Automated Doorway Detection for Assistive Shared-Control Wheelchairs

Matthew Derry[1] and Brenna Argall[2]

Abstract— **This work presents an algorithm for rapid, automated detection of open doorways using 3D point cloud data. The algorithm has been developed in the context of shared-control powered wheelchairs in which adaptive assistance is provided to individuals who otherwise might not possess the fine motor control necessary to handle potentially challenging activities, such as doorway traversal. In this context it is important to go beyond the 2D laser scanner for open doorway detection, for both safety reasons as well as opportunities for improved shared-control behavior development. We evaluate the doorway detection by systematically testing the performance on several doors and door configurations from varied view points, using point clouds generated by a Microsoft Kinect.**

I. Introduction

Spasticity, paresis, and tremor in the upper limbs are just a few of the diagnoses that can limit the prescription of a powered wheelchair to patients due to insufficiently accurate steering commands. With self-controlled mobility being so important to quality of life, it is critical to continue to develop enabling solutions for this class of patient.

One candidate solution is the use of assistive robots employing a shared-control paradigm, where navigation planning and steering control is shared between the powered wheelchair and the user. To support the development of a shared-control paradigm for assistive wheelchairs, a first step is to enhance the 3D perception capabilities of the system, so that it is able to perceive the situations in which navigation assistance would be beneficial and safe.

To that end, this work presents a system able to identify a very common navigational challenge in driving a powered wheelchair: traversing doorways. Locating these doorways allows for navigation planning that can provide both customizable and smoother trajectories through doorways based on the desired goals of the human. For instance, the NavChair has a door passage mode that acts to center the NavChair within a doorway, then steering the chair through [1].

Previous work in automating wheelchairs largely relies on *a priori* knowledge of the environment or engineered environments in order to identify points of interest to the human, including doors and doorways. This information is used in different ways, from navigation planning [2], [4] to estimation of the human's intent [3], [5]. A goal of our work is to facilitate operation in novel environments, thus lifting the requirement of an *a priori* map, fiducial markers, or specialized sensor setups. As such, our algorithm uses 3D point cloud data from a Microsoft Kinect and doorway size constraints specified in the American's with Disabilities Act (ADA) [21], to identify doors without prior knowledge of the environment or fiducial markers.

The choice to focus on open doorways is motivated by the fact that in most situations wheelchairs are not expected to include a robotic arm and the wheelchair is meant to only assist locally when a situation arises in which assistance would be beneficial. Without a manipulator on the wheelchair to handle door opening, in terms of navigation it is less important for a closed door to be perceived by the wheelchair.

It is important to go beyond just looking for gaps in a 2D laser scan that are wide enough for the user to pass through, because not all gaps that meet this criterion would be considered points of interested by a human (e.g. if the gap didn't actually have vertical clearance for the wheelchair), so already there would be disagreement about possible targets in the nearby vicinity. Additionally, being able to identify the open doorway location and orientation is information that can be incorporated into intelligent behaviors, for example that consider multiple target locations instead if just going straight through the door.

In this paper, we present an algorithm for detecting open doorways using 3D point cloud data. The result is a rapid and robust detection of the location and orientation of doorways through which a wheelchair can pass. This algorithm uses only the depth data in the point cloud provided by the sensor, so it is less prone to being affected by changes in lighting conditions. The remainder of the paper is formatted as follows. Related work is overviewed in Section II, and our algorithm for open doorway detection is presented in detail in Section III. Section IV describes our validation approach while Section V presents the experimental results. The paper is concluded with Section VI along with some directions for future work.

II. Related Work

Here we overview the background literature in both automatic doorway detection and shared control for wheelchairs that supports this work.

