https://doi.org/10.2352/EI.2023.35.5.IRIACV-323

3D Joist Perception, Detection, and Climbing for Hexapod Robot

Yibin Li[1] and Avideh Zakhor[1]

[1]Department of Electrical Engineering and Computer Science
University of California, Berkeley

Abstract

Avoiding obstacles is challenging for autonomous robotic systems. In this work, we examine obstacle avoidance for legged hexapods, as it relates to climbing over randomly placed wooden joists. Our main motivation is to enable robots to navigate inside tight attic spaces of single-family residential homes in the U.S., which typically contain rows of 4, 6, or 8 inch tall joists placed 16 inches apart from each other. We formulate the task as a 3D joist detection problem, and propose a detect-plan-act pipeline using a simultaneous localization and mapping (SLAM) algorithm to generate a pointcloud and a grid map to expose high obstacles such as joists. A line detector is applied on the grid map to extract parameters of the joist, such as height, orientation, and distance; based on this information the hexapod plans a sequence of leg movements to either climb over the joist or move sideways if the joist is too high. We show that our perception and path planning modules work well on the real-world joists with different heights and orientations.

Introduction

Attics are one of the largest sources of energy loss in residential homes. As such attic air sealing and insulation can result in a substantial reduction in home energy costs and its carbon footprint. One way to do this is through spray foaming which simultaneously helps to prevent insect infection, reduces energy costs, and keeps the main component of a home at a comfortable temperature. Despite its effectiveness, spray foaming the attic is challenging: the insulation material contains toxic substances so a protective suit must be worn. Figure 1 shows a typical attic during construction. As seen, attics are typically tight spaces with rows of joists and as such are uncomfortable and difficult for a human operator to work in. For example, if a human operator by mistake steps in the space between two joists, she or he could fall through the attic floor into the main compartment of the house, resulting in serious injuries. As such, legged robots are ideal candidates for navigating inside attics and carrying out various tasks such as vacuuming or air-sealing. While there has been a great deal of recent work on two and four legged robot locomotion, in this work, we will focus on using six legged robots for two reasons: first, hexapods are inherently more stable than bi-pedal or quadruped robots; second, their lower heights compared to bi-pedals and quadruped robots makes them more suitable for roaming around tight spaces for example at the corners of an attic. Since most single family residential home attics in the U.S. contain 4, 6, or 8 inch joists that are 16 inches apart, it is important for a hexapod to be able to detect, and climb such joists inside an attic. In this paper, we will focus on developing methods for a hexapod equipped with a depth camera to autonomously detect and climb joists of various heights and orientations.

Figure 1: Unfurnished attic with joists.

Hexapod and quadruped robots have been studied for many years. Back in 1990, Mcghee *et al.* [8] proposed a set of rules to navigate a hexapod in a simulated terrain. Putz *et al.* [10] proposed a 3D navigation path for mobile robots in uneven terrain, but their work is mainly for wheeled robots and does not consider the constraints of hexapod robots. More recently, Nguyen *et al.* [9] experimented with what they called a "library of gaits", a sequence of different leg gaits, on bipedal robots. Carlo *et al.* [6] used a more sophisticated convex Model Predictive Control (MPC) to control and plan locomotion on the quadruped robot dynamic system. Frankhauser *et al.* [5] developed a universal elevation map library in ROS for hexapod and quadruped. Frankhauser *et al.* [4] also applied the previous elevation map on their quadruped robots and reported solid results on quadruped robot navigation. Their trajectory planning algorithm spends a great deal of time balancing the quadruped robots, which is not an issue in our case since hexapods are more stable than quadruped.

For a hexapod to climb joists inside an attic, it must be able to use its perception system to both detect and parameterize the height and orientation of joists, so as to plan its path and climbing leg movement sequence. There have been attempts to use deep learning-based approaches on 3D pointcloud data to detect objects, but as Wang *et al.* [13] states, 3D object detection tasks require significantly more data to train than 2D. It is also expensive to acquire open-sourced 3D labels [14]. Most of the open-sourced 3D labeled data are released by self-driving companies and only focus on vehicles. Therefore, it would be time-consuming to label data and train a new 3D object detection model for joists from scratch.

