CoPilot: Autonomous Doorway Detection and Traversal for Electric Powered Wheelchairs

Tom Panzarella, Dylan Schwesinger (✉) and John Spletzer

Abstract In this paper we introduce CoPilot, an active driving aid that enables semi-autonomous, cooperative navigation of an electric powered wheelchair (EPW) for automated doorway detection and traversal. The system has been cleanly integrated into a commercially available EPW, and demonstrated with both joystick and head array interfaces. Leveraging the latest in 3D perception systems, we developed both feature and histogram-based approaches to the doorway detection problem. When coupled with a sample-based planner, success rates for automated doorway traversal approaching 100% were achieved.

1 Introduction

The U.S. Department of Health and Human Services reports that the number of people over the age of 65 will increase from 40.4 million people in 2010 to over 70 million by 2030 [18]. This rapid growth in the U.S. elder population will also increase the number of people with age-related symptoms that hamper their mobility. Such common symptoms include visual impairments, dementia, and Alzheimer's disease [16]. Providing electric-powered wheelchairs (EPWs) to seniors (and others) is a significant step in helping them live at home and maintain independent mobility. However, it is not without its own challenges. Maintaining straight paths and avoiding obstacles is often challenging - especially for drivers using alternative controls

Tom Panzarella
Love Park Robotics, LLC, 2025 Washington Ave, Suite 217, Philadelphia, PA USA 19146 e-mail: tpanzarella@loveparkrobotics.com

Dylan Schwesinger (✉)
Lehigh University, 27 Memorial Drive West, Bethlehem, PA 18051, e-mail: dts211@lehigh.edu

John Spletzer
Lehigh University, 27 Memorial Drive West, Bethlehem, PA 18051, e-mail: josa@lehigh.edu

such as sip-and-puff devices, switch driving systems, chin controls, or short-throw joysticks. Additionally, traditional joystick users with impaired hand control and those who rely on "latched driving" modes (i.e., cruise control) for independence and function may require additional assistance to ensure safe and comfortable mobility. To realize the home health benefits of EPWs while also maintaining safety, active safety systems for EPWs could be deployed.

To this end, we have developed CoPilot, an active driving aid that enables semi-autonomous, cooperative navigation of an EPW. Similar to active driver-assist systems in automobiles, the driver remains in primary control of the vehicle, while in the background, CoPilot uses intelligent sensing and drive control systems that work in cooperation with the driver to aid in avoiding obstacles/collisions and fine precision driving tasks. The motivation is that as an individual begins to lose cognitive, perceptive, or motor function due to age, injury, or disease, CoPilot can augment that loss because it can interpret the user's intent by seeing into the environment. This exteroceptive sensing capability is enabled by leveraging the latest in three-dimensional (3D) imaging technology. While being developed with a suite of semi-autonomous driving behaviors in mind, the focus of this paper is automated doorway detection and traversal. This functionality was motivated by discussions with physical and occupational therapists in the wheelchair space who prioritized doorway navigation as a capability that would provide real value to EPW users. CoPilot provides near 100% effectiveness in this application.

2 Related Work

Doorway detection using 3D sensing has been accomplished in various ways. Rusu et al. used 3D point clouds to locate doors [15]. The goal was to find doors for the purpose of opening or closing them with a robotic manipulator. When the robot was at a door location, a planar model was fit to the point cloud data. The models were validated based on geometric constraints. More recently, RGB-D data has been used for the task of parsing indoor scenes [6, 13]. The goal of which is to detect and correctly label objects in indoor environments. This is a more difficult task than looking for a single category of object, in our case doorways. These algorithms are based on learning classifiers where the feature vectors are largely inspired from computer vision techniques, such as histograms of oriented features. In our work, we also leverage computer vision approaches for some aspects of doorway detection.

Early approaches of wheelchair systems capable of doorway traversal include [9, 19, 11]. For navigation, Levine et al. [9] and Yanco [19] both utilized an array of sonar sensors and Parikh et al. [11] used a planar laser scanner. While these works yielded successful demonstrations, the limitations of the sensors were not necessarily suitable for use in cluttered environments. For example, depending on sensor placement, these approaches might be susceptible to navigating through a table because the table legs could be detected but not the table top.

The work most similar to our own is Derry and Argall [3], where the goal was to detect open doorways suitable for wheelchair traversal. Their approach involved

processing point cloud data to fit planar models under the assumption that gaps in the planar model correspond to doorways if they meet certain geometric criteria. A key difference in approaches is that while their focus was in processing point clouds, our algorithms emphasize processing the depth images directly. Furthermore, their investigation was limited to doorway detection. In contrast, CoPilot provides a complete solution for automated doorway navigation.

