Joist Detection and Climbing Method for Hexapod Robots

Yibin Li
Avideh Zakhor, Ed.

Technical Report No. UCB/EECS-2022-166
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-166.html

May 22, 2022

Acknowledgement

I would like to thank my research advisor Professor Avideh Zakhor for the tremendous support she has given me in the past two years to learn and grow as a student and researcher. I learned substantially from her and other students in the Berkeley Video and Image Processing Lab, whose knowledge and dedication inspired me to keep learning on this field. I am also grateful for the invaluable advice Professor Yi Ma provided to me in my research journey. Most of all, I would like to thank my family and friends for their support throughout my education.

Joist Detection and Climbing Method for Hexapod Robots

by

Yibin Li

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Spring 2022

Joist Detection and Climbing Method for Hexapod Robots

by Yibin Li

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Avideh Zakhor
Research Advisor

5/22/2022

(Date)

* * * * * * *

Yi Ma

Professor Yi Ma
Second Reader

May 23, 2022

(Date)

Joist Detection and Climbing Method for Hexapod Robots

Abstract

Joist Detection and Climbing Method for Hexapod Robots

by

Yibin Li

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Avideh Zakhor, Chair

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. We formulate the task as a 3D joist detection problem, and propose a detect-plan-act pipeline using a 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 parametric information of the joist, such as height, width, orientation, and distance; based on this information the hexapod plans a sequence of leg movements to either climb over the joist or move sideways. We show that our perception and path planning module works well on the real-world joists with different heights and orientations.

Acknowledgments

I would like to thank my research advisor Professor Avideh Zakhor for the tremendous support she has given me in the past two years to learn and grow as a student and researcher. I learned substantially from her and other students in the Berkeley Video and Image Processing Lab, whose knowledge and dedication inspired me to keep learning on this field. I am also grateful for the invaluable advice Professor Yi Ma provided to me in my research journey. Most of all, I would like to thank my family and friends for their support throughout my education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Attics are one the biggest sources of energy loss in residential homes. As such attic air sealing and insulation can result in a substantial reduction in home energy costs. One way to do this is through spray foaming which simultaneously helps to prevent insect infection, reduce energy costs, and keep the main component of a home at a comfortable temperature. Despite its importance, spray foaming the attic is challenging: the insulation material contains toxic substances so the protective suit must be worn. Figure 1.1 shows a typical attic during construction. The attic's narrow and dark space makes it difficult for an operator to apply spray foam. Therefore, we consider using a robot to complete this job. With six legs easily crawling between joists, a hexapod robot is a perfect candidate for this task. It carries a spray foam gun, walks autonomously in the attic, and sprays insulation foam between gaps.

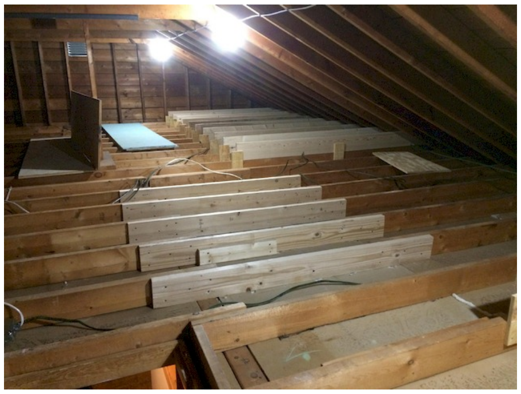


Figure 1.1: Unfurnished Attic with Joists

Inspired by attic spray foaming work, we intend to develop obstacle avoidance method for hexapods inside attics by developing perception and legged locomotion solutions.

Nevertheless, obstacle avoidance remains a hard challenge for robotics systems. Although this topic has been well-studied and many path planning algorithms have been proposed [9] [14], the gap between robots seeing obstacles and planning paths continues to exist.

The most common approach to avoid obstacles is detect-plan-act: the robot first detects and identifies the obstacle, then plans possible trajectories, and finally moves the actuator to the desired place. Robotic vision is essential in this process, and luckily, with recent developments in deep learning and sensors, robot vision has become easier and more affordable. The stereo cameras enable the robot to compute the depth and construct its surrounding environment with Simultaneous Localization And Mapping (SLAM). Nevertheless, as we will discuss in Chapter 2, traditional algorithms have many limitations and do not work in all scenarios. For example, the reconstructed SLAM maps typically do not carry semantic information, making it difficult for the robot to know which part of the pointcloud to pay attention to.

This work aims to present elementary building blocks for obstacle avoidance solutions for the legged hexapod. We use an indoor simulated attic to test whether the hexapod could detect and climb over joists. The outline of this work is as follows: in Chapter 1 we discuss the motivation and background, in Chapter 2 we present the approaches and experiments on hexapod robots joist climbing, and in Chapter 3 we conclude this project with our achievements and future works.

1.1 Related Work

Hexapod and quadruped robots have been studied for many years. Back in 1990, Mcghee et al. [10] proposed a set of rules to navigate hexapod in a simulated terrain. Putz et al. [12] 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. [11] experimented with what they called a "library of gaits", a sequence of different leg gaits, on bipedal robots. Carlo et al. [8] 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. also [4] 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, in our case, is not a major issue. Our hexapod is more stable than quadruped since it has six rather than four legs.

Another challenge is 3D object detection to detect and parametrize joists. Joist parameterization supplies the necessary information to the path planning algorithm. There have been attempts to use a deep learning-based approach on 3D pointcloud data to detect objects, but as Wang et al. [16] states 3D object detection tasks require significantly more data to train than 2D. It is also expensive to acquire open-sourced 3D labels [17]. 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 fresh 3D object detection model for joists from scratch.

Chapter 2

Joist Detection and Climbing

In this chapter, we present the joist climbing solution for a hexapod robot. This Chapter is organized as follows: In Section 2.1, we introduce the setup of our hexapod robot; in Sections 2.2 and 2.3, we describe the algorithm for the vision system and joist climbing algorithm; in Section 2.4, we discuss the ROS message synchronization and delay issues; in Section 2.5, we test the integrated system with real-world joists.

2.1 Setup

The hexapod used in this experiment is a Widow X from Trossen Robots. 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.

The onboard Raspberry Pi is primarily for Hexapod control and does not have enough computing power for the perception module. To solve this, we add an Intel NUC as our additional computing unit. Although NVIDIA's Jetson AGX Xavier features a much better CUDA framework and a much more powerful GPU for deep learning computing than Intel NUC, Intel's CPU computing power beats NVIDIA Xavier in sequential tasks, as illustrated in [3]. Since our algorithm does not run any deep learning modeling and we are performing "online" sequential trajectory 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 2.1.

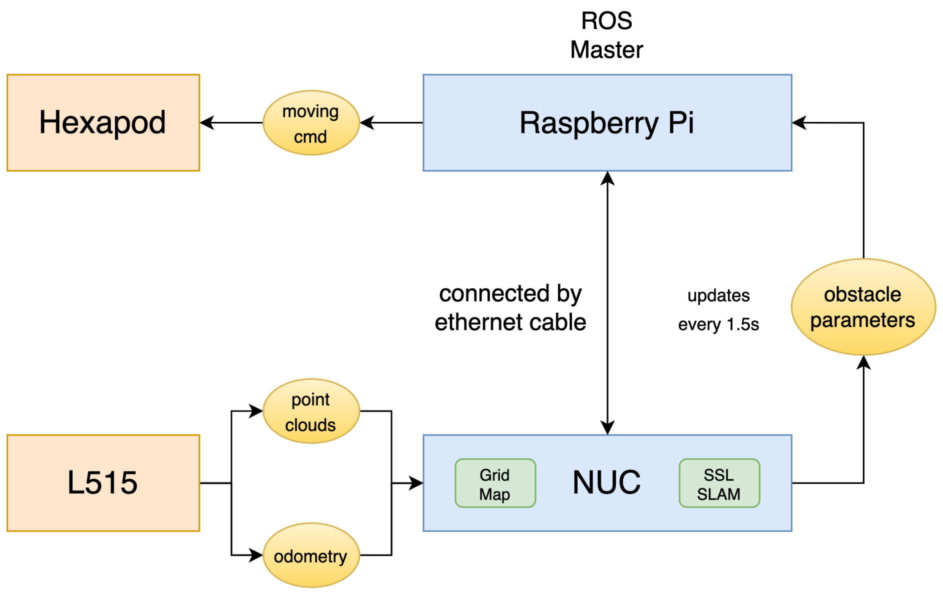


Figure 2.1: Communication diagram between devices

The Widow X does not include any perception sensors. In order for the perception system to work in 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. On the hexapod, we use a single L515 camera to understand surroundings in a near-dark or low-light environment, where a traditional stereo camera is incapable of performing 3D depth measurement. Once we know the depth map of the surroundings, we plan trajectories accordingly to avoid obstacles or climb over joists.

One challenge is the location for mounting the L515 camera. There is no convenient space on the hexapod to tightly hold the L515 camera, but it is crucial to place the L515 high from the ground to see things far away, so we 3D printed a rig to mount it. The complete hardware setup is illustrated in Figure 2.2.

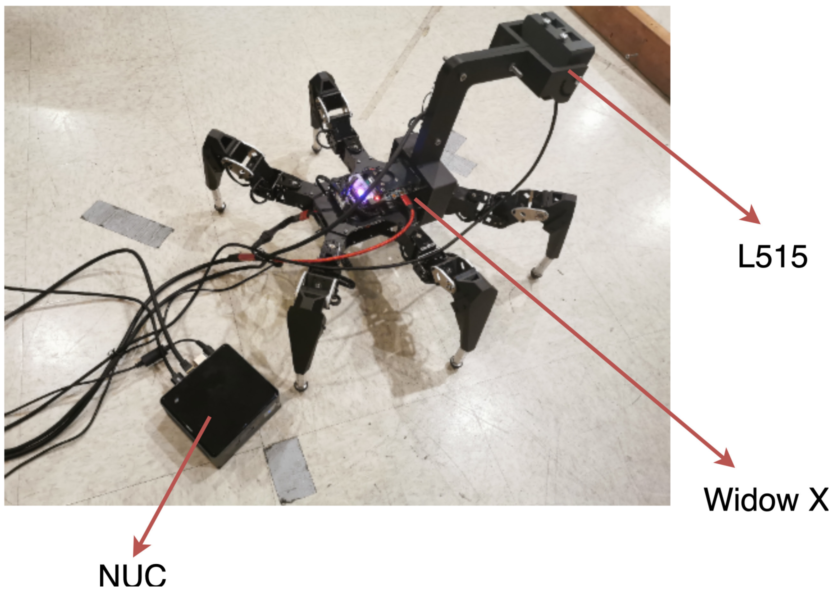


Figure 2.2: Hexapod, NUC, and L515 Depth Camera

Since we are mostly interested in ground obstacles near the hexapod, it is useful to tilt the depth camera downwards rather than having its optical axis parallel to the ground. To achieve that, we designed a simple nob shown in Figure 2.3 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 degrees L515 is completely looking down and at 90 degrees L515 is looking straight ahead.

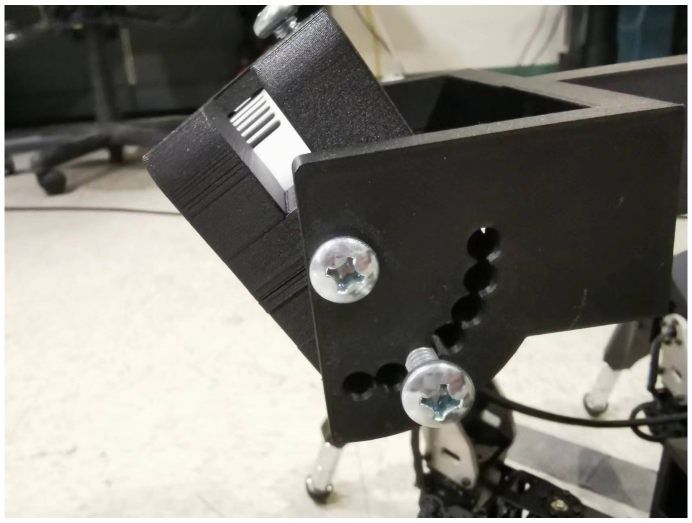


Figure 2.3: Pitch Angle Adjustable Nob for L515 Depth camera

2.2 Hexapod Vision System

Our vision system is a three-stage pipeline shown in Figure 2.4. SLAM and Grid-Map are the two core components in the hexapod vision system.

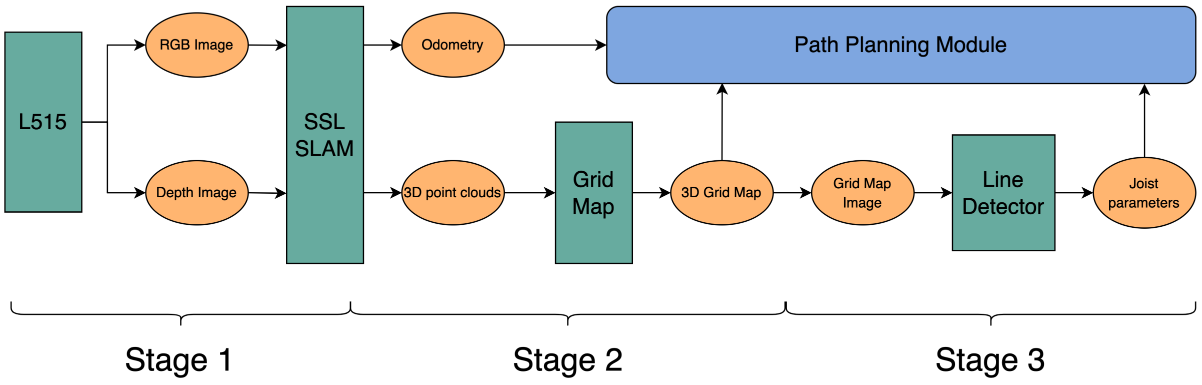


Figure 2.4: Three-Stage Vision Pipeline

2.2.1 SSL-SLAM and Grid-Map

SSL-SLAM [15] is a visual SLAM algorithm: it 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 serves 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, 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, where the hexapod is at the bottom middle, as shown in Figure 2.5. 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 0.01, meaning each cell is a square of 0.01m by 0.01m. 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.

