Automated Incline Detection for Assistive Powered Wheelchairs

Mahdieh Nejati[1] and Brenna D. Argall[2]

Abstract— **This work presents an algorithm for automated real-time ramp detection using 3D point cloud data in the context of shared-control powered wheelchairs. Limitations in the interfaces available to those with severe motor impairments can make basic maneuvering tasks with powered wheelchairs difficult. Although a significant amount of work has been done on obstacle detection and avoidance, much less attention has been given to algorithms for the safe and reliable detection of ramps and inclines; even though navigating these structures is an important part of urban life. We provide an algorithmic solution for accurately detecting traversable inclines for applications with powered wheelchairs using the Point Cloud Library (PCL) within the Robotics Operating System (ROS) framework. All algorithms are implemented first in simulation and later evaluated on data obtained from indoor and outdoor urban environments. We measure the performance of our algorithm with systematic testing on several different ramp structures, observed from varied viewpoints. Results show that our algorithm is successful in detecting the orientation, slope, and width of traversable ramps with up to 100% accuracy and an average detection accuracy of 88%.**

I. INTRODUCTION

Powered wheelchairs are a solution to providing independent mobility to people who cannot operate a manual wheelchair for reasons of limited strength or impairment. Shoulder forces and moments more than double when ascending ramps with manual wheelchairs [1]. Electric wheelchairs eliminate this physical burden. However, a different set of challenges are introduced with binary control interfaces. For those with sever motor impairments, commercial powered wheelchairs are commonly equipped with binary interfaces such as Electronic Head Array systems or Sip-and-Puff (SNP) devices. These interfaces are often cumbersome to operate, and the difficulty in using these devices can make certain tasks that require fine control difficult.

Ramps require particularly careful consideration [2]. Consider a SNP wheelchair user traversing a ramp into a van. SNP devices work as switches, and using them for modulating the speed and power in a way that is necessary for ramp traversal is extremely difficult. The speed and heading commands are non-proportional, and do not scale with the magnitude of the breath exhalation/inhalation issued by the user. Speed may be changed only by first navigating to a menu (typically four speeds are available). So before driving up the ramp, the user must give the correct number of binary turning commands in order to be properly aligned and to prevent crashing with the edges. Driving up a ramp at the lowest speed level is not possible (insufficient power). On the other hand, using anything other than the lowest speed level after driving up the ramp while inside the van is dangerous and may result in crashing into walls.

Detecting inclines also has implications for the broader field of mobile robots. Traditionally, mobile robots have been restricted to traversing level ground. Much attention has been paid in mobile robotics research to obstacle detection and avoidance while traversing level ground. This restriction presents a significant limitation to real-world applications and urban navigation, such as sacrificing detecting ramps and traversing highly uneven floors [3]. Significantly less work has been done on detecting drop-offs and inclines, although detecting these are just as crucial and complex.

One solution to the disparity between demand and current state-of-the-art is the use of shared-controlled assistive robotic wheelchairs, where the task of navigation and steering is shared between the user and the robot. Assistive tasks previously addressed on our robotic wheelchair platform (Fig. 1) include automated open doorway detection [4] and automated perception of safe docking locations [5]. The aim of this work is to expand on this paradigm with automated ramp detection and alignment information.

In this paper, we provide a method for detecting ramps and traversable inclines without any *a priori* knowledge of the environment. We use point cloud data collected from an RGB-D sensor (camera image + depth), using the Point Cloud Library (PCL) within the Robot Operating Systems (ROS). Furthermore, the information from the RGB-D sensor can be used for purposes other than safety, for example, object recognition. We update the traversability costmap based on information from the algorithm. The result is a robust and rapid detection of the location and orientation of navigable ramps over which a wheelchair can drive.

We evaluate our system quantitatively by constructing local safety maps for different datasets and comparing them against ground truth maps to measure error. We also measure the accuracy of the ramp detection process.

In the following section, we overview related work on ramp detection and robotic wheelchair perception (Sec. II). In Section III we describe our algorithm for detecting traversable ramps. We describe our experimental setup and discuss the results in Section IV, followed by conclusions and future work in Section V.