A. Doorway Detection

Automated doorway detection has been accomplished through a variety of sensor technologies. A single camera detects door frames [7], by first filtering (Gaussian and Sobel) and transforming (Hough) the image, and then searching for horizontal lines to confirm the door frame and identify the state of the door. Performance numbers are not provided,

[1]M. Derry is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA mderry@u.northwestern.edu

[2]B. Argall is jointly appointed with the Faculty of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA and the Rehabilitation Institute of Chicago, Chicago, IL 60611, USA brenna.argall@northwestern.edu

but vision in this context can be susceptible to environmental conditions, such as ambient lighting, colors, and contrast. A similar processing approach transforms instead a point cloud from a Kinect into a 2D depth map, wherein the depth values are transformed in to the RGB space [15]. Once this 2D map has been calculated, an image discontinuity map is created, which highlights edges where a sufficient discontinuity in depth exists, and the Hough transform is applied to look for two vertical lines of a certain length connected by a horizontal line at the top. The reported detection rate of this approach was 69%.

Two 2D laser scanners are used in conjunction with a panoramic camera to capture shape, color, and motion properties between frames in Anguelov et al. [6]. This data is then passed to a probabilistic model that segments the environment into door and wall objects and learns their properties, such as color, which direction the door opens, and the current state of the door. In this work, the self-pose estimates are provided to the robot by an external software system, the map is known, and the door detection success rate is 84%.

As part of the work by Rusu et al., using only depth data from the 3D point clouds doors were located, with the eventual goal of opening or closing them [10]. In this work the focus was on the door, and not the doorway, and so once the robot was placed near the door, their technique applied geometric constraints (defined by the dimensions of a typical door) to a plane model and then attempted to fit that plane model in the point cloud. The best matches were deemed to be doors. Several algorithms in this work that were used for tasks such as wall segmentation are similar to the algorithms used in our work.

More recently, promising object detection algorithms based on RGB-D input have been developed in the context of detecting people in a 3D scene [11], [12]. These algorithms are largely based on learning classifiers for objects using histograms of oriented features, such as depth gradients or surface normal vectors as inputs to the classifier. These are extensions to 3D of the Histogram of Oriented Gradients (HOG) approach presented in [13]. Additionally, template matching of a human head model is used for human detection in [14]. In contrast, our proposed algorithm uses subsampling that is informed by geometric contraints provided by doorways instead of applying specific templates or learned models to identify the doorway.

B. Shared-Control Paradigms for Wheelchair Navigation

Here we present a handful of shared-control wheelchair systems, and how the methods of perception they use enable or enhance the shared-control behaviors.

A well-known example of shared-control being used to improve the driving capacity of humans in wheelchairs is the NavChair [1]. A number of modes of assistance are provided, including door passage. In this mode, the user drives the chair towards a door, then activates the door passage mode, and data from a ring of sonar sensors are used to attempt to center the wheelchair within the door as the user drives it through. This approach is very effective when set up facing the doorway directly, but if the user activates the door passage mode while at a significant angle, the algorithm will actually drive the chair away from the door in an attempt to center on the door, because the doorway is not actually actively detected; instead, an assumption is made about the doorway being directly ahead.

Zeng et al. developed a system called the Collaborative Wheelchair Assistant [2]. In this system, a wheelchair is equipped with wheel encoders for dead reckoning and a bar code scanner for global positioning. In this system, a helper can program a path by demonstration and then the user can drive this path controlling the speed, while the chair enforces the direction. In this system there are no sensors to detect collisions and in tight spaces such as doorways, it relies on the driving skill of the user and the accuracy of the localization to make it through the doors. In the evaluation, only one doorway was tested and the path to traverse it was pre-set before the performance was tested.

A system that provides adaptive assistance through the prediction of user intent while driving a wheelchair is presented by Carlson and Demiris [3]. In this work, the system operates in a set of rooms with 3 doorways, equipped with fiducial markers at regular intervals for localization and provided with a map *a priori*. As the user drives the wheelchair around, each target doorway is given a confidence score based on Euclidean distance to the goal and the heading of the chair compared to the angle of the target. Once this confidence score crosses a threshold, the automation steps in to assist the user in sticking to a safe trajectory through the door. An extension of this system to incorporate a mechanism for automated doorway detection could potentially lift the dependence on an *a priori* map and markers.

