Fast Staircase Detection and Estimation using 3D Point Clouds with Multi-detection Merging for Heterogeneous Robots

Prasanna Sriganesh[1], Namya Bagree[2], Bhaskar Vundurthy[1], and Matthew Travers[1]

***Abstract*— Robotic systems need advanced mobility capabilities to operate in complex, three-dimensional environments designed for human use, e.g., multi-level buildings. Incorporating some level of autonomy enables robots to operate robustly, reliably, and efficiently in such complex environments, e.g., automatically "returning home" if communication between an operator and robot is lost during deployment. This work presents a novel method that enables mobile robots to robustly operate in multi-level environments by making it possible to autonomously locate and climb a range of different staircases. We present results wherein a wheeled robot works together with a quadrupedal system to quickly detect different staircases and reliably climb them. The performance of this novel staircase detection algorithm that is able to run on the heterogeneous platforms is compared to the current state-of-the-art detection algorithm. We show that our approach significantly increases the accuracy and speed at which detections occur.**

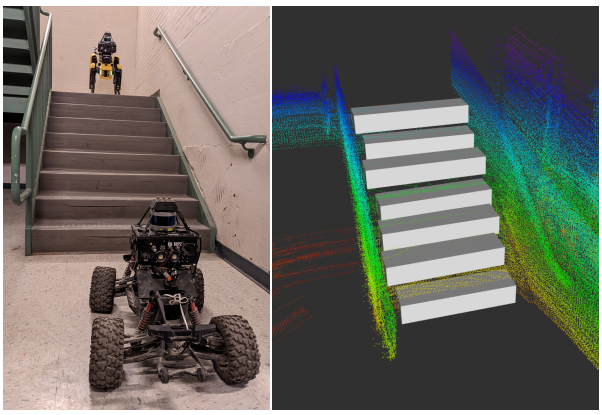
***Index Terms*— Robot Perception, Staircase Detection, Point Cloud Estimation**

I. Introduction

Mobile autonomy and advanced mobility are active research areas within the broader robotics community. However, they have primarily been treated as separate research fields. Approaching this as a tightly-coupled system will help augment the capabilities of mobile robot systems to operate in complex, three-dimensional environments designed for humans.

In the 2020 DARPA Subterranean Challenge, teams of heterogeneous robots, comprising wheeled, legged, and aerial robots, explored unknown urban spaces. Challenges like these typically operate in complex environments composed of underground spaces and multi-level buildings. Operatorless environments require the robots to perceive, analyze and make decisions autonomously. Environment-aware autonomy allows for coordinating robot actions by communicating information about traversable terrains like troughs, inclines, and staircases. Navigating staircases is a significant component of handling multi-floor environments. Knowledge of staircases from wheeled robots that scout enables legged robots to navigate staircases and explore environments that are not reachable by the scout robots. Achieving such real-time coordination with advanced mobile robots requires a fast, robust method of staircase detection.

Staircase detection and characterization are particularly challenging for autonomous robot perception systems. Diversity in staircases, for instance, spiral or hollow staircases, aggravates autonomous detection even further. Existing approaches have emphasized accuracy with a compromise on the speed of detection. Specifically, these approaches can be too slow when implemented in compute-constrained mobile robots. In this work, we focus on detecting a variety of staircases as fast as possible on mobile robots. Further, we aim to accurately indicate the number of steps in a staircase and the size of the steps to help assess if the staircase can be traversed.



(a) Two heterogeneous robots around a staircase (b) White marker shows the detected staircase

Fig. 1: Staircase environment and detection

The main contributions of this work are as follows:

- An algorithm to detect different types of staircases in real-time and estimate their geometry.
- An algorithm that combines two detection instances of the same staircase obtained by different robots or different viewpoints.

II. Related Work

There has been considerable research carried out in the field of staircase detection. All the methods either use images or point clouds as their primary modality. Once the staircases are detected, the next task is estimating the features of all detected stairs. While detection is fairly indifferent between images and point clouds, estimation benefits immensely from the latter.