This work aims to develop elementary building blocks for joist climbing of the legged hexapod. We test our proposed solution in a small row of joists assembled in our laboratory. The out-

line of this paper is as follows: we first discuss the hardware and software setup for our hexapod, then present the hexapod perception and actuation methods, followed by our experiments and results, and conclude the paper with final remarks and future works.

Setup

The hexapod used in our experiments is a Widow X from Trossen Robots shown in Figure 2. It features eighteen ultra fast DYNAMIXEL AX-18A series robot servos, six three-degree-of-freedom legs, a Raspberry Pi, and Trossen open-sourced SDK. Figure 2 also illustrates the complete hardware setup.

The onboard Raspberry Pi is primarily for hexapod control and does not have sufficient computing power for the perception module. To overcome this, we add an Intel NUC as an additional computing unit. Although NVIDIA's Jetson AGX Xavier features a better CUDA framework and a more powerful GPU for deep learning computing than Intel NUC, the latter computing power outperforms the former in sequential tasks, as illustrated in [3]. Since our algorithm does not run any deep learning models and we are performing "online" sequential path planning, Intel NUC is our choice for computing. To share the ROS network between NUC and Raspberry Pi, an Ethernet cable connects the two devices, and the IP addresses have been configured in such a way that the Raspberry Pi is the ROS master. Our planned communication schema between two devices is summarized in Figure 3.

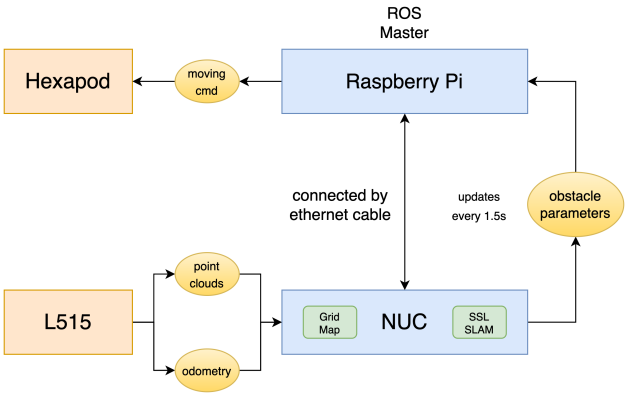


Figure 3: Communication diagram between devices.

The Widow X is not pre-loaded with any perception sensors. In order to perceive low light conditions and not to consume substantial precious computing power, the Intel L515 camera is chosen for the onboard perception sensor. L515 is a LiDAR camera with 3D depth, RGB, and IR output streams. We considered other stereo and IR cameras but L515 meets our needs best since it works well under low-light conditions. Once we capture the depth map of the surroundings, we plan trajectories accordingly to avoid obstacles or climb over joists.

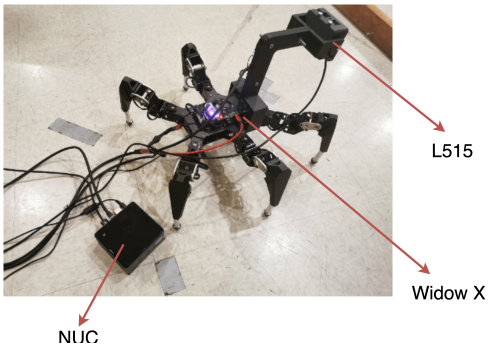


Figure 2: Hexapod, NUC, and L515 depth camera.

Since we are mostly interested in ground obstacle such as joists near the hexapod, it is useful to tilt the depth camera downward rather than having its optical axis parallel to the ground. To achieve that, we designed a simple nob to adjust the tilted angle of the mount. The nob holds the mount and can be easily set to 7 different pitch configurations: 0, 15, 30, 45, 60, 75, and 90 degrees, where at 0 and 90 degrees L515 is completely looking downward and looking straight ahead respectively.

Our overall approach can be summarized as follows: we use the perception system to detect the orientation and height of joists in the vicinity of the hexapod. If the estimated height of the nearest joist is too large for the robot to climb, it moves sideways. Otherwise, it approaches the joist and executes a set of pre-specified leg movements to climb over it.

Hexapod Vision System

Our vision system is a three-stage pipeline shown in Figure 4. Simultaneous Localization And Mapping (SLAM) and Grid-Map are the two core components in the hexapod vision system.