III. Open Doorway Detection Algorithm

We now present our algorithm for the automated detection of open doorways. Our approach acts directly on the depth (D) portion of the RGB-D point cloud data provided by the Kinect, ignoring the RGB data for everything except visualization purposes. In using only the depth data, the influence of ambient lighting changes on the performance of the algorithm is minimized. At a high level, the algorithm scans for walls in a scene. For each wall detected, a horizontal stripe of the wall is scanned for gaps that would be within the width range of a door. When a gap of the appropriate width is identified, a count is made of the number of points within a space bounded by the gap location, wheelchair (or expected door) height, wall depth and ground plane This constrained volume should contain very few points if it is an open doorway (it should actually contain zero points, but a small threshold is used to accommodate sensor noise). Once a door is detected, the algorithm then provides 3D annotations of open doorway locations from the geometric information captured from the point cloud.

The output of the algorithm is the location of each identified open doorway with respect to the camera coordinate system, the normal vector indicating the orientation of the

doorway, and its width. An important note about the camera coordinate frame is that the Y-axis is perpendicular to the ground, the Z-axis is back to front, and the X-axis is left to right. This information can be transformed onto the world coordinate frame and used for map updates.

Algorithm 1 presents the primary computational steps used for doorway detection. First, a note on parameters. Parameter α is a scaling factor that sets a lower bound on what size point cloud is considered too sparse to search for walls (line 2). Parameters h_d and h_s are the expected height of the doorway (line 18) and the height at which to scan the wall (line 7), respectively. The parameter δ defines the nearest neighbor search radius during the wall stripe scan (line 7).

Wall Detection: The first step of the algorithm improves the runtime speed by taking the input point cloud and, using an Octree structure [17], creates a down-sampled representation P. The down-sampled point cloud is then searched for the dominant wall, P_{wall}, which is identified by using a Random Sample Consensus (RANSAC) estimator [16] to fit a plane model ($ax + by + cz + d = 0$) in which the plane is parallel to the camera Y-axis (which is also perpendicular to the ground plane). The normal for P_{wall} is then calculated. The inliers of the model found are projected onto the plane (line 4). From these, a stripe of points constrained in the Y-axis are extracted (line 7).

Stripe Scan: The region of the stripe being scanned is defined by end points p_{curr} and p_{end}. These are initialized to the point on the wall with the minimum x value and the maximum x value, respectively (line 9). The scan continues along the plane of the wall (line 25), at height h_s. At each step along the scan, a count of the neighboring points within a radius δ is performed (lines 11-12). When a candidate region is found for which there are 0 neighboring points and the previous region had neighboring points, the scan logs the current point as a gap start and then continues scanning (line 14). The next time a region is found for which the neighbor count is greater than 0, the previous region is stored as the end of the gap (line 16).

Doorway Detection: Using the ADA accessibility guidelines for door widths, if the gap is between 82 cm ($\tau_{w_{min}}$) and 164 cm ($\tau_{w_{max}}$) in length, then the gap end points, and expected doorway height, are used to establish a rectangular region in the plane of the wall, P_{wall}; which should contain no points if the region is indeed an empty doorway. The projected wall point cloud P_{wall} is searched within the new boundary space for occupied points (line 18) and if a sufficiently small number of points are found within the search volume (to account for noise in the sensor), then a door is found and a structure indicating the position, orientation, and width of the door is created and stored (lines 20-21). Once the scan is complete for the current wall, P_{wall}, then P_{wall} is removed from P (line 27) and the new, smaller P is searched for the next best wall match, until there are no remaining wall matches found or the number of points in P falls below some threshold N_P. The set of detected doors D then is returned.[1]

