. Every time the reference leg takes a step, the algorithm forward-simulates the robot's body, as if it were to continue following the curve, without actually moving the robot. The algorithm figures out the distance the robot body travels between the current step, and the next step that the reference leg will have to take. The algorithm then tells the other three legs to take a step at ¼, ½, and ¾ of the distance between the two steps of the reference leg. This ensures that no two legs step too soon after one another, causing the robot to fall over. When the reference leg is on the inside of a turn, the reference leg travels less relative to the legs on the outside of the turn. This results in the inside-legs staying in their range, but the outside-legs overextending. A further improvement was made to switch the reference leg between the left or right side of the robot. When a leg that is a candidate for a reference leg, but is not currently one, takes a step, the robot again forward-simulates the robot's body. The algorithm calculates the distance the body would move in between the current non-reference leg step and its next, as well as the distance between the reference leg's next step and the step after that. It picks the one with the shortest distance, meaning the leg moves the fastest, ensuring that the reference leg picked is on the outside of a turn (Figure 31). In the process of creating this algorithm, a visual simulation tool was made to aid in debugging (Figure 32).

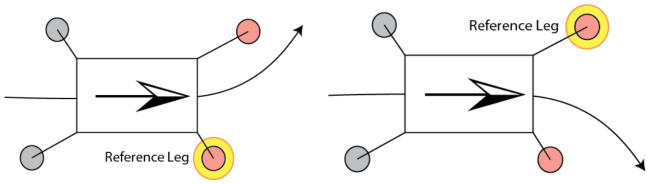


Figure 31: Chosen reference legs.

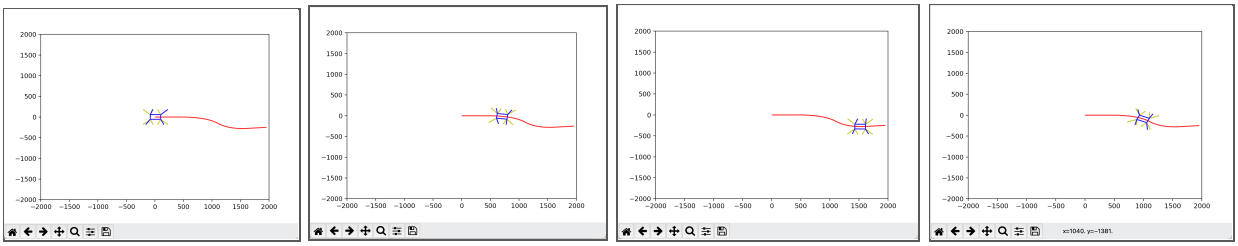


Figure 32: Gait visualization tool.

5 Results

With all the subsystems integrated into a full control system, various tests have been conducted, as listed below:

- **Basic instruction testing:** straight line walking with various gaits, spot turn, self-leveling, ground detection.

- **Integrated testing:** single to multi-obstacle avoidance, in a variety of path shapes (Figure 33), flat vs. sloped path planning (Figure 34), and stairs and slopes traversal (Figure 35).

- **Real-world testing:** Indoor and outdoor navigation.

The results show that the robot can detect and avoid obstacles, while choosing the best path for it to follow. This means that the robot avoids being too close to obstacles, as well as bumpy or very sloped terrain. However, if necessary, it can still walk over slopes and stairs. Listed below in table 1 are the robot's current performance specifications.