[1]Mahdieh Nejati is with the McCormick School of Engineering Master of Science in Robotics Program, Northwestern University, Evanston, IL, USA m.nejati@u.northwestern.edu

[2]Brenna D. Argall is jointly appointed with the Department of Electrical Engineering and Computer Science, the Department of Mechanical Engineering, and the Department of Physical Medicine and Rehabilitation, Northwestern University, Evanston, IL, USA and the Rehabilitation Institute of Chicago, Chicago, IL, USA brenna.argall@northwestern.edu

II. Related Work

Here we present prior work on perception and incline detection for mobile robots and robotic wheelchairs.

A. Ramp and Drop-off Detection

Examples of ramp detection use a variety of methods and sensors. Some provide a general solution to identifying ramp structures in the scene, but they do not identify the precise location and orientation information about the ramp [6–9].

Rankin *et al.* [6] use geometry and thermal cues to perceive negative obstacles (such as ditches, holes, and other depressions) on cross-country terrain for unmanned ground vehicle autonomous navigation at night.

Cockrell *et al.* [7] use RGB-D point cloud data to generate a gradient height map. The algorithm can identify sudden changes in gradients, and using two threshold values categorizes each gradient cell into three categories: ramps, obstacles, or level floors. The algorithm is able to distinguish between a ramp and a level floor. However, the output does not provide the necessary information for identifying the exact location and the edges of ramps or slopes, or whether the gradients are safe for traversal.

Murarka *et al.* [8], [9] use motion cues, stereo-based scene reconstruction and a color segmentation stereo algorithm to categorize observed terrain into five classes: ground, below ground, above ground, drop-off edge, and unkown. In their algorithm, edges are identified by comparing consecutive stereo image pairs captured by the camera, and the output is affected by the speed and motion of the wheelchair. The remaining classes are determined by setting thresholds on the elevation of cells in the gradient map. Additionally, the color segmentation stereo algorithm can be susceptible to environmental conditions.

The approaches described above provide a binary solution to the problem of ramp detection. Other works simplify the problem by focusing on customized ramp structures [10–12].

As part of designing a climbing tracked robot, Li *et al.* [10] provide an algorithm for detecting ramps. However, the algorithm is specific to their experimental set up (a level plane with a single ramp) and does not generalize to a powered wheelchair planner suitable for urban navigation.

Brossette et al. [11] use RGB-D point cloud data to provide an understanding of a ramp and staircase environment for their humanoid robot. They describe an overview of steps for detecting ramps, however performance numbers and detection rates are not presented.

Detecting an incline plane in the vicinity of the wheelchair is not sufficient for detecting navigable ramps. Moreover, information about the width, location and orientation of the ramp can be used to seed target locations for a motion planner. Furthermore, some approaches are limited in application areas due to their susceptibility to environmental conditions such as ambient light, colors and contrast. In order to identify safe ramps for powered wheelchair users, we need to extend these detection algorithms further.

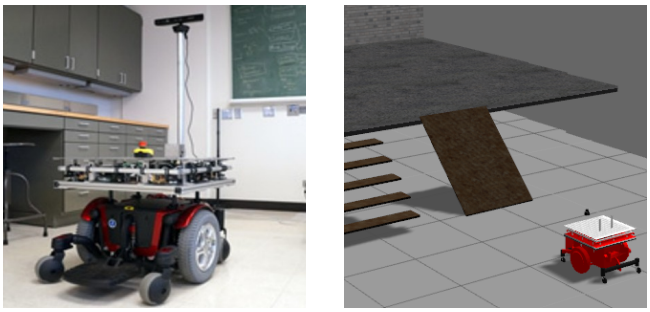

Fig. 1: Our robotic wheelchair system in real hardware (*left*) and the Gazebo simulation environment (*right*).

B. Perception for Robotic Wheelchair Navigation

Perception in robotic wheelchairs is done using an array of different sensors, and for different purposes.