One of the first image based approaches to staircase detection was put forth by Cong et al. [1] where the authors detect the edges formed by stairways in image space [2][3][4]. Murakami et al. [5] used both RGB images and depth images to segment edges and detect both ascending and descending staircases. Even contemporary learning based

[1]Prasanna Sriganesh, Bhaskar Vundurthy and Matthew Travers are from The Robotics Institute, Carnegie Mellon University, USA. {pkettava, pvundurt, mtravers}@andrew.cmu.edu

[2]Namya Bagree is from the Department of Mechanical Engineering, Carnegie Mellon University, USA nbagree@andrew.cmu.edu

(a) Ascending Staircase (b) Hollow Staircase (c) Spiral Staircase (d) Descending Staircase (e) Staircase with Debris

Fig. 2: Different types of staircases found in an urban environment including industrial and damaged staircases

algorithms have been used to detect staircases from images [6][7].

The only advantage of image based methods is the speed of the detections. Some shortcomings include reliance on images, which makes the system environment dependent. In low-light conditions, the robustness of the system goes down. Learning methods require massive datasets to work reliably. Moreover, the lack of depth information in RGB images prohibits accurate geometry or location estimation. The camera positioning also impacts the field of view of detection.

The primary idea of detecting staircases with point clouds is by segmenting planes formed by the stairs as they have a fixed geometry. Point clouds help also estimate the geometry and location of the staircase which can be used as an input for navigation. Point clouds are typically collected from LiDAR sensors and Depth cameras. Oßwald et al. [8] first explored and experimented on two different plane segmentation methods to detect risers of the staircase.

Different variations of Random Sample Consensus (RANSAC) has also been extensively used to segment planes and then detect staircases [9][10]. RANSAC based methods have been used to estimate staircase location for navigation by different robots [11][12][13]. Fourre et al. [14] even devised a way to localize industrial stairways which have no risers. Even though RANSAC is a simple and efficient algorithm, it is non-deterministic. It does not guarantee a best-solution or have a fixed time-bound. This is not a great way to detect stairways in scenarios that are time critical. These algorithms also have prerequisites on which part of the staircase needs to be visible to the sensor.

Westfechtel et al. [15] were the first to successfully achieve detection and estimation of staircases in all direction(360^o). They compared three different segmentation methods to detect planes in LiDAR point clouds and used a graph-based strategy to detect staircases of all types. Their estimates of the staircase location and the geometry were the best among all the previous work. Consequently, we can consider this work to be the current state-of-the-art for staircase detection. Although, their biggest drawback was the speed of the detection. The robot was expected to be static during the entire process, and plane analysis took around 4-8 seconds. This is entirely not feasible in search-and-rescue scenarios where every second is crucial.

None of the plane-based methods discuss detection speed, which is essential to us. All previous works look at staircase detection as a one-off algorithm. There has not been much research conducted on multiple robot use scenarios. Two robots with different viewpoints can help achieve a better estimation of a staircase by fusing both instances. These are two aspects we intend to address.

III. Methodology

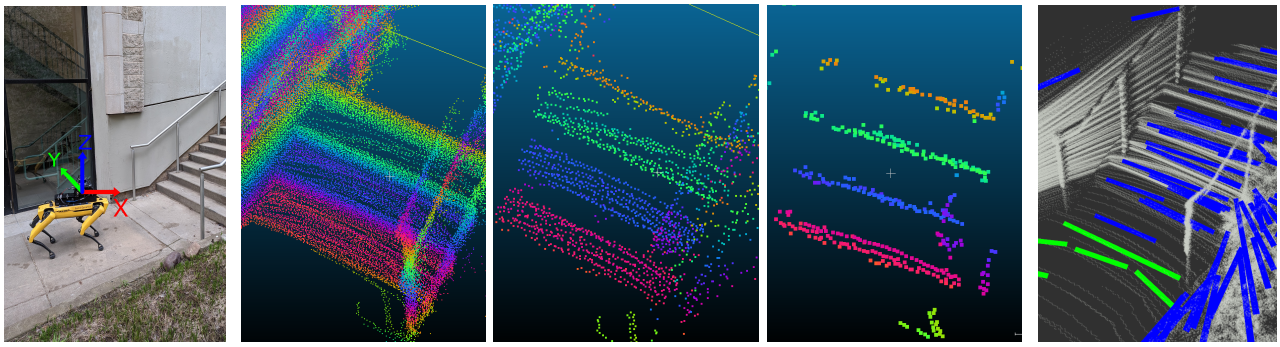
In this section, we present an algorithm that detects staircases from LiDAR point clouds and accurately estimates the staircase parameters such as location, height, and depth. The main intuition behind our algorithm is to segment only the edges formed by the staircase surfaces. The pipeline has three major steps: pre-processing, segmentation, and detection.

