

Cheap or Robust? The Practical Realization of Self-Driving Wheelchair Technology

Maya Burhanpurkar*, Mathieu Labbé†, Charlie Guan‡, François Michaud†, and Jonathan Kelly*

Abstract—To date, self-driving experimental wheelchair technologies have been either inexpensive or robust, but not both. Yet, in order to achieve real-world acceptance, both qualities are fundamentally essential. We present a unique approach to achieve inexpensive and robust autonomous and semi-autonomous assistive navigation for existing fielded wheelchairs, of which there are approximately 5 million units in Canada and United States alone. Our prototype wheelchair platform is capable of localization and mapping, as well as robust obstacle avoidance, using only a commodity RGB-D sensor and wheel odometry. As a specific example of the navigation capabilities, we focus on the single most common navigation problem: the traversal of narrow doorways in arbitrary environments. The software we have developed is generalizable to corridor following, desk docking, and other navigation tasks that are either extremely difficult or impossible for people with upper-body mobility impairments.

I. INTRODUCTION

Electric power wheelchairs are often prescribed to individuals with mobility challenges. For many millions of users who suffer from a range of upper-body motor disabilities, such as quadriplegia, age-related hand tremors, Parkinson's disease, multiple sclerosis, and amyotrophic lateral sclerosis, it is impossible to operate an electric wheelchair using the standard joystick interface. These individuals must instead rely on other types of assistive control devices, such as sip-and-puff switches, which are typically extremely difficult to use. This results in degraded mobility, a lack of meaningful engagement with society, and ultimately reduced life expectancy [1].

A robotic navigation system for electric wheelchairs, which would allow the chairs to self-navigate in home and workplace environments, would dramatically improve users' mobility. Currently, there are over 5 million power wheelchairs in use in Canada and the United States and an estimated 20 million such chairs deployed in G20 countries. Thus, the design of a cost-effective navigation system that can be retrofitted to these existing chairs is of great importance. Such a technology would substantially increase quality

This research was supported in part by funding from the National Sciences and Engineering Research Council of Canada (NSERC).

*M. Burhanpurkar and J. Kelly are with the STARS Laboratory, Institute for Aerospace Studies, University of Toronto, Ontario, Canada, maya.burhanpurkar@robotics.utoronto.ca, jkelly@utoronto.ca.

†M. Labbé and F. Michaud are with the Interdisciplinary Institute for Technological Innovation (3IT), Université de Sherbrooke, Québec, Canada, {mathieu.m.labbe, francois.michaud}@usherbrooke.ca.

‡C. Guan is with the Robust Robotics Group, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Massachusetts, USA, cguan@mit.edu.

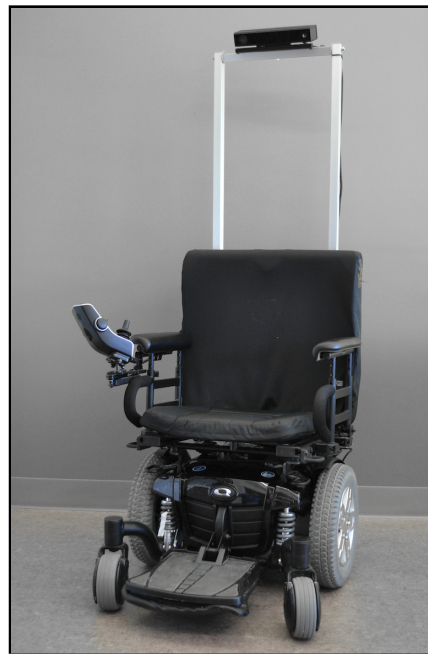


Fig. 1. Autonomous wheelchair prototype, with the Microsoft Kinect 2 sensor mounted on an elevated frame to ensure an adequate field of view. The onboard navigation computer is mounted behind the seat.

of life and societal engagement for this extremely vulnerable segment of society.

Past research efforts have focused on the development of self-driving wheelchair technologies using cost-prohibitive industrial sensors such as 3D laser scanners and advanced stereoscopic depth cameras, as well as high-performance computing hardware for data processing [2]. This cost profile makes past research on self-driving wheelchairs infeasible for use in near-term consumer applications. Furthermore, previously-explored systems have lacked the requisite robustness for a consumer device.

In this paper, we present a robotic navigation system for electric wheelchairs that provides a unique combination of attributes essential to a solution that is both economical and robust. By design, the system must be:

- (a) inexpensive — uses only consumer grade-electronics and operates with reasonable computational demands;
- (b) portable — able to be retrofitted to existing wheelchairs;
- (c) robust and reliable — does not rely on vision algorithms that are susceptible to poor lighting or difficulties with clutter, and uses specialized path planning to ensure ideal alignment for tasks such as doorway traversal;

- (d) rapidly deployable — does not require an *a priori* map, allowing for deployment in unfamiliar environments;
- (e) non-proprietary — uses open source software, leveraging existing open source libraries and components to reduce cost; and
- (f) readily extensible — able to be extended to add new capabilities such as corridor following, desk docking, etc.

Herein, we describe our efforts to address all of the above points above simultaneously, with the aim of developing a low-cost, practical system suitable for mass adoption and capable of robust performance under real-world operating conditions. Our prototype successfully achieves many of the design requirements while still meeting severe cost constraints, requiring less than \$2,000 US worth of components. As an example of the system’s capabilities, we focus much of our discussion on the difficult task of doorway traversal.