Historically, perception is used for localization, to detect and avoid obstacles, to perceive other humans or identify certain landmarks, and finally vision-based navigation [13–17]. Depending on the application, different sensors are utilized. Demeester *et al.* [13] use front and back laser scanners and wheel encoders to localize the robot and generate planned paths while avoiding any and all protrusions in the path. Trahanias *et al.* [14] use a combination of sonar range sensors and a camera to constantly check for obstacles in the direction of motion. Similarly, Horn *et al.* [15] use a combination of ultrasonic sensors and a camera to locate the wheelchair in a known environment.

Researchers at McGill university conducted surveys to determine powered wheelchair design recommendations from users, which highlighted the ability to detect the edges of ramps and the ability to intelligently adjust the speed and angle to account for the detected ramps [18]. Ensuring safety is an essential role of a smart powered wheelchair in enabling the operator to perform the aforementioned tasks.

Our algorithm not only identifies the exact location of the ramps, but does so without clearing other obstacles, and returns the orientation and dimensions of the ramp, the amount of incline and the length of the ramp. This information is useful for providing navigation goals and controller commands to safely drive the identified ramp. Our work extends the prior state of the art by not only detecting inclined planes in a typical urban setting with non-uniform surfaces, but also determining whether an incline is safe for traversal by a powered wheelchair as specified by the Americans with Disabilities Act (ADA) [19] accessibility guidelines for ramps. In addition, we add artificial obstacles to the side of open ramps to ensure the planners will prevent the wheelchair from getting too close to the edges.

III. Ramp Detection Algorithm

At a high level, our algorithm consists of two steps. The first step is to find inclined planes in the vicinity of the robot. The second step is to determine if such planes are navigable. More specifically, our algorithm first identifies inclined planes, finds the convex hull of the plane and subsequently the four corners that define the incline (Sec.

(a) (b) (c) (d) (e) (f) (g)

Fig. 2: [Best viewed in color.] Illustration of the sequential steps in the algorithm. (a) Point Cloud of the scene with image overlay. (b) Candidate inclined surface extraction (blue region). (c) Convex hull (red dots). (d) Corner locations (red markers). (e) Horizontal edges representing the start and end of the ramp (horizontal red lines). (f) Left and right edges of the ramp (vertical red lines). (g) Identified starting location of the ramp-ascension pose (green marker) and direction of travel (green arrow).

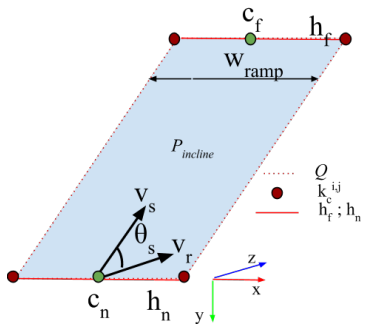

Fig. 3: Diagram illustrating variables used in Algorithm 1. $P_{incline}$: Pre-processed input cloud with convex hull Q. h_n and h_f: Flush transitions of the ramp to landings, with centers at $\boldsymbol{c_n}$ and $\boldsymbol{c_f}$ respectively. $\vec{\boldsymbol{v}}_{\boldsymbol{s}}$ and $\vec{\boldsymbol{v}}_{\boldsymbol{r}}$: Unit vectors in the direction of slope and run, respectively. θ_s: Slope angle.

III-B), and then determines whether the ramp is safe for navigation or not (Sec. III-C).

The input to our algorithm is a continuous RGB-D point cloud stream, ignoring the RGB data. The output of the algorithm is the location of each identified navigable ramp with respect to the camera coordinate system, a set of normal vectors describing the orientation of the ramp plane and the estimated width of the ramp. In the camera coordinate frames, the Y-axis is top-to-bottom and perpendicular to the ground plane, the X-axis is left-to-right, and the Z-axis is back-to-front (Fig. 3). This information is then transformed into the world coordinate frame. Once determined safe for navigation, the points associated with the ramp are removed from the point cloud stream sent to the costmap server, which updates the local and global costmaps. Ramp edges are inflated and kept within the costmap. Information regarding the ramp goal location and orientation is sent to the path planner for safe ramp navigation.

The computation steps used for navigable ramp detection are provided in Algorithm I. Note that we restrict our search to ramp structures that are rectangular in geometry.

Figure 2 illustrates the sequential steps of the algorithm.

