Detection of Docking Locations at Desks for Automated Wheelchairs

Andy Everitt
Department of Mechatronic Engineering
University of Canterbury
aev39@uclive.ac.nz

Supervisor: Richard Green
Department of Computer Science and Software Engineering
University of Canterbury
richard.green@canterbury.ac.nz

Abstract— **This paper proposes a method of identifying the edge of a rectangular desk using an Intel D435 depth camera. The average distance and angle of the desk relative to the camera are calculated, any obstacles beneath the desk are detected and the largest potential docking location for an automated wheelchair is identified. Several computer vision algorithms are used in accomplishing this task, including but not limited to: convolution, Gaussian blur, depth thresholding, morphological opening and closing operations, contours, and pixel to point deprojection. Testing the program showed a successful detection rate between 85-100% in real time at 15 frames per second for a standard rectangular desk in a variety of environments. The accuracy of the calculated distance and angle is within +/-0.01 m and +/-1^o respectively. The detection of the front of the desk and identification of obstacles is improved from prior attempts.**

I. Introduction

Electric wheelchairs are commonly controlled using a joystick with limited speed control. This can make controlling the movement inconvenient for disabled users when navigating tight spaces where a series of small manoeuvring adjustments are required. One situation that causes difficulty for wheelchair users is accurately approaching and docking at desks and tables, being able to automate this process would improve the users comfort in day-to-day life.

Without extensive product testing, it is unclear how willing elderly people would be to use a fully automated wheelchair. It might therefore be preferable to give forced feedback through the wheelchair joystick to indicate which direction the wheelchair suggests would be optimum. This would require additional hardware and cost to be implemented. A software alternative could be used to reduce the cost and complexity of the wheelchair such that a faster movement speed would be used if the user moves in the direction suggested by the wheelchair and a slower speed if they move in a different direction. This principle has been explored by D. Vanhooydonck et al. [1], and B. Faria et al. [2].

Several methods have been used previously to detect potential docking locations. This paper will consider an approach utilising the intel D435 depth camera [4]. It should be able to detect a docking location in situations where the entire desk might not be visible, or the desk only occupies part of the frame. Potential obstructions beneath the surface of the desk such as drawers, boxes, or anything that would prevent the wheelchair from successfully docking also should be detected to aid visually impaired users and in situations where the user might not be able to see beneath the desk. After a docking location has been identified, the system should calculate the required manoeuvres for the wheelchair, after each manoeuvre the docking location should be reassessed, and the required manoeuvres updated.

In association with Dynamic Controls [3]

II. Background

Many of previous attempts at wheelchair and robot docking used either fiducial markers, landmarks, or customised docking structures. They were therefore not optimal for real world use as the desired docking location requires treatment before it can be used. One such paper uses a landmark with a distinctive colour to find the desk, this results are promising however it relies on a pre-treated and uncluttered environment [5].

The system should work with no prior knowledge of the environment. The closest prior papers which use a similar solution will be referenced for this paper [6]–[11].

Comparable systems have been developed for docking automated wheelchairs at objects and locations other than desks. It is worth noting that similar principles have been used by these papers to identify the object of interest [12]–[14].

Lucas Martins [6] and Elliot Langdon [7] both used the Intel R200 depth camera [15], placed in an unobtrusive position on the wheelchair, under the arm in [6], and “somewhere on the front” in [7]. This positioning gives a frontal view of the desk, allowing for features such as the desk top, legs, and drawers to be identified. The size of any available docking space can also be assessed from these viewpoints.

Martins and Langdon take different approaches to identifying a docking area. Langdon uses the colour image from the camera to detect the desk with a Canny edge detector to identify edges, a probabilistic Hough transform then detects the lower edge of the desk by making a series of assumptions that a standard height, rectangular desk is present. The depth image is used to calculate the size of the docking area but is not used in the identification of the docking area itself. If there is a break in the line generated by the Hough transform, the lines are connected. This makes the method more robust however it also means that potential obstacles, such as a chair, are ignored which could be detrimental to the user, this is shown in Fig. 2. Because only the colour image is used it also makes the detection susceptible to the lighting conditions in the room.