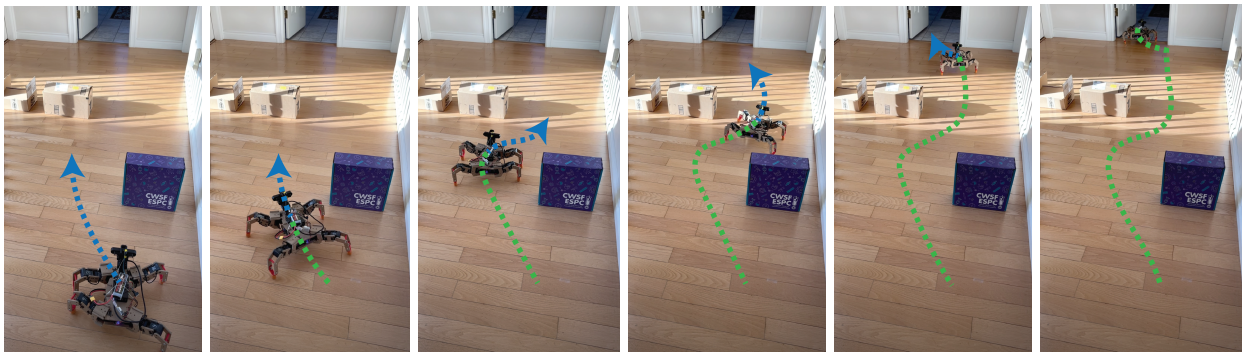


Figure 33: Robot avoiding boxes by walking in S-curve. The blue line shows the detected path, the green line shows the already-walked path.

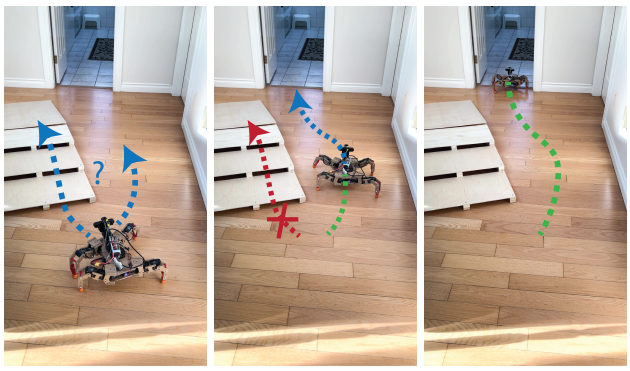


Figure 34: Robot choosing flat path over stairs.

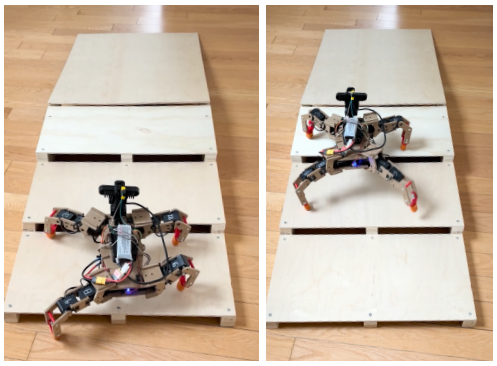


Figure 35: Robot walking over stairs.

Table 1: list of robot performance specifications

Trotting Gait Speed	0.5m/s
Crawling Gait Speed	0.25m/s
Rough Terrain Gait Speed	0.1m/s
Max Slope Degree	16°
Max Stair Height	10cm
Straight Line Accuracy	±5cm
Spot Turn Accuracy	±6%
Vision Processing Time	2s

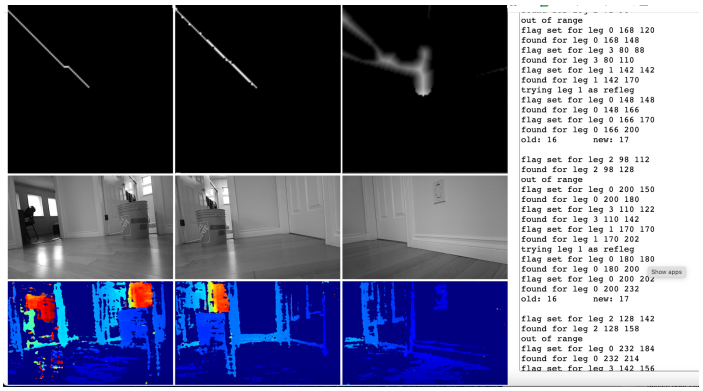


Figure 36: Web UI and log for live debugging.

6 Future Improvements

There are a few future improvements that could be made. Firstly, the ability to track the robot's current global position would be greatly beneficial. Further research into Simultaneous Localization and Mapping (SLAM) algorithms could possibly result in the development of such a system. This would improve the robot's walking and turning accuracy, and would allow for the robot to merge data from multiple locations and rotations, leading to a full map of the environment, as currently, there is a chance for the robot to get stuck in a loop, as it cannot record the areas that it has already been in. Global positioning would also allow for concurrent imaging and walking, aiding in the detection of dynamic and non-stationary obstacles. Secondly, a more extensive motor and sensor feedback system, allowing for reactions in unexpected situations, such as in losing stability or being flipped over. A platform for various attachments would greatly increase the possible use cases of this robot, as well as allowing for more sensors and limbs. Finally, exploration into object detection and recognition could greatly increase the robot's capabilities in more complex tasks.