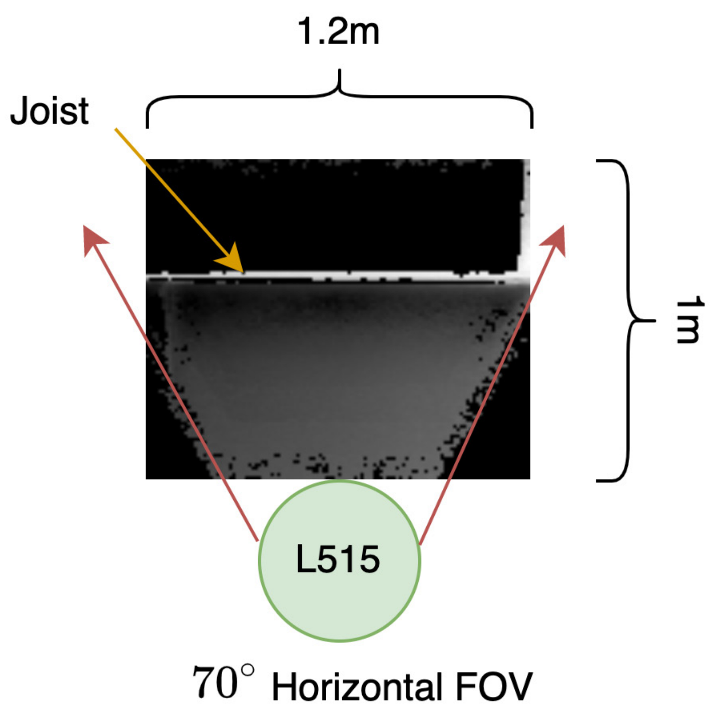
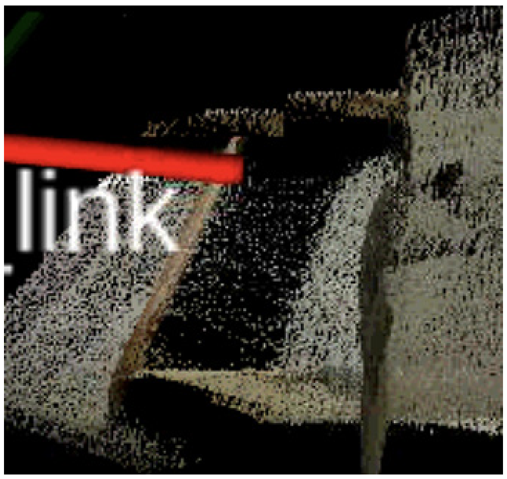


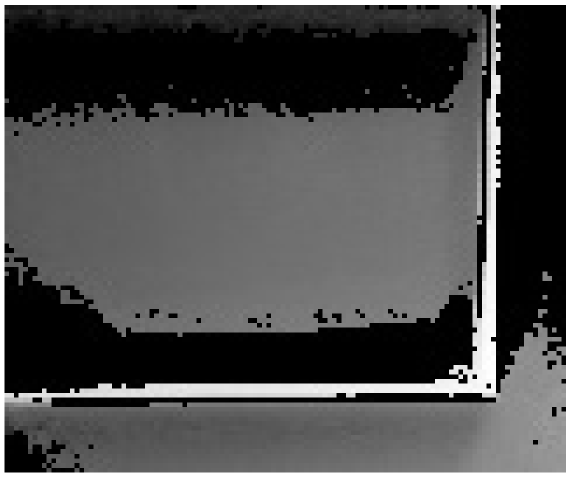
Figure 2.5: L515 camera is placed at bottom middle of the local grid map

Similar to all SLAM algorithms, the SSL-SLAM is limited to the sensor field of view (FOV), environmental lighting, and feature points in the image. Because the SLAM algorithm is based on feature point detection and matching, shadows and reflections could make feature matching more difficult from the robotics perspective. Furthermore, obstructed regions behind objects will not have assigned depth value due to occlusion. As a result, in the process of converting the pointcloud into grid map, these regions with undefined height values are assigned not a value (NAN). Overall, the NaN regions are extremely problematic. Figure 2.6 shows a pointcloud with the region behind a joist completely occluded, resulting in a NaN region in the grid map. Figure 2.7 (a) shows the reflections in front of a joist in the RGB image, which result in points below the ground plan with negative height in the point cloud shown in Figure 2.7 (b).

These NaN regions need to be handled with caution. We cannot simply assume all NaN regions are the floor and assign 0 height to them in the grid map. It might be shadows and reflections of the joist that the hexapod should avoid. One way to handle NaN would be to interpolate nearby not NaN value. This can be done by the Fast Marching Inpaint method introduced by Telea et al. [13], which uses a binary mask of spots to be filled in values in an input image. In our case, the mask is a zero matrix with the same size as the grid map. All the locations of the NaN region are marked as 1 on the mask matrix, and this mask matrix

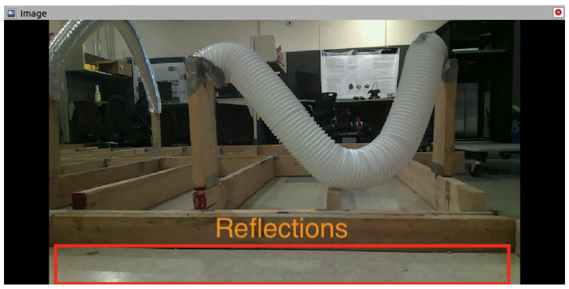


(a) The region behind joist has no or very few points in the pointcloud

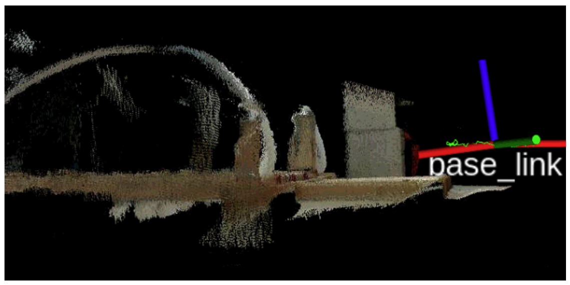


(b) The black region in a 2D grid map corresponds to the NaN value

Figure 2.6: NaN region behind a joist. (a) pointcloud indicating shadow of the joist. On 2D grid map image (b), grey region corresponds to the floor and bright white region corresponds to the joist.



(a) RGB image from L515 depth camera



(b) Side-view of the pointcloud visualization

Figure 2.7: (a) Reflections in front of the joist and have negative height in the pointcloud (b)

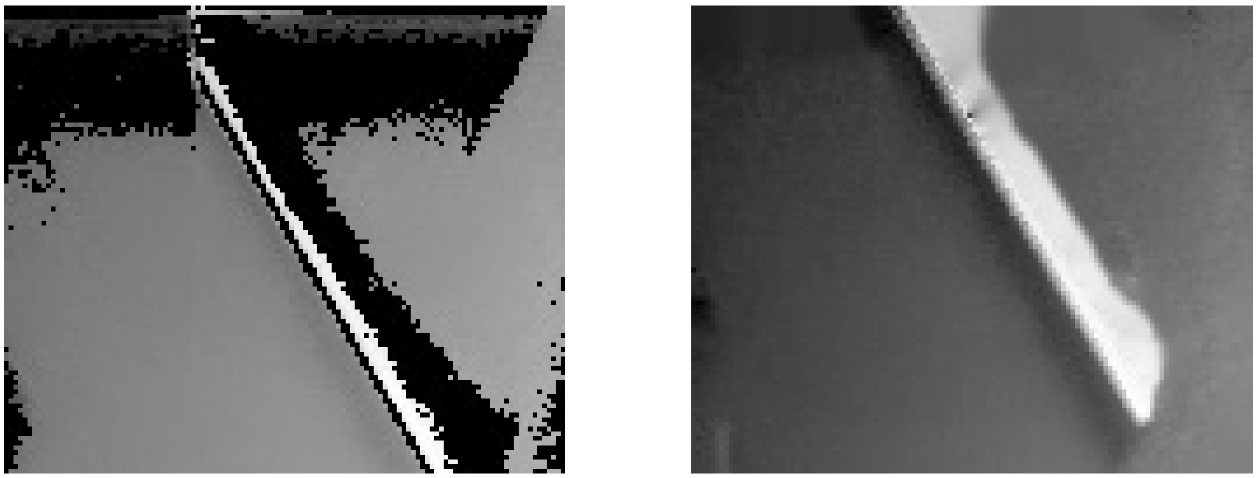

(a) 2D grid map image (b) Inpainted 2D grid Map

Figure 2.8: (a) Before and (b) after inpainting to remove NaN value.

with a 2D grid map is fed into the inpainting algorithm. Figure 2.8 shows the 2D grid map before and after inpainting.

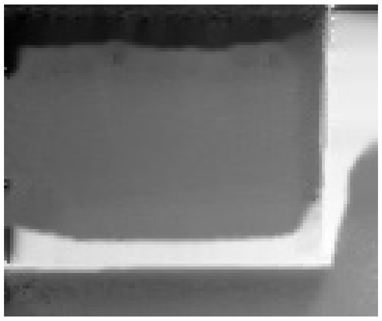
2.2.2 Line Detection

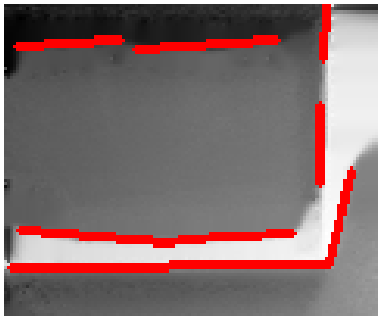
In the two stages described above, we have created a grid map for the surroundings, but we do not know where the obstacles are. In fact, the joists are well-presented in the 3D pointcloud and grid map, but the lack of semantic information is a challenge for our path planning and climbing sequence. Moreover, previous work of 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. Therefore, we must develop other methods to detect joists. This is the third stage of our vision pipeline.

Our approach to detection joist relies on traditional 2D line detection. A Grid map can be easily converted to a 2D image, which is a top-down view of the world with each cell filled with the height value. The line detection idea is based on the assumption that if we could detect the joist from a top-down 2D view, we could transfer them back to the real 3D world coordinates and deliver joist parametric data such as height, orientation, and distance information to the hexapod. More specifically, our joist detection algorithm follows these three steps:

1. Grid map is converted to a top-down 2D image. ED-line detector 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 the lines is estimated. Since we are interested in lines on the actual joists, detected lines that


(a) 2D inpainted grid map image of a joist


(b) All line detection results in red lines

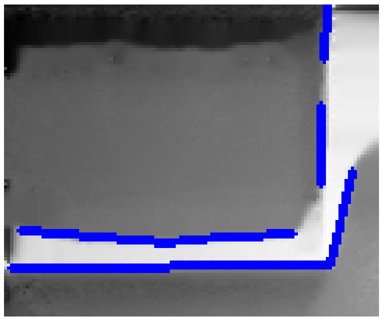

(c) Filtered line detection results in blue lines

Figure 2.9: 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 dark grey colors respectively. (a) The grid map image after inpainting. (b) The ED-Line detector incorrectly detects two line segments on the top; these two line segments are filtered out based on underlying grid map height value, as shown in (c)

have average height less than 5cm are assumed to be not joists and filtered out, so that these lines are not picked as the closest line in Step 3.

3. Joist distance is measured by the vertical pixel distance from the bottom of grid map image to the center of the line. We calculate the joist distance for each line and the line with the smallest distance is chosen and returned.

In **Step 1**, not all detected lines are on joists, as illustrated in Figure 2.9. Some of these lines are formed between the floor in color grey and inpainted NaN in color black. These lines are filtered out based on average height, in Step 2.

In **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

$$\hat{h} = \frac{1}{(x_2 - x_1 + 4)(y_2 - y_1 + 4)} \times \sum_{i=x_1-2}^{x_2+2} \sum_{j=y_1-2}^{y_2+2} g(i, j) \tag{2.1}$$

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 2.1, we traverse over a line and compute the 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. Two other common line detectors, Hough transform (Hough-line) [7] and Line Segment Detector (LSD) [6], however, did not work as well as ED-line. We speculate that this is due to our data which is unorganized with little structure. Figure 2.9 shows that the ED-Line detection results on the inpainted grid map image in red and the results after height filtering in blue,.

These three stages summarize our vision system. Once the robot detects joists, it has to climb over or avoid joists.

2.3 Hexapod Joist Climbing

Foothold is the point of contact on the terrain where the leg can firmly support the body. Foothold selection of is critical for holding the body stance and maintaining balance. The foothold selection process is commonly done on an elevation map where the algorithm loops through all possible small grid cells and determines the best foothold for the next movement. We have developed a heuristic algorithm to estimate the distance, height, and orientation of the joist and it is used to prevent collisions with the joist. In our case, collision prevention is done through a scan of the 3D global map and adjusting the hexapod's pose accordingly. During the climbing sequence, the hexapod leaves enough space between its legs and joist so that the legs do not hit on the joist.

By default, the hexapod follows the 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 also 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. Therefore, to plan hexapod motion, we need to take joist height into account. Once we estimate the height of the joist 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.

It is worth mentioning the coordinate system used in our joist climbing algorithm. Our vision and joist climbing algorithms use slightly different coordinate systems due to the nature of the vision sensor and hexapod body.