Martins method can work from the depth image alone. The Canny edge and Harris corner detection algorithms are used to identify the four external corners of the desk, it is therefore required that the entire desk is in view with each corner occupying a different quadrant of the image for this method to work. Since this will be hard for the user to identify when these criteria have been met and the desk location will change as the wheelchair approaches the desk during the docking manoeuvre, this method is not practical. Obstructions are detected by searching for Harris corners within the external bounding region of the desk. This method shows a 90-100% success rate for detecting the desk in the two ideal examples given in the paper.

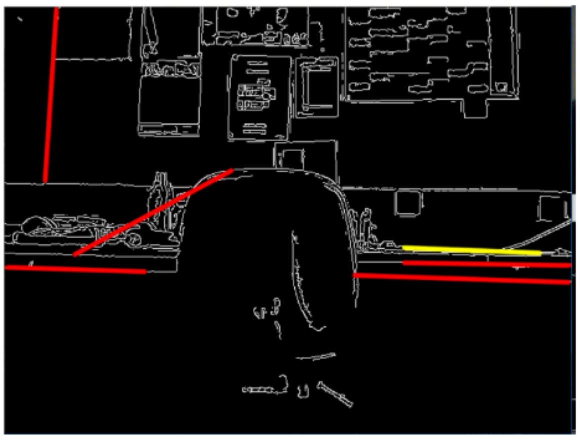
Fig. 1. Hough lines found using Langdon's method.

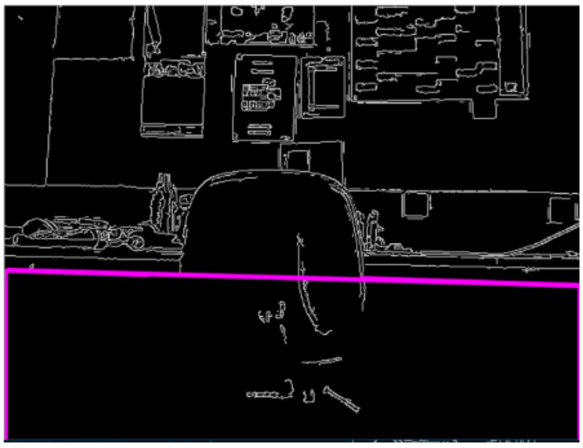
Fig. 2. Docking location identified using Langdon's method. The unidentified obstruction (chair) can be seen.

Langdon's method shows a similar success rate however it has the advantage of not requiring the entire desk to be in view. The main disadvantages of this method are the lack of obstruction detection, and the limited performance when there is a low contrast between the desk and the surroundings.

Another approach was to use a Microsoft Kinect camera mounted high above the wheelchair to get an aerial depth frame of the desk and surroundings [8]. This was shown to produce an effective detection rate of 80-100% for both circular and rectangular desks by generating a point cloud from the depth frame, identifying the boundary of the desk assuming a desk is present, and finally trying to place a virtual box under the edge of the desk and identifying how many points intersect the box therefore finding the most suitable docking location. This implementation used an obtrusive camera position and would struggle to identify objects under the desk due to the obstructed view.

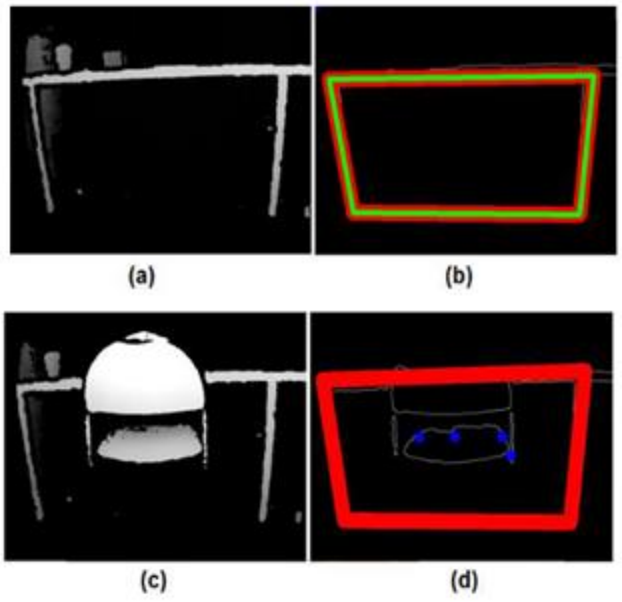

Fig. 3. Martin's method for identifying a valid docking location (green) beneath a detected desk (red), without an obstruction (a&b), and with an obstruction (c&d).