SSL-SLAM and Grid-Map

SSL-SLAM [12] is a visual SLAM algorithm which uses successive frames to reconstruct a scene using structure from motion, estimates motion via visual odometry, and localizes the robot in a global world map. The output of the SSL-SLAM algorithm is a dense colored pointcloud, in the format of ROS msg/pointcloud2. Grid-Map [5] is a two-dimensional grid map, namely a smooth surface quantized into grid cells, with multiple data layers. It is typically used as a central map information system for foothold search and trajectory planning and can be customized with useful data layers such as surface normal vectors and traversability. Grid map is stored as a 2D Eigen-matrix and can be converted from and to pointcloud, octomap, costmap_2d, and 2D images. In addition, the Grid-Map-OpenCV package provides a convenient interface to process Grid-Map images with OpenCV functions.

In the first stage of our pipeline, the SSL-SLAM algorithm takes in successive frames captured by L515 and outputs a dense colored pointcloud, estimated odometry data, and hexapod's position. In the second stage, due to the bandwidth of the computing unit NUC, the dense colored pointcloud is cropped into a $1m \times 1.2m \times 0.4m$ chunk around the hexapod. Next, this dense colored pointcloud chunk is processed by the Grid-Map library and converted to a grid map with one single layer of the cell's height. The grid map has a resolution of 0.01m, meaning each cell

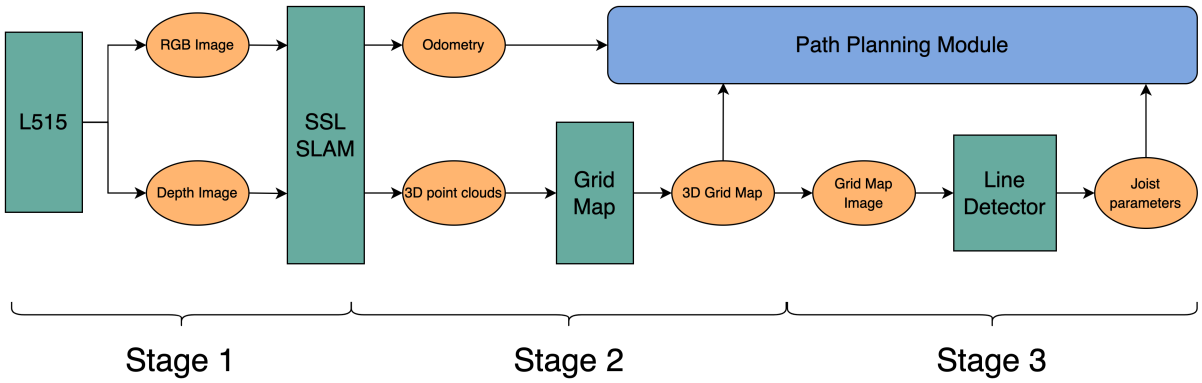
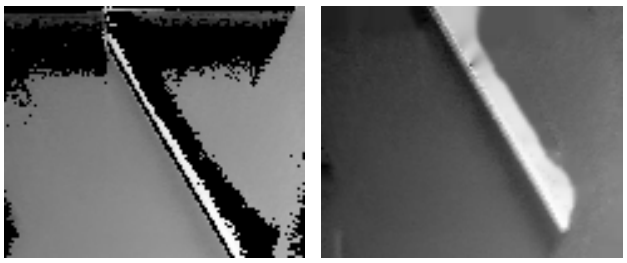


Figure 4: Three-Stage vision pipeline.



(a) (b)

Figure 5: 2D grid map (a) before and (b) after inpainting to fill missing points.

is a square of 1×1 cm. Next, this grid map goes through 12 layers of filters to calculate additional information such as elevation, variance, color, friction coefficient, foothold quality, surface normal, traversability, etc. After filtering, new information is stored as different data layers in the grid map.

There are missing points in the pointcloud due to occlusions, reflections, and other depth sensor imperfections, as shown in black in Figure 5a. To interpolate these missing points, we applied Fast Marching Inpaint method introduced by Telea *et al.* [11], which uses a binary mask of spots to fill in missing values in an image. In our case, the mask is a zero matrix with the same size as the grid map. The region of interest is marked as 1 in the mask matrix, which along with a 2D grid map is fed into the inpainting algorithm. Figure 6a shows the 2D grid map after inpainting.