A. Data Pre-processing

Working with raw point clouds is costly due to the large number of datapoints. To be useful in real-time, we enhance the performance and speed of the algorithm by preprocessing the incoming stream of point cloud data. We first create a downsampled representation of the point cloud P using the VoxelGrid filter, which is an implementation of an Octree structure; we next remove points that are further than a threshold τ from the sensor (here τ is the sensor-specific threshold after which data becomes unreliable and thus a potential source of error in the next steps; in our implementation τ = 5.0 m). Finally we reduce the noise in P by removing outliers. This significantly improves the runtime speed of the algorithm. In our implementation we take advantage of the libraries and filters provided by the Point Cloud Library (PCL) [20].

B. Finding Inclined Planes

The first step of the algorithm is to identify all of the candidate inclined planes within the vicinity of the robot. In order to identify inclined planes, we first find the normals N for all the points in P. Then we can take advantage of a Region Growing algorithm to merge the points in P that are close enough in terms of the smoothness and curvature constraints imposed on the point normals N (Algorithm 1, line 2). This is an important step, especially when it comes to identifying ramps from real-world data, which can be quite noisy, in addition to the fact that many outdoor ramps often have coarse surfaces. In our implementation we used a smoothness threshold of 5° and a curvature threshold of 2°.

After the clusters are segmented, we use the point normals to find the dominant inclined plane (line 4). The incline plane is fit to the clusters using a Random Sample Consensus (RANSAC) estimator. Any cluster whose point normals are (i) parallel to the ground plane normals, (ii) within an offset angle for plane matching (here we accept offsets up to 5°), and (iii) with an acceptable number of inlier points (here threshold is set to 10) is considered for incline fitting.

Precise knowledge of all the datapoints contained in the plane model is not necessary for further modeling. Similar to [11], after identifying an incline plane, we can further reduce the point cloud to its convex hull Q (line 6) without any loss of important information, ignoring concave polygons. In this way, we can also represent the plane as a structure composed of a normal vector to the plane model, an origin point that is the centroid of the set of points and a small set of points that define its convex hull. After determining that Q has enough points to encapsulate the full plane, we find the four corners of this convex hull (line 7) in clockwise order, starting from the top left corner.

The width w_{ramp} of the inclined plane is estimated using the corner values (line 8). Additionally, we identify the incline plane edges (the horizontal line representing the flush transition from slope run to level planes at the bottom h_n and top h_f of the slope) (line 9). The centroids $\boldsymbol{c_n}$ and $\boldsymbol{c_f}$ of h_n and h_f are calculated (line 10). See Figure 3 for illustration.