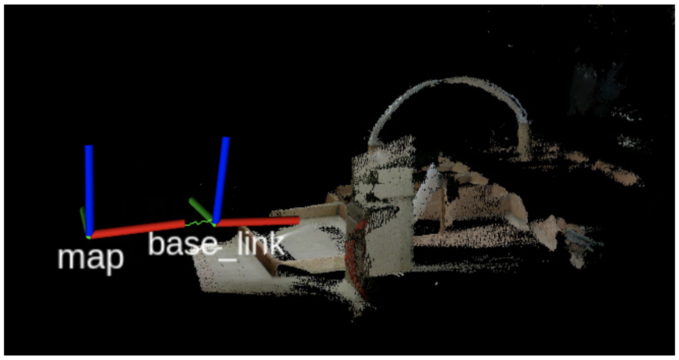


Figure 2.10: Hexapod Vision Coordinate System

In Figure 2.10, "map" is the center of the global coordinate system, as defined by the starting position and orientation of L515 at time 0. On the other hand, "base_link" is the moving L515 coordinate system after time 0. Both "map" and "base_link" follow the right-handed coordinate system where red is the positive X-axis (+X), green is the positive Y-axis (+Y), and blue is the positive Z-axis (+Z). L515 is pitched at an angle θ around +Y axis. At the up-right position $\theta = 90°$ and at the facing-down position $\theta = 0°$.

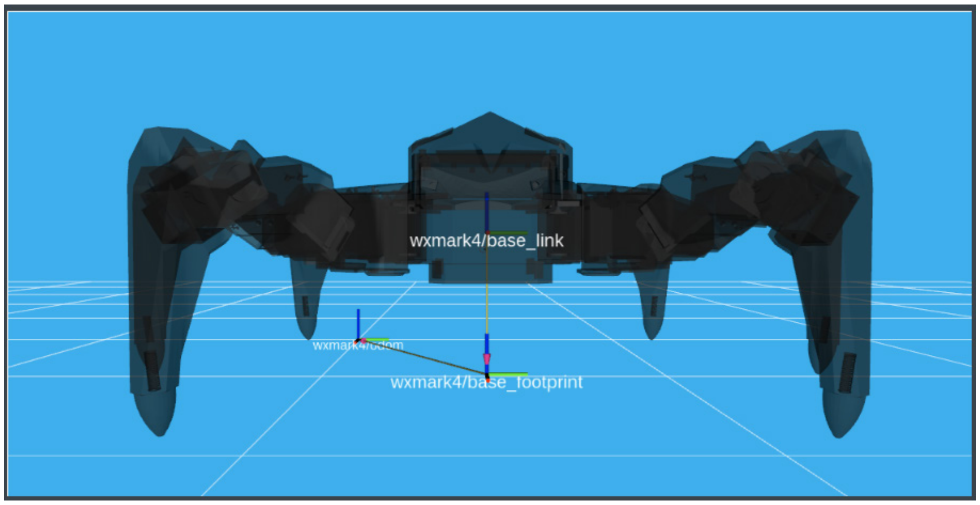
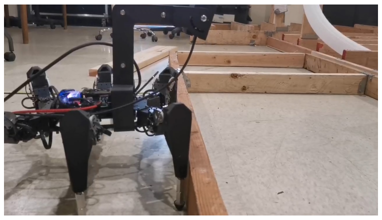
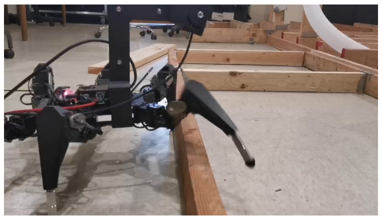
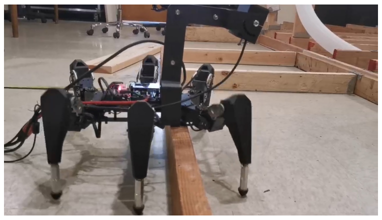


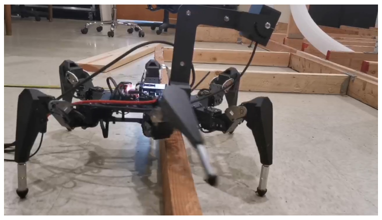
Figure 2.11: Hexapod Robot Coordinate System. Source: Interbotix

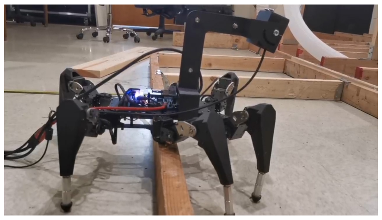
The Hexapod Robot coordinate shown in Figure 2.11 is defined slightly differently, but the axis orientation is the same. The "robot/base_link" is at the center of the hexapod body, the "robot/base_footprint" is at the projection center of the robot body on the ground, and the "robot/odom" is the center of the global coordinate system, as defined by the starting position and orientation of the hexapod at time 0. The transformation between "robot/base_link" and "robot/base_footprint" is a pure translation on -Z, but this transformation relationship

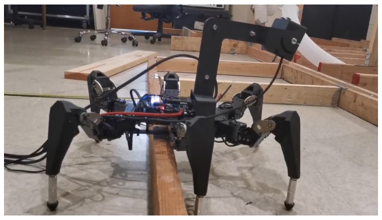

(a) Pose 1: stop near joist, calculate trajectory

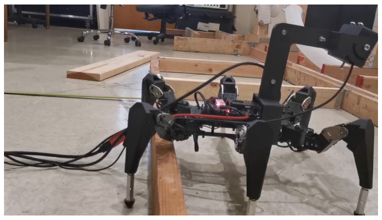

(b) Pose 2: move front legs up in the air

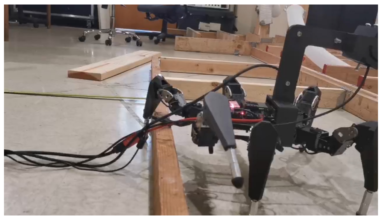

(c) Pose 3: reset body position


(d) Pose 4: move middle legs up in the air


(e) Pose 5: lean body forwards


(f) Pose 6: reset body position


(g) Pose 7: walk 1 gait cycle and lean body forward


(h) Pose 8: move back legs up in the air

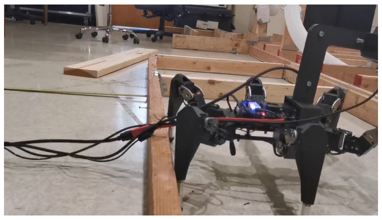

(i) Pose 9: reset body and all legs position

Figure 2.12: Illustration of Hexapod Climbing Sequence

could change when the hexapod raises its body. On the other hand, the transformation between either "robot/base_link" and "vision/base_link" or "robot/odom" and "vision/map" is always fixed with 20cm translation in +Z and 25cm translation in +X.

Figure 2.12 illustrates the hexapod climbing sequence over a joist. Once the hexapod moves 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.

2.4 ROS Message Synchronization and Delay

There are 6 ROS message topics running on the ROS server. They form the sequential message flow of the hexapod in the time domain and execute each callback function when necessary.

- **/hexapod/joist_cmd**: receives commands to move hexapod around
- **/camera/color/image_raw** and **/camera/depth/color/points**: output of L515
- **/map**: 3D global pointcloud map, output of SSL-SLAM
- **/grid_map_from_pointcloud**: grid map from **/map** pointcloud
- **/odom**: visual odometry, output of SSL-SLAM
- **/joist**: joist parametric data, namely distance, height, and orientation of the closest joist

During the experiments, we realized that there is a 0.2s processing time in processing raw L515 data and converting it to pointcloud data, 0.15s processing time in joist detection and calculation, and 1.5-2s processing time for grid map generation. The grid map processing time depends on the density of pointcloud. The processing time delay issue has impacts on two things:

1. Grid map and odometry data time synchronization. In order to localize the hexapod and plan new paths, we need both grid map information and odometry data. However, **/grid_map_from_pointcloud** and **/odom** are running at two different rates. In fact, it is computationally intensive to construct a grid map and as such **/grid_map_from_pointcloud** runs between 0.5 to 0.6HZ, while **/odom** runs at a much faster frequency of 5Hz. Therefore, we first synchronize **/odom** message and **/grid_map_from_pointcloud** message with ROS message filter in the main callback function to obtain the most up-to-date data of odometry and grid map. After synchronization, the difference between the two topic's message timestamps is less than 0.2s.

2. Joist distance estimation. The hexapod is moving forward at a speed v. Assume at time (t_0) the L515 captures joist image, at $(t_0 + 0.2)$ SSL-SLAM sends out pointcloud data, at $(t_0+0.2+t_1)$ grid map finishes processing, at $(t_0+0.2+t_1+0.15)$ line detection returns detected line results and logs joist detection result on screen, which by now corresponds to the joist distance estimation from a few seconds ago, during which hexapod has moved forward $v(t0 + 0.2 + t1 + 0.15)$. To address this, we use odometry data to compensate the joist detection distance at time $(t_0 + 0.2 + t_1 + 0.15)$, such that if the odometry measurement difference between (t_0) and $(t_0 + 0.2 + t_1 + 0.15)$ is O and the joist detection distance at time (t_0) is J, then the actual joist distance we use at time $(t_0 + 0.2 + t_1 + 0.15)$ is $J - O$.

2.5 Experiments and Results

To understand the choice of L515 pitch angle and to verify our vision and climbing algorithm, 4 experiments are performed. Experiment descriptions are summarized in the Table 2.1.

Table 2.1: Experiments Description

Experiment	Description
1	Understand how the L515 pitch angle affects horizontal and vertical views
2	Determine correlation between pitch angle, NaN region, and joist width
3	Measure accuracy of the line detector
4	Test integrated vision and climbing algorithm in 7 settings

2.5.1 Experiment 1: L515 Camera Angle

The first experiment is to understand how the L515 pitch angle affects horizontal and vertical views of the robot. Considering the pitch angle would also affect the SSL-SLAM quality, we ran the SSL-SLAM algorithm with different pitch angles and observed the quality of pointcloud. We found that at 45° pitch, about 80% of the area in RGB image from L515 is on the ground with planar depth and very few feature points. Unsurprisingly, the SSL-SLAM algorithm fails to reconstruct the pointcloud after only 5 seconds. Therefore, in the following experiments, we will only consider the pitch angles of 60°, 75°, and 90°.

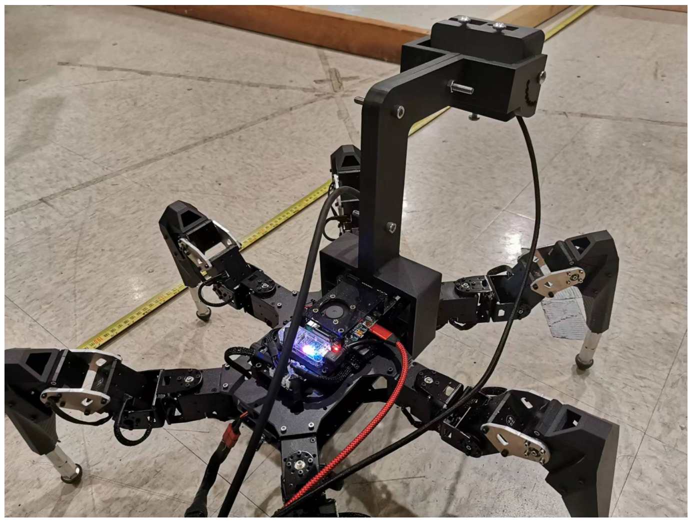
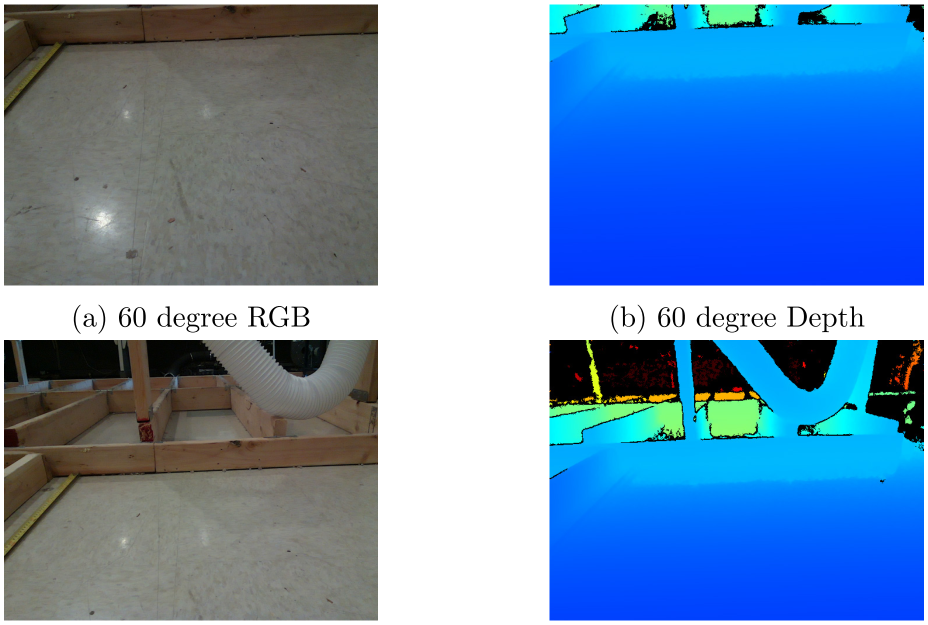


Figure 2.13: Experiment to evaluate the vertical and horizontal FOV of image with different pitch angle configurations


(a) 60 degree RGB (b) 60 degree Depth

(c) 75 degree RGB (d) 75 degree Depth

Figure 2.14: Depth and RGB image at 1m distance away from joists

Next, we evaluate the FOV on different pitch angles. The technical spec information of L515 indicates that it has a vertical FOV of 55° and a horizontal FOV of 70°. In our experiments, a tape measure is placed next to the hexapod so that the horizontal distance can be measured as seen in Figure 2.13. L515 is fixed on the mount 0.35m above ground. The pitch angles adjustable nob is inserted at 90, 75, and 60 degrees respectively. Then, the L515 camera is turned on and the observable horizontal distance from the RGB image is recorded as shown in Table 2.2.