Algorithm 1 *Open Doorway Detection*

1: Given *pointcloud* P
2: *initialize* $D \leftarrow \{\}, N_P \leftarrow \alpha \cdot |P|$
3: **while** $N_P < |P|$ **do**
4: Find dominant plane within P (normal: $\langle n^x, n^y, n^z \rangle$). Extract nearby points $P_s \subseteq P$. Project onto plane ($\rightarrow P_{wall}$).
5: *initialize* $P_{stripe} \leftarrow \{\}$
6: **for** each $p_i \in P_{wall}$ **do**
7: $P_{stripe} \leftarrow \{P_{stripe} \cup p_i\}$ if $p_i^y \in [h_s - \delta, h_s + \delta]$
8: **end for**
9: *initialize* $N_{prev} \leftarrow 1$
$p_{curr} \leftarrow \underset{p_i \in P_{wall}}{\texttt{argmin}}\ p_i^x$
$p_{end} \leftarrow \underset{p_i \in P_{wall}}{\texttt{argmax}}\ p_i^x$
10: **while** $\tau_{done} < \|p_{curr} - p_{end}\|$ **do**
11: $\leftarrow$ sphere centered at p_{curr} with radius δ
12: $N_{curr} \leftarrow |\{p_i\} \in s|,\ \forall p_i \in P_{stripe}$
13: **if** $N_{curr} = 0$ and $N_{prev} > 0$ **then**
14: $p_c \leftarrow p_{curr}$
15: **else if** $N_{curr} > 0$ and $N_{prev} = 0$ **then**
16: $p_e \leftarrow p_{curr}$
17: **if** $\|p_c - p_e\| \in [\tau_{w_{min}}, \tau_{w_{max}}]$ **then**
18: $r \leftarrow$ rectangle with corners (p_c^x, h_d, p_c^z) (p_e^x, h_d, p_e^z) $(p_c^x,\ 0\ , p_c^z)$ $(p_e^x,\ 0\ , p_e^z)$
19: **if** $|\{p_i\} \in r| = 0,\ \forall p_i \in P_{wall}$ **then**
20: $door \leftarrow \{$ position= $(\frac{p_c^x + p_e^x}{2},\ 0\ , \frac{p_c^z + p_e^z}{2})$
orienation= $\langle n^x, n^y, n^z \rangle$
width= $\|p_c - p_e\|$ $\}$
21: $D \leftarrow \{D \cup door\}$
22: **end if**
23: **end if**
24: **end if**
25: $p_{curr} \leftarrow p_{curr} + \langle n^z, 0, -n^x \rangle$
26: **end while**
27: $P \leftarrow P \setminus P_w$
28: **end while**
29: **return** D

Key: $|\cdot|$ = size (number of elements), $\|\cdot\|$ = distance (Euclidean)
h_s = stripe scan height, h_d = door height, δ = search parameter
τ_{done} = termination threshold (distance), $\alpha \in (0, 1)$
$\tau_{w_{min}}, \tau_{w_{max}}$ = min/max thresholds on doorway width

Finally, we note that to extend this approach to identify and reject a partially opened door would be trivial, despite the fact that the door would not show in the wall plane projection. The search volume could simply be extended—in the direction of the normal of the doorway—by some portion of the length of a door. A fast-running search can be implemented using simple pass-through filters on the point cloud with the appropriate boundaries.

IV. Experimental Methods

This section describes the validation approach used to evaluate the effectiveness and accuracy of the doorway detection in real-world situations. Experiments to validate

[1] In our implementation, the following parameter values were arrived at experimentally: $\alpha = 0.4$, $h_s = 1$ m, $h_d = 2.13$ m, $\delta = 0.02$ m, $\tau_{done} = 0.01$ m.