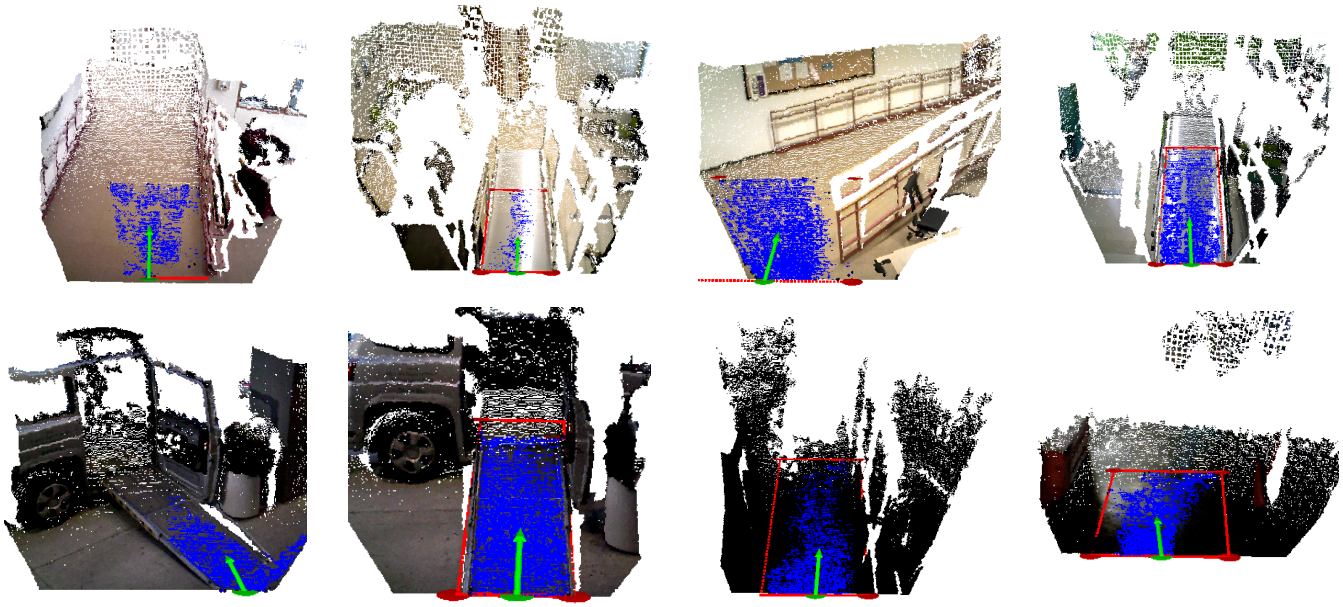

Fig. 4: Examples of the algorithm working correctly on different scenes with a variety of ramps and landing configurations. *Top row:* Indoor ramps. *Bottom row:* Outdoor ramps. Two leftmost images taken during the day; two rightmost images during the evening. The red border around the plane indicates where the algorithm believes the convex hull of the ramp is. The green marker represents a safe starting position for ramp ascent/descent. The green arrow points towards the direction of travel.

Algorithm 1 Incline Detection

1. Given Pointcloud P
   Find Ramp Cloud
2. $C \leftarrow findclusters(P)$
3. **for** $c_i \in C$ **do**
4. $\quad P_{incline} \leftarrow extract\_incline(c_i, N)$
5. **end for**
6. $Q \leftarrow extract\_hull(P_{incline})$
7. $r \leftarrow extract\_rectangle(Q)$
8. $w_{ramp} \leftarrow get\_width(r)$
9. $h_n \leftarrow get\_horizontal\_near(Q)$
   $h_f \leftarrow get\_horizontal\_far(Q)$
10. $\boldsymbol{c_n} \leftarrow get\_center(h_n)$
    $\boldsymbol{c_f} \leftarrow get\_center(h_f)$
11. $\vec{\boldsymbol{v_s}} = \boldsymbol{c_f} - \boldsymbol{c_n}$
12. $\vec{\boldsymbol{v_r}} = < \boldsymbol{c_n}^x, \boldsymbol{c_n}^y, (\boldsymbol{c_n}^z + 1) >$
13. **if** $\theta_s = arccos(\frac{\vec{\boldsymbol{v_s}} \cdot \vec{\boldsymbol{v_r}}}{||\vec{\boldsymbol{v_s}}|| ||\vec{\boldsymbol{v_r}}||}) < \theta_{max}$ **and** $w_{ramp} > w_{min}$
14. $\quad P_{ramp} \leftarrow P_{incline}$
15. **end if**
    Extract Side Edges
16. $S_r \leftarrow get\_right\_edge(Q)$
    $S_l \leftarrow get\_left\_edge(Q)$
17. $R \leftarrow (P_{ramp} - S_r - S_l)$
18. return $P - R$

Key: $||\cdot||$ = distance (Euclidean), h_n = nearest horizontal edge, h_f = furthest horizontal edge, θ_{max} = max allowable slope (5°)

C. Finding Navigable Ramp

Our determination of what constitutes a traversable ramp is informed by the 405 Americans with Disabilities Act (ADA) [19] wheelchair accessibility guidelines for ramps. According to these guidelines, the running slope must not exceed 1:8 (i.e. 7.13°) [12]. This means that for every inch of height change, there should at least be 8 inches of ramp run. To check for this, we define two vectors (lines 11-12): vector $\vec{\boldsymbol{v_s}}$, from the center of the closest horizontal edge to the furthest horizontal edge of Q (the ramp *slope*); and vector $\vec{\boldsymbol{v_r}}$, the unit vector pointing from the center of the closest horizontal edge of Q projected forward in the Z-axis direction (the ramp *run*). If the angle θ_s between $\vec{\boldsymbol{v_s}}$ and $\vec{\boldsymbol{v_r}}$ is less than $\theta_{max} = 7.13°$, then the slope meets the ADA specifications.