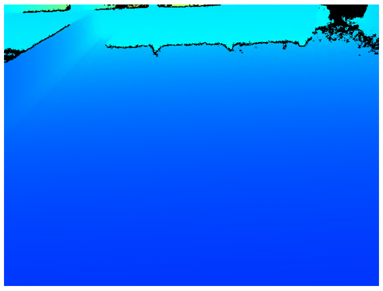
Table 2.2: Vertical FOV Min/Max Ground Distance

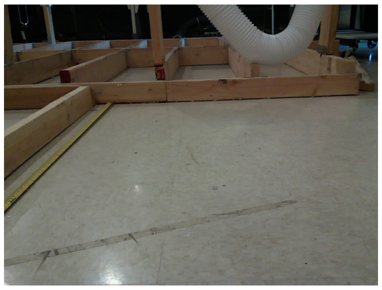
Pitch angles	Minimum distance (m)	Maximum distance (m)
90°	0.7	>4*
75°	0.4	>4*
60°	0.25	1.5

*: The max distance we can measure is 4m due to space constraints.

According to Table 2.2, at 90° and 75° the L515 has a relatively large horizontal view; at 60° the L515 is able to see the ground as close as 0.25m, but cannot see ground as far as at 90° and 75°. We need more experiment data to choose between 75° and 60° pitch angle.

(a) 60 degree RGB


(b) 60 degree Depth


(c) 75 degree RGB

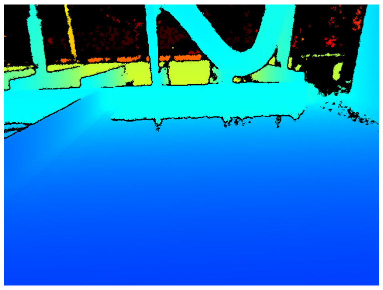

(d) 75 degree Depth

Figure 2.15: Depth and RGB image at 1.5m distance away from joists

To do so, we recorded the depth and RGB images for 75° and 60° pitch angles at 1m and 1.5m, as we plan to use these two distances in later experiments. Figures 2.14 and 2.15 show that at 60° pitch there is very little depth distinction and about 80% of image is ground. This might have an impact on the quality of SLAM map and could result in the SLAM algorithm failing occasionally due to limited feature points. From this perspective, 75° pitch angle might be preferable. However, at 60°, we might have a smaller NaN region and hence more accurate joist width data after inpainting. Therefore, before we deciding on the pitch angle, we need experiment 2 to determine whether decreasing the pitch angle improves NaN region handling.

2.5.2 Experiment 2: Joist Width Estimation

The next experiment is to determine the correlation between the size of the NaN region and the L515 pitch angle, the effectiveness of the grid map inpainted method to fill in the NaN value, and the joist width estimation.

In this experiment, the hexapod starts 1m away from a joist with 0.1m height and 0.04m width, walks towards it, and stops when the front legs hit the joist. While moving, the grid map is being constantly updated and each grid map frame is saved. When the hexapod stops, we measure the width of the NaN region and joist on the grid map image. Since each grid map image pixel corresponds to a $1cm \times 1cm$ square, the width measurement is done by manually counting the number of pixel sizes. As shown in Table 2.3, we observe that the primary NaN region is behind the joist and the joist width is overestimated after inpainting.

Table 2.3: Pitch Angle and Inpainting Effect on Grid Map

θ	W_{NaN} (m)	$\widetilde{w}$ (m)	$\widetilde{w}'$ (m)	W_{joist} (m)
90°	0.2	0.02	0.13	0.04
75°	0.15	0.02	0.1	0.04
60°	0.13	0.03	0.08	0.04

θ: pitch angle
W_{NaN}: width of NaN region
$\widetilde{w}$: estimated joist width by assuming NaN is zero height
$\widetilde{w}'$: estimated joist width with inpainting
W_{joist}: actual joist width

In Table 2.3, $\widetilde{w}$ corresponds to the estimated joist width, assuming NaN behind the joist are assigned to zero height in the grid map. Even though in the example the assumption is valid, in general it might not be true, since the occluded margin could have any height. This is the main motivation for considering inpainting the NaN values.

As seen in Table 2.3, before inpainting $\widetilde{w}$ is underestimated about 0.01m. Though not linear, W_{NaN} decreases as pitch angle decreases. The decrease in the size of the NaN region width is 0.05m from 90° to 75° pitch angles, while the decrease is only 0.02m from 75° to 60°. On the other hand, 75° and 60° pitch angles lower $\widetilde{w}'$ by 0.03m and 0.02m respectively. In summary, we observe that the NaN region width and overestimated joist width improves mostly from 90° to 75°, and marginally from 75° and 60°. Based on these results above, we determine that 75° is the best pitch angle configuration in consideration of horizontal distance FOV, depth map quality, SLAM reliability, and decrease in the size of NaN region and joist width estimation. Note that all results in Table 2.3 are computed manually and not algorithmically.

2.5.3 Experiment 3: Line Detection Accuracy

In the third 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 leg hits the joist. Figure 2.16 shows the experimental setup.

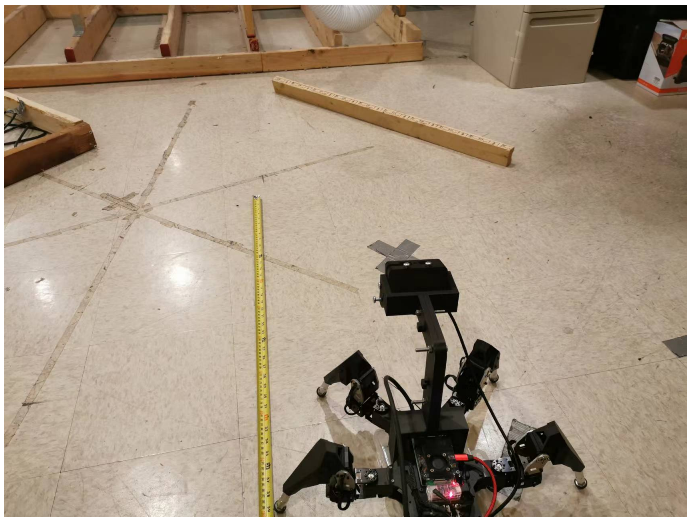


Figure 2.16: Joist are oriented at 1m in front of the hexapod.

The joist orientation angle, ψ, is measured from the perpendicular line to the direction of the motion of the hexapod, such that when the joist is parallel to the direction of the motion of the hexapod, $\psi = 90°$; when the joist is perpendicular to the direction of the motion of the hexapod, $\psi = 0°$. Joist orientation is estimated by trigonometry formula:

$$\psi = arctan(\frac{abs(y_{start} - y_{end})}{abs(x_{start} - x_{end})}) \tag{2.2}$$

where (x_{start}, y_{start}) and (x_{end}, y_{end}) are two vertex coordinates defining the line.

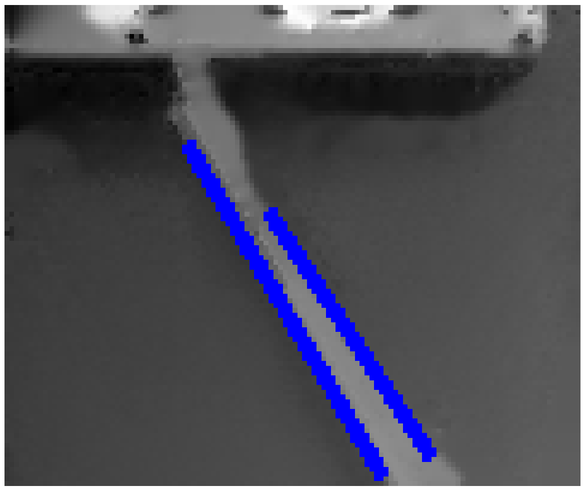


Figure 2.17: Line detection result on an oriented joist.

Table 2.4: Estimated Joist Heights and Orientation Angle

ψ (°)	ψ'_{front} (°)	ψ'_{back} (°)	h' (m)	H (m)
60	55	59	0.083	0.1
45	42	47	0.085	0.1
30	33	31	0.081	0.1
60	52	58	0.187	0.18
45	41	47	0.19	0.18
30	31	34	0.195	0.18

where ψ: ground truth joist orientation yaw angle
ψ'_{front}: estimated yaw angle on the front of the joist
ψ'_{back}: estimated yaw angle on the back of the joist
h': estimated joist height
H: ground truth joist height

The result of this experiment is summarized in the Table 2.4 and an example image is shown as Figure 2.17. We can see that detected line segments estimates joist orientation relatively accurately, with a tolerance of ± 3°. Moreover, the estimated height is close to the actual height with about a 0.01m difference, after inpainting to fill in the NaN value.

2.5.4 Experiment 4: Climbing over Joists

For Experiment 4, 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 performed, as shown in Table 2.5.

Table 2.5: Experiment 4 Setup

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

In Test 1, the hexapod starts at 1m away from the joist. The joist in this test is measured at 10cm in height, 4cm in width, and 83cm in length. Joist is placed perpendicular to the direction of the motion of the robot.

Test 1: Hexapod 1m away from 10cm joist

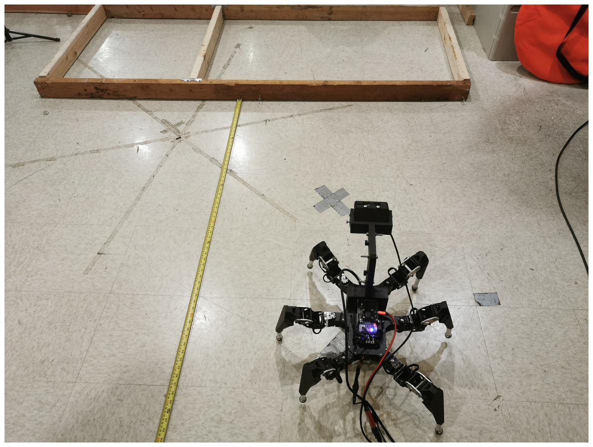


Figure 2.18: Test 1: hexapod starts at 1m from 10cm joist. The resulting grid map images and line detections for this experiment are shown in Appendix A. Demo video.

The hexapod walks from 1m starting location and detects a joist at 0.65m. Hexapod keeps moving forward until the joist distance falls within the 0.55m threshold. Next, the hexapod moves forward with tripod gait and odometry data to get as close to the joist as possible, where the climbing sequence starts. During the climbing sequence, the hexapod first takes two front legs over the joist, plans again based on the current pose, then takes two middle legs over the joist, plans new trajectories, and finally moves the back legs across the joist.

Test 2: Hexapod 1.5m away from 10cm joist

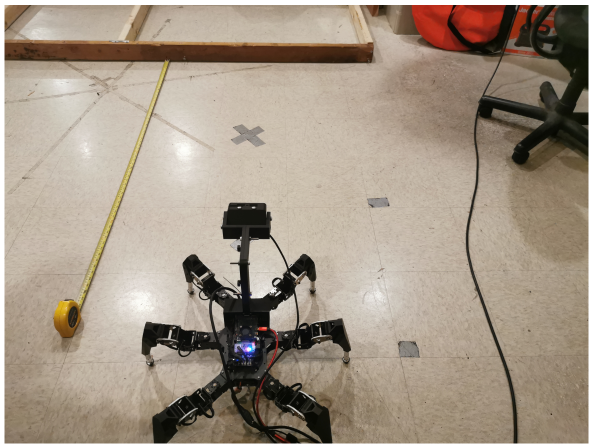


Figure 2.19: Test 2: hexapod starts at 1.5m from 10cm joist. The resulting grid map images and line detections for this experiment are shown in Appendix B. Demo video.

Similar setup as Test 1, except that the hexapod starts further away. In Test 2, the hexapod is placed 1.5m away from the horizontal joist. The hexapod detects the joist at 0.95m, 0.56m, and 0.35m; once the distance to the closest joist falls under the 0.55m threshold, it steps out of the walking forward loop, approaches the joist as close as possible, and climbs over the joist.

Test 3: Hexapod 1m away from 18cm joist

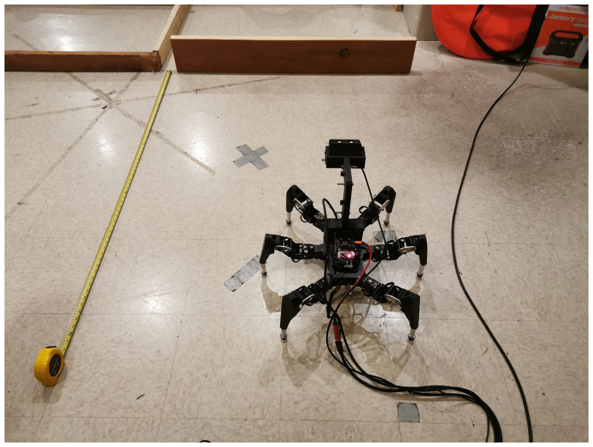


Figure 2.20: Test 3: hexapod starts at 1m from 20cm joist. The resulting grid map images and line detections for this experiment are shown in Appendix C. Demo video.

Similar setup as Test 2, except that the 10cm joist is replaced with a 18cm joist. The hexapod starts at 1m away from the joist. The joist is placed perpendicular to the direction of the motion of hexapod. The hexapod detects the joist at 0.68m and immediately realizes the joist height, estimated 0.15m, is too high to climb. It then moves sideways.