3 Development Platform

The development platform used in this research was based on the Quantum Q6 Edge electric powered wheelchair (EPW) shown in Fig. 1. The Q6 features motors with integrated encoders for measuring wheel velocities. To access these for odometry purposes, we interfaced an on-board embedded computer with the EPW's motor controller over the CAN bus. It also enabled the regulation of the EPW's linear and angular velocities via a software-based PID.

Fig. 1 CoPilot integrated into an Quantum Q6 Edge EPW.

Exteroceptive sensing was from two Primesense Carmine 1.09 sensors. The Carmine 1.09 is the shorter range version of the Primesense structured lighting sensor. It has an advertised effective range between 35-140 cm (compared to 80-350 cm for the standard range Carmine 1.08). The decision to use the short range variant was to ensure that doorways and obstacles remained visible in close proximity to the chair. However, the maximum range of 1.4 m was extremely limiting. We addressed this through an intrinsic calibration procedure which extended the effective range to approximately 3 meters with little degradation in accuracy. This is discussed in detail in Section 4.1. Two sensors were used in order to increase the total field of view. This ensured better coverage of the chair footprint (to avoid collisions with obstacles), as well as facilitated doorway detection at a range of chair orientations. The mounting positions of the sensors are depicted in Fig. 1. Note that the sensors are mounted vertically rather than horizontally as this was found to be a less obstructive configuration.

The software was developed using the Robot Operating System (ROS) [12] framework and modularized based on ROS' message passing paradigm. For basic image processing and point cloud manipulation, we leveraged the OpenCV [2] and Point Cloud Library (PCL) [14] projects respectively. Processing was via a separate onboard computer with a 2.2 GHz Intel Core i7 processor and 8 GiB of RAM.

4 CoPilot Perception

4.1 Intrinsic Sensor Calibration

As alluded to in Section 3, the maximum advertised range of the Carmine 1.09 (1.4 meters) was insufficient for effective doorway detection. While objects at depths farther than 1.4 meters could still be detected, the triangulation based nature of structured light sensors induces a nonlinear noise model of the form $|\delta z| \propto z^2|\delta d|$, where δz is the error in the depth observation, z is the actual depth, and δd is the error in disparity. In other words, errors grow quadratically with depth. This can be mitigated by using an appropriate error model and adjusting the depth measurements accordingly. Unfortunately, global distortion models used for traditional camera calibration are of limited use as sensors based on the Primesense appear to have irregular distortion patterns unique to each individual sensor [17]. While they propose an unsupervised procedure to intrinsic calibration in [17], we use an alternate approach that while supervised, is fast to use and significantly less complex to implement.

Starting at the minimum effective range of the sensor, the user captures a depth image of a nominally flat wall. The sensor is then moved incrementally farther from the wall, and a new image is captured out to the maximum sensor range. For example, if the minimum and maximum ranges of interest were 0.5 m and 3.0 m respectively, depth images would be captured at nominal depths of $\mathbf{z}$ = [0.5,1.0,1.5,2.0,2.5,3.0] meters. Note that the exact spacing is not critical. However, the accuracy of the depths $\mathbf{z}$ *is* the basis for the calibration, and must be measured accurately. This can be readily accomplished using standard tools (e.g., a tape measure or laser distance measurer). It is also important that the sensor's optical axis be roughly normal to the wall surface. To ensure this, we developed an application that provides visual feedback of the alignment error between the sensor's optical axis $\mathbf{o}$ and the wall's surface normal $\mathbf{n}_w$. This is estimated by using RANSAC [4] to automatically segment the wall plane in real-time. The user then adjusts the sensor orientation until $||\mathbf{o} \times \mathbf{n}_w|| \approx 0$. In practice, an alignment error of ≤ 1 degree is adequate for calibration, and easily obtained.

Given a set of k point cloud images $\mathbf{P} = [P_1, \ldots, P_k]$ and corresponding ground truth depth measurements $\mathbf{z}$, the remainder of the calibration process is completely automated. For each $P \in \mathbf{P}$, we recover the parameters for the respective wall planes $\Pi = [\Pi_1, \ldots, \Pi_k]$ where the relative orientation is again estimated using RANSAC and the translation using the depth measurements $\mathbf{z}$. Given robust estimates of the actual wall's relative positions and orientations Π, the point clouds $\mathbf{P}$ are adjusted to ensure that each point $p_i(i, j) \in P_i$ lies on its respective plane Π_i. This is accomplished by generating a set of scaling coefficients $\mathbf{K} = K_1, \ldots, K_k$ for each point of each point cloud. We denote the corrected point cloud set as $\mathbf{P}^*$.