The 405 ADA guidelines also specify the required ramp width (w_{min}) to be at least 36 inches (91.5 cm). If the convex hull Q meets both the slope requirements and width specifications, then we can mark this cloud cluster $P_{incline}$ as a traversable ramp P_{ramp} (lines 13-15). Once a ramp cloud is identified, we can use Q to define a rectangular region for the ramp plane. The position, orientation, and width of this region is stored and sent to the path planner and controllers.

Finally, the side edges of P_{ramp} are identified (line 16). Left and right stripes describing the inflated edges of the ramp are then removed from P_{ramp} (line 17), and the remaining points are removed from the point cloud (line 18). This point cloud is cleared from the costmap, thus safely clearing the ramp from the list of observed obstacles.

IV. Experiments

The algorithm was developed for the wheelchair platform in Figure 1. This platform is equipped with a suite of sensors, from RGB-D to laser range finders to ultrasonic sensors. Perception capabilities include open doorway [4] and safe

docking [5] location detection. A crucial next step to support further development in enhancing the 3D perception capabilities of the system for situations where robotic assistance is beneficial and safe includes autonomous ramp detection. We use the RGB-D sensor as the source of perception input, which provides a large amount of information about the scene at a continuous rate.

A. Data Collection and Methods

The data used for evaluation consists of a variety of ramp-structures located around the Rehabilitation Institute of Chicago (RIC) and the city of Evanston, IL, USA. The ramps used in validation consists of structures commonly encountered by powered wheelchair users in urban settings, including indoor ramps, van ramps, small outdoor ramps, long outdoor ramps, wide ramps and narrow ramps.

The performance of the algorithm is tested at different angles (ranging from 0°-90°) and distances (ranging from 0-2.5m) to the ramps. For evaluation purposes, we also include data that contains no ramps to ensure the algorithm does not give false positives.

Our experiments to test the ramp detection algorithm are carried out on a PC connected to an ASUS Xtion Pro sensor. The PC has an Intel Core i5 processor, 8GB of RAM, and was running Ubuntu 14.04, ROS Indigo, the OpenNI driver (version 0.2.2), and the Point Cloud Library (version 1.7.1).

A ramp is considered correctly classified if the algorithm estimated the four corners of the ramp (indicated visually with red markers) and the ramp edges, and returns a safe starting point and direction arrow for driving on the ramp (indicated visually with a green marker and green arrow). Each configuration is evaluated for at least 15 frames.

B. Experimental Results

The number of correctly classified frames from each data input stream was manually counted by reducing the frequency of the data to 10 Hz. Once the data was collected and tallied, the statistics were confirmed by running the data at full speed, approximately 530 Hz.

The results are discussed in terms of the distance to the ramp and the offset angles, and the external environment (indoor or outdoor) at detection. Figure 4 presents a number of screen shots from correctly classified ramps. We also present some examples of misclassification in Figure 5.

1) Overall Performance: Table I summarizes the results of the true and false positive detection rates, for 10 runs of the algorithm.

Overall, the detection rates were quite good, with an average 88% accuracy rate for the full dataset collected. In each of the cases, the algorithm accurately detected the ramps for the majority of the frames, with very few incorrectly classified ramps and no missed ramps.

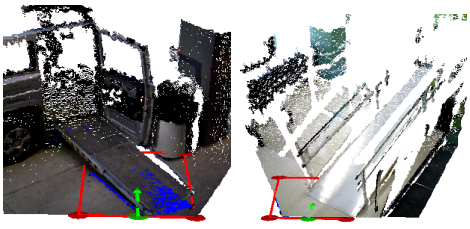

Fig. 5: False positives. *Left:* Outdoor van ramp. *Right:* Indoor ramp.

2) Detection Distance: The results showed that the effective range of the algorithm is approximately 0 to 1.8 meters from the ramp. The ability of the algorithm to reliably segment the ramp planes becomes significantly less reliable the farther away the sensors are from the ramp.