A. Pre-Processing

Let's first define the robot's frame. As shown in Fig. 3a, the x-axis is in line with the robot orientation, the z-axis faces upwards and y-axis is to its left. The input point cloud (accumulated from multiple LiDAR scans) is converted into a 3D voxel grid with fixed leaf size. This input cloud is shown in Fig. 3b.

The first step in pre-processing is to retain points with maximum height for a given X-Y cell index. Specifically, for each column in the Z direction, we retain only the one with the highest z-value. This step retains all the points visible from a top view of the point cloud. All the planes perpendicular to the ground are reduced to a line parallel to the ground. We apply this technique to the point cloud shown in Fig. 3b. The resulting point cloud is shown in Fig. 3c.

Next we organise this point cloud into a 2D array around the robot using cylindrical coordinates. The z-axis is discretised into rows of the array, while the columns are indexed using the azimuth angle θ which is given by $tan^{-1}(\frac{y}{x})$. If there are 2 or more points with similar z and θ, we then compute the range ρ $(= \sqrt{x^2 + y^2})$ of the point and retain the point with smallest range value. This essentially reduces all the horizontal planes (stairs) into a single edge.



(a) Robot and its reference frame (b) Input point cloud of the staircase (c) Point cloud after first step of pre-processing (d) Result point cloud after all pre-processing steps (e) Segmented lines, grouped based on height

Fig. 3: Preprocessing and segmentation overview for stair detection

The two pre-processing steps should reduce the 3D point cloud to points that belong to the stair edges. Fig. 3d shows the final processed cloud.

B. Segmentation

As mentioned before, the point cloud is organized into a 2D array structure that can be thought of as an unwrapped cylinder with the robot at its center. Each row index corresponds to a fixed z height, and the column indices represent the azimuth angle spanning from $-180°$ to $180°$. Consequently, a full row of this array can be treated as single 2D laser scan. Furthermore, since all points in each row have approximately equal z-values, we can fit lines in the xy space and add the height information later.

To extract lines from a 2D laser-scan, we use a modified version of Iterative-End-Point Fit [16][17]. Given N points in a scan, we fit a line between the first and last points to start the splitting. We then find the point with the maximum distance to this line. If the distance is more than a threshold (d_p), the points are split into two groups until the number of points in each set is greater than a limit (N_{min}) or if all the points are at a distance less than a threshold d_p.

Typically, if all the points are close to the line, the loop stops and returns the line. We modify this behavior to use weighted line fitting to estimate a line with all the points. Weighted line fitting is a version of least square line fitting with the uncertainty of a point used as weights. We refer to [18] for a detailed discussion of the problem.

The merging is usually done if the points forming the two lines are collinear. We modify this to use the merging criteria provided by weighted line fitting. The main advantage of weighted line fitting is that it outputs a covariance for each line. This allows us to estimate the similarity between two lines better and merge it better. We represent each line fit using this method by the parameters below:

- 3D start point of the line - $\bar{p_s}$
- 3D end point of the line - $\bar{p_e}$
- Orientation α - angle between the line with the xy plane
- Covariance Matrix P_L

This line segmentation algorithm is run separately on each row (2D scan) of the point cloud array. All the resulting lines are added to a single list $\mathcal{L}$. As a result, we get lines parallel to the ground plane. Since staircase edges also have to be parallel to the ground, it eliminates the need to check for line altitudes. If the point cloud has Z rows and T columns, the time complexity of this algorithm is $O(ZTlog(T))$.

After all the lines are segmented, we group them into three groups based on their height in the 3D space. Assuming that the robot has a fixed height, it is trivial to group lines that are above the ground ($\mathcal{L}_{ag}$), below the ground ($\mathcal{L}_{bg}$), and that are part of the ground plane ($\mathcal{L}_g$). This makes it easier to search for staircases that are ascending and descending. Fig. 3e shows the lines detected by our algorithm in that scene. Blue represents lines above the ground, and the green represents lines on the ground plane.

C. Detection

We start searching for staircases in the segmented lines from the previous step. We first describe the model of our staircase as follows. Fig. 4a also describes these parameters.