Line Detection

The third stage of our pipeline involves detecting joists within the grid map. The joists are well-presented in the 3D pointcloud and grid map, but lack of semantic information is a challenge for our path planning and climbing sequence. Previous work on the grid map library Frankhauser *et al.* [4] only tests the quadruped walking algorithm on a semi-flat terrain with a grid map but not in a world with random joist-like obstacles.

Our approach to detect joists relies on traditional 2D line detection. A Grid map can be easily converted to a 2D image, representing a top-down view of the world with each cell filled with a height value. If we detect a joist from a top-down 2D view, we could compute its 3D world coordinates and deliver its parametric data such as height, orientation, and distance to the hexapod. More specifically, our joist detection algorithm follows these three steps:

1. Grid map is converted to a top-down 2D image. Edge Drawing line detector (ED-line) [1] is applied on a 2D grid-map image and all possible joist contour lines are returned.
2. Detected lines are overlaid on the original grid map and the average height of each line is estimated. Since we are interested in lines on actual joists, detected lines with average height of less than 5cm are assumed not to correspond to joists and are filtered out.
3. Joist distance is measured by the vertical pixel distance from the bottom of the grid map image to the center of the line as shown in Figure 7a. We calculate the joist distance for each line and the line with the smallest distance corresponding to the closest joist is chosen and returned.

Step 1, not all detected lines are on joists, as illustrated in Figure 6b. Some are formed between the floor and the inpainted region due to reflections. These lines are filtered out based on average height, in Step 2.

Step 2, the average height of a line $\hat{h}$ is estimated as follows: Given a grid map g with each cell's height of $h_{cell} = g(i,j)$, and given a line starting at (x_1,y_1) and ending at (x_2,y_2), the estimated average height is given by

$$\hat{h} = \frac{1}{(x_2-x_1+4)(y_2-y_1+4)} \times \sum_{i=x_1-2}^{x_2+2} \sum_{j=y_1-2}^{y_2+2} g(i,j) \qquad (1)$$

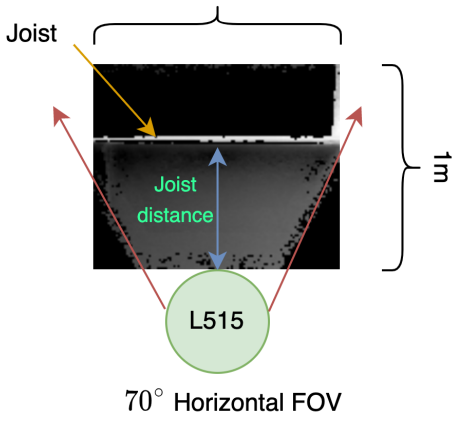
where $x_1 \leq x_2$, $y_1 \leq y_2$, and all (i,j) are within grid map g's boundary. Intuitively, in Equation (1), we traverse over a line and compute its average height over a stripe of 4 pixels width with 2 pixels on each side of the line. We use Edge Drawing line detector (ED-line) [1] to detect lines in the 2D grid map image as shown in Figure 6c.

Step 3, the method of computing joist distance is illustrated in Figure 7a. The joist orientation angle ψ is defined as the angle between the line defining the joist and the line perpendicular to the direction of the motion of the robot. Joist orientation as shown in Figure 7b is estimated by Equation (2) below, where (x_{start},y_{start}) and (x_{end},y_{end}) are two vertex coordinates defining the detected line for the joist:

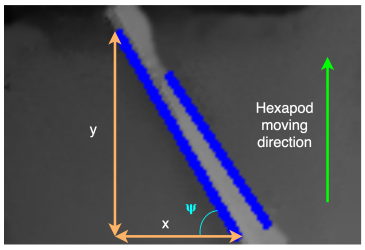
$$\psi = arctan(\frac{abs(y_{start}-y_{end})}{abs(x_{start}-x_{end})}) \qquad (2)$$

(a) (b) (c)

Figure 6: The result of ED-Line detection on a $1m \times 1.2m$ grid map. The L515 depth camera is at the bottom. Joist, floor, and joist reflection are in bright white, grey, and black colors respectively. (a) the grid map image after inpainting; (b) the ED-Line detector incorrectly detects two line segments on top; (c) these two line segments are filtered out based on underlying grid map height value.