A more complete automated wheelchair was developed by J. Wang, W. Chen, and W. Liao [10]. This used an improved particle filter localisation algorithm to build a map of the environment using a Laser Range Finder and encoder mounted on the wheelchair. The primary purpose of this paper was automating a wheelchair to navigate narrow and crowded environments however it was shown that this implementation could be used to automatically dock at a desk. The user would select the desired destination on a touchscreen and an A* search algorithm continually calculated the optimum route accounting for any new obstacles that might enter the path such as pedestrians. It is unclear from the paper how exactly the desk was identified other than the user input, however the success rate was 100% for the single desk it was tested on.

Neural networks can be trained to detect objects reliably and in a wider range of situations than traditional computer vision algorithms, however they required extensive training to reach this level. One such implementation of a neural network used to detect a desk is shown by C. Weber, S. Wermter, and A. Zochios [11].

A depth camera will be used to implement the desk detection algorithm. The depth stream produced by the R200 camera was inconsistent, making features hard to identify; Fig. 4 illustrates this issue.

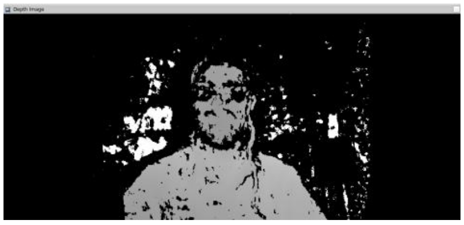

Fig. 4. R200 depth image, black areas are lacking a depth value [16].

The Intel D435 depth camera [4] provides a more accurate and consistent depth image with fewer missing values compared to the R200. The larger operating range, resolution, and field of view should prove advantageous in improving the reliability of docking location detection.

III. Method

This paper proposes a new method to identify the edge of a rectangular desk using a D435 depth camera mounted on the front of the wheelchair at the same height as the arms.

As the D435 has multiple sensors to capture the depth and colour frames, the two streams must first be aligned so that the pixels correlate between the frames, allowing for the results from the depth frame analysis to be accurately represented on the colour frame. All the algorithms used are applied only to the depth frame.

A. *Identifying the Front of the Desk*

An assumption is initially made that a desk is present in the frame and that there are no significant obstructions blocking the desk from view. The first step is to find the height of the desk. This is done by using horizontal convolution to average the depth value of each row in the frame. The result is blurred using a Gaussian blur algorithm with a kernel size of (1,5) to reduce the impact of noise in the frame that originates from missing depth values. A circular buffer then calculates minimum depth from the 3 most recent blurred frames. The depth value gained from this method gives a reasonable estimation of the distance of the desk if it occupies more than 50% of the frame horizontally.

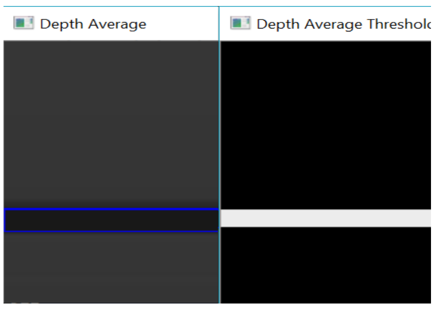

Fig. 5. (left) The blurred depth frame after horizontal convolution. (right) The thresholded depth frame using the minimum depth value +/- 20% as the upper and lower limits

The minimum average depth value is then used to threshold the blurred depth frame as shown in Fig. 5. This produces a binary image that a contour algorithm can be applied to, allowing the largest white area to be identified which is assumed to be the front edge of the desk. The contour algorithm is based off Greens Theorem [17] followed by calculating the image moments.

After the height of the desk has been found, the width of the desk must also be identified. The method used is to threshold a specific region of the depth frame using the same upper and lower limits as in the previous step. Morphological opening and closing operations are applied to the thresholded image to reduce the impact of noise. The region of interest (ROI) used are the rows of the depth frame identified to contain the front edge of the desk. By finding the largest contour of the thresholded region, a bounding rectangle of the front of the desk is identified.