The scaling coefficients $\mathbf{K}$ are to this point limited to the discrete set of ranges $\mathbf{z}$ where calibration data were collected. These are generalized to continuous space by modeling the scaling coefficients as a quadratic function of scene depth, i.e.,

$$K(i, j, z) = A(i, j)z^2 + B(i, j)z + C(i, j) \tag{1}$$

where (i, j) are the pixel coordinates of the point cloud. Thus, every sensor pixel has it's own specific quadratic function $k(i, j, z)$ that is used to determine the scaling factor at a given depth z. The quadratic coefficients $[A(i, j), B(i, j), C(i, j)]$ for each pixel (i, j) are recovered as a least squares solution minimizing the residuals between $\mathbf{P}$ and $\mathbf{P}^*$. The coefficients are calculated offline, and stored in three Look Up Tables (LUTs) A, B, C corresponding to the respective quadratic coefficients.

A point cloud P of $m \times n$ points can be described through its Euclidean coordinates $X, Y, Z \in \mathbb{R}^{m \times n}$ where each matrix entry corresponds to the x, y, z coordinates of the respective point. To calculate the corrected points, the following operations are performed on the streaming point cloud:

$$
\begin{aligned}
K(i,j) &= A(i,j) * Z(i,j)^2 + B(i,j) * Z(i,j) + C(i,j) \quad \forall \ (i,j) \\
X^*(i,j) &= K(i,j) * X(i,j) \\
Y^*(i,j) &= K(i,j) * Y(i,j) \\
Z^*(i,j) &= K(i,j) * Z(i,j)
\end{aligned}
$$

where X^*, Y^*, Z^* denote the corrected point set. Thus, online intrinsic calibration can be performed at a cost of only several floating point operations and array look ups per point.

We have used the calibration procedure extensively over the past year, and performance has been very good. A sample calibration run is shown at Figure 2. The left sub-figure shows a point cloud before (top in red) and after (bottom in blue) calibration. Qualitatively, we see that both the distortion and dispersion of the points were significantly reduced. This is also reflected quantitatively in the center-right sub-figures, which show the mean error and mean standard deviation of the points vs. scene depth (pre-calibration and post-calibration). The reductions in both error and variance were significant, clearly demonstrating the efficacy of the approach.

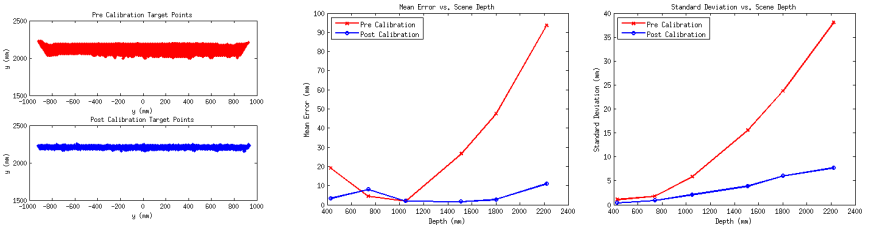

Fig. 2 (Left) Sensor points before (top) and after (bottom) intrinsic calibration. Note that both point distortions and dispersion is reduced. This is also reflected in the mean error (center) and standard deviation (left) of the points.

4.2 Depth Image Warping & Fusion

Our approach to doorway segmentation relies heavily upon the observation that doorway border features are strongly vertical. We further observe that computationally, these features can be extracted most efficiently if the sensor frame is aligned vertically with the world frame, i.e., the gravity vector. An analogy would be the

motivation for rectification of stereo image pairs. As a result, we warped and fused the depth image pair as a pre-processing stage.

Given two point clouds P_L, P_R associated with the left and right sensors, respectively, the first step was to warp the points to a common coordinate frame **F**. We chose **F** to be centered between the actual sensor positions, and with an orientation identical to the EPW vehicle frame. Using the extrinsic calibration relating the sensor and vehicle frames, we recovered the rigid transformation between the frames and transformed the points in each point cloud

$$\hat{P}_L = {}^{C}R_L P_L + {}^{C}\mathbf{t}_L \tag{2}$$

$$\hat{P}_R = {}^{C}R_L P_R + {}^{C}\mathbf{t}_R \tag{3}$$

where $({}^{C}R_L, {}^{C}\mathbf{t}_L)$ and $({}^{C}R_R, {}^{C}\mathbf{t}_R)$ were the rigid transformations relating the left and right sensor frames to **F**. Since most of our processing will be in the depth image space, we next calculated the back projection of $\hat{P}_L, \hat{P}_R$ to form the fused depth image I_D. In doing so, a couple of subtleties needed to be addressed. First, the back projection of points do not lie on exact pixel boundaries. As a result, we use a nearest neighbor interpolation scheme to form the depth image. Second, there was the potential that a point in both $\hat{P}_L$ and $\hat{P}_R$ would warp to the same pixel $I_D(i, j)$. In this event, the depth of the closer point was used.