(a)



(b)

Figure 7: Joist (a) distance and (b) orientation angle ψ estimation.

These three stages summarize our vision system. Once the robot detects a joist, it has to climb over or avoid it depending on its height [7].

Hexapod Joist Climbing

During the climbing sequence, the hexapod leaves enough space between its legs and the joist so that the legs do not hit the joist. The hexapod follows a simple "move straight" command and keeps moving forward in the world with a tripod gait, until the joist distance is less than our chosen threshold, 0.55 meters. The joist height is estimated from the vision system while the hexapod is moving. Joist height estimation helps to identify joists that are too high to climb over. In our attic setup, we have two types of joists with different heights, 10 cm and 18 cm. Our experiments have shown that the 10 cm joist is climbable while the 18 cm one is not due to the relative dimensions of the robot limb with respect to the joist height. Therefore, to plan hexapod motion, we need to take joist height into account. Once we estimate the joist height from the grid map, the rest of the logic is as follows: if the joist height is over 14 cm, the hexapod stops any climbing attempt and moves sideways; if the joist height is less than 14 cm, the hexapod continues climbing with caution.

(a) Pose 1: stop near joist, compute path. (b) Pose 2: move front legs up in the air. (c) Pose 3: reset body position

(d) Pose 4: move middle legs up in the air. (e) Pose 5: lean body forward. (f) Pose 6: reset body position.

(g) Pose 7: walk 1 gait cycle, lean forward. (h) Pose 8: move back legs up in the air. (i) Pose 9: reset body and all legs position.

Figure 8: Illustration of hexapod climbing sequence.

Figure 8 illustrates the hexapod climbing sequence over a joist. Once the hexapod is close to the joist, the onboard inverse kinematic (IK) solver computes trajectories for the front legs to cross over the joist through pre-computed waypoints. Next, the leg actuator executes the selected leg trajectories while the hexapod leans forward to compensate for the change of gravity center. Once this process is completed, the front two legs climb over the joist and now support the body from the other side of the joist. At this point, the hexapod moves forward with one cycle of tripod gait to bring two middle legs close to the joist. This process repeats again for the middle and back legs.

Experiments and Results

To verify our vision and climbing algorithm, two experiments are performed. Experiment descriptions are summarized in the table below.

Experiment	Description
1	Measure accuracy of the line detector against ground truth
2	Test integrated vision and climbing algorithms in seven settings

Experiment 1: Line Detection Accuracy

In this experiment, we investigate the accuracy of the line detector. Joists with different heights and orientation are tested in this experiment. The hexapod starts at 1m away from the joist and moves forward until its legs hit the joist. Figure 9a shows the experimental setup.

The result of this experiment is summarized in Table 1 and an example image is shown as Figure 9b. We can see that de-

(a) (b)

Figure 9: Experiment 1 for line detection accuracy; (a) joists are oriented at 1m in front of the hexapod; (b) line detection result on an oriented joist.

tected line segments estimates joist orientation relatively accurately, with a tolerance of $\pm$ 3°. Moreover, the estimated height is close to the actual height with about 0.01m difference after inpainting.

ψ (°)	$\overline{\psi'}$ (°)	ψ_e (°)	H (m)	h' (m)	H_e (m)
60	57	3	0.1	0.083	0.017
45	44.5	0.5	0.1	0.085	0.015
30	32	-2	0.1	0.081	0.019
60	55	5	0.18	0.187	-0.007
45	44	1	0.18	0.19	-0.01
30	32.5	-2.5	0.18	0.195	-0.015

Table 1: Estimated joist heights and orientation angle.