B. *Calculating the Distance and Angle of the Desk*

By successfully segmenting the front edge of a rectangular desk from the image, the distance and angle of the desk relative to the camera can be found. The depth values from the gained from the D435 must first be converted into real world coordinates. Pixel to point deprojection allows (x,y,z) coordinates, in meters, of a specific pixel relative to the camera to be identified. Deprojection converts the 2D depth stream back into the 3D point cloud originally captured by the D435 and is dependent on the camera intrinsics. Two points on the front of the desk are used to determine the distance and angle, highlighted by the green boxes in Fig. 6. If no depth value is present at the corresponding pixel then the point is moved progressively further away from the centre horizontally until a depth value is found.

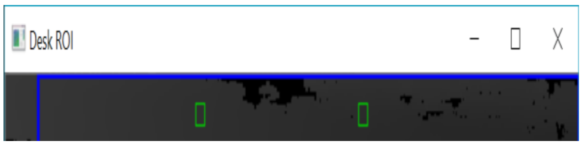

Fig. 6. Depth values of the front of the desk ROI. Green boxes indicate the points used to calculate the distance and angle of the desk relative to the D435.

Trigonometry is used to calculate the average distance and angle of the front of the desk is meters and degrees respectively. Fig. 7 illustrates how these dimensions relate to the real world.

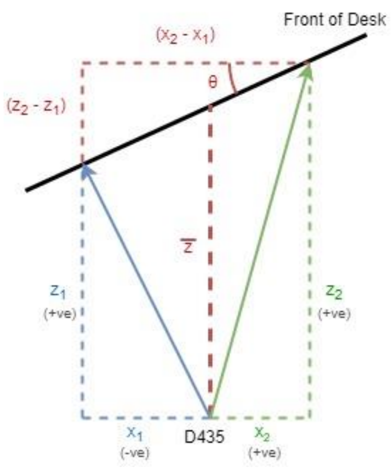

Fig. 7. Plan view of the desk relative to the D435, showing the dimensions that are known of the two points (blue & green), and the dimensions that are desired (red).

$$\overline{z} = (z_2 + z_1) / 2 \quad (1)$$

$$\theta = \tan^{-1}((z_2 - z_1)/(x_2 - x_1)) \quad (2)$$

C. *Obstacle Detection*

To ensure that there is a viable docking location beneath the desk for the wheelchair, any obstacles must first be detected. The area beneath the desk is assumed to be from the bottom of the depth frame up to the lower edge of the desk. When fully docked at a desk, it is typical for the wheelchair to be partially under the desk; the exact distance the wheelchair will be past the front of the desk is specific to the wheelchair. In this paper the upper limit for the range where obstacles are detected is 120% the average distance of the desk. The lower limit is 0, by thresholding the depth frame of the new ROI beneath the desk using these limits, any obstacle between the camera and just past the front of the desk is detected.

Contours are again used to detect the white regions of the ROI shown in Fig. 8.

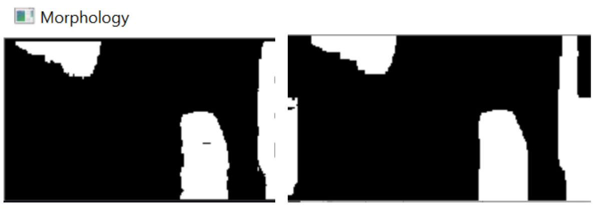

Fig. 8. (left) Thresholded depth frame beneath the desk. (right) After morpholocial opening & closing operations are applied to the thresholded image.

D. *Finding a Docking Location*

To reduce the amount of computational time required to find a possible docking location for the wheelchair, each column of pixels in the ROI under the desk is analysed sequentially. If there is an obstruction present in a column then it is marked unviable. Adjacent viable columns are grouped, and the largest group is determined to be the best docking location.

Once the docking location has been found, pixel to point deprojection is used to measure the size in meters and compare it to a known width for the wheelchair.