7 Bibliography

Dudzik, T., Chignoli, M., Bledt, G., Lim, B., Miller, A., Kim, D., & Kim, S. (2020). Robust Autonomous Navigation of a Small-Scale Quadruped Robot in Real-World Environments. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3664–3671. doi:10.1109/IROS45743.2020.9340701

Görner, M., Chilian, A., & Hirschmüller, H. (08 2010). *Towards an Autonomous Walking Robot for Planetary Surfaces*.

Carsten, J., Ferguson, D., & Stentz, A. (2006). 3D Field D: Improved Path Planning and Replanning in Three Dimensions. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3381–3386. doi:10.1109/IROS.2006.282516

Gu, J., & Cao, Q. (05 2011). Path planning for mobile robot in a 2.5-dimensional grid-based map. *Industrial Robot: An International Journal*, *38*, 315–321. doi:10.1108/01439911111122815

Gu, J., Cao, Q., & Huang, Y. (11 2008). Rapid Traversability Assessment in 2.5D Grid-based Map on Rough Terrain. *International Journal of Advanced Robotic Systems*, *5*. doi:10.5772/6233

Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., & Jurišica, L. (12 2014). Path Planning with Modified a Star Algorithm for a Mobile Robot. *Procedia Engineering*, *96*. doi:10.1016/j.proeng.2014.12.098

Lewis, M., & Bekey, G. (05 2002). Gait Adaptation in a Quadruped Robot. *Auton. Robots*, *12*, 301–312. doi:10.1023/A:1015221832567

Bledt, G., Powell, M. J., Katz, B., Di Carlo, J., Wensing, P. M., & Kim, S. (2018). MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2245–2252. doi:10.1109/IROS.2018.8593885

Lategahn, H., Derendarz, W., Graf, T., Kitt, B., & Effertz, J. (2010). Occupancy grid computation from dense stereo and sparse structure and motion points for automotive applications. *2010 IEEE Intelligent Vehicles Symposium*, 819–824. doi:10.1109/IVS.2010.5548078

Garrote, L., Rosa, J., Paulo, J., Premebida, C., Peixoto, P., & Nunes, U. J. (2017). 3D point cloud downsampling for 2D indoor scene modelling in mobile robotics. *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 228–233. doi:10.1109/ICARSC.2017.7964080

Yu, C., Cherfaoui, V., & Bonnifait, P. (06 2015). *Evidential occupancy grid mapping with stereo-vision*. 712–717. doi:10.1109/IVS.2015.7225768

Ghazouani, H., Tagina, M., & Zapata, R. (11 2011). *Robot Navigation Map Building Using Stereo Vision Based 3D Occupancy Grid*.

Harms, H., Rehder, E., Schwarze, T., & Lauer, M. (2015). Detection of ascending stairs using stereo vision. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2496–2502. doi:10.1109/IROS.2015.7353716

Iloie, A., Giosan, I., & Nedevschi, S. (2014). UV disparity based obstacle detection and pedestrian classification in urban traffic scenarios. *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 119–125. doi:10.1109/ICCP.2014.6936963

Li, Y. (12 2013). *Stereo vision and LIDAR based Dynamic Occupancy Grid mapping : Application to scenes analysis for Intelligent Vehicles*.