Test 4: Hexapod 1m away from $45°$ oriented 10cm joist

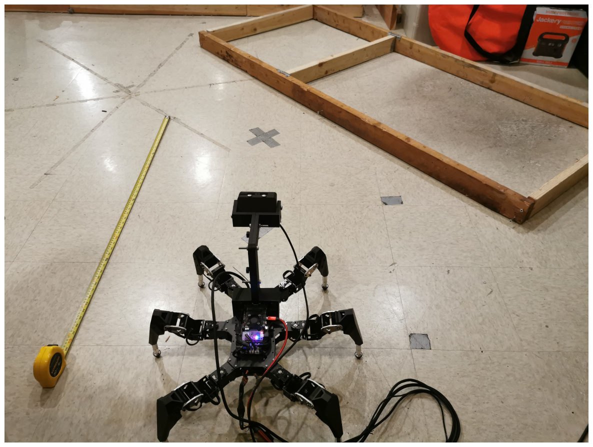


Figure 2.21: Test 4: hexapod starts at 1m from 45° oriented joist. The resulting grid map images and line detections for this experiment are shown in Appendix D. Demo video.

Test 4 has a similar setup as Test 1, except that the 10cm joist is no longer perpendicular to the direction of the motion of hexapod. The hexapod starts at 1m away from the bottom of the joist. The joist is placed at 45° perpendicular to the direction of the motion of hexapod. The hexapod detects the oriented joist at 0.5m and estimates the orientation angle, θ, of 41.5°. It moves forward to get as close as possible to the joist, where it rotates clockwise 41.5° and orients itself perpendicular to the joist. It successfully climbs over the joist.

Test 5: Hexapod 1m away from 60° oriented 10cm joist

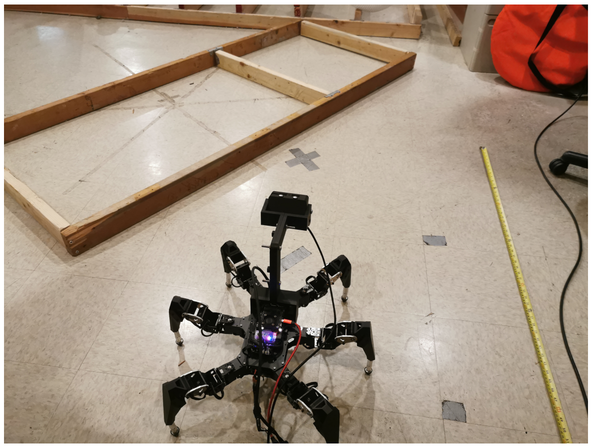


Figure 2.22: Test 5: hexapod starts at 1m from 60° oriented joist. The resulting grid map images and line detections for this experiment are shown in Appendix E. Demo video.

Test 5 has a similar setup as Test 1, except that the 10cm joist is no longer perpendicular to the direction of the motion of the hexapod. The hexapod starts at 1m away from the joist. The joist is placed at 60° perpendicular to the direction of the motion of the hexapod. The hexapod detects the oriented joist at 0.59m and estimated orientation angle, θ, of 41.5°. Since the joist distance, 0.59m, is larger than the threshold of 0.55m, it keeps moving forward until the second estimation came in. The second estimation is at 0.53m distance and 58.9° of orientation. It moves forward to get as close to the joist as possible, where it rotates counter-clockwise 58.9° degree and orients itself perpendicular to the joist. It successfully climbs over the joist.

Test 6: Hexapod 1m away from 75° oriented 10cm joist

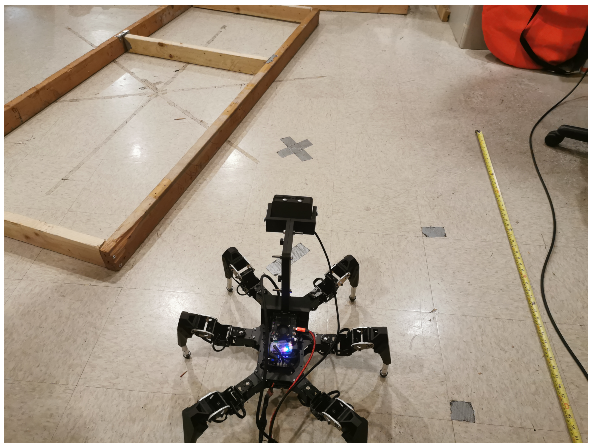


Figure 2.23: Test 6: hexapod starts at 1m from 75° oriented joist. The resulting grid map images and line detections for this experiment are shown in Appendix F. Demo video.

Test 6 has a similar setup as Test 1, except that the 10cm joist is no longer perpendicular to the direction of the motion of the hexapod. The hexapod starts at 1m away from the joist. The joist is placed at 75° perpendicular to the direction of the motion of the hexapod. The hexapod detects the oriented joist at 0.66m and estimates the orientation angle $\theta = 73.9°$. Since the joist distance 0.66m is larger than the threshold of 0.55m, it keeps moving forward until the second estimation comes in. The second estimate is at 0.46m distance and 72.5° of orientation. It moves forward to get as close to the joist as possible, where it rotates counter-clockwise 72.5° and orients itself perpendicular to the joist. It successfully climbs over the joist.

Test 7: Hexapod 2.7m away from 18cm joist

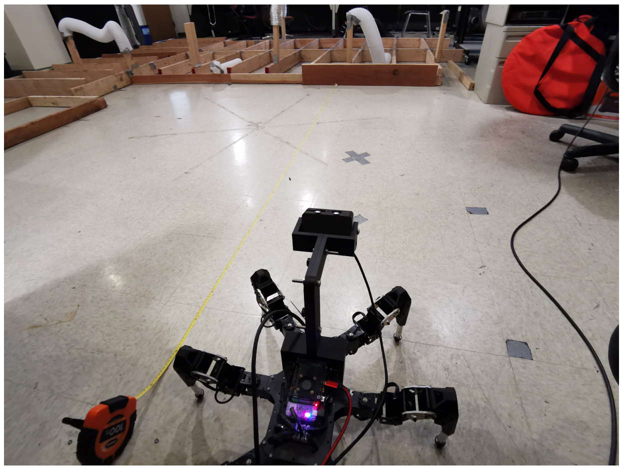


Figure 2.24: Test 7: hexapod starts at 2.7m from 18cm joist. The resulting grid map images and line detections for this experiment are shown in Appendix G. Demo video.

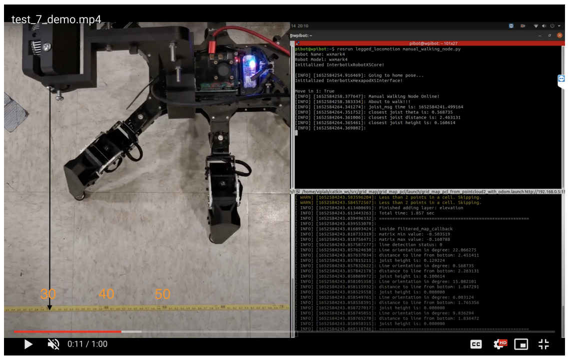
In test 7, we measure the accuracy of our joist distance estimation. The input pointcloud size is changed from $1m \times 1.2m \times 0.4m$ to $3m \times 1.2m \times 0.4m$ to ensure that joist is always in the view. The hexapod starts at 2.7m and moves forward for 1.8m. As soon as a new joist distance estimation shows up, we immediately record the ground truth distance using a tape measure as seen in Figure 2.25. After moving for 1.8m, the hexapod moves to the left, signaling it detects a high joist of 18cm. Throughout the experiment, we collect 6 measurements. The joist distance error is summarized in Table 2.6:

Measurement	Estimated distance (m)	Ground truth distance (m)	Error (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

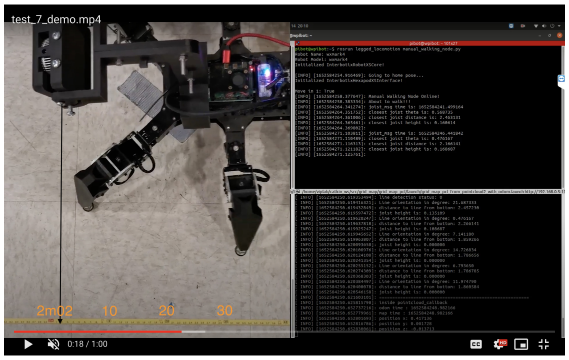
Table 2.6: The snapshots of video indicating the measurements shown in Figure 2.25

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

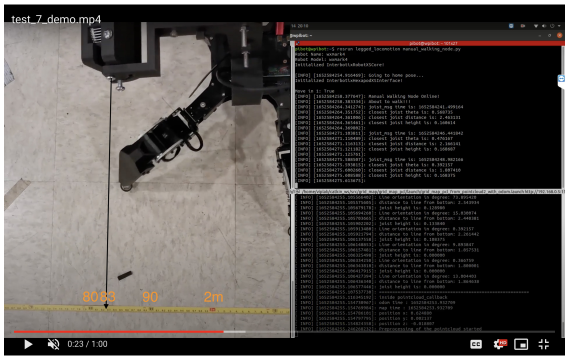
1. The grid map processing delay: the grid map processing time highly depends on the density of the input pointcloud, as mentioned in Section 2.4.4. Although we use syn-



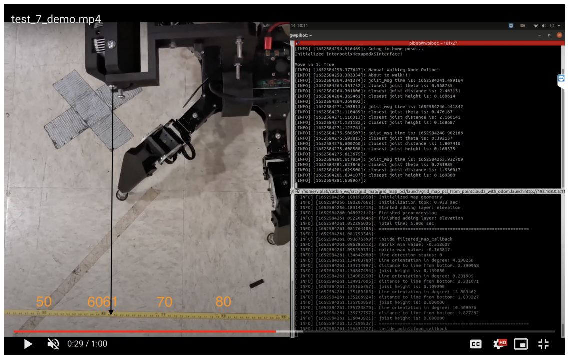
(a) Measurement 1



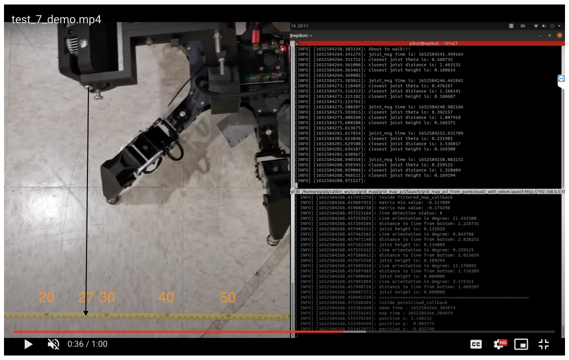
(b) Measurement 2



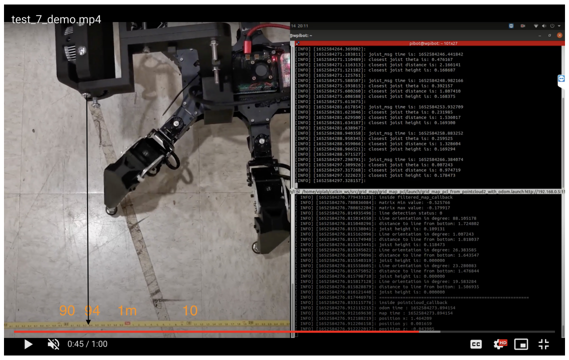
(c) Measurement 3



(d) Measurement 4



(e) Measurement 5



(f) Measurement 6

Figure 2.25: Six snapshots from demo video where measurements are taken place

chronized odometry data to compensate for this processing delay so that the joist distance between 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 accurate line detection. However, when the hexapod is far away, the pointcloud might be sparse. Consequently, the line detection might be slightly inaccurate.

3. The delay inside ROS logging system between the time joist distance estimation is sent and the time it is actually logged on the screen might underestimated the distance.

Chapter 3

Conclusions and Future Works

We analyze the problem of joist climbing for a hexapod robot. We use multiple wooden joists as the obstacles and test whether the hexapod could detect the joist and take proper actions. We examine the possibility to use one single L515 camera and an Intel NUC to detect joists on the floor and move the hexapod to the desired location. The experiments show that the hexapod is able to detect obstacles 2 meters ahead, plan trajectories accordingly, and successfully climb over obstacles or move sideways when encountering unclimbable obstacles.

From the experiments, we discover the limitations of our algorithms. Even though we show promising results for hexapod moving near joists, the environment is special and unique. The material of obstacles could severely impact our detection algorithm. Moreover, the delay from processing units is non-negligible.

Generally speaking, obstacle avoidance remains the hardest part of the autonomous robotics system. For self-driving cars on a crowded highway with more than hundreds of vehicles considered ”obstacles”, obstacle avoidance is crucial yet difficult. As discussed in Chapter 1, the traditional way of obstacle avoidance is detect-plan-act. Unsurprisingly, with observations from our experiment, detect-plan-act is not enough; every unit in the pipeline needs to wait for the previous stage to complete. Some recent research work has been 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]. Demo videos show that the robot is able to navigate through complex indoor environments with RGB video. However, this method is far from perfect. One typical issue in the outdoor environment is that objects could be extremely far away, making all current depth estimation algorithms fail.

On the other hand, we should still be optimistic that these problems could be addressed soon. With next-generation LiDARs and cameras, robots can see objects further away and more clearly. With powerful edge-computing devices such as NVIDIA Xavier and Intel NUC, more computations could be performed onboard. Thus, a single end-to-end network for vision and action is probably not out of reach.

Bibliography

[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] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010.

[7] P. V. C. Hough. Method and means for recognizing complex patterns, U.S. Patent 30696541962, Nov. 1969.

[8] 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.

[9] Oussama Khatib. *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, pages 396–404. Springer New York, New York, NY, 1990.

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

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

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

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

[14] I. Ulrich and J. Borenstein. Vfh+: reliable obstacle avoidance for fast mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 2, pages 1572–1577 vol.2, 1998.

[15] 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.