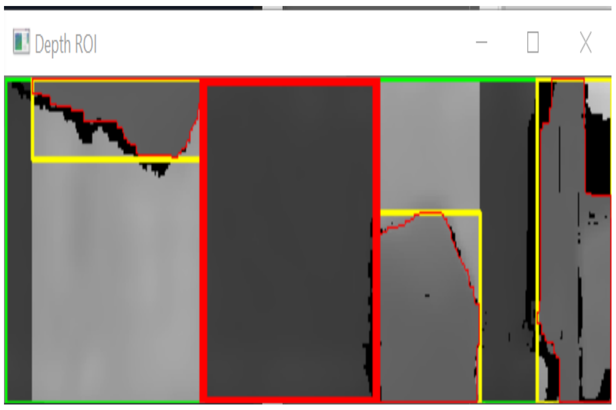

Fig. 9. Depth frame beneath a desk with multiple obstacles highlighted by yellow bounding rectangles. Viable docking locations have been shaded in dark grey, the largest has been identified and measured, if the location is large enough for the wheelchair then it appears as a bold white rectangle, else it is a bold red rectangle.

IV. RESULTS

The algorithm was programmed using Python 3.6 with the OpenCV 3.4.1 library used to provide the relevant computer vision algorithms and the Intel RealSense SDK 2.0 library to interface with the D435 camera. The program was tested on two Windows 10 computers using a core m3-6Y30 with 4GB of RAM, and a core i7-4720HQ with a GTX980m graphics card and 16GB of RAM respectively.

A. *Distance and Angle*

The method developed in this paper can be used to successfully identify a desk and the obstacles beneath it even if the desk is partially obscured. The average distance and angle of the desk relative to the D435 calculated by the program has been compared to the real distance and angle measured by a ruler and protractor respectively. The results show that the distance is reliable from 0.2 m to 2.5 m +/-0.01 m, below this range the D435 has a higher percentage of false depth values, above the upper limit program struggles to accurately find the desk in a typical room. The angle calculated by the program was reliable to +/-1^o within the range of +/-15^o, outside of these limits the full width of the desk was not properly detected, and the calculated angle gave spurious results. The results are summarised in TABLE I. The method of measuring the accuracy is crude, with the chance of significant human error, however it gives an initial view showing that the both the calculated distance and angle are giving viable results for a rectangular desk irrespective of the level of clutter on and around the desk. The two desks used to measure the accuracy and reliability are shown in Fig. 12 and Fig. 11.

Compared to previous attempts at implementing desk detection, neither L. Martins [6] or E. Langdon [7] showed evidence of calculating the distance in meters, and neither made an attempt at calculating the angle of the desk. The work done in [8] shows effective identification of docking locations using point clouds and as such it could be implied that accurate distance and angle calculations could be made, however no such verification is covered in the paper.

TABLE I. Distance and Angle Accuracy

Situation	**Distance (m)**		**Angle (º)**	
	Calculated	***Measured*** [a]	***Calculated***	***Measured*** [b]
Plain desk	0.336	0.345	-1.781	-1.5
	0.457	0.465	2.503	3.0
	0.841	0.830	5.021	4.5
	0.202	0.205	-10.658	-9.0
	0.559	0.665	16.141	12.0
Cluttered desk	1.202	1.180	0.782	0.5
	0.862	0.890	-2.388	-2.0
	0.514	0.510	0.238	0.5

a. Accuracy of the ruler used was +/-0.005 m

b. Accuracy of the protractor used was +/-0.5º

B. *Obstacle Detection & Docking Locations*

The maximum number of obstacles detected by the program is unlimited, this is an improvement over both L. Martins [6] and E. Langdon [7]. The method used to find a viable docking location could be improved to ignore small obstacles towards the top or bottom of the area beneath the desk if it is found that there is enough space for the wheelchair to fit. This would require more processing power for the program to run in real time.

Obstacles are often detected accurately however if multiple obstacles are close then the program can group them, as in Fig. 12, using a single contour which makes a potentially valid docking location become invalid. Another limitation is the program does not have any memory, hence as the wheelchair approaches the desk and obstacles are removed from the field of view, they are considered to exist no longer. Changing the location of the D435 could improve this.