II. RELATED WORK

A wide variety of assistive wheelchair navigation systems have been developed over the past 30 years, and we provide only a brief survey of related literature. Low-cost solutions are generally either limited and semi-autonomous [3], [4], require an external localization system [5], [6], are able to operate over short distances only without global localization [7], or lack global planning capabilities [8]. Fully autonomous navigation approaches typically rely on expensive laser scanner (lidar) sensors [9]–[13] (which do have the advantage of being able to operate outdoors [14]–[16]). The goal of existing systems has usually been to provide a working solution in the target environment, without attempting to minimize cost.

The specific problem of doorway traversal, in particular, presents many challenges—no truly robust and cost-effective door detection and navigation system currently exists [2]. Recent approaches utilize visual properties extracted from images, but such methods suffer from difficulties with lighting conditions as well as susceptibility to a variety of artifacts, leading to computationally demanding and unreliable solutions [17]. For example, the three-camera visual approach in [18] relies on edge detection and vanishing point identification to extract trapezoidal (door-like) structures in the environment; similarly, the method in [19] uses two cameras to identify occupied and free space. Neither approach is robust to poor lighting or visual clutter. Further, the planning mechanisms implemented in both cases may be insufficient for tight doors because the chair may not begin in a pose that is well aligned for traversal. In addition, the approach in [19] lacks a definitive mechanism for navigating the common case of a corridor on the other side of the door.

One possible alternative to visual sensing is to employ active scanning devices. In [20], multiple 2D laser scanners are used to match input data to three potential doorway configurations. While successful in benign environments, 2D lasers are unable to detect objects above and below the plane of the laser, resulting in false positives. For example, desks and chairs may be detected as doors. Planar laser-based

methods also are unable to verify that a detected doorway is traversable along the vertical axis.

In contrast to the above, the system we have developed makes use of a commodity RGB-D sensor to provide rich and reliable 3D spatial data, leverages custom and open source software for sophisticated data processing to ensure reliability, and considers unique characteristics of the hardware platform (the power wheelchair) during planning to enable successful operation in cases where other approaches fail.

III. SYSTEM DESCRIPTION

Our prototype navigation system is based on a standard commercial electric wheelchair, shown in Figure 1, to which we have retrofitted a Microsoft Kinect 2 sensor, wheel encoders, and an onboard computer (Intel i7 processor, 4 cores). The total cost of the additional hardware is less than \$2,000 US (retail), and would be even lower in an OEM production scenario. While previous research has focused on varying aspects of autonomy, including wall following, obstacle avoidance, and doorway traversal, modern simultaneous localization and mapping (SLAM) software enables the unification of these functions within a common navigation framework.

Figure 2 illustrates the system architecture. We make significant use of the Robot Operating System (ROS) [21] to provide underlying support for navigation-related tasks. However, we stress that ROS alone provides only a limited portion of the software used to drive the chair—the majority of the code is bespoke and carefully tuned for our application. The navigation system is divided into four layers: 1) a sensor and actuator (hardware) layer (green and red); 2) a low-level processing layer (blue) which filters and translates raw sensor and actuator data streams into more useful formats; 3) a high-level processing layer (orange) containing three submodules: SLAM, navigation, and door traversal; and 4) a user interface layer (purple) that provides visualization and navigation control.

In a typical deployment scenario, an initial map of the environment is built by an operator who manually guides the wheelchair to visit all of the locations where the platform will be expected to navigate. During the mapping process, the SLAM module generates a 2D floor plan, while automatically detecting when the wheelchair revisits a previously-traversed area (to maintain map consistency). The floor plan is then validated by the operator and corrected, if necessary, using an interface tool currently in development. Once the map is available, a wheelchair user can click on a destination location, and the navigation software will compute a collision-free trajectory to reach the goal. While navigating, any dynamic obstacles along the path are automatically avoided. If the wheelchair is shut down and restarted, the SLAM module can re-localize anywhere in the existing map, without the need to start from the same initial location.

The door traversal module is a recent addition to the navigation system, and is able to operate independently of a full environment map. We have found that robust door traversal requires specialized processing beyond that of the