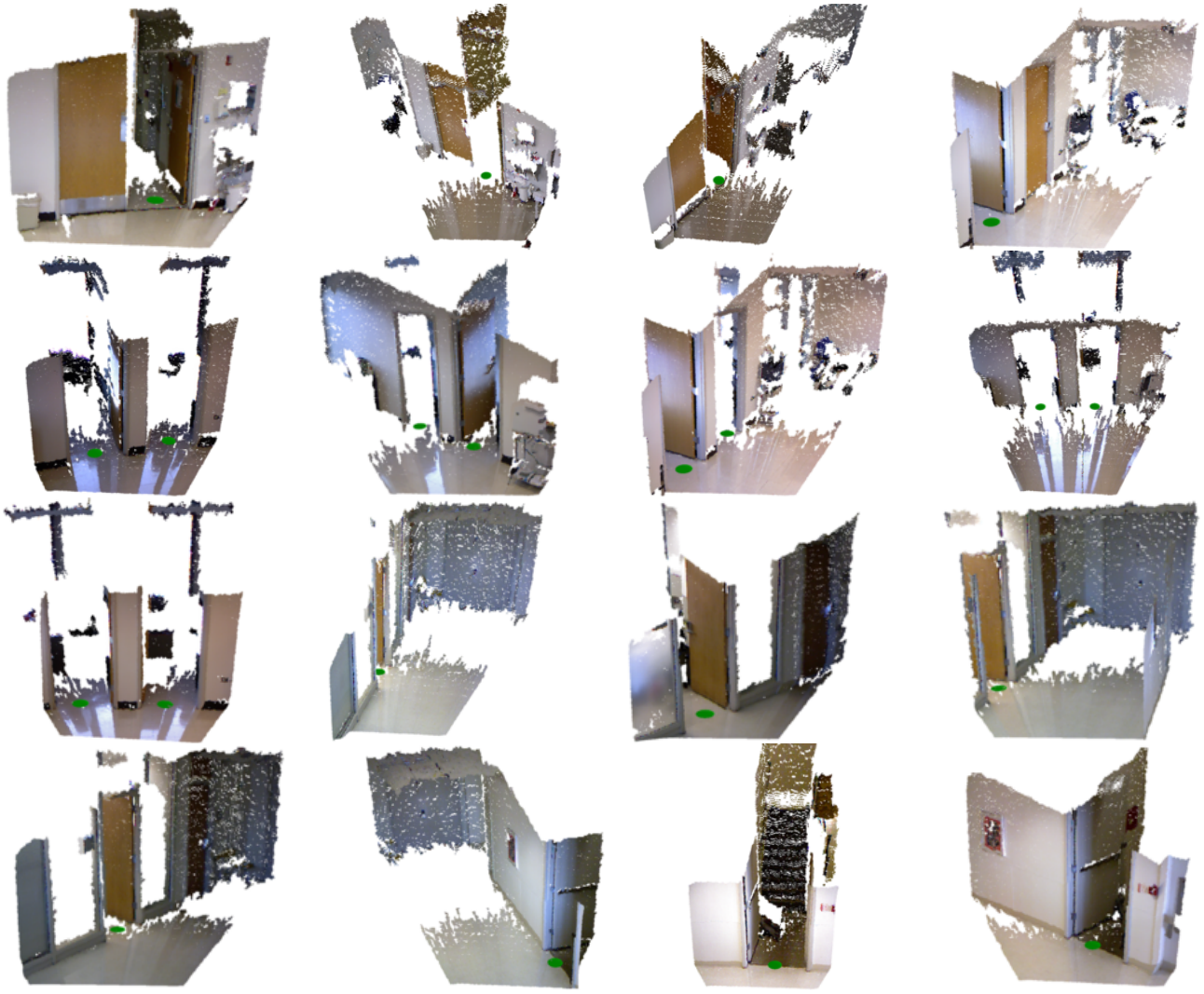

Fig. 1. Sixteen examples of the algorithm working correctly on different scenes with a variety of doorways and doorway configurations. The green dot on the ground plane indicates where the algorithm believes the center of the doorway is.

the doorway detection algorithm were carried out on a PC with a Microsoft Kinect sensor connected to it. The PC had an Intel Core i7 Quad-Core processor, 8GB of RAM, and was running Ubuntu 12.04, ROS (Robot Operating System) Fuerte [18], Openni Kinect Driver, and the PCL (Point Cloud Library, version 1.6.0) [19].

A systematic approach to collecting data on doorways in several different locations around the Rehabilitation Institute of Chicago (RIC) was used in order to characterize the performance of this algorithm. For the doorway configurations evaluated, each was tested at several different distances and angles ranging from 0 degrees offset, to 75 degrees offset.