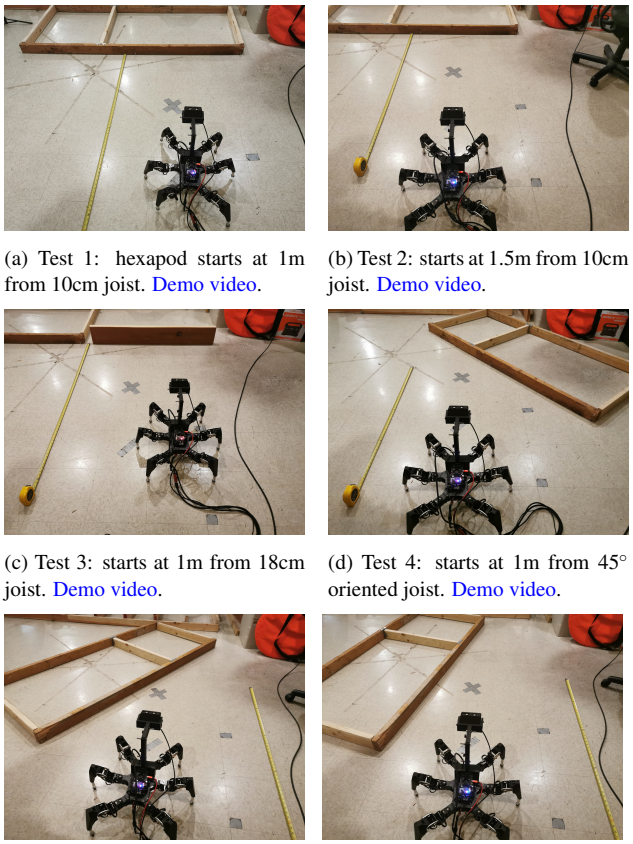
ψ: ground truth joist orientation yaw angle
$\overline{\psi'}$: estimated yaw angle of the joist
ψ_e: joist yaw angle estimation error
H: ground truth joist height
h': estimated joist height
H_e: joist height estimation error

Experiment 2: Climbing over Joists

For the next experiment, we combine the path planning and vision modules to determine whether the hexapod could correctly detect joists' parametric information and climb over joists. Seven tests are conducted, as shown in Table 2. The setup pictures of tests 1-6 are in Figure 10. In tests 1, 2, and 4-6, the hexapod successfully and safely climbed over joists placed in various configurations. In tests 3 and 7, the hexapod correctly identified the joist height and took the proper action to move sideways.

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
1	1	0°	10
2	1.5	0°	10
3	1	0°	18
4	1	45°	10
5	1	60°	10
6	1	75°	10
7	2.7	0°	18

Table 2: Experiment 2 setup. Seven tests are conducted to evaluate integrated vision and climbing algorithms.



(a) Test 1: hexapod starts at 1m from 10cm joist. Demo video.

(b) Test 2: starts at 1.5m from 10cm joist. Demo video.

(c) Test 3: starts at 1m from 18cm joist. Demo video.

(d) Test 4: starts at 1m from 45° oriented joist. Demo video.

(e) Test 5: starts at 1m from 60° oriented joist. Demo video.

(f) Test 6: starts at 1m from 75° oriented joist. Demo video.

Figure 10: Tests 1-6 setup from joist parameter estimation.

Test 7: Hexapod 2.7m away from 18cm joist

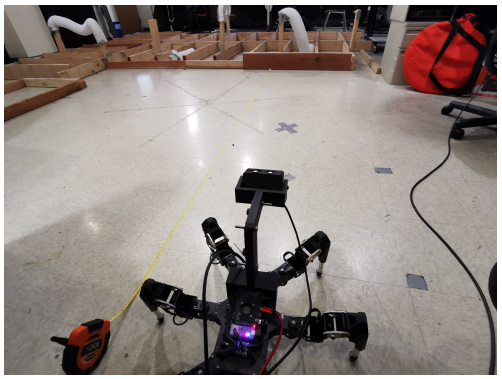


Figure 11: Test 7 starts at 2.7m from 18cm joist. Demo video.