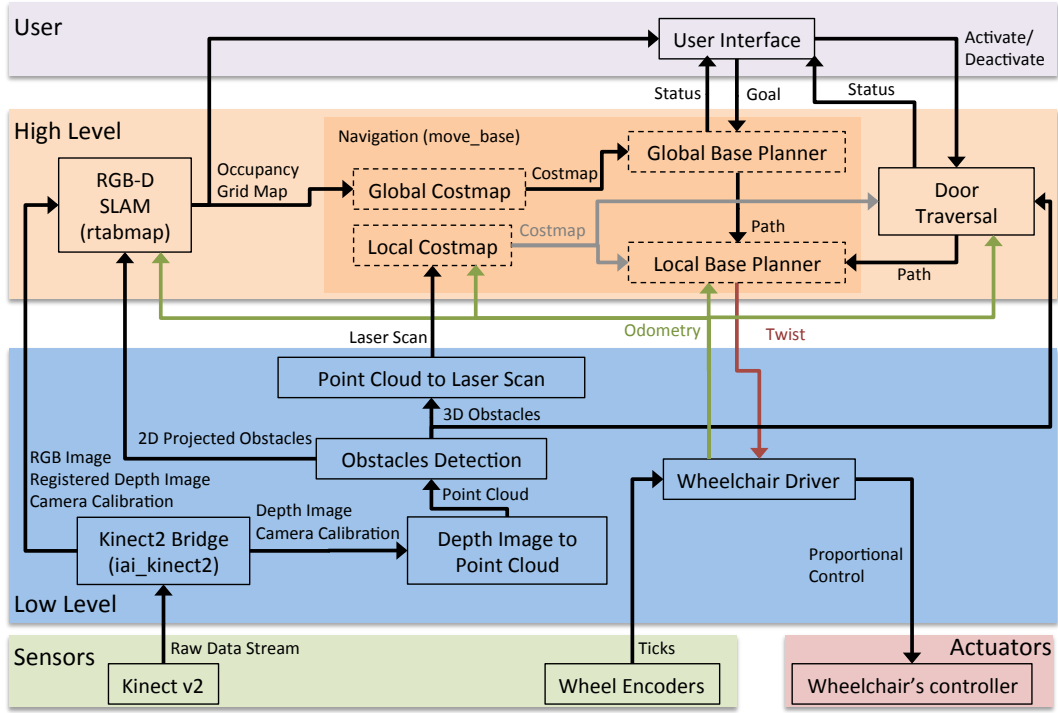


Fig. 2. System architecture, designed in four layers: *sensing and actuation*, *low-level processing*, *high-level processing*, and *user interaction*. The individual software components are built around the Robot Operating System (ROS), which facilitates easy integration and allows us to leverage open source libraries.

baseline SLAM solution. Our doorway traversal software is capable of reliably negotiating doorways and other narrow passages while following a smooth and predictable path to the destination location. Testing has shown that the system operates well in complex environments with diverse geometries and scales.

The following subsections describe the low-level processing layer and the three key submodules of the high-level layer.

A. Low-Level Processing Layer

Low-level processing is performed by five software components: the Kinect 2 Bridge, the Depth Image to Point Cloud module, the Obstacle Detection module, the Point Cloud to Laser Scan module and the Wheelchair Driver module.

Kinect data is acquired using the `iai_kinect2` ROS package [22]. The maximum frame rate is set to 10 Hz to limit CPU usage. However, the module still requires 100% of one CPU core to decode the USB 3 data stream and to register the depth images. Registered depth images are required by the RGB-D SLAM module so that loop closure transformations can be correctly computed.

Depth images from the Kinect are projected into 3D space by the Depth Image to Point Cloud module, generating a point cloud for obstacle detection. The depth image has a resolution of 512×424 pixels, which results in a cloud of 217,088 points. To reduce CPU usage, the depth image is decimated by a factor of four (to 256×212 pixels), so that the resulting cloud has a maximum of 54,272 points. Further reductions in size are achieved using a voxel filter with a size of 5 cm [23]. For navigation in most environments, a

5 cm voxel size represents a good tradeoff between computational load and sufficient precision for obstacle avoidance (based on the wheelchair dimensions and minimal clearance requirements).

Filtered point clouds are used by the Obstacle Detection module to segment the ground plane and identify potential navigation hazards. A normal vector is computed for each point, to determine if the point lies on a plane parallel to the ground—any vector that lies more than 20° away from the ‘upward’ axis is considered to belong to an obstacle. The 20° threshold was selected to be large enough to be robust to Kinect’s noise and small enough to avoid labelling real obstacles as part of the ground. Remaining planar surfaces with centroids at a height of less than 10 cm are labelled as ‘ground’. Points belonging to obstacles are projected onto the ground plane for use in 2D mapping.

Movement commands are sent to the wheelchair through the Wheelchair Driver module, which translates velocity messages (twists) from the Navigation module into the format required by the wheelchair’s onboard controller (in a PID loop). The module also computes wheel odometry information using knowledge of the wheel sizes. This odometry data is then published to other modules and is used as feedback for the PID controller. The onboard computer is connected to the wheelchair’s existing (factory) control interface, emulating the standard wheelchair joystick. The control loop runs at 10 Hz, matching the maximum rate the wheelchair’s existing controller can sustain; the proprietary internal controller of the wheelchair cannot be configured to run faster than 10 Hz. Increasing this control rate would help to enable even more accurate path following.

B. RGB-D SLAM

The RGB-D SLAM module provides autonomous mapping capabilities for the wheelchair. Based on the open source RTAB-Map ROS package [24], the module creates a detailed 2D occupancy grid map from RGB-D images. The map is a graph, where each node stores a synchronized RGB image, a registered depth image, a set of 2D projected obstacles, and odometry information. Using the poses in the graph, a global occupancy grid can be generated by assembling all of the local occupancy grids. The complete map is updated at 1 Hz to limit computational load. At each map update, a new RGB image is visually compared with past images in the map to identify loop closures, based on a bags-of-words approach [25]. When a loop closure is found, any accumulated drift in the odometry is corrected. The local occupancy grid maps are then re-assembled with the corrected wheelchair poses.

The module can be configured in one of two modes: SLAM or localization. The SLAM mode is used to create the initial map of the environment. After creating the map, localization can be activated, without continuously adding new data to the map. This limits computational and memory requirements. Note that it is possible to switch back to SLAM mode to remap an area that has changed significantly or to extend the current map to new locations.

Although incremental motion could be computed using visual odometry (VO), we have found that, with a single camera setup, VO is insufficiently robust for our application. Because of the limited field of view of the camera and because the wheelchair sometimes has to navigate in areas with insufficient visual features, VO can become ‘lost’ easily. We utilize wheel odometry instead, even if it is slightly worse than VO in environments that are visually rich. To reduce odometric drift, 2D projected obstacles from consecutive graph (map) nodes are used to refine the pose-to-pose transformations. The 2D projected obstacle data is also employed to refine any identified loop closures.