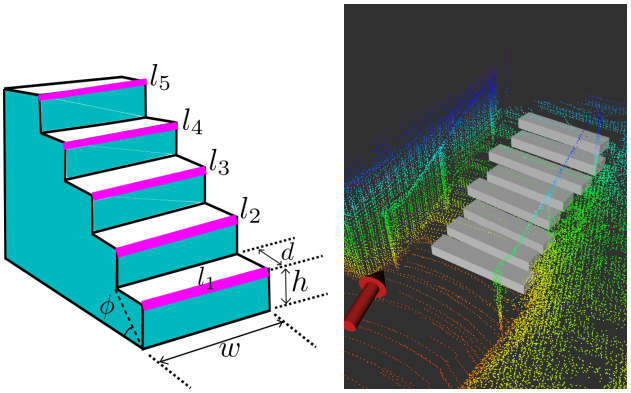
- Step Height, h
- Step Depth, d
- Step Width, w
- List of Lines $\mathcal{L}$, Each line l_i represents a stair and is described by 3D start ($\bar{p_s}^{(i)}$) point, 3D end ($\bar{p_e}^{(i)}$) point and it's orientation ($\alpha^{(i)}$) in the XY plane
- Stair Slope, $\phi = tan^{-1}(h/d)$

In order to start grouping a set of lines that form a staircase, we define a stair case based on the standards set by OSHA[19].

Definition 1: Given two lines $l_1(\bar{p_s}^{(1)}, \bar{p_e}^{(1)}, \alpha^{(1)})$ and $l_2(\bar{p_s}^{(2)}, \bar{p_e}^{(2)}, \alpha^{(2)})$, we compute the height h_i given by the difference in z between the lines, the depth d_i given by the xy distance between the lines and the slope $\phi_i = tan^{-1}(h_i/d_i)$. Any two lines that satisfy the five conditions is defined as a **stair**:

1) $0.11 \text{ m} \leq h_i \leq 0.30 \text{ m}$
2) $0.15 \text{ m} \leq d_i \leq 0.45 \text{ m}$
3) $25° \leq \phi_i \leq 60°$
4) $|\alpha^{(1)} - \alpha^{(2)}| \leq 10°$
5) There is no other line in between l_1 and l_2 ■

Depending on whether we want to detect an ascending or descending staircase, we pick the appropriate list ($\mathcal{L}_{ag}$ or $\mathcal{L}_{bg}$) of segmented lines. The algorithm to detect staircases



(a) Staircase model with parameters, pink lines represent stair edges to be segmented

(b) Detected staircase, each white marker corresponds to a stair

Fig. 4: Staircase model and successful detection result

has two parts, initialization, and extension. In the initialization stage, we create a subset of lines that are below 2.5 h_{max} in the z-direction called the initialization list. Here h_{max} is the maximum step height allowable (0.3 m). We then look for two lines in this initial list that can form a staircase as per Definition 1. If two such lines exist, we add them to a staircase list $\mathcal{S}$.

Once we have the first two steps of the staircase, we perform an extension. For every line in the increasing z-direction, we check if it forms a staircase with the previous line in the staircase set using Definition 1. If it complies, we add the new line to the set and repeat the extension until all lines are exhausted. If there are more than four lines in the staircase list, it is a successful staircase detection, and use this to estimate the staircase geometry.

If the initialized staircase does not extend, i.e., the set only has two stairs, we remove the lines from the initialization list and repeat the process until the initialization list is empty. The pseudo-code of this method is presented in **Algorithm 1**. To detect descending staircases, we switch the set of lines to L_{bg} and run the same algorithm, but the lines are checked in decreasing z-direction. Fig. 4b shows the detected staircase.

D. Estimation

After detection, we estimate the model parameters as shown in Fig. 4a. Our detection gives us a list of lines $\mathcal{S}$ with k lines. Assuming that all stairs in a stairway have similar dimensions, we compute the parameters as shown below:

$$h = \frac{\sum_{n=1}^{k-1} ||\bar{p_s}^{(i+1)} - \bar{p_s}^{(i)}||_z + ||\bar{p_e}^{(i+1)} - \bar{p_e}^{(i)}||_z}{2k}$$

$$d = \frac{\sum_{n=1}^{k-1} ||\bar{p_s}^{(i+1)} - \bar{p_s}^{(i)}||_{xy} + ||\bar{p_e}^{(i+1)} - \bar{p_e}^{(i)}||_{xy}}{2k}$$