The algorithm was validated on five different doorway configurations. Each doorway configuration was chosen to test different cases that might be commonly encountered, such as a single doorway, two doorways in the same scene, a wider doorway, doorways along a hallway, and doorways with non-uniform frames, such as a double door with one door shut; see Figure 1. In the case with two doors both open, each door was tallied individually and the mistakes were not systematically biased towards one door or the other.

During this process, the point cloud stream was throttled to 1 Hz to facilitate online manual counting of correctly identified doorways, falsely identified doorways, and missed doorways. A doorway was considered correctly identified if the system estimated the doorway center (indicated visually by placing a green marker on the ground plane) as roughly midway between the two vertical edges of the open doorway. Each doorway configuration was evaluated over 15 frames and then the experimental setup was reconfigured to the next distance and angle.

Once this data was collected and tallied, the throttle on the point cloud stream was removed and the statistics were confirmed running at full speed, approximately 7.5 Hz.

V. Experimental Results

In each of the described cases, the algorithm performed quite well and succeeded in detecting the open doorways, with relatively few missed doorways or incorrectly identified

Door Type	Offset Angle (°)	Distance (m)	Number of Doors	True Positives		False Positives	
				#	%	#	%
One single door (glass wall)	20	1.8	15	14	93.3	0	0
	40	2.2	15	15	100	0	0
	50	3.5	15	15	100	0	0
	60	4.2	15	9	60.0	1	6.7
	75	3.2	15	10	66.7	1	6.7
One single door (white wall)	0	2.3	15	15	100	0	0
	40	2.8	15	14	93.3	0	0
	60	2.5	15	10	66.7	0	0
Two single doors (both open)	0	2.7	30	30	100	0	0
	0	4.2	30	29	96.7	0	0
	45	3.5	30	30	100	0	0
	60	2.7	30	29	96.7	1	3.3
Two single doors (one closed)	0	2.7	15	15	100	2	13.3
	0	4.2	15	14	93.3	0	0
	45	3.5	15	15	100	0	0
	60	2.7	15	13	86.7	1	6.7
Double doors	0	3.1	15	15	100	0	0
	20	3.1	15	14	93.3	1	6.7
	50	3.1	15	15	100	1	6.7

TABLE I

THE DETECTION RATES FOR TRUE AND FALSE POSITIVES AT VARYING OFFSET ANGLES AND DISTANCES FOR EACH DOORWAY SCENE.

doorways. Table I presents the true positive and false positive detection rates for each of these doorway configurations, and the various distances and angles at which they were evaluated. Figure 1 presents a number of examples of the algorithm at work for each of the configurations. Lastly, Figure 2 presents some examples of doorway misidentification.

The effective range of the algorithm proved to be approximately 1.5 m to 4.2 m. Outside of this range the wall segmentation became less reliable: from the smaller samples of wall planes in the near case, and from decreasing accuracy in the sensor's distance calculations in the far case. Increasing the range between the doorway and sensor also magnified differences between the camera reference frame and the world frame; caused by the motion of the robot platform on which the sensor is mounted. This operational range is useful and appropriate for the intended task of assisting wheelchair users through doorways, as the initial target wheelchair platform has a top speed of 2.2 m/s, which would allow the algorithm to detect the doorway up to 7 or 8 times while operating at 7.5 Hz. This allows for detection and confirmation while moving at top speed. Once closer than 1.5m, it is expected that an additional, fast-responding, sensor or suite of sensors will handle perception for collision avoidance and driving execution.

Overall, detection rates were quite good, at better than 90% for the full dataset collected. The biggest contributing factor to missed doorways was the angle at which the doorway was detected. When looked at straight on, open doorways were detected in the high 90% range, as the angle grew more extreme open doorway detection became less reliable with only 60% detection at 75 degrees off straight on. This was the lowest measure recorded, and 60% detection can still be useful for planning purposes.