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

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

Appendix A

Joist Detection Figures For Test 1

The exact joist parameters in Test 1 is summarized in Table A.1. Figure A.1 shows the grid map image with NaN values shown in green for this test. The time span between frame 1 and frame 12 in Figure A.1 is about 43 seconds. The time between each successive frame is about 3.5 seconds. Figure A.2 shows the inpainted grid map corresponding to the frames in Figure A.1 and Figure A.3 shows line detection results on the inpainted frame of Figure A.2. As shown in Figure A.3, the front and the back of the closest joist is detected in all frames except for 3 - 7. In frames 8 - 12, there is an uneven region behind the test joist, resulting in degraded inpainting images and line detections. This region is presumably the result of inpainting on an array of the joists, circular objects, and the NaN region on the floor, as shown in Figure A.4.

Table A.1: Test 1 Setup

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
1	1	0°	10

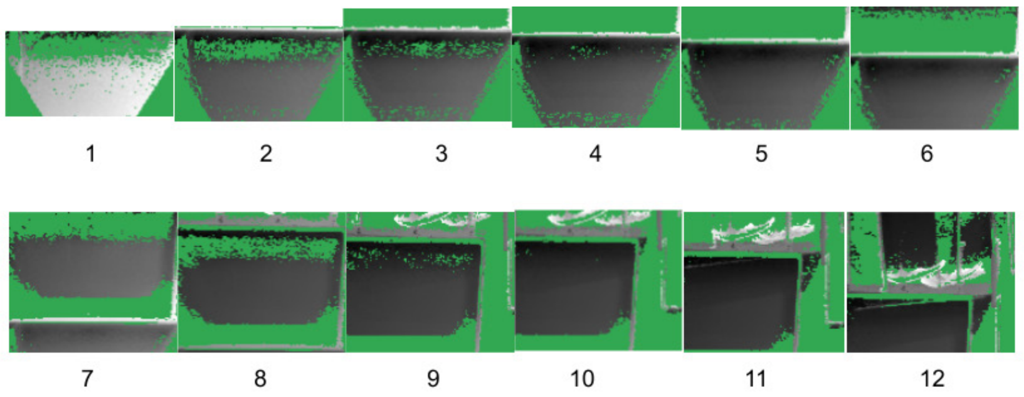


Figure A.1: Grid map images with all NaN value painted in green

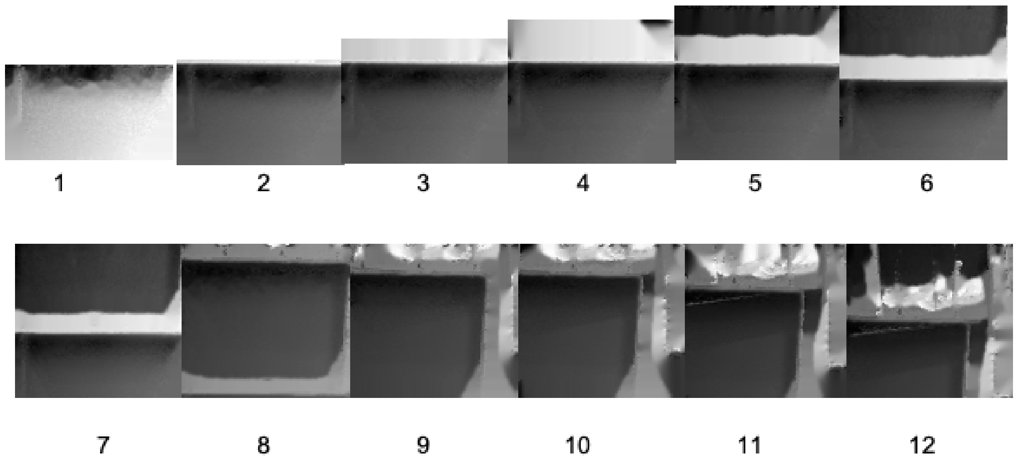


Figure A.2: Inpainted grid map images

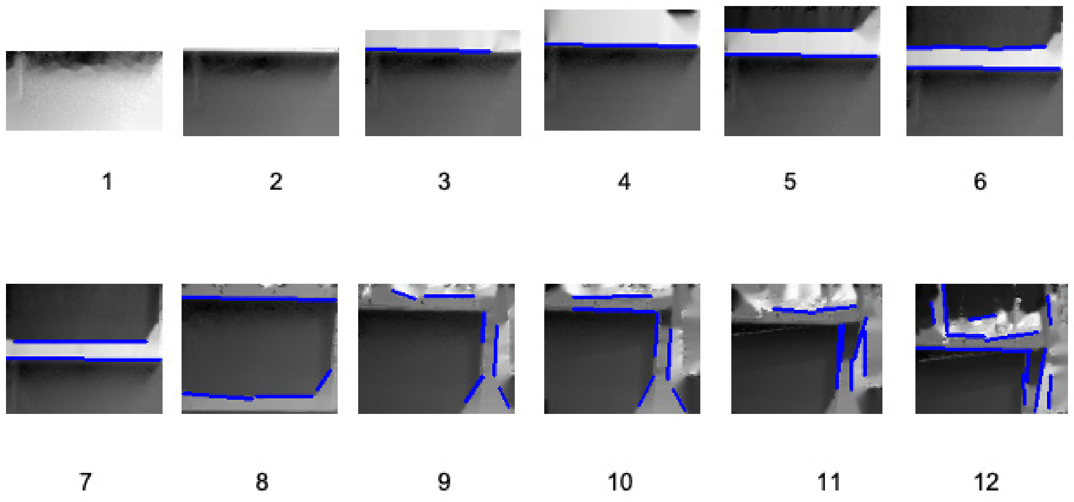


Figure A.3: Detected joist overlaid on inpainted grid map images

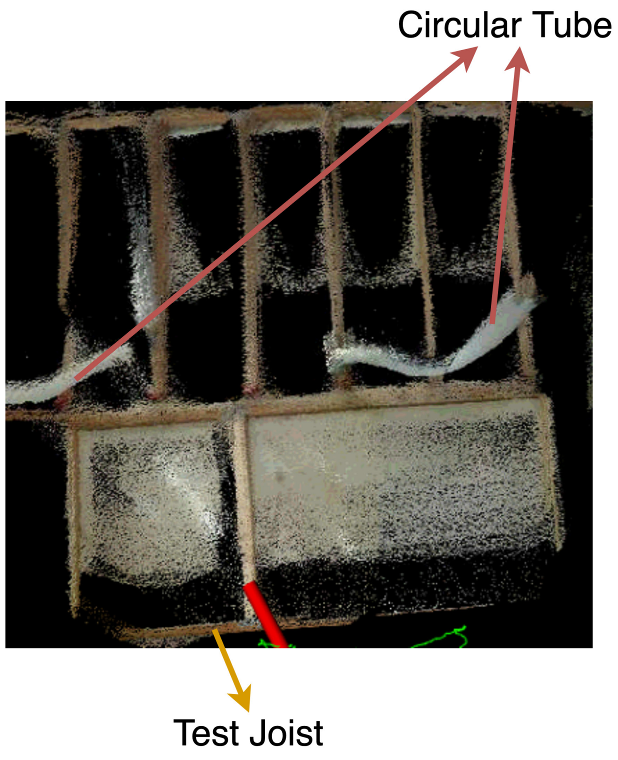


Figure A.4: Top down view of the region behind the test joist

Appendix B

Joist Detection Figures For Test 2

The exact joist parameters in Test 2 is summarized in Table B.1. Figure B.1 shows the grid map image with NaN values shown in green for this test. The time span between frame 1 and frame 12 in Figure B.1 is about 54 seconds. The time between each successive frame is about 4.5 seconds. Figure B.2 shows the inpainted grid map corresponding to the frames in Figure B.1 and Figure B.3 shows line detection results on the inpainted frame of Figure B.2. As shown in Figures B.3, the closest joist is detected in all frames except for 1, 2, 11, and 12. In frames 8 - 12, there is an uneven region behind the test joist, resulting in degraded inpainting images and line detections. One could hypothesize that this issue is due to the joist reflections, circular objects, and NaN regions on the floor.

Table B.1: Test 2 Setup

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
2	1.5	$0°$	10

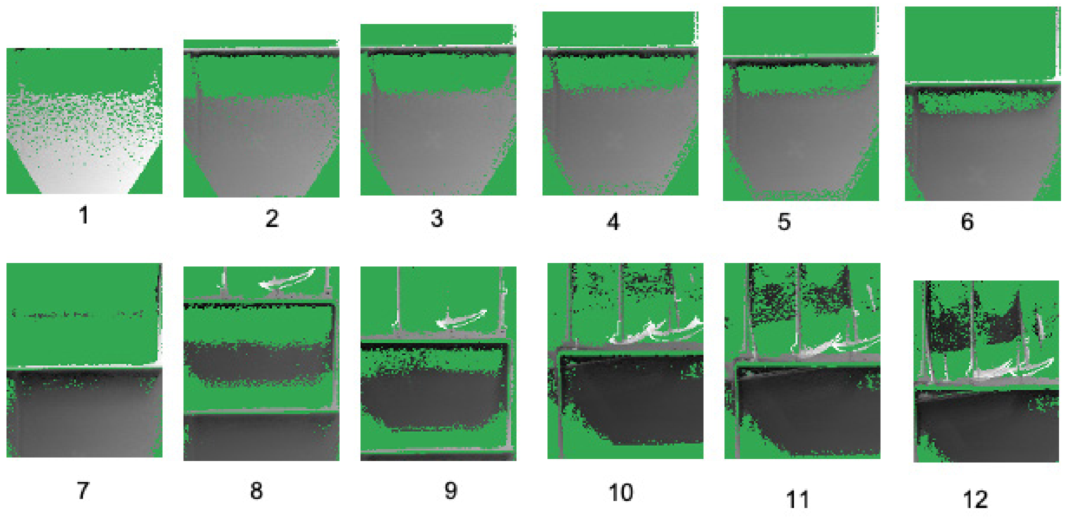


Figure B.1: Grid map images with all NaN value painted in green

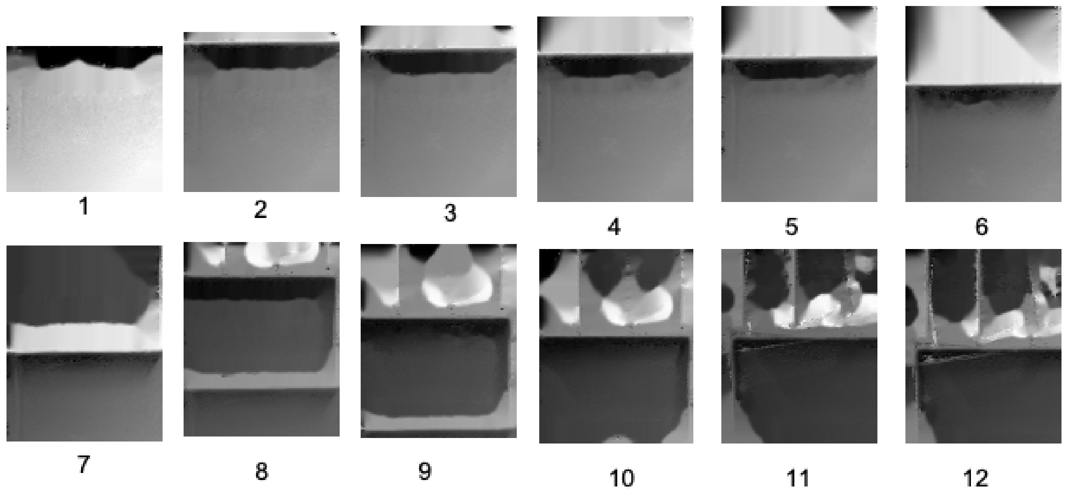


Figure B.2: Inpainted grid map images

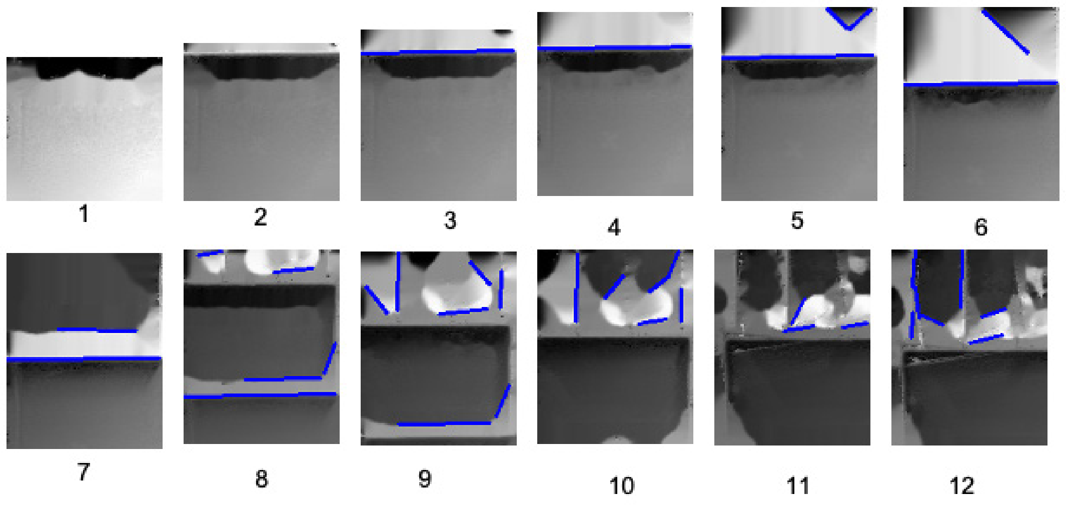


Figure B.3: Detected joist overlaid on inpainted grid map images

Appendix C

Joist Detection Figures For Test 3

The exact joist parameters in Test 3 is summarized in Table C.1. Figure C.1 shows the grid map image with NaN values shown in green for this test. The time span between frame 1 and frame 6 in Figure C.1 is about 20 seconds. The time between each successive frame is about 3.2 seconds. Figure C.2 shows the inpainted gridmap corresponding to the frames in Figure C.1 and Figure C.3 shows line detection results on the inpainted frame of Figure C.2. As shown in Figure C.3, only the front of the test joist is detected. The inpainted grid map image shows a degraded region behind the joist in frame 5. One could hypothesize that this issue is due to very few points behind the 18cm joist and NaN on the floor.