The process is reflected in Figure 3. The left-center sub-figures show the raw depth images from the left and right sensors. Note that when mounted on the EPW, the sensors were rolled approximately 90 degrees which explains the vertical orientation of the depth images. The right sub-figure shows the resulting depth image I_D after transforming and fusing the point clouds. All subsequent image and point cloud processing is done using this image as input.

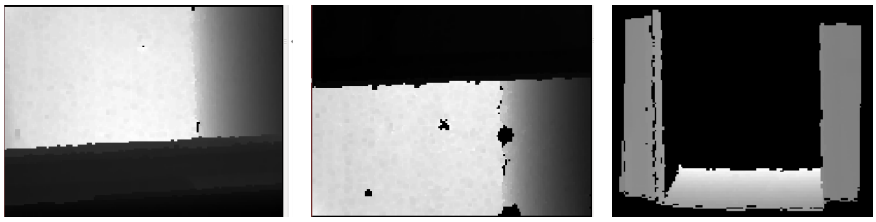

Fig. 3 (Left-Center) Raw depth images of a doorway from the left and right sensors. (Right) Fused depth image.

4.3 Real-time Doorway Detection

After the transformation outlined in Section (4.2), vertical edges in the real-world map to vertical columns in I_D. The doorway detection procedure exploits this fact to efficiently find doorway boundaries based on salient features in the depth image. We evaluated two approaches to finding doorway boundaries, a feature based approach and a histogram based approach. After a set of doorway boundaries was obtained

(from either approach), they were then validated based upon geometric constraints. We now describe the process in detail.

4.3.1 Feature Based Doorway Boundary Detection

Doorways are transition features between interior and exterior space. When viewed within a depth image I_D, they appear as spatial discontinuities. This is to be expected, as there must be sufficient free space to accommodate pedestrian (or EPW) traffic across the spaces. We leveraged techniques traditionally used in 2D image processing to localize this discontinuity, and by association the doorway edges. To enhance these edges, we convolved I_D with a $[-1,0,1]$ kernel to generate the horizontal gradient image, and then thresholded based upon the size of the depth discontinuity to generate an edge image E_D. The next step was to identify edges of sufficient length to be classified as a doorway edge. Note that simply summing the edge pixels for each column of E_D would produce incorrect results for two reasons: (i) the edges could actually be at different depths in 3-space, possibly corresponding to multiple objects, and (ii) the resulting sum would be biased towards objects close to the sensor because they subtend more pixels.

The first problem was mitigated by calculating the median depth $\bar{z}_k$ of each column k of I_D and generating a copy of the depth image, M_D, where values in column k are set to zero if they are not within some specified distance to $\bar{z}_k$. The idea was that true doorway edges would represent the majority of the edge length in the column, and the median value would therefore lie upon this edge. The second problem was addressed by weighting the depth measurements with the height of the unit pixel p_h subtended at the respective depth. The approach can be expressed concisely as

$$\Phi = \mathbf{1}^T (p_h \cdot E_D \odot M_D) \tag{4}$$

where $\mathbf{1}$ is a column vector of all ones, $\odot$ denotes elementwise multiplication, and Φ defines a row vector where each component corresponds to the edge height in each column. Each component in Φ was evaluated based on a minimum height requirement. The set of columns that meet the threshold were marked as potential doorway boundaries at a depth of $\bar{z}_k$.

The process is illustrated in Figure 4. The left sub-figure shows the edge image E_D. The center image shows edge pixels overlaid on the fused RGB-D image. The right image shows edge clusters projected to the $x-y$ plane. Note that each cell represents a potential doorway boundary, so that multiple candidates can be obtained from a single doorway image. Discriminating the correct edge (e.g., the front doorway edge vs. the rear) will be discussed in Section 4.3.3.

We quickly determined that by themselves, doorway edges were an insufficient feature for doorway detection. For example, an inward opening door may not offer a strong edge on the hinge side as the door face can provide a smooth transition into the room. As a result, we also integrated corner features into our classifier. To do this, we first generated a 2D histogram $H(x,y)$ that bins points in 3-space to the ground plane. After applying the Harris operator to $H(x,y)$ [7], we identified the set of bins $\mathbf{C}$ in $H(x,y)$ that corresponded to corner features using an appropriate