3) Detection Angle: The effect of angle on detecting the ramp varied according to the ramp structure. For ascending ramps, if the sides of the ramp were not occluded by walls, the algorithm was very successful in detecting the ramps for all measured offset angles, with the lowest rate of detection about 87%. However, for ascending ramps with offset angles greater than 75° we start to see a decline in the accuracy of the estimated ramp corners. For ascending ramps with occluded edges (brick walls, etc.), the ability of the algorithm decreases as the angle increases from 45° (head on) to 75°. For descending ramps, angles greater than 75° hindered the algorithm's ability to accurately identify ramps, and resulted in missed detections. As the sensor is oriented more directly in front of the ramp, the accuracy increases significantly.

When looked at straight on, the rates of detection for most ramps were very high, with the exception of the sidewalk ramp. The difficulty in detecting this could be attributed to the curved surface and the lack of a well defined outline. The lowest detection rate was around 66% in this case. For planning purposes, a detection rate of 66% is still useful.

4) Detection Setting: The accuracy of detecting indoor ramps was much higher than outdoor ramps. The accuracy of detection for outdoor cases furthermore increased when there was less sunlight (i.e. evening, night). The worst scenario was detecting outdoor ramps covered in snow at noon, where the infrared reflections obstructed the sensor readings.

The accuracy of detecting the van ramp was very high (average of 96%) even though it was parked outdoors. This can be attributed to the sides of the ramp being unobstructed by walls, such that the ramp is clearly visible and identifiable from a range of angles on either side. Additionally, the surface of the van ramp was clean and non-reflective. The van data also was captured during a less sunny day.

V. CONCLUSIONS AND FUTURE WORK

We have introduced in this paper a novel method for the autonomous detection of safely traversable ramps using 3D point cloud data, without any visual fiducial or environment customization requirements. Through evaluations on different ramp structures in varying configurations and environmental settings, our algorithm has demonstrated good performance and was shown to be effective in the identification of the orientation and location of safe traversable ramps, and safe starting poses for driving up the ramps. Finding traversable ramps is important in the context of assistive robotic wheelchairs, and also in the broader area of autonomous mobile robots. By detecting traversable ramp locations, along with alignment information, custom trajectories can be planned and executed by a path planner and

TABLE I: Performance evaluation at varying offset angles and distances to a variety of ramp structures.

Structure	Offset Angle (°)	Distance (m)	Frames	True Positives #	True Positives %	False Positives #	False Positives %
Indoor Ramps (ascending)	0	0.0	15	15	100.0	0	0.0
	10	0.5	15	15	100.0	0	0.0
	20	2.0	15	14	93.3	0	0.0
	30	0.8	15	15	100.0	0	0.0
	45	0.8	15	14	93.3	1	6.7
	45	1.8	15	13	86.7	2	13.3
	90	1.0	15	10	66.7	5	33.5
Indoor Ramps (descending)	0	0.0	15	15	100.0	0	0.0
	45	0.8	15	14	93.3	0	0.0
	75	0.5	15	13	86.7	2	13.3
Outdoor Ramps (ascending)	0	0.5	30	29	96.7	0	0.0
	0	2.5	30	20	66.0	1	3.3
	20	0.0	30	28	92.4	0	0.0
	30	0.5	30	28	92.4	1	3.3
	45	0.8	30	22	72.6	2	6.6
	45	1.8	30	20	66.0	6	19.8
Outdoor Ramps (descending)	0	0.5	15	14	93.3	1	6.7
	45	1.8	15	12	80.4	2	13.3
	45	2.5	15	10	66.7	4	26.6
Van Ramp	0	1.8	30	15	100.0	0	0.0
	45	1.5	30	29	96.7	1	3.3
	45	2.0	30	28	92.4	2	6.6
Mean					87.9%		7.1%

controller to comfortably and safely drive up ramps, which can be a difficult and dangerous task if using binary control interfaces such as SNP devices. This work was done with the aim of benefiting assistive wheelchair technologies by realizing the difficult problem of ramp traversal.

Future work includes dealing with differently shaped top and bottom ramp landings, as well as dealing with infrared light during daylight outdoor navigation on clear days, as this accounts for most of the errors in our perception algorithm.