Table C.1: Test 3 Setup

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
3	1	0°	18

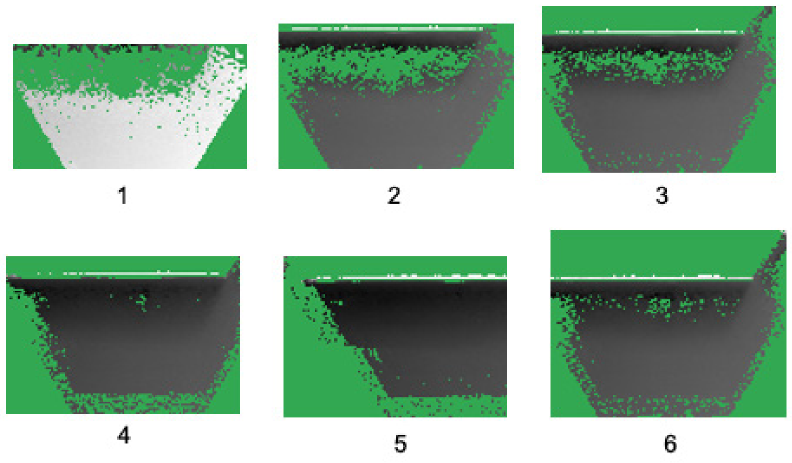


Figure C.1: Grid map images with all NaN value painted in green

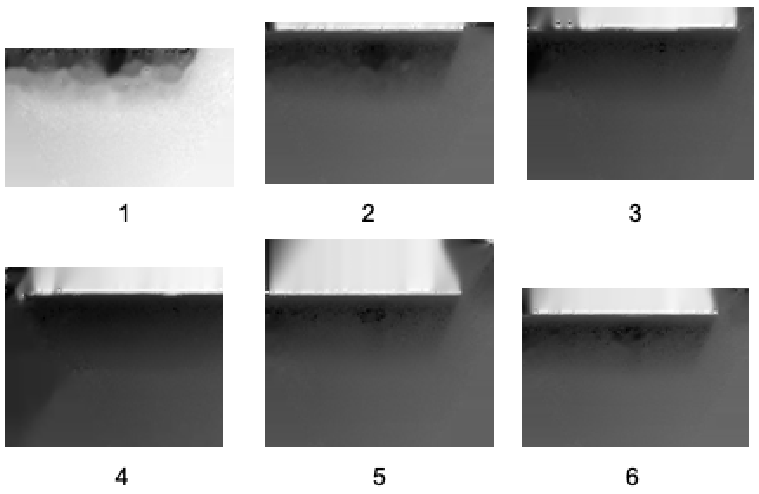


Figure C.2: Inpainted grid map images

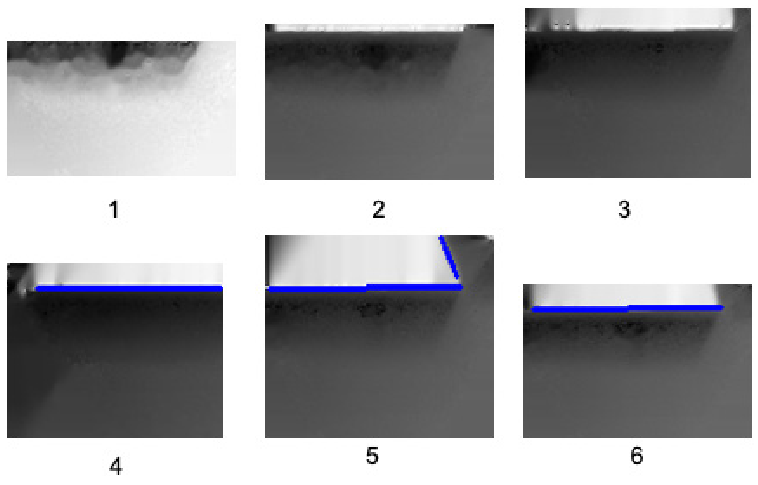


Figure C.3: Detected joist overlaid on inpainted grid map images

Appendix D

Joist Detection Figures For Test 4

The exact joist parameters in Test 4 is summarized in Table D.1. Figure D.1 shows the grid map image with NaN values shown in green for this test. The time span between frame 1 and frame 12 in Figure D.1 is about 39 seconds. The time between each successive frame is about 3.2 seconds. Figure D.2 shows the inpainted gridmap corresponding to the frames in Figure D.1 and Figure D.3 shows line detection results on the inpainted frame of Figure D.2. As shown in Figure D.3, the front and back of the test joist is detected in all grid map images except for frames 1 and 10. As the hexapod approaches the joist, the joist detection becomes more accurate.

Table D.1: Test 4 Setup

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
4	1	45°	10

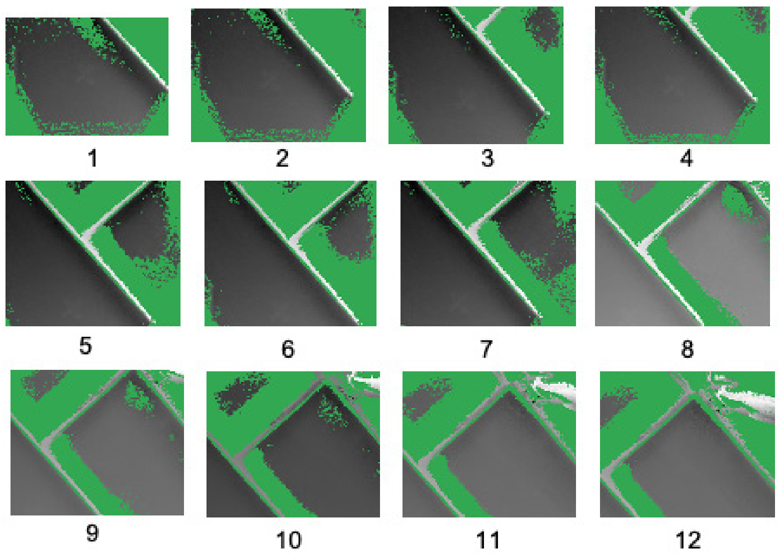


Figure D.1: Grid map images with all NaN value painted in green

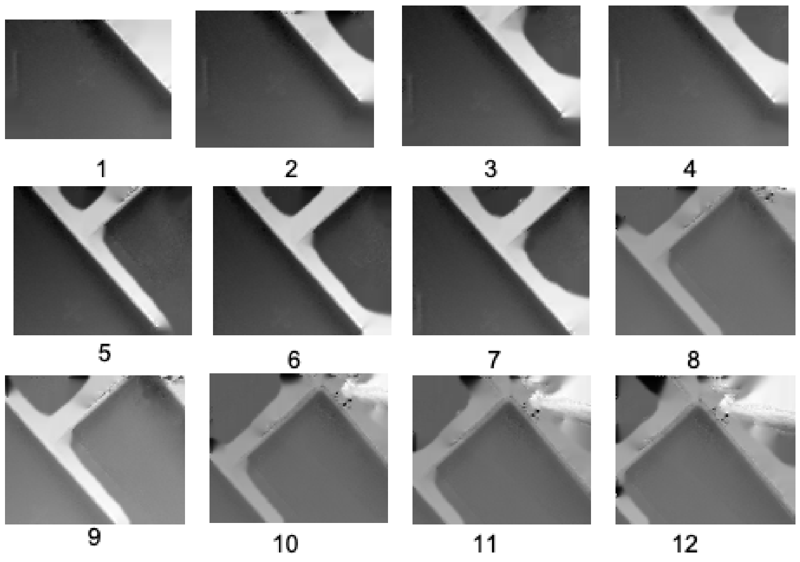


Figure D.2: Inpainted grid map images

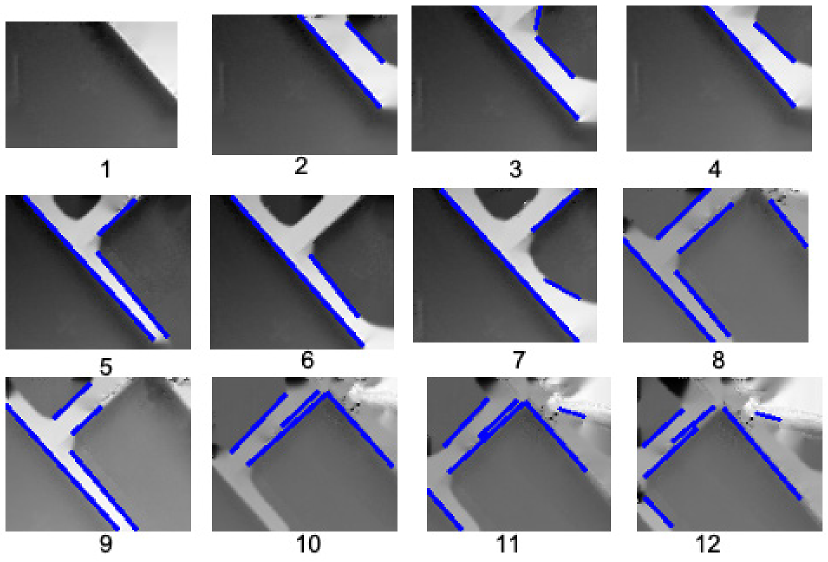


Figure D.3: Detected joist overlaid on inpainted grid map images

Appendix E

Joist Detection Figures For Test 5

The exact joist parameters in Test 5 is summarized in Table E.1. Figure E.1 shows the grid map image with NaN values shown in green for this test. The time span between frame 1 and frame 14 in Figure E.1 is about 42 seconds. The time between each successive frame is about 3 seconds. Figure E.2 shows the inpainted gridmap corresponding to the frames in Figure E.1 and Figure E.3 shows line detection results on the inpainted frame of Figure E.2. As shown in Figure E.3, the front and back of the test joist is detected in all grid map images except for frames 6, 11, and 14. The middle joist is also detected in frames 9 - 14.