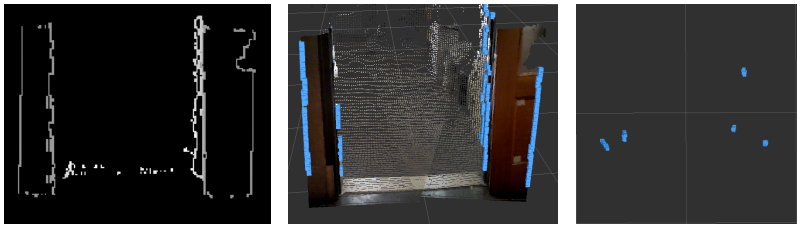

Fig. 4 (Left) Edge image of doorway. (Center) Edge pixels identified in the scene. (Right) Top down view of door edge coordinates.

threshold. Marking a column as a potential doorway based on $\mathbf{C}$ required a small amount of effort since measurements from multiple columns could fall into the same bin. For each $C_k \in \mathbf{C}$, we found the data point $\mathbf{x}$ closest to the centroid of the bin and marked the associated column as a potential doorway boundary at a depth equal to the distance to $\mathbf{x}$.

The corner detection process is illustrated in Figure 5. The left sub-figure shows the fused depth image. The center image shows corner pixels overlaid on the fused RGB-D image. The right image shows valid corners projected to the $x-y$ plane.

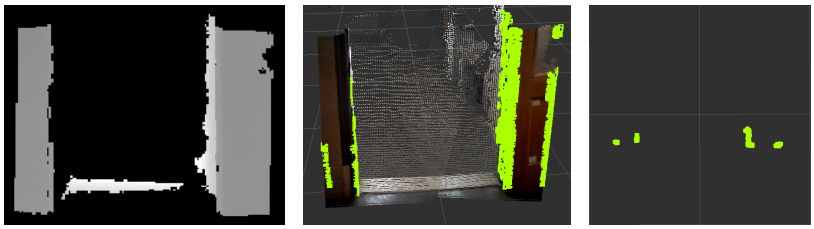

Fig. 5 (Left) Fused depth image of doorway. (Center) Corner pixels identified in the scene. (Right) Top down view of doorway corner coordinates.

In practice, our feature based approach was very successful at segmenting doorways. However, its computational complexity - dominated by the corner segmentation component - was of concern. This motivated our investigation into the histogram-based approach described below.

4.3.2 Occupancy Histogram Doorway Boundary Detection

Our feature-based approach attempts to directly identify the doorway boundaries, leaving only a small number of candidates as input to the validation procedure outlined in Section 4.3.3. However, this comes at the expense of significant up-front computation. As a result, we investigated a simpler descriptor. It is based upon the observation that the segmented edge and corner features were subsets of all columns largely occupied by a vertical object. For edge and corner features, we expend significant computational resources verifying that neighboring columns in 3-space are not occupied. But what if we simply identified each column that had a high occupancy rate as a potential doorway boundary? Undoubtedly this would lead to a much

larger number of candidates for validation, but in practice the computational savings in image and point cloud processing more than makes up for this expense.

In effect, the depth image was reduced to a 1-D occupancy histogram. To accomplish this, we simplified the approach summarized in (4) to

$$\Phi = \mathbf{1}^T (p_h \cdot M_D) \tag{5}$$

which yielded a row vector where each component was the height of the object in each column corresponding to the median value $\bar{z}_k$. In other words, where in (4) we accumulated edge lengths, in (5) we are accumulating object height. Φ is now a 1D histogram of heights per bin where each bin corresponds to a column in M_D. Thresholding each component of Φ on a minimum height requirement segments every column that corresponds to a large vertical object.

When combined with the validation procedure in Section 4.3.3, this approach worked surprisingly well in practice. Compared to the feature-based approach, the implementation is far simpler as neither edge nor corner detection is required. It is also more efficient computationally. With a Primesense at VGA resolution (640×480), the feature based approach detected doorways at 12 Hz on the computer in Section 3. By comparison, the histogram approach ran at frame-rate (30 Hz). In the current version of CoPilot, the occupancy histogram approach is used exclusively.

4.3.3 Doorway Validation

Given the columns marked as candidate doorway boundaries and the associated depth depth values, the role of the doorway validation procedure is to find the best estimate of the relative position and orientation of the doorway. Algorithm 1 VALIDATE-DOORS outlines the procedure of reducing the set of doorway boundaries to a set of doorway candidates **D**. Each pair of doorway boundaries must meet geometric constraints based on the width of the doorway (line 5), the orientation of the EPW to the doorway (line 5) and the amount of free space beyond the sill of the doorway (lines 10-14). Guided by the American's with Disability Act (ADA) [1] accessibility guidelines, minimum and maximum doorway widths were set to 82 cm and 162 cm, respectively. The orientation constraint was set to $\pm 45^\circ$ and the free space beyond the doorway had to be sufficient to accommodate the EPW footprint. Doorway width and orientation validation are performed by the GEOMETRIC-VALIDATION sub-procedure.