C. Navigation

Our Navigation module is based on the `move_base` ROS package [26]. Given the wheelchair’s configuration (i.e., geometry, differential drive, maximum acceleration, and speed), the Local Base Planner module can provide appropriate velocity commands to reach a goal location while avoiding obstacles in the local cost map. The cost map is a 2D occupancy grid where obstacles are inflated by a fixed radius (e.g., generally the robot radius) so that the planner can determine how far away it should pass to safely avoid collisions.

The local cost map has a fixed size of 4 m \times 4 m (the robot is always at the centre) and is updated with the latest sensor readings. To handle dynamic environments, the local cost map is ‘cleared’ of obstacles at each update if possible (e.g., if an obstacle has disappeared). To do so, 2D ray tracing (when using a 2D sensor like a 2D laser scanner) or 3D ray tracing (when a 3D point cloud is used) is performed. The latter technique is the most computationally intensive, and

so for efficiency we use the first approach, relying on the `pointcloud_to_laserscan` ROS package [27] to carry out the 3D to 2D conversion and ray tracing.

For fully autonomous navigation, the Global Base Planner module of `move_base` is utilized [28]. Given the global occupancy grid created by RTAB-Map, the current pose, and the desired goal, a path is planned through the empty cells in the global map. If the goal coincides with an obstacle, or if the location cannot be reached (e.g., a new obstacle is blocking the way), the planning step fails and the user is notified. When a complete path is able to be computed, it is sent to the Local Base Planner module and a series of sub-goals are selected in incremental local cost maps. The Local Base Planner is thus not constrained to exactly follow the global plan, allowing for maneuvering around dynamic obstacles. When RTAB-Map corrects the global map or re-localizes, the global path is recomputed accordingly from the current map location.

D. Door Detection and Traversal

In general, previous attempts at developing an assistive wheelchair with door traversal capabilities have failed due to sensor limitations or difficulties in planning for correct wheelchair-to-door alignment. Our approach utilizes 3D depth data to identify doorways by exploiting their planar nature. The method is motivated by two principles: 1) doorways must contain free space, and 2) the walls that support a door are almost always planar, or contain planar regions even if cluttered with furniture or other objects. Using these constraints, we apply a subsampling and clustering approach, which leads to robust traversal. This is similar to the detection method described in [29], although we consider a wider variety of possible door configurations.

Conceptually, the algorithm first checks the surroundings for walls, then searches for free space within or between them, generating potential doorway candidates. If a gap of sufficient width is detected between the walls and free space is detected along the plane of the door, a doorway is validated. The detection algorithm provides the doorway edges to the door traversal planning routine. Using the normal vector to the plane of the door, the door traversal routine sends a series of global goals to the ROS navigation stack, perpendicular to the plane of the door to ensure alignment for successful traversal. The process can be segmented into four steps: filtering and wall extraction, door candidate identification, validation, and traversal. The first three steps are repeated until a door is successfully found or until the number of points remaining in the input point cloud falls below a threshold value.

Filtering and Wall Extraction: The input to the algorithm is the Kinect 3D point cloud, processed by the Point Cloud Library [23]. The plane of the most prominent wall remaining in the input cloud parallel to the z -axis (vertical) is then identified using a Random Sample Consensus (RANSAC) estimator [30]. In order to limit the influence of obstacles near the ground, the wall plane cloud is filtered in z , such that only the midsection of the wall remains.

Door Candidate Identification: Filtered wall planes are stored in a tree and clustered using a Euclidean cluster extraction routine. These clusters define potential wall segments surrounding the door. Each valid cluster is added to an array of candidate clusters. Typically, one or two clusters are added per iteration of the algorithm—two in the case of a door defined by a gap in a single plane, and one in the case of a door contained between two planes. During each iteration of the door detection algorithm, every previously untested combination of two clusters is checked as a potential door candidate. If the candidate planes are parallel to within a pre-defined tolerance, they are passed to a Single Plane Detection routine; if they intersect, they are passed to a Double Plane Detection routine. In this way, any configuration of door can be detected.

- (a) *Single Plane Detection:* This routine identifies doorways bounded by a single plane, such as hallway entrances and doorways with non-protruding hinged doors. First, the direction of chair tilt is determined. Next, the rightmost cluster is identified by comparing the minimum and maximum y -values for points in each cluster. From this information, the left and right edges of the door are extracted from the cluster clouds through a minimum-maximum search within each cluster. The left and right door coordinates are returned.
- (b) *Double Plane Detection:* This routine identifies doorways bounded by two planes, such as protruding hinged doors that occlude a section of the wall surrounding a doorframe. The intersection of the planes at the height $z = 0$ is found, and used in a nearest neighbours search over the points in each plane to determine the left and right door edges.

Validation: The door edges are passed to a validation routine that checks whether the resulting door width is acceptable and whether there is sufficient free space for the chair to pass. If the number of ‘stray’ points between the door planes is below a threshold value (non-zero due to sensor noise) the doorway candidate is validated.

Traversal: Using the left and right door edges, a series of goals up to, including, and exceeding the centre of the door frame, is sent to the Local Base Planner. This is done to ensure proper alignment with the door (tests conducted by sending only one goal at the midpoint of the left and right door edges often failed in low clearance doors, whereas sending multiple intermediate goals results in robust performance). The number of goals sent varies depending on the starting distance from the door. After each goal is sent and reached, the input obstacle cloud is filtered along the x -axis by the distance the chair has moved in the x direction, to prevent obstacles or walls on the opposing side of the doorway from being detected, as the chair continuously re-evaluates the position of the door.