Table E.1: Test 5 Setup

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
5	1	60°	10

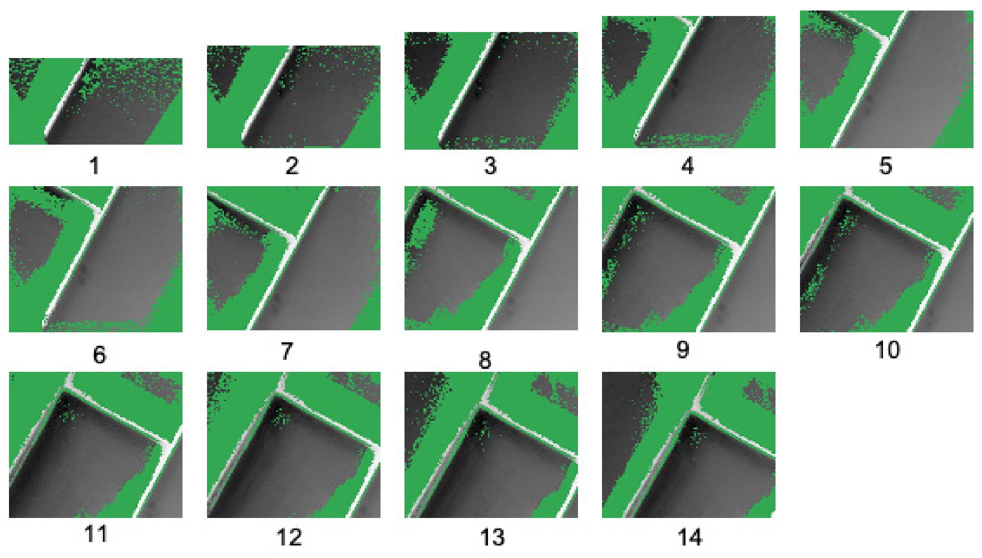


Figure E.1: Grid map images with all NaN value painted in green

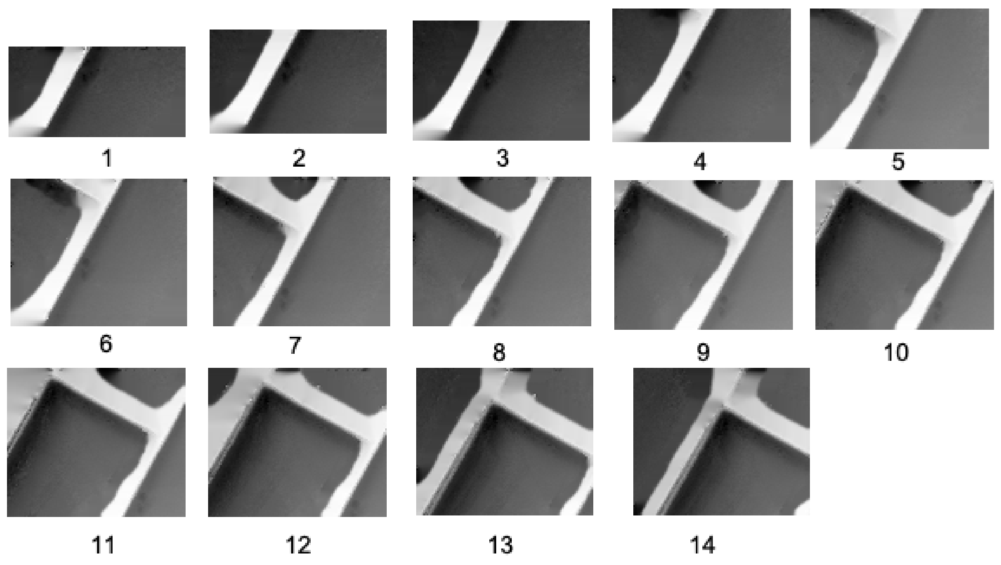


Figure E.2: Inpainted grid map images

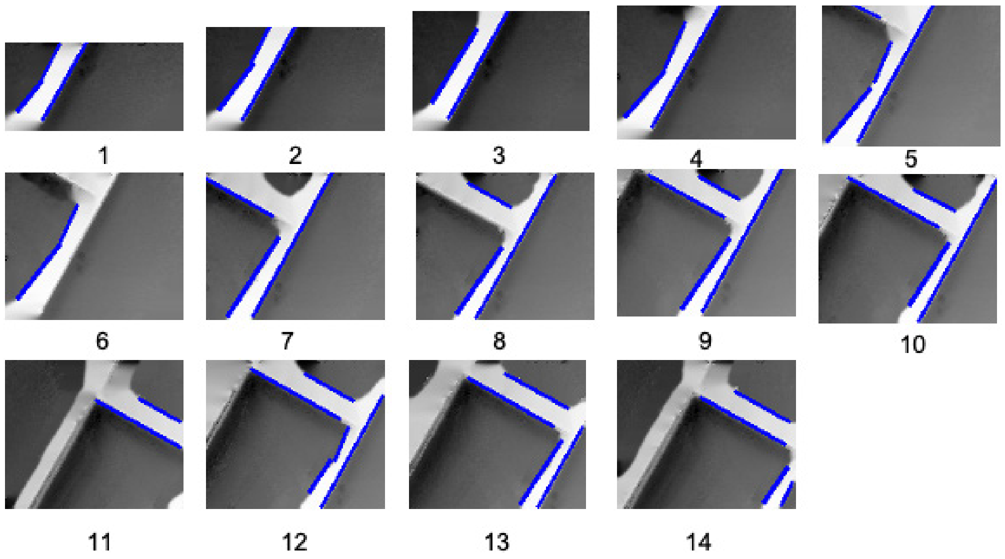


Figure E.3: Detected joist overlaid on inpainted grid map images

Appendix F

Joist Detection Figures For Test 6

The exact joist parameters in Test 6 is summarized in Table F.1. Figure F.1 shows the grid map image with NaN values shown in green for this test. The time span between frame 1 and frame 15 in Figure F.1 is about 40 seconds. The time between each successive frame is about 2.7 seconds. Figure F.2 shows the inpainted gridmap corresponding to the frames in Figure F.1 and Figure F.3 shows line detection results on the inpainted frame of Figure F.2. As shown in Figure F.3, the front and back of the test joist is detected in all grid map images except for frames 1 and 4. The line detector is also able to detect the joist that is perpendicular to the 75° joist.

Table F.1: Test 6 Setup

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
6	1	75°	10

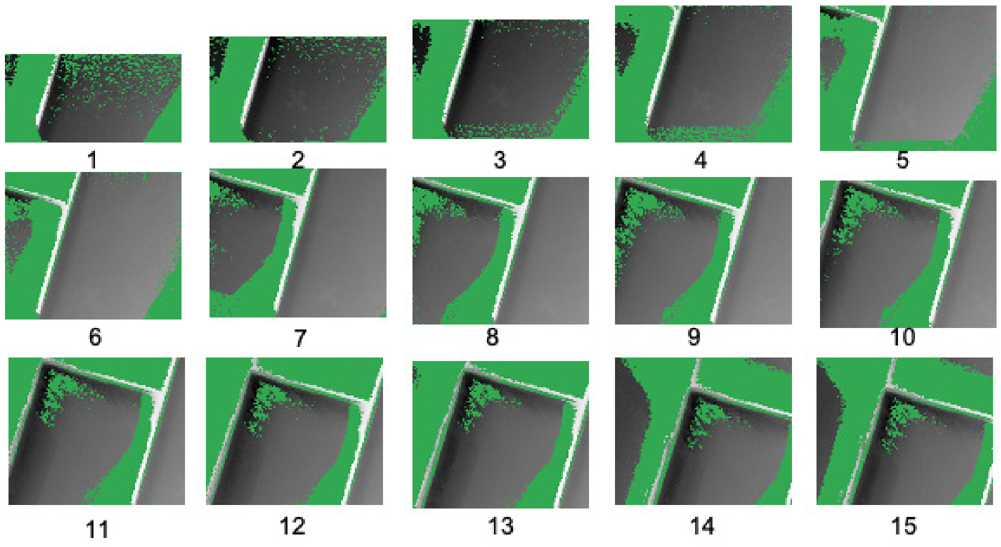


Figure F.1: Grid map images with all NaN value painted in green

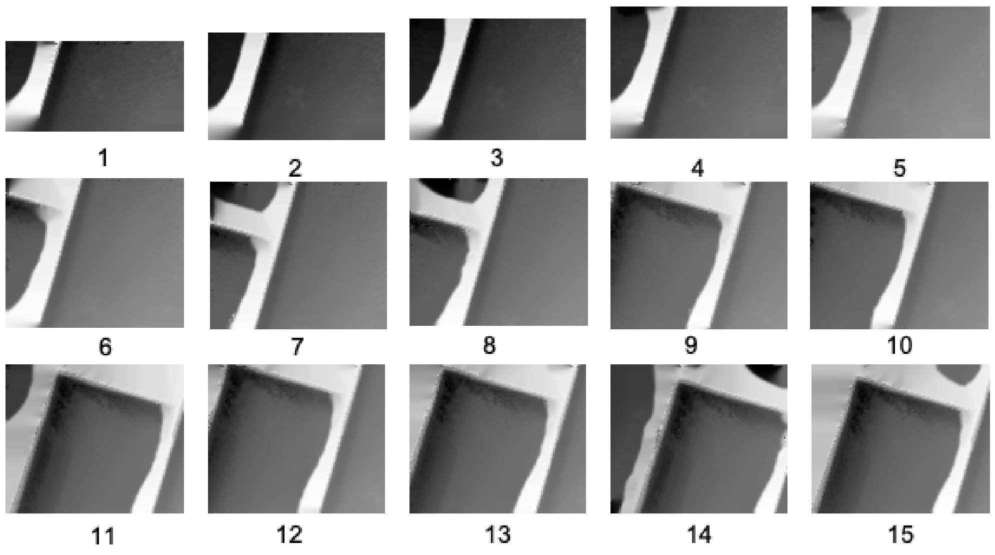


Figure F.2: Inpainted grid map images

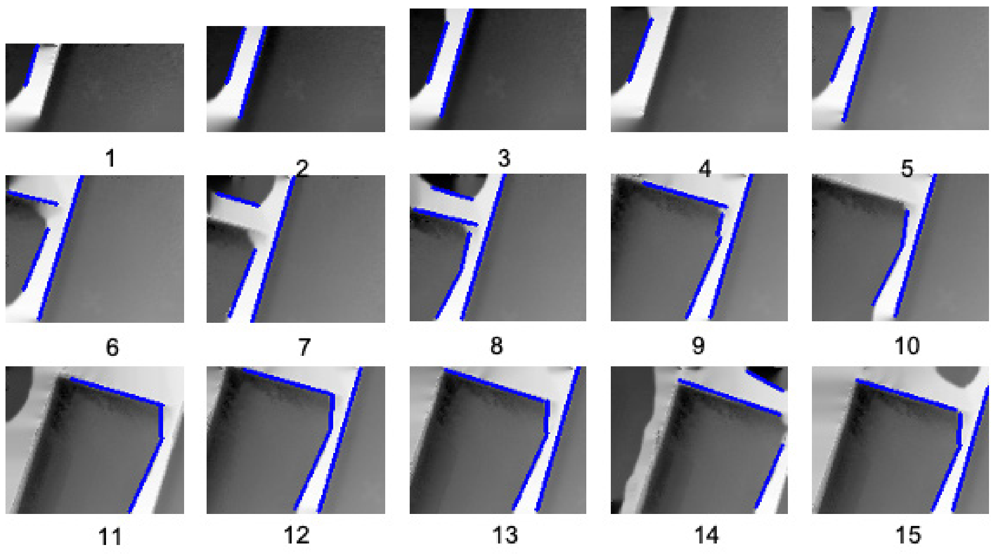


Figure F.3: Detected joist overlaid on inpainted grid map images

Appendix G

Joist Detection Figures For Test 7

The exact joist parameters in Test 7 is summarized in Table G.1. Figure G.1 shows the grid map image with NaN values shown in green for this test. The time span between frame 1 and frame 6 in Figure G.1 is about 55 seconds. The time between each successive frame is about 10 seconds. Figure G.2 shows the inpainted gridmap corresponding to the frames in Figure G.1 and Figure G.3 shows line detection results on the inpainted frame of Figure G.2. As shown in Figure G.3, the front of the test joist is detected in all grid map images, but there is a degraded inpainting region behind the test joist. This region is presumably the result of inpainting on an array of the joists, circular tubes, and the NaN region on the floor.

Table G.1: Test 7 Setup

Test	Joist Distance (m)	Joist Orientation	Joist Height (cm)
7	2.7	$0°$	18

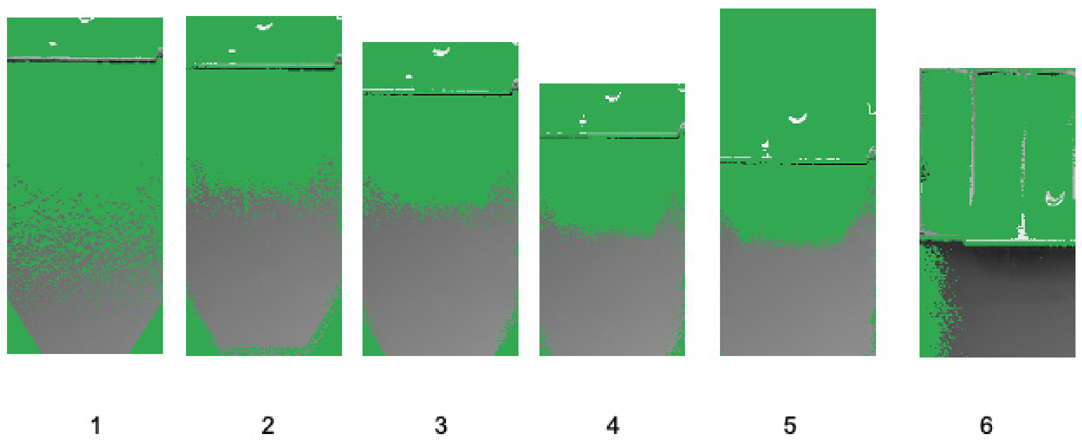


Figure G.1: Grid map images with all NaN value painted in green

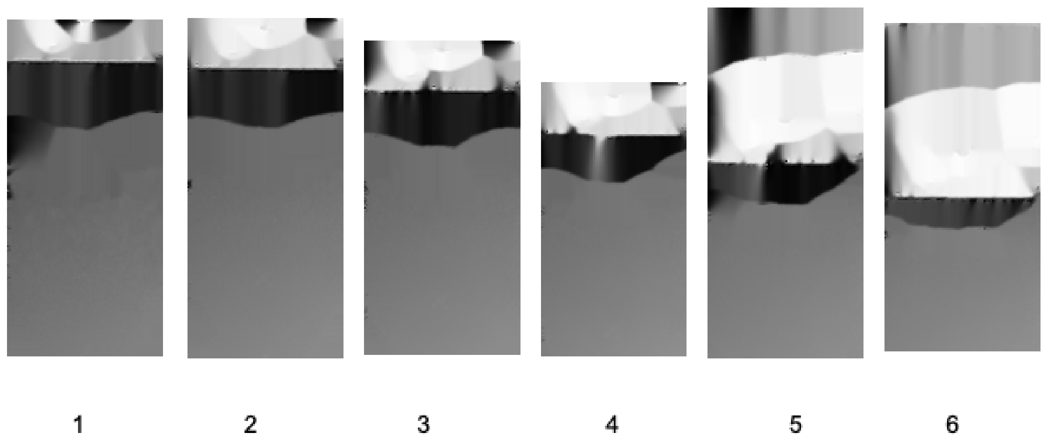


Figure G.2: Inpainted grid map

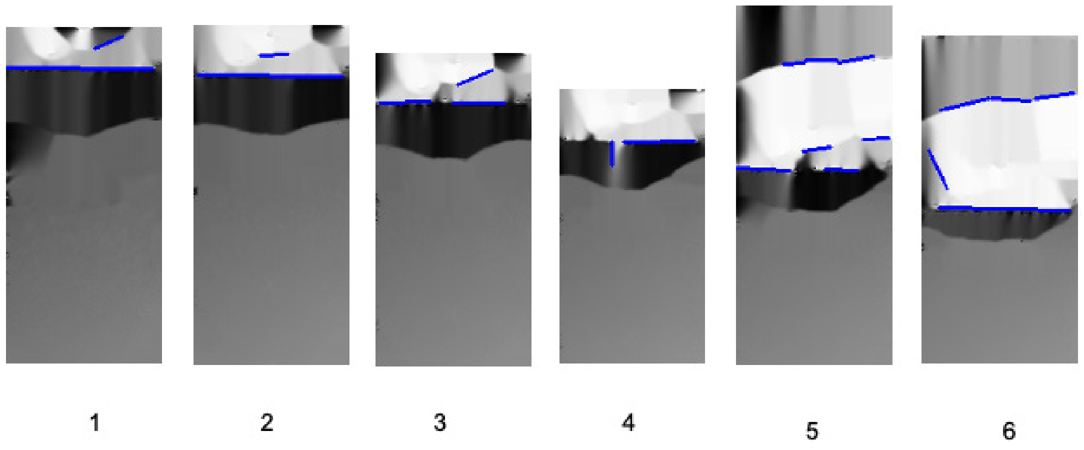


Figure G.3: Detected joist overlaid on inpainted grid map images