Computationally, the most expensive part of VALIDATE-DOORS is the INTERSECT sub-procedure which verifies that sufficient free space exists beyond the candidate doorway via ray-tracing. In theory, there could be $O(n^2k)$ calls, where n is the number of columns and k is the number of free space tracing operations per doorway boundary pair. In practice, this will not happen due to constraints on doorway width, sensor field-of-view, and wheelchair orientation. When benchmarked with a single Primesense at VGA resolution, VALIDATE-DOORS had a mean run time of approximately 3 ms with a standard deviation of approximately 1 ms.

The doorway validation procedure returns the set of valid doorways **D** with the relative position of the doorway's center and its orientation. Note there is high prob-

Algorithm 1 Door Validation

```
1:  procedure VALIDATE-DOORS(O, B)        ▷ O: obstacle coordinates, B : boundary coordinates
2:      D ← ∅                                                          ▷ set of valid doorways
3:      for i ← 0 to n − 1 do
4:          for j ← i + 1 to n do
5:              if GEOMETRIC-VALIDATION(B[i], B[j]) then
6:                  continue
7:              end if
8:              is_valid ← true
9:              for k ← i to j do                                      ▷ trace free space
10:                 p ← INTERSECT(B[i], B[j], k)                       ▷ line segment intersection point
11:                 if ‖O[k]‖ < ‖p‖ then
12:                     is_valid ← false
13:                 end if
14:             end for
15:             if is_valid = true then
16:                 D ∪ {[x, y, θ]^T}                                  ▷ add doorway pose
17:             end if
18:         end for
19:     end for
20: end procedure
21: Note: The loops on B continue early when the column has no associated doorway boundary.
```

ability that the classifier will return multiple doorway candidates. However, these will typically be variants of the actual doorway opening (e.g., front edge to rear edge, front corner to rear edge, door stop to front corner, etc.). To ensure consistent position and orientation estimates, we wish to identify only the front edges/corners of the doorway. To this end, we use a heuristic of choosing the closest doorway candidate. In practice, this has worked quite well for detecting the actual doorway.

The process is illustrated in Figure 6. The center sub-figure shows the valid doorway candidates (red arrows), and the right sub-figure the chosen doorway. The latter well approximates the doorway position and orientation.

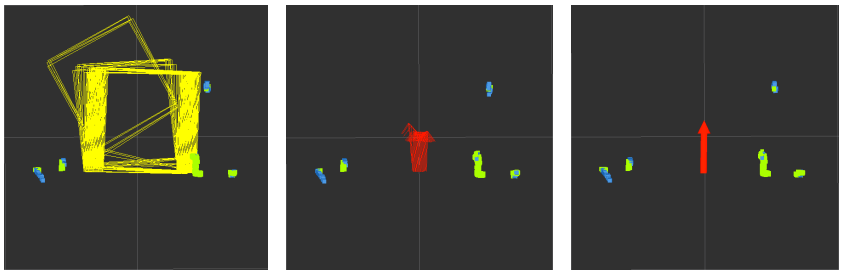

Fig. 6 (Left) Free space check for feature pairs. (Center) The set of valid doorway features. (Right) The final doorway chosen using the "nearest doorway" doorway heuristic.

5 Autonomous Doorway Navigation

At the user level, the CoPilot interface is very intuitive. The user switches the EPW controller drive mode to "CoPilot" and manually drives towards the door. As soon as CoPilot detects the doorway, an icon appears on the LCD control panel. The user then pushes a single button to effect doorway traversal. Note also that the user can also steal back control from CoPilot at any time by simply touching the joystick.

At the software level, doorway navigation is decomposed into two primary subtasks: mapping the environment, and given such a map perform real-time planning and control of the EPW for safe and reliable doorway traversal.

5.1 Mapping

The local map was a 2D occupancy grid centered at the current EPW pose. We leveraged ROS for populating and clearing cells in the local map through ray tracing techniques [12]. For navigation purposes, 3D points were projected down to a 2D costmap M where the individual cells were categorized as either occupied (*i.e.*, obstacles), free, or unknown. Each cell $M(x,y)$ was also assigned a cost $C(x,y)$ based on its proximity to obstacle cells taking into account the vehicle footprint. If the EPW were to occupy a cell (x,y), and any portion of its footprint would overlap with an obstacle cell, $C(x,y)$ would be assigned a lethal cost making it untraversable by the local planner. Otherwise, obstacles were modeled by exponential functions. The resulting costmap was then input to the local planner for trajectory planning. In our implementation, map updates were done asynchronously whenever a scan from either of the sensors was available, with an objective feedback rate of 15 Hz.