Obstacles near a door, such as free standing coat racks or water fountains, do not confound the algorithm since a) an object separated from the wall plane in the x -axis will likely not be included in the plane identified by RANSAC,

b) in the event that it is, the obstacle likely will not contain enough points for it to pass clustering validation, and c) if it is accepted as a cluster, it will be rejected by the validation routine. Finally, if Door Candidate Identification or Validation fails, the most recent wall being examined is removed from the 3D point cloud and a new wall is searched for with the remaining points.

IV. EXPERIMENTAL RESULTS

We have carried out a comprehensive series of experiments to evaluate and validate all aspects of the wheelchair platform. This includes a major simulation campaign, made possible by fully modelling the chair in the Gazebo simulator [31], and several studies at the Université de Sherbrooke and the University of Toronto. Below, we report on our results to date, which involve unmanned testing to ensure all safety concerns are addressed; in Section V we review a number of the challenges that remain, and in Section VI we describe plans for upcoming trials with potential end users.

A. Simultaneous Localization and Mapping

The SLAM module, based on RTAB-Map, has an extensive heritage (e.g., winning the IROS 2014 Microsoft Kinect Challenge), and has been shown to operate robustly in a wide range of environments. As an example, Figure 3 shows the map produced after a standard mapping run at the Université de Sherbrooke. Light grey pixels are empty space and black pixels are obstacles. Since the wheelchair has a front-facing camera only, it cannot see behind itself and thus cannot localize when navigating the same corridor in reverse direction. This problem sometimes creates a double-wall effect when the odometry drifts significantly. During the mapping phase, we usually carry out a series of 360° rotations so that loop closures can be detected at locations seen from the opposite direction. To produce the map shown, full rotations were carried out at the corners of the corridors. The zoomed-in section of the map shows that the wooden object lying on the floor is correctly detected during mapping (the wall is thicker and the local cost map is suitably inflated).

B. Autonomous Navigation

The full navigation system has been tested through more than 10 km of autonomous driving, in varying situations and with an assortment of static and dynamic obstacles. This testing has enabled us to refine the system, such that it now operates very reliably in the vast majority of cases. An example of dynamic obstacle detection and avoidance is shown in Figure 4. Referring to Figure 3, the orange line represents the path planned by the Global Base Planner using the available global map. A user-specified goal is shown at the top left (red arrow). This initial path is sent to Local Base Planner to be executed. Figure 4 shows the system reacting to a (previously unseen) person walking directly on top of the original path (purple), obstructing progress. The cost map is automatically updated with the dynamic obstacle and a new plan is formulated (orange). The green line represents the

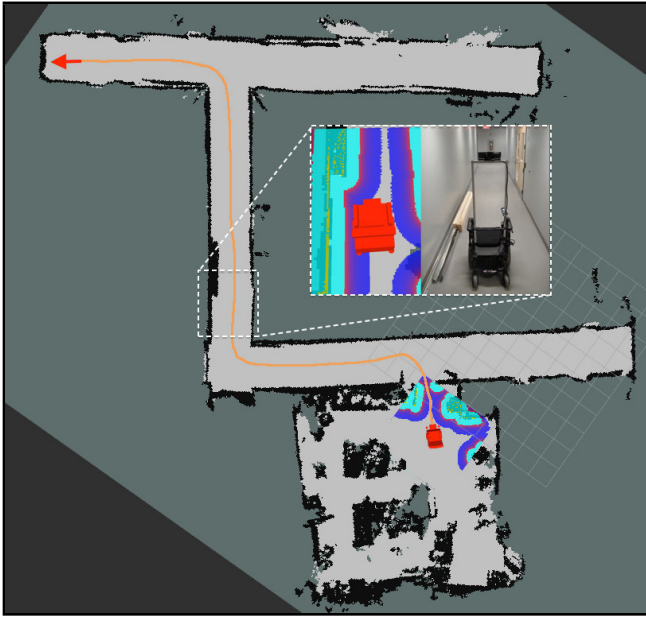


Fig. 3. Map created using SLAM. The orange line is an example of a planned path from the bottom of the map to the goal, represented by the red arrow at the top left (pose with desired orientation). The close-up view shows how the wooden object on the ground is detected. The cost map is represented in colour: yellow pixels are obstacles and cyan represents regions where the planner cannot plan a path because of the risk of collisions.

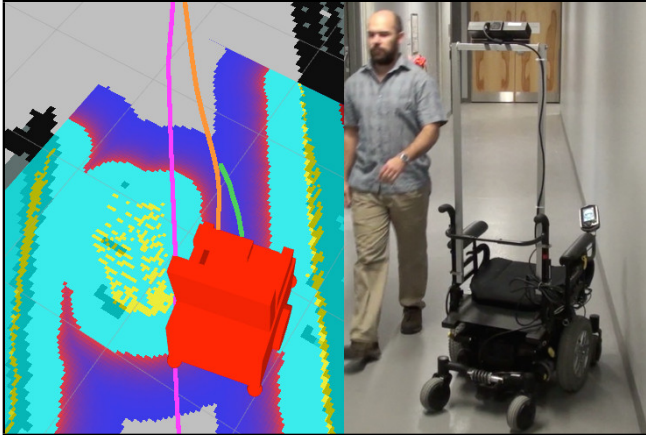


Fig. 4. The wheelchair autonomously avoids a person walking down the corridor. The purple line is the original planned path; the orange is the new global plan, updated with the new obstacle detected. The green line is the actual velocity command sent by the Local Base Planner to the chair controller.

actual velocity commands sent by Local Base Planner to the wheelchair controller.