Beyond 75 degrees, the algorithm could no longer determine if the doorway was open, closed, or not even a doorway at all (such as a slightly recessed window), and as such would generate many false positives. As this is due to the occlusions present at these high angles, this was considered to be of relatively low import, since the scene is constantly changing for a mobile robot. Eventually, the robot would likely reach a pose that would allow the algorithm to correctly detect the doorway. If the false positives prove to be an issue in the future, then this high-angle situation could be identified and door detections repressed due to uncertainty. Within the angle constraint, the false positive points identified as open doorways were seldom seen at only 2.3% over the entire dataset. Upon further inspection, these false positives often stemmed from noise in the Kinect sensor or from incorrectly identifying the wall plane.

Within these angle/distance constraints, there were a few situations which were not able to be handled by this algorithm. These can be broken into groups: sensor-related deficiencies, situational deficiencies, and algorithm-related deficiencies. The sensor-related deficiencies are beyond the scope of this algorithm, such as floor to ceiling glass walls which are undetectable by the projected-IR light technique used by the Kinect. These must be handled using a different sensing modality that is not susceptible to the same deficiencies, such as ultra-sonic range finders. The situational deficiencies are caused by occlusions. These can come up in places like hallways, where the angle and distance from an opening in the wall may initially be detected as a doorway, when in reality it may be a closed door that is set back a few centimeters from the detectable wall plane and obstructed from view by the door frame. This gets handled as the robot moves closer to the door, and the angle and distance becomes sufficient for detection; but it can lead to an initial false positive doorway identification. The final group of

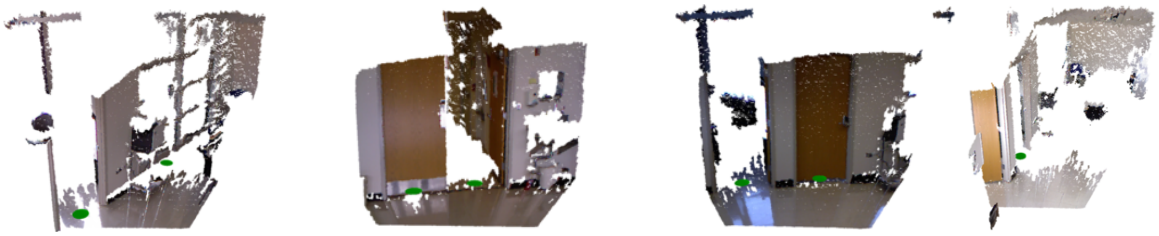

Fig. 2. Examples of the algorithm finding false positive doors (1st three images, left to right), and a false negative (rightmost image).

deficiencies are algorithm-related. Largely, they stem from the wall detection portion of the algorithm. In order to achieve good run-time speed, the RANSAC algorithm is used to do the plane-fitting. This is a non-deterministic plane fitting algorithm and as such, can occasionally lead to somewhat strange wall selections. This will sometimes result in missed doorways, or in a false positive identification of a doorway. With overall detection rates greater than 90%, and false positive rates less than 3%, this was deemed to be an acceptable trade off for the increased run-time speed.

Since this algorithm is intended for use on a shared-control powered wheelchair, it is important for the algorithm to work while in motion as well. To that end, a series of videos were recorded demonstrating the performance of the algorithm while in motion.[2] These videos were not analyzed to the same extent as the above results, but the performance appears consistent with the observed statistics.

VI. Conclusions and Future Work

This paper presents a fast, robust method for detecting open doorways using 3D point cloud data. Going beyond the 2D laser scanner is important, particularly when height and ground clearances need to be confirmed if the robot platform is to be carrying human passengers, as in the case of a shared-control wheelchair. Additionally, by detecting the location and orientation of the doorway, custom trajectories can be planned and executed in conjunction with a human.

The next step in this work is to incorporate this algorithm with a path planner and shared-control paradigm on a semi-autonomous, shared-control wheelchair base. On this platform, further validation can be performed on the algorithm in motion in conjunction with different peoples driving styles (i.e. smooth vs. jerky driving, slow vs. fast driving, etc.). Additionally, improving the plane-fitting portion of the algorithm is a priority as many of the errors that are present are related to the plane-fitting portion of the algorithm. Improvement may be seen by using different sample consensus methods, such as MSAC (M-estimator SAmple and Consensus) or MLESAC (Maximum Likelihood Estimation SAmple and Consensus) [20]. Finally, it is likely that significant run-time speed improvements could be realized by implementing the wall segmentation and search in parallel using GPU hardware as seen in [11].