$$w = \frac{\sum_{n=1}^{k} ||\bar{p_s}^{(i)} - \bar{p_e}^{(i)}||}{k}$$

where $||p_1 - p_2||_z$ *is the 1D distance in z axis*

where $||p_1 - p_2||_{xy}$ *is the euclidean distance in xy plane*

We can also define the location of the staircase ($\bar{S_p}$) at the center of the first stair, given by:

$$\bar{S_p} = (\bar{p_s}^{(1)} + \bar{p_e}^{(1)})/2$$

Algorithm 1 Detect ascending staircase from list of lines

Input: Set of above ground lines - $\mathcal{L}_{ag}$
Output: Staircase $\mathcal{S}$ with N lines

Initialize list $\mathcal{L}_I$ with all lines ($\mathcal{L}_{ag} \leq 2.5 h_{max}$)
while $\mathcal{L}_I$ not empty **do**
  Initialize empty staircase set $\mathcal{S}$
  $stair_{init} \leftarrow false$
  while $stair_{init}$ is $false$ **do**
    Pick 2 lines, l_1, l_2 from $\mathcal{L}_I$
    if l_1 and l_2 form a stair (Definition 1) **then**
      $stair_{init} = true$
      Add l_1 and l_2 to S
    end if
    if No valid pair exits **then**
      return $\mathcal{S}$
    end if
  end while
  Reorder lines in $\mathcal{L}_{ag}$ by ascending order of height(z)
  for Every line l_{curr} in $\mathcal{L}_{ag}$ **do**
    $l_{prev} \leftarrow$ last line in $\mathcal{S}$
    if l_{curr} and l_{prev} form a stair (Definition 1) **then**
      Add l_{curr} to Set $\mathcal{S}$
    end if
  end for
  if Total stairs in $\mathcal{S} \geq 4$ **then**
    return $\mathcal{S}$
  else
    Remove the initialized stair lines from $\mathcal{L}_I$
  end if
  Initialize empty staircase list $\mathcal{S}$
end while
return $\mathcal{S}$

E. Multi-Detection Merging

This section describes a simple algorithm that can merge two different detection instances of the same staircase. The detections can be either by the same robot during different time instances or by using different robots with different viewpoints. We would first like to define a criteria to classify stairs as similar.

Definition 2: Given two stairs $l_1(\bar{p_s}^{(1)}, \bar{p_e}^{(1)}, \alpha^{(1)})$ and $l_2(\bar{p_s}^{(2)}, \bar{p_e}^{(2)}, \alpha^{(2)})$, we first compute the height h_i given by the difference in z between the stairs, the depth d_i given by the xy distance between the stairs. Any two stairs that satisfy the following three conditions are considered to be the **same stair**:

1) $h_i \leq 0.05$ m
2) $d_i \leq 0.05$ m
3) $|\alpha^{(1)} - \alpha^{(2)}| \leq 10°$ ■

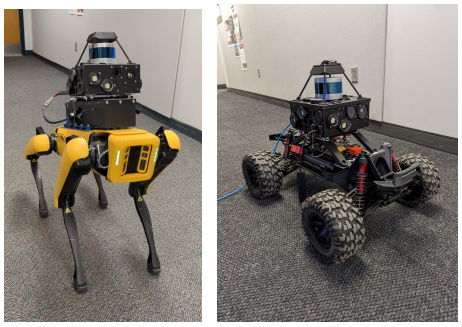
Algorithm 2 describes the way to combine two detection instances. The main idea is to find the location of the intersection for two detections. We do this by iteratively

Algorithm 2 Merging staircases

Input: Two Staircase Detections $\mathcal{S}_a$ with k stairs and $\mathcal{S}_b$ with m stairs with $k \leq m$

Output: Fused Staircase $\mathcal{S}$ if successful, else NO_MATCH

$match \leftarrow false$
for Every stair l_a in $\mathcal{S}_a$ **do**
  for Every stair l_b in $\mathcal{S}_b$ **do**
    if l_a and l_b are similar stairs (Definition 2) **then**
      $match \leftarrow true$ ▷ Stair match found
      $i \leftarrow index(l_a)$ and $j \leftarrow index(l_b)$
      break and **goto** 10
    end if
  end for
end for
if $match$ is $false$ **return** NO_MATCH
$\mathcal{S} \leftarrow \{l_1, \cdots, merge(l_i, l_j), merge(l_{i+1}, l_{j+1}), \cdots, l_m\}$ where 2 stairs are merged using [18]
Re-estimate parameters of Staircase $\mathcal{S}$ and **return** $\mathcal{S}$