C. Doorway Traversal

Since door traversal is a critical capability for an practical autonomous or semi-autonomous assistive wheelchair platform, substantial testing of the traversal algorithm was first carried out in simulation (where a wide variety of doorways could be evaluated). Based on these results, we conducted more than 100 tests with real-world doors in buildings at the University of Toronto.

TABLE I
DOORWAY TRAVERSAL EXPERIMENTAL RESULTS

Evaluation Metric	Door A	Door B	Door C
Number of Tests	56	19	21
Detection Success Rate	100%	100%	100%
Traversal Success Rate	100%	100%	100%
Mean Perceived Door Width	0.89 m	1.50 m	0.89 m
True Door Width	0.89 m	1.51 m	0.89 m
Door Width Std. Dev.	0.05 m	0.04 m	0.03 m
Max. Distance Tested	2.91 m	3.17 m	3.20 m
Min. Distance Tested	1.37 m	1.08 m	1.00 m

In simulation, a map with 20 unique door configurations, allowing for 40 unique traversal situations, was constructed. Figure 5 shows an example of one simulated environment. Five tests were conducted on each side of each door with a 100% success rate for all 200 tests. Additionally, several maps were constructed that included obstacles (such as bookshelves and coffee tables) near walls surrounding the doors, to ensure the effectiveness of the algorithm even in the presence of clutter. In these cases, door detection and traversal once again performed with a 100% success rate.

At the University of Toronto, the chair was tested on 8 different door configurations dozens of times each, again with a 100% success rate (after initial failures that led to revisions to the algorithm). Formal tests were conducted on three representative doors, as shown in Table 1 and Figure 6. Door A was a doorway with a hinged door protruding, testing the Double Plane Detection method. Door B was a double doorway, testing the ability to detect large openings. Door C was a doorway without a hinged door, testing the Single Plane Detection method. Tests were conducted at a range of angles and distances from the door; plots of the initial wheelchair locations and orientations for each door are shown in Figure 6. The chair was able to navigate effectively throughout the entirety of the range of situations in which it was tested, including cases where only one plane could be detected.

V. DISCUSSION AND ONGOING CHALLENGES

While we have developed an initial prototype system that performs well in the majority of situations, a series of failure modes and corner cases remain to be investigated. For instance, as with any sensor, the Kinect 2 has some critical limitations. In particular, the unit can have difficulty registering accurate depth information in certain environments. Highly reflective surfaces may cause the sensor to return false depth data, while light-absorbent materials often produce a very low return. Transparent and translucent objects also cause erratic performance. Further, the Kinect 2 is unable to operate outdoors in bright sunlight, due to saturation of the IR receiver. Many of these issues could be mitigated by augmenting the system with other sensor types, although costs would increase.

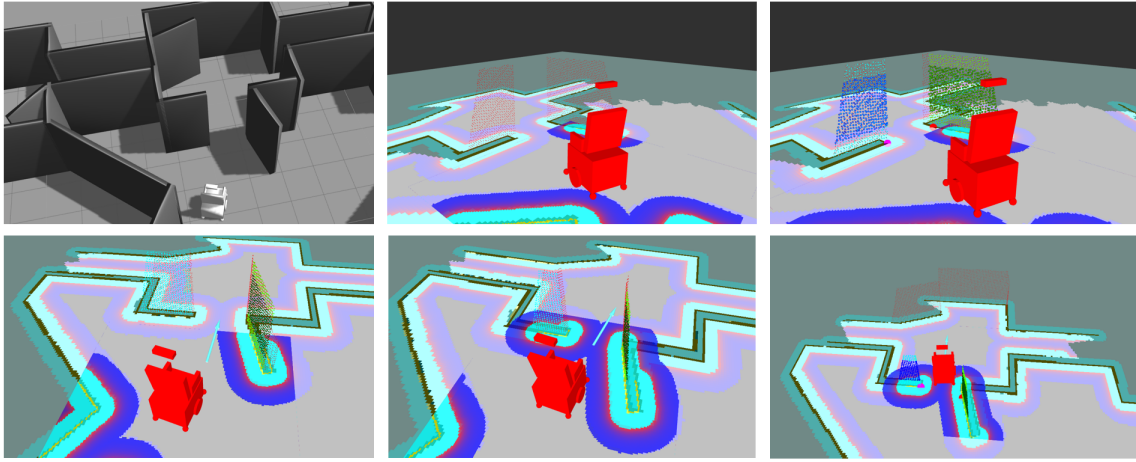


Fig. 5. A door traversal simulation experiment. Top row (left to right): simulation environment in Gazebo; the obstacle cloud in RViz, and the cloud processed by the door detection algorithm (light blue = plane 1, dark blue = plane 1 midsection, light green = plane 2, dark green = plane 2 midsection, pink and red circles = left and right edges of the door respectively determined using the point of intersection of plane 1 and plane 2 at $z = 0$). Bottom row: The chair traverses the door in three steps: first aligning itself with the door, then traveling to the middle of the door frame, and finally traveling an additional distance forward from the frame.