[2]The videos can be found at `http://smpp.northwestern.edu/research/a2-r2/`

References

[1] S.P. Levine, D.A. Bell, L.A. Jaros, R.C. Simpson, and Y. Koren, "The NavChair assistive wheelchair navigation system," IEEE Transactions on Rehabilitation Engineering, vol. 7, no. 4, pp. 443-451, 1999.

[2] Q. Zeng, E. Burdet, B. Rebsamen, and C. L. Teo. "Evaluation of the collaborative wheelchair assistant system," in Proceedings of IEEE Conference on Rehabilitation Robotics, The Netherlands, 2007.

[3] T. Carlson and Y. Demiris, "Human-wheelchair collaboration through prediction of intention and adaptive assistance," in Proceedings of ICRA, Pasadena, CA, 2008.

[4] T. Taha, J. V. Mir, and G. Dissanayake, "Pomdp-based long-term user intention prediction for wheelchair navigation," In Proceedings of ICRA, Pasadena, CA, 2008.

[5] E. Demeester, A. Hntemann, D. Vanhooydonck, G. Vanacker, H. Van Brussel, M. Nuttin, "User-adapted plan recognition and user-adapted shared control: A Bayesian approach to semi-autonomous wheelchair driving," Autonomous Robots, v.24 n.2, p.193-211, February 2008

[6] D. Anguelov, D. Koller, Parker E., and S. Thrun, "Detecting and modeling doors with mobile robots," In Proceedings of ICRA, 2004.

[7] E. Aude, E. Lopes, C. Aguiar, and M. Martins, "Door Crossing and State Identification Using Robotic Vision," in Proceedings of the IFAC Symposium on Robot Control (SYROCO), 2006.

[8] A. Jain and C. C. Kemp, "Behaviors for Robust Door Opening and Doorway Traversal with a Force-Sensing Mobile Manipulator," in RSS Manipulation Workshop: Intelligence in Human Environments, 2008.

[9] A. Y. N. Ellen Klingbeil, Ashutosh Saxena, "Learning to Open New Doors," in RSS Workshop on Robot Manipulation, 2008.

[10] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz, "Laser-based perception for door and handle identification," in Proceedings of ICAR, 2009.

[11] L. Spinello and K. O. Arras, People detection in RGB-D data, in Proceedings of IROS, 2011.

[12] S. Tang, X. Wang, X. Lv, T. X. Han, J. Keller, Z. He, M. Skubic, and S. Lao, Histogram of Oriented Normal Vectors for Object Recognition with a Depth Sensor, in Proceedings of 11th Asian Conference on Computer Vision (ACCV 2012).

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," In Proceedings of CVPR, 2005.

[14] L. Xia, C.-C. Chen, and J.K. Aggarwal, "Human Detection Using Depth Information by Kinect" In CVPR Workshop on Human Activity Understanding from 3D Data (HAU3D 2011).

[15] C. Zheng and R. Green, "Feature Recognition and Obstacle Detection for Drive Assistance in Indoor Environments," in Proceedings of Image and Vision Computing New Zealand (IVCNZ), Auckland, New Zealand, Nov 2011.

[16] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," CACM, 24(6):381-395, June 1981.

[17] D. Meagher, "Geometric modeling using octree encoding," Computer Graphics and Image Processing, Vol. 19, Iss. 2, 1982, Pages 129-147.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in ICRA workshop on Open-Source Software, 2009.

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in Proceedings of ICRA, Shanghai, China, 2011.

[20] P. H. S. Torr and A. Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry," Computer Vision and Image Understanding, 78:138-156, 2000.

[21] American's with Disabilities Act Standards for Accessible Design, 2010 `http://www.ada.gov/2010ADAstandards_index.htm`