(a) Spot Legged Robot (b) Autonomous Ground Vehicle

Fig. 5: Mobile robot platforms used to test our algorithm

finding two lines, one from each detection, that are similar. This common line (stair) between detections will act as an anchor point to merge the two detections. Once merged, if any more stairs do not have a corresponding pair, it is appropriately appended to the end/start of the list. The staircase parameters are re-estimated after this merge.

This algorithm allows the merging of any two detections irrespective of the time of the detection or the robot's viewpoint as long as the point clouds are registered in the same global frame, and there is at least one common stair between the detections.

IV. Experimentation and Results

A. Experimentation

1) System Overview: We make use of heterogeneous robots with identical perception sensor payloads. Fig. 5a shows the Boston Dynamics Spot legged robot. This robot is capable of climbing staircases once its location has been estimated. Fig. 5b shows the unmanned ground vehicle. The vehicle is capable of moving at up to 6m/s autonomously. The sensor payload has four fish-eye cameras on each side and a LIDAR sensor to obtain 3-dimensional scans of the environment. It also houses a Jetson AGX Xavier as the processor that performs SLAM using Super Odometry [20] and other autonomy tasks.

2) Setup: We tested our algorithm's performance on staircases shown in Fig. 2 and performed a comparison with the state-of-the-art [15]. The leaf size of the voxel grid was set to 2.5 cm. As the robots move around these five staircase types, we create a dataset that spans multiple distance points and orientations in front of the staircase. We aim to detect all five kinds of staircases, followed by estimating the geometric parameters of the detected stairs. An essential aspect of our comparison is the time taken to obtain a detection.

B. Results

Fig. 6 summarizes all the detections from our algorithm in five kinds of staircases. The white markers represent the detected stairs, which are resized to match our algorithm's estimation results. It is worth noting that the algorithm detects all staircases with an average computation time of $20.64\ ms$ as opposed to the $1511.33\ ms$ using SOTA.

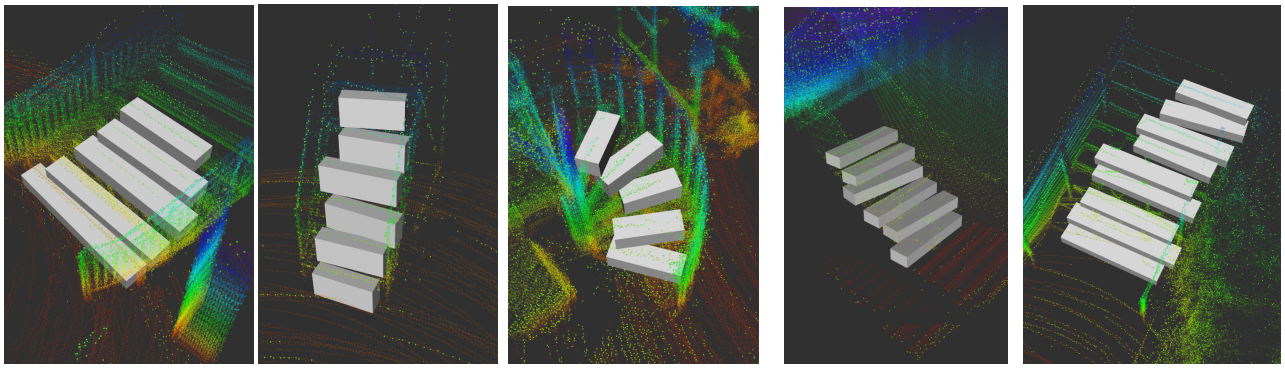
TABLE I summarizes our algorithm's estimation accuracy and detection speed and compares it to the SOTA. We recorded the average error with respect to the ground truth in estimating height, depth, and width. Green indicates better accuracy for that parameter for the given algorithm. As a general trend, we estimate the height and depth better than the SOTA. While both algorithms trade blows with standard ascending and descending staircases, we perform much better than the SOTA in other cases. Our algorithm has significantly higher estimation accuracy when it comes to hollow or spiral staircases. We also performed relatively better when the staircases had some debris on them. Our detection speeds were faster by at least two orders of magnitude.