For door traversal specifically, tuning was required initially to ensure reliable performance. Doors with reflective surfaces (even lower metal ‘bumper’ strips) sometimes made proper plane extraction difficult. Door traversal also relies on stable incremental motion estimates. In cases where VO failed, the doors would successfully be traversed, but this would sometimes require the chair to rotate in place, which is undesirable in real-world usage. In addition, because of the sensor noise, the cost map inflation radius must be relatively high to ensure safety, resulting in a very narrow section of the map in which goals can be sent through the door frame. This is, in part, due to downsampling of the point cloud data and the fact that most doors only provide a few centimetres of clearance on either side of the chair. The result is a need to reevaluate incremental planner goals, searching for nearest neighbour points that do not lie within the inflated safety radius. The heuristic could be relaxed if the point cloud were not downsampled, but processing requirements would increase.

At present, our system assumes that the wheelchair is navigating on a planar floor surface (i.e., in 2D), without any ramps or elevators. It would be possible with the current framework to support a 3D surface by using 6-DOF odometry instead of the 3-DOF odometry currently employed (i.e., wheel odometry). More precise odometry would be also needed, as the chair would be able to move in a larger configuration space.

As mentioned in Section IV-A, having a rear-facing camera would assist in localization when traversing the same area in the reverse direction. No 360° rotations would be required to correctly optimize the map, although it is always desirable to have many different views of the same environment for better localization. We note, however, that our computing resources are already used at nearly maximum capacity (e.g., processing the data stream from one Kinect 2 uses 100% of one core of the CPU).

The frequency and latency of the sensing and control

loops necessitate limiting the wheelchair velocity to a modest walking pace, to ensure sufficient time to respond to dynamic obstacles. The system is also dependent on the latency of the proprietary wheelchair controller, which we do not have direct access to. For example, even when issuing commands directly with the standard wheelchair joystick, delays between 100 ms to 400 ms are observed. The wheelchair also has caster wheels that create friction. If the caster wheels are perpendicular to the frame of the chair and a forward velocity command is sent, the wheelchair will rotate slightly until the casters are aligned with the desired travel direction. When navigating around nearby objects at a low speed, commands sent by the Local Base Planner are poorly executed, making difficult to maintain precise control under such conditions.

VI. CONCLUSIONS AND FUTURE WORK

This paper presented a cost-effective and robust autonomous navigation system for existing power wheelchairs. Based on an inexpensive sensor suite (an RGB-D sensor and wheel odometry), the various modules of the system (SLAM, navigation, and door traversal) function synergistically to enable reliable operation under real-world conditions. Our goal has been to develop a system that satisfies the characteristics listed in Section I. We continue to work to address various confounds and corner cases, with rigorous testing and validation. Full navigational autonomy has the potential to improve the safety of users and those around them, while greatly reducing operator fatigue.

We have now begun testing of our development platforms in busy home, office, and retail environments, in order to assess performance with end users. This testing is being carried out under the guidance of trained occupational therapists, ensuring that we meet the needs of the target community.

ACKNOWLEDGMENT

This work was supported in part by NSERC Canada and by Cyberworks Robotics, Inc. The authors wish to thank

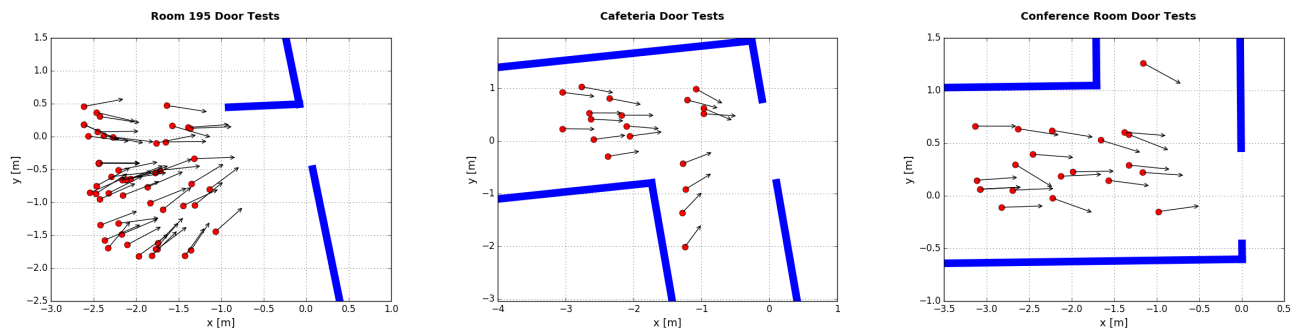


Fig. 6. Doorway traversal tests carried out at the University of Toronto. Each plot shows a top-down view of the door configuration (see Section IV-C). Planar surfaces are shown in blue. Each red dot specifies the position of the centre of the wheelchair at the start of an experiment, while the corresponding arrow indicates the initial orientation of the chair at the start of the traverse.

Rodolphe Perrin and Daniel Elbirt for their assistance with data collection.