Yiruo, D., Wenjia, W., & Yukihiro, K. (2013). Complex ground plane detection based on V-disparity map in off-road environment. *2013 IEEE Intelligent Vehicles Symposium (IV)*, 1137–1142. doi:10.1109/IVS.2013.6629619

Cong, Y., Peng, J.-J., Sun, J., Zhu, L.-L., & Tang, Y. (05 2010). V-disparity Based UGV Obstacle Detection in Rough Outdoor Terrain. *Acta Automatica Sinica*, *36*, 667–673. doi:10.1016/S1874-1029(09)60029-X

Soquet, N., Aubert, D., & Hautiere, N. (2007). Road Segmentation Supervised by an Extended V-Disparity Algorithm for Autonomous Navigation. *2007 IEEE Intelligent Vehicles Symposium*, 160–165. doi:10.1109/IVS.2007.4290108

De Cubber, G., Doroftei, D., Nalpantidis, L., Sirakoulis, G., & Gasteratos, A. (01 2009). *Stereo-based Terrain Traversability Analysis for Robot Navigation*.

Tsiogas, E., Kostavelis, I., Giakoumis, D., & Tzovaras, D. (11 2019). *V-Disparity Based Obstacle Avoidance for Dynamic Path Planning of a Robot-Trailer*. doi:10.1007/978-3-030-34995-0_14

Zhao, J., Whitty, M., & Katupitiya, J. (2009). Detection of non-flat ground surfaces using V-Disparity images. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4584–4589. doi:10.1109/IROS.2009.5354207

Labayrade, R., Aubert, D., & Tarel, J.-P. (01 2002). *Real time obstacle detection on non flat road geometry through v-disparity representation, in 'IEEE Intelligent Vehicles Symposium*.

Herghelegiu, P., Burlacu, A., & Caraiman, S. (2016). Robust ground plane detection and tracking in stereo sequences using camera orientation. *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, 514–519. doi:10.1109/ICSTCC.2016.7790717

Zhang, C., Li, Z., Cheng, Y., Cai, R., Chao, H., & Rui, Y. (2015). MeshStereo: A Global Stereo Model with Mesh Alignment Regularization for View Interpolation. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2057–2065. doi:10.1109/ICCV.2015.238

Song, H., Yoo, J., Kwak, S., Lee, C., & Choi, B. P. (10 2013). *3D mesh and multi-view synthesis implementation using stereo cameras and a depth camera*. 1–3. doi:10.1109/3DTV.2013.6676645

Chiang, S.-Y. (09 2016). Vision-based obstacle avoidance system with fuzzy logic for humanoid robots. *The Knowledge Engineering Review*, *32*, 1–11. doi:10.1017/S0269888916000084

Pandey, D. A. (01 2014). *Path Planning Navigation of Mobile Robot With Obstacles Avoidance Using Fuzzy Logic Controller*. doi:10.1109/ISCO.2014.7103914

You, B., Qiu, J., & Li, D. (2008). A novel obstacle avoidance method for low-cost household mobile robot. *2008 IEEE International Conference on Automation and Logistics*, 111–116. doi:10.1109/ICAL.2008.4636130

Seki, H., Shibayama, S., Kamiya, Y., & Hikizu, M. (2008). Practical obstacle avoidance using potential field for a nonholonmic mobile robot with rectangular body. *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, 326–332. doi:10.1109/ETFA.2008.4638414

Hidalgo-Paniagua, A., Vega-Rodríguez, M. A., Melero, J. F., & Pavón, N. (2017). Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach. *Soft Computing*, *21*, 949–964.

Chen, Y., Han, J., & Wu, H. (07 2012). Quadratic Programming-based Approach for Autonomous Vehicle Path Planning in Space. *Chinese Journal of Mechanical Engineering*, *25*. doi:10.3901/CJME.2012.04.665

Hassani, I., Maalej, I., & Rekik, C. (06 2018). Robot Path Planning with Avoiding Obstacles in Known Environment Using Free Segments and Turning Points Algorithm. *Mathematical Problems in Engineering*, *2018*, 1–13. doi:10.1155/2018/2163278

Shao, X., Huang, Q., Wang, Z., Cai, Q., & Wang, W. (2014). Motion planning and compliant control for a quadruped robot on complicated terrains. *2014 IEEE International Conference on Mechatronics and Automation*, 1587–1594. doi:10.1109/ICMA.2014.6885937

Laible, S., Khan, Y. N., Bohlmann, K., & Zell, A. (09 2012). *3D LIDAR- and Camera-Based Terrain Classification Under Different Lighting Conditions*. doi:10.1007/978-3-642-32217-4_3