In TABLE II, we recorded the performance of both algorithms. Both algorithms detect all staircases for standard ascending and descending staircases, but the SOTA has a lot of false positives in the same scene. In the hollow staircase scenario, the SOTA could barely detect it successfully. In addition to this, we were also able to run our algorithm in real-time on the robots. Fig. 8 shows the result we detected both an ascending and descending staircase simultaneously.

We also evaluated the performance of our merging algorithm. We were successfully able to merge two different detection instances into one. Fig. 7 shows the overview of this process. In this case, Fig. 7(a) was the detection result of the Spot when it was on top of the staircase looking down, and 7(b) was the output of the ground vehicle from the bottom. We can clearly see a better merged staircase detection in Fig. 7(c). This algorithm is not only limited to multiple robots but will also work with two different instances of a detection on the same robot.

V. Conclusions and Future Work

In this paper, we presented an algorithm to detect and estimate staircases in real-time and merge detections of the same staircase from different robots. We successfully demonstrated our algorithms on two heterogeneous robots and evaluated the accuracy of our estimation. Compared to



(a) Ascending Staircase (b) Hollow Staircase (c) Spiral Staircase (d) Descending Staircase (e) Staircase with Debris

Fig. 6: Staircase detection results for different types of staircases shown in Fig. 2.

Staircase Type	Our Method				Westfechtel et al. [15]			
	Average Time Taken (ms)	Height Error (cm)	Depth Error (cm)	Width Error (cm)	Average Time Taken (ms)	Height Error (cm)	Depth Error (cm)	Width Error (cm)
Ascending Staircase	21	1.101	2.804	38.125	1569	1.527	2.564	7.023
Hollow Staircase	18	0.982	0.839	15.121	1009	7.673	13.760	41.233
Spiral Staircase	15	0.490	3.467	11.040	1005	3.357	2.687	27.667
Descending Staircase	10	2.367	2.653	27.773	623	1.517	4.470	15.437
Staircase with Debris	48	2.980	2.362	29.612	4282	3.660	3.736	93.270

TABLE I: Comparison of parameter estimation errors and time taken by our method vs [15] for 5 categories of staircases

	Samples	Our Method		Westfechtel et al.[15]	
		Detection	False Positives	Detection	False Positives
Ascending Staircase	15	15	0	15	10
Hollow Staircase	17	17	1	3	1
Spiral Staircase	15	15	0	15	14
Descending Staircase	14	14	0	12	3
Staircase with Debris	10	10	1	10	10

TABLE II: Performance comparison of our method vs [15] for five categories of staircases

the state-of-the-art, our algorithm is faster by an order of two magnitudes while maintaining similar accuracy for the staircase parameters.

One of the limitations of our algorithm is in scenarios where the risers or treads of the stairs are sloped or if the stair edges are curved. In these cases, the segmented lines would be noisy, resulting in failed detections. We want to address these scenarios in future work. Further, we would like to improve the merging algorithm by potentially using a Kalman filter that can help in noisy environments.

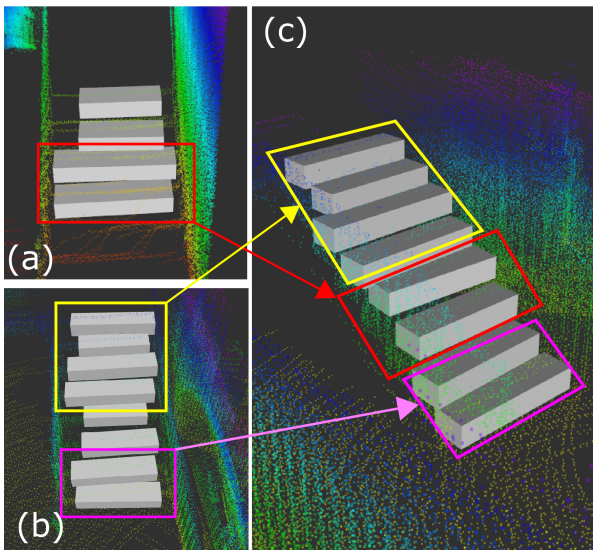


Fig. 7: Two detections [(a), (b)] get merged into one good detection (c) using Algorithm 2.