REFERENCES

- [1] E. Keeler, J. M. Guralnik, H. Tian, R. B. Wallace, and D. B. Reuben, "The impact of functional status on life expectancy in older persons," *J. Gerontology: Series A*, vol. 65A, no. 7, pp. 727–733, 2010.
- [2] R. C. Simpson, "Smart wheelchairs: A literature review," *J. Rehabilitation Research and Development*, vol. 42, no. 4, pp. 423–436, Jul. 2005.
- [3] D. P. Miller and M. G. Slack, "Design and testing of a low-cost robotic wheelchair prototype," *Autonomous Robots*, vol. 2, no. 1, pp. 77–88, Mar. 1995.
- [4] J. Shen, B. Xu, M. Pei, and Y. Jia, "A low-cost tele-presence wheelchair system," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2016, pp. 2452–2457.
- [5] J. C. Garcia, M. Marron, J. Ureña, and D. Gualda, "Intelligent wheelchairs: Filling the gap between labs and people," *Assistive Technology: From Research to Practice*, vol. 33, pp. 202–209, 2013.
- [6] B. K. Kim, H. Tanaka, and Y. Sumi, "Robotic wheelchair using a high accuracy visual marker LentiBar and its application to door crossing navigation," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2015, pp. 4478–4483.
- [7] C. A. Rockey, "Low-cost sensor package for smart wheelchair obstacle avoidance," Ph.D. dissertation, Case Western Reserve University, 2012.
- [8] B. F. Wu, C. L. Jen, W. F. Li, T. Y. Tsou, P. Y. Tseng, and K. T. Hsiao, "RGB-D sensor based SLAM and human tracking with Bayesian framework for wheelchair robots," in *Proc. Int. Conf. Advanced Robotics and Intelligent Systems*, 2013, pp. 110–115.
- [9] C. Mandel, K. Huebner, and T. Vierhuff, "Towards an autonomous wheelchair: Cognitive aspects in service robotics," in *Proc. Towards Autonomous Robotic Systems*, 2005, pp. 165–172.
- [10] C. D. la Cruz, T. F. Bastos, F. A. A. Cheein, and R. Carelli, "SLAM-based robotic wheelchair navigation system designed for confined spaces," in *Proc. IEEE Int. Symp. Industrial Electronics*, 2010, pp. 2331–2336.
- [11] R. Tang, X. Q. Chen, M. Hayes, and I. Palmer, "Development of a navigation system for semi-autonomous operation of wheelchairs," in *Proc. IEEE/ASME Int. Conf. Mechatronic and Embedded Systems and Applications*, 2012, pp. 257–262.
- [12] Y. Morales, N. Kallakuri, K. Shinozawa, T. Miyashita, and N. Hagita, "Human-comfortable navigation for an autonomous robotic wheelchair," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2013, pp. 2737–2743.
- [13] L. Cavanini, F. Benetazzo, A. Freddi, S. Longhi, and A. Monteri, "SLAM-based autonomous wheelchair navigation system for AAL scenarios," in *Proc. IEEE/ASME Int. Conf. Mechatronic and Embedded Systems and Applications*, 2014, pp. 1–5.
- [14] C. Gao, M. Sands, and J. R. Spletzer, "Towards autonomous wheelchair systems in urban environments," in *Field and Service Robotics: Results of the 7th International Conference*, ser. Springer Tracts in Advanced Robotics. Springer, 2010, vol. 62, pp. 13–23.
- [15] M. Yokozuka, N. Hashimoto, K. Tomita, and O. Matsumoto, "Development of autonomous wheelchair for indoor and outdoor traveling," in *International Internet of Things Summit*, 2014, pp. 91–96.
- [16] D. Schwesinger, A. Shariati, C. Montella, and J. Spletzer, "A smart wheelchair ecosystem for autonomous navigation in urban environments," *Autonomous Robots*, vol. 41, no. 3, p. 519538, Mar. 2017.
- [17] D. Anguelov, D. Koller, E. Parker, and S. Thrun, "Detecting and modeling doors with mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 4, 2004, pp. 3777–3784.
- [18] F. Pasteau, V. K. Narayanan, M. Babel, and F. Chaumette, "A visual servoing approach for autonomous corridor following and doorway passing in a wheelchair," *Robotics and Autonomous Systems*, vol. 75, Part A, pp. 28–40, 2016.
- [19] T. Panzarella, D. Schwesinger, and J. Spletzer, "CoPilot: Autonomous doorway detection and traversal for electric powered wheelchairs," in *Field and Service Robotics: Results of the 10th International Conference*, ser. Springer Tracts in Advanced Robotics. Springer, 2016, vol. 113, pp. 233–248.
- [20] F. Cheein, C. De La Cruz, T. Bastos, and R. Carelli, "SLAM-based cross-a-door solution approach for a robotic wheelchair," *Int. J. Adv. Robotic Syst.*, vol. 7, no. 2, pp. 155–164, Sep. 2010.
- [21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA 2009 Workshop on Open Source Software*, 2009.
- [22] T. Wiedemeyer, "IAI Kinect2," https://github.com/code-iai/iai_kinect2, Institute for Artificial Intelligence, University Bremen, accessed January 10, 2017.
- [23] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2011, pp. 1–4.
- [24] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014, pp. 2661–2666.
- [25] —, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Trans. Robotics*, vol. 29, no. 3, pp. 734–745, Jun. 2013.
- [26] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The Office Marathon: Robust navigation in an indoor office environment," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010, pp. 300–307.
- [27] P. Bovbel and T. Foote, "Point Cloud to Laser Scan," http://wiki.ros.org/pointcloud_to_laserscan, accessed January 10, 2017.
- [28] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1, 1999, pp. 341–346.
- [29] M. Derry and B. Argall, "Automated doorway detection for assistive shared-control wheelchairs," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2013, pp. 1254–1259.
- [30] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Comm. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.
- [31] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2004, pp. 2149–2154.