In test 7, we measure the accuracy of our joist distance estimation. The input pointcloud size is changed from $1m \times 1.2m \times 0.4m$ to $3m \times 1.2m \times 0.4m$ to ensure that a joist is always in the view. The hexapod starts at 2.7m and moves forward for 1.8m. As soon as a new joist distance estimation shows up, we record the ground truth distance using a tape measure. After moving for 1.8m, the hexapod moves to the left, signaling it has detected a high joist of 18cm. Throughout the experiment, we collect 6

measurements. The joist distance error is summarized in Table 3, indicating estimated joist distance error is between 2cm and 16cm:

Measurement	d_E (m)	d_G (m)	E (m)
1	2.46	2.3	0.16
2	2.17	2.02	0.15
3	1.81	1.83	-0.02
4	1.54	1.61	-0.07
5	1.33	1.27	0.06
6	0.97	0.94	0.03

Table 3: Joist distance measurements and ground truth. d_E denotes estimated distance, d_G denotes ground truth distance, and E denotes error.

For Test 7, one can hypothesize possible causes of the error as follows:

1. The grid map processing delay: the grid map processing time highly depends on the density of the input pointcloud. Although we use synchronized odometry data to compensate for this processing delay so that the distance between the hexapod and joist is up to date, the delay between synchronizing grid map message and odometry message is non-negligible.
2. The quality of pointcloud: if the input pointcloud is dense, we should be able to compute a high-quality grid map and detect lines accurately. However, when the hexapod is far away, the pointcloud is sparse, resulting in inaccurate distance estimation.
3. The delay within ROS logging system between the time joist distance estimation is sent and the time it is actually logged on the screen might underestimate the distance.

Conclusions and Future Work

We investigated the problem of joist climbing for a hexapod robot. We used multiple wooden joists as obstacles and showed that the hexapod could detect and climb over joists successfully. We used a L515 camera and an Intel NUC to detect joists on the floor. The experiments showed that the hexapod is able to detect obstacles 2 meters ahead, plan trajectories accordingly, and successfully climb over joists or move sideways when encountering unclimbable ones.

From the experiments, we determine the limitations of our algorithms as follows. The material of obstacles could severely impact our detection algorithm. Moreover, the delay from processing units is non-negligible. Every unit in the pipeline needs to wait for the previous stage to complete. Recent work has proposed to combine the three stages using a single deep learning network, such that the robot "knows" where to "go" when it "sees" the obstacle [2]. Such an approach could potentially reduce both the delay and computation load.

References

[1] Cuneyt Akinlar and Cihan Topal. Edlines: Real-time line segment detection by edge drawing (ed). In *2011 18th IEEE International Conference on Image Processing*, pages 2837–2840, 2011.

[2] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. 2019.

[3] Houliston T. Mendes A. Chalup S.K. Biddulph, A. Comparing computing platforms for deep learning on a humanoid robot. *International Conference on Neural Information Processing (ICONIP)*, 2018.

[4] Marko; Bellicoso Dario; Miki Takahiro; Hutter Marco Fankhauser, Péter; Bjelonic. Robust rough-terrain locomotion with a quadrupedal robot. *The IEEE International Conference on Robotics and Automation*, 2018.

[5] Péter Fankhauser and Marco Hutter. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In Anis Koubaa, editor, *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, chapter 5. Springer, 2016.

[6] Benjamin Katz Gerardo Bledt1 Jared Di Carlo, Patrick M. Wensing and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[7] Yibin Li. Joist detection and climbing method for hexapod robots. Master's thesis, EECS Department, University of California, Berkeley, May 2022.

[8] S.H. Kwak R.B. McGhee. Rule-based motion coordination for a hexapod walking machine. *Advanced Robotics*, pages 263–282, 1990.

[9] William Martin Hartmut Geyer Quan Nguyen, Ayush Agrawal and Koushil Sreenath. Dynamic bipedal locomotion over stochastic discrete terrain. *Robotics: Science and Systems (RSS)*, 2017.

[10] Jochen Sprickerhofh Joachim Hertzberg Sebastian Putz, Thomas Wieman. 3d navigation mesh generation for path planning in uneven terrain. *International Federation of Automatic Control*, 2016.

[11] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9, 01 2004.

[12] H. Wang, C. Wang, and L. Xie. Lightweight 3-d localization and mapping for solid-state lidar. *IEEE Robotics and Automation Letters*, 6(2):1801–1807, 2021.

[13] Yilin Wang and Jiayi Ye. An overview of 3d object detection, 2020.

[14] Zhen Yang, Chi Zhang, Huiming Guo, and Zhaoxiang Zhang. Manual-label free 3d detection via an open-source simulator. 2020.