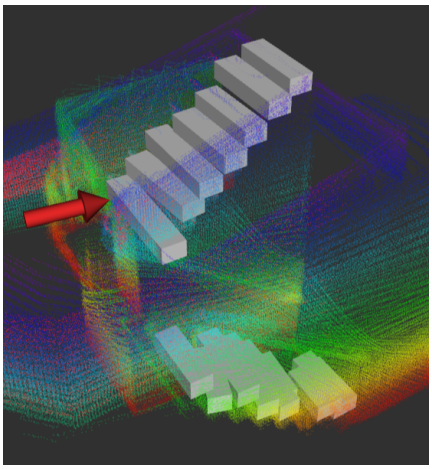


Fig. 8: Ascending and descending staircases detected by our algorithm simultaneously

References

[1] Y. Cong, X. Li, J. Liu, and Y. Tang, "A stairway detection algorithm based on vision for ugv stair climbing," in *2008 IEEE International Conference on Networking, Sensing and Control*, IEEE, 2008, pp. 1806–1811.

[2] C. Zhong, Y. Zhuang, and W. Wang, "Stairway detection using gabor filter and ffpg," in *2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, IEEE, 2011, pp. 578–582.

[3] H. Harms, E. Rehder, T. Schwarze, and M. Lauer, "Detection of ascending stairs using stereo vision," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 2496–2502.

[4] W. Samakming and J. Srinonchat, "Development image processing technique for climbing stair of small humaniod robot," in *2008 International Conference on Computer Science and Information Technology*, IEEE, 2008, pp. 616–619.

[5] S. Murakami, M. Shimakawa, K. Kivota, and T. Kato, "Study on stairs detection using rgb-depth images," in *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, IEEE, 2014, pp. 1186–1191.

[6] U. Patil, A. Gujarathi, A. Kulkarni, *et al.*, "Deep learning based stair detection and statistical image filtering for autonomous stair climbing," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, IEEE, 2019, pp. 159–166.

[7] M. Ilyas, A. K. Lakshmanan, A. V. Le, and R. E. Mohan, "Staircase recognition and localization using convolution neural network (cnn) for cleaning robot application," *Preprints*, 2018.

[8] S. Oßwald, J.-S. Gutmann, A. Hornung, and M. Bennewitz, "From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2011, pp. 93–98.

[9] M. Vlaminck, L. Jovanov, P. Van Hese, B. Goossens, W. Philips, and A. Pižurica, "Obstacle detection for pedestrians with a visual impairment based on 3d imaging," in *2013 International Conference on 3D Imaging*, IEEE, 2013, pp. 1–7.

[10] X. Qian and C. Ye, "Ncc-ransac: A fast plane extraction method for 3-d range data segmentation," *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2771–2783, 2014.

[11] B. Sharma and I. A. Syed, "Where to begin climbing? computing start-of-stair position for robotic platforms," in *2019 11th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, 2019, pp. 110–116.

[12] S. Woo, J. Shin, Y. H. Lee, *et al.*, "Stair-mapping with point-cloud data and stair-modeling for quadruped robot," in *2019 16th international conference on ubiquitous robots (UR)*, IEEE, 2019, pp. 81–86.

[13] J. A. Sánchez-Rojas, J. A. Arias-Aguilar, H. Takemura, and A. E. Petrilli-Barceló, "Staircase detection, characterization and approach pipeline for search and rescue robots," *Applied Sciences*, vol. 11, no. 22, p. 10 736, 2021.

[14] J. Fourre, V. Vauchey, Y. Dupuis, and X. Savatier, "Autonomous rgbd-based industrial staircase localization from tracked robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 10 691–10 696.

[15] T. Westfechtel, K. Ohno, B. Mertsching, *et al.*, "Robust stairway-detection and localization method for mobile robots using a graph-based model and competing initializations," *The International Journal of Robotics Research*, vol. 37, no. 12, pp. 1463–1483, 2018.

[16] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer graphics and image processing*, vol. 1, no. 3, pp. 244–256, 1972.

[17] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2005, pp. 1929–1934.

[18] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, IEEE, vol. 1, 2003, pp. 1304–1311.

[19] *1910.25 - Stairways. — Occupational Safety and Health Administration*, `https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.25`, Accessed: 2022-09-11.

[20] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 8729–8736.