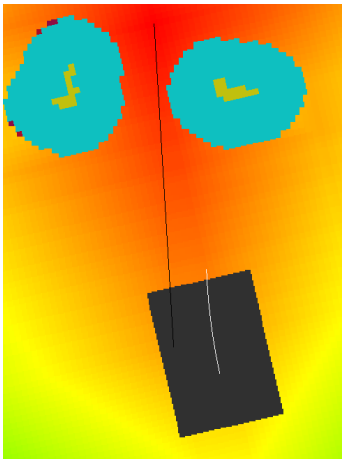

Fig. 7 Costmap $C(x,y)$ of the EPW at a doorway. The black line denotes the desired path.

Figure 7 shows a top-down view of the costmap for the EPW staring at a doorway. The doorway edges are inflated by the potential function to define traversable regions in the costmap. Cyan colored cells correspond to regions with lethal cost, while the transition region from red to dark blue is traversable with decaying cost.

5.2 Planning

The global planner for doorway navigation is very intuitive. Given a doorway position and orientation, the it constructs an objective path down the doorway centerline with the same orientation as the doorway itself. A goal pose $G = [x_g, y_g, \theta_g]$ is then placed on this path the length of the EPW through the doorway.

For local planning, we employed a traditional sample based approach on the input space of the linear and angular control velocities (v, ω) [8]. The range of velocities

sampled was $v \in [0.1, 0.4]$ m/s, and $\omega \in [-0.3, 0.3]$ rad/s. Each sampled trajectory T_i was then evaluated against a cost function

$$C(T_i, M) = C_{obst} + C_{goal} + C_{path} \tag{6}$$

C_{obst} was the maximum obstacle cost of any cell along the specified trajectory. If $C_{obst} > C_{max}$, the obstacle cost was considered fatal and the associated trajectory discarded. C_{goal} was proportional to the distance from the current EPW position to G. Similarly, C_{path} was proportional to the to the distance from the EPW position to the path derived from the doorway's centerline. The optimal trajectory $T^* = \arg\min C(T, M)$ was then selected, and the associated velocity command $(v^*, \omega^*) \in T^*$ was issued to the CoPilot motor controller.

6 Experiments

The doorway navigation behavior for CoPilot is extremely effective. ADA compliant doorways can be navigated with near 100% reliability. The mapping capability also allows CoPilot to identify both static and dynamic obstacles in the environment, and react to these accordingly (i.e., by avoiding the obstacle or stopping when necessary). As additional anecdotal evidence of its performance, CoPilot was recently demonstrated at the headquarters of a major EPW manufacturer. The system was fully integrated into an EPW with a user-friendly interface. When placed in CoPilot driving mode, an icon would appear on the EPW's control display when a doorway was detected. The user then simply pressed a button to initiate door traversal. Although no data was collected during the demonstration, the system was tested by numerous company representatives across a large population of doors. CoPilot successfully traversed every door that the participants attempted.

To support this paper, a more formal experiment was conducted over the course of several days at various locations around the Lehigh University campus. During this time, the EPW was operated in a natural fashion with no attempt to specifically align the wheelchair into a favorable pose. A total of 100 traversals of 100 unique doorway instances were attempted. All were successful. Fig. 8 depicts a sample of the doorways that were traversed. Note that CoPilot was even successful navigating through doorways where structured lighting systems might be expected to struggle, e.g., doorways with glass doors. Fig. 9 shows the variety of starting EPW poses and a probability mass function of the door widths. Note also that the large majority of the doorways were at the lower range of ADA compliant doorway widths.

In terms of "failure modes," the doorway detection system used in CoPilot is susceptible to false positives in that clustered vertical objects meeting the geometry constraints could be interpreted as doorways. For example, two tall file cabinets with a sufficient opening in between would be segmented as a doorway. However, while some may consider this a false positive, others might consider it a feature as it generalizes CoPilot to traversing a larger range of narrow openings. We should emphasize that since migrating to the occupancy histogram approach to doorway segmentation, no false positives have been observed when attempting an actual doorway traversal.

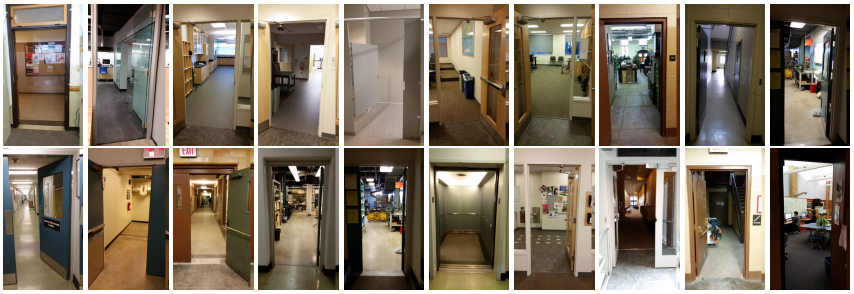

Fig. 8 Examples of the variety of doors successfully traversed

Finally, videos demonstrating the use of CoPilot can be found at `http://loveparkrobotics.com/?p=993` and `http://loveparkrobotics.com/?p=997`. The latter shows CoPilot integrated with a head array controller, an input device not well suited for the doorway navigation tasks. With the EPW in CoPilot mode and the doorway detected (i.e., when it puts the icon on the screen), a momentary tap of the rear switch embedded in the head-array will signal CoPilot to initiate door traversal. Just as with the Joystick mode of operation, the user can steal back control at any time by pushing the head-array switches.