ACKNOWLEDGMENT

The authors thank Jarvis Schultz for discussions during the algorithm development.

REFERENCES

[1] K. Kulig, S. S. Rao, S. J. Mulroy, C. Newsam, J. K. Gronley, E. L. Bontrager, and J. Perry, "Shoulder joint kinetics during the push phase of wheelchair propulsion," Clinical Orthopaedics and Related Research, vol. 354, pp. 132–143, 1998.

[2] "Wheelchair ramp navigation." [Online]. Available: http://blog.rollmobility.com/2010/08/23/wheelchair-ramp-navigation/

[3] S. Boucher, "Obstacle detection and avoidance using turtlebot platform and xbox kinect." Rochester Institute of Technology, Tech. Rep., 2012.

[4] M. Derry and B. Argall, "Automated doorway detection for assistive shared-control wheelchairs," in Proceedings of the IEEE International Conference on Robotics and Automation, 2013.

[5] S. Jain and B. Argall, "Automated perception of safe and oriented docking locations with alignment information for assistive wheelchairs," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014.

[6] A. Rankin, A. Huertas, and L. H. Matthies, "Nighttime negative obstacle detection for off-road autonomous navigation," Unmanned Systems Technology IX, 2007.

[7] S. Cockrell, L. Gregory, and W. Newman, "Determining navigability of terrain using point cloud data," in Proceedings of the IEEE International Conference on Rehabilitation Robotics, 2013.

[8] A. Murarka, M. Sridharan, and B. Kuipers, "Detecting obstacles and drop-offs using stereo and motion cues for safe local motion," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008.

[9] A. Murarka and B. Kuipers, "A stereo vision based mapping algorithm for detecting inclines, drop-offs, and obstacles for safe local navigation," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

[10] C. Tseng, I. Li, Y. Chien, M. Chen, and W. Wang, "Autonomous stair detection and climbing systems for a tracked robot," in Proceedings of the International Conference on System Science and Engineering, 2013.

[11] S. Brossette, J. Vaillant, F. Keith, A. Escande, and A. Kheddar, "Point-cloud multi-contact planning for humanoids: Preliminary results," in Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics, 2013.

[12] C. Lutz, F. Atmanspacher, A. Hornung, and M. Bennewitz, "NAO walking down a ramp autonomously," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012.

[13] E. Demeester, E. V. Poorten, A. Huntemann, and J. D. Schutter, "Wheelchair navigation assistance in the FP7 project RADHAR: Objectives and current state," in The IROS 2012 workshop on Navigation and Manipulation Assistance for Robotic Wheelchairs, 2012.

[14] L. Trahaniaa and S. Argyros, "Vision-based assistive navigation for robotic wheelchair platforms." [Online]. Available: http://www.academia.edu/12411515/vision-based_assistive_navigation_for_robotic_wheelchair_platforms

[15] O. Horn and M. Kreutner, "Smart wheelchair perception using odometry, ultrasound sensors, and camera," Robotica, vol. 27, no. 02, pp. 303–310, 2008.

[16] A. Siadat, K. Djath, M. Dufaut, and R. Husson, "A laser-based mobile robot navigation in structured environment," in in Proceedings of the European Control Conference, 1999.

[17] A. M. Sabatini, V. Genovese, E. Guglielmelli, A. Mantuano, G. Ratti, and P. Dario, "A low-cost, composite sensor array combining ultrasonic and infrared proximity sensors," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1995.

[18] D. Kairy, P. Rushton, P. Archambault, E. Pituch, C. Torkia, A. E. Fathi, P. Stone, F. Routhier, R. Forget, and L. Demers, "Exploring powered wheelchair users and their caregivers perspectives on potential intelligent power wheelchair use: A qualitative study," International Journal of Environmental Research and Public Health, vol. 11, no. 2, pp. 2244–2261, 2014.

[19] "American's with disabilities act standards for accessible design, 2010," http://www.ada.gov/2010ADAstandards_index.htm.

[20] "Point cloud library," http://docs.pointclouds.org/trunk/modules.html.