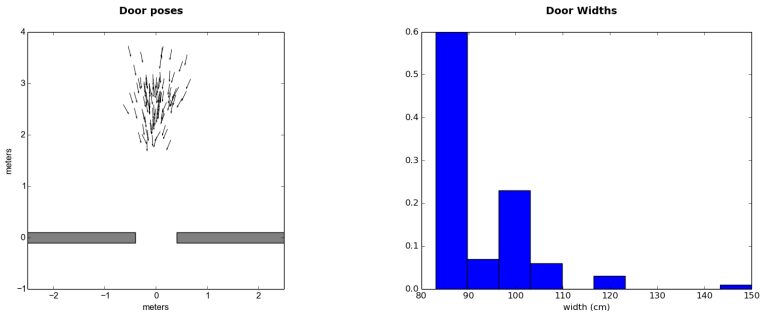

Fig. 9 (Left) visualization of EPW starting poses with respect to a doorway centered at the origin with an orientation of -90° and (right) the probability mass function of the traversed door widths.

7 Conclusion

In this paper we introduced CoPilot, an active driving aid that enables semi-autonomous, cooperative navigation of an EPW for automated doorway detection and traversal. The system was fully integrated into a Quantum Q6 Edge EPW using both joystick and head array controls. For doorway detection, we investigated both feature and histogram based approaches. The latter exhibited at least as good performance with significantly lower computational burden. Coupled with a sample-based planner, CoPilot demonstrated near 100% reliability in detecting and traversing a large population of doorways when employed by a range of users. We are currently

investigating the integration of additional driving aids for CoPilot, to include active braking for real-time collision avoidance and corridor following.

References

1. American's with disability act standards for accessible design. http://www.ada.gov/2010ADAstandards (2010)
2. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
3. Derry, M., Argall, B.: Automated doorway detection for assistive shared-control wheelchairs. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, pp. 1254–1259. IEEE (2013)
4. Fischler, M., Bolles, R.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In: Communications of the ACM (1981)
5. Gerkey, B.P., Konolige, K.: Planning and control in unstructured terrain. In: In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2008)
6. Gupta, S., Arbelaez, P., Malik, J.: Perceptual organization and recognition of indoor scenes from rgb-d images. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 564–571. IEEE (2013)
7. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey vision conference, vol. 15, p. 50. Manchester, UK (1988)
8. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
9. Levine, S.P., Bell, D.A., Jaros, L.A., Simpson, R.C., Koren, Y., Borenstein, J.: The navchair assistive wheelchair navigation system. Rehabilitation Engineering, IEEE Transactions on **7**(4), 443–451 (1999)
10. Marder-Eppstein, E.: costmap_2d. http://ros.org/wiki/costmap_2d. Accessed: 2014-12-14
11. Parikh, S.P., Rao, R., Jung, S.H., Kumar, V., Ostrowski, J.P., Taylor, C.J.: Human robot interaction and usability studies for a smart wheelchair. In: Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 4, pp. 3206–3211. IEEE (2003)
12. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
13. Ren, X., Bo, L., Fox, D.: Rgb-(d) scene labeling: Features and algorithms. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 2759–2766. IEEE (2012)
14. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China (2011)
15. Rusu, R.B., Meeussen, W., Chitta, S., Beetz, M.: Laser-based perception for door and handle identification. In: Advanced Robotics, 2009. ICAR 2009. International Conference on, pp. 1–8. IEEE (2009)
16. Simpson, R., LoPresti, E., Cooper, R.: How many people would benefit from a smart wheelchair? Journal of Rehabilitation Research & Development **45**(1), 53–72 (2008)
17. Teichman, A., Miller, S., Thrun, S.: Unsupervised intrinsic calibration of depth sensors via SLAM. In: Robotics: Science and Systems (2013)
18. U.S. Department of Health and Human Services, Administration on Aging: A profile of older americans: 2011 (2011)
19. Yanco, H.A.: Wheelesley: A robotic wheelchair system: Indoor navigation and user interface. In: Assistive technology and artificial intelligence, pp. 256–268. Springer (1998)