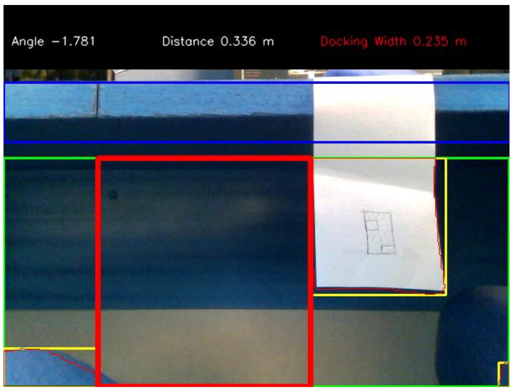

Fig. 10. Example desk and docking location detection using the depth frame and overlaid onto the colour frame.

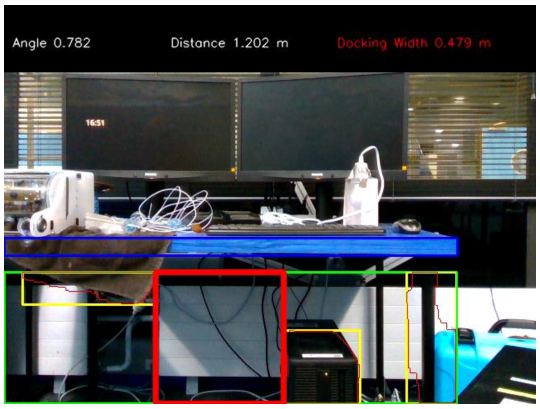

Fig. 11. A cluttered desk successfully identified and the optimum docking location highlighted.

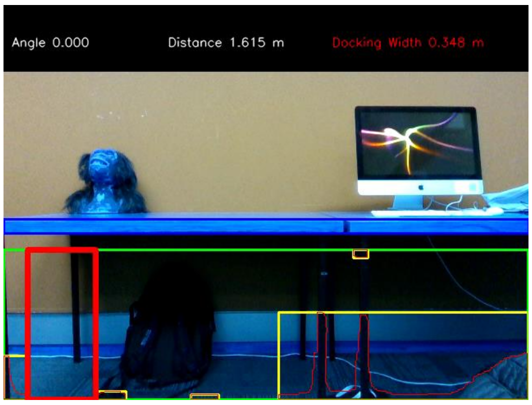

Fig. 12. Cables on the floor have been detected as an obstacle.

TABLE II. Legend for the desk detection colour frame output.

Colour	**Object**
Blue	Front edge of the desk
Green	Area beneath the desk
Yellow	Bounding rectangle of obstacles beneath the desk
Red (thin)	Contours of obstacles beneath the desk
White	Largest docking location that is wide enough for the wheelchair
Red (bold)	Largest docking location that is not wide enough for the wheelchair

C. *Performance*

The program in this paper has been coded using Python, this limits its ability to work in real time, however a resolution of 480x640p at a frame rate of 5-15 frames per second (fps) can be achieved on both a low spec and mid-high spec Windows 10 computer. This quality is enough for a proof of concept and could be improved by using C as the coding language. When testing the ability for the program to detect a desk and the obstacles beneath it for a single frame, there was no significant difference between 480x640p and 720x1280p. The frame rate when using the higher resolution was significantly worse when attempting to run in real time with the low spec PC unable to achieve 1 fps.

The accuracy of the program at detecting a desk in real time was comparable to previous attempts, a summary of tests done on different desks in different environments is given in TABLE III. A “True Positive” indicates where a desk has been successfully detected, a “False Positive” relates to when the program detects the desk in the wrong location, a “True Negative” is when the program correctly detects that no desk is present, a “False Negative” is when the program cannot find any desks when one exists in the frame.

TABLE III. Desk Detection Accuracy

Situation	***True Positives***	***False Positives***	***True Negatives***	***False Negatives***
Plain desk, 2m -> 0.4m	100%	0%	n/a	0%
Angle test, +/- 10 degrees, 0.5m	87.3%	0%	n/a	13.7%
Really cluttered desk 1.2m -> 0.4m	94.2%	2.1%	n/a	3.7%
No movement, docking location detection	100%	0%	n/a	0%
Desk with 2 worksurfaces & human leg	72.9%	27.1%	0%	0%
Round table	50.7%	10.2%	n/a	39.1%

An example of a “False Positive” for the cluttered desk is shown in Fig. 13, in this situation the largest contour in the thresholded horizontal convolution frame is not the desk. Round tables can also be detected; however the performance of the program is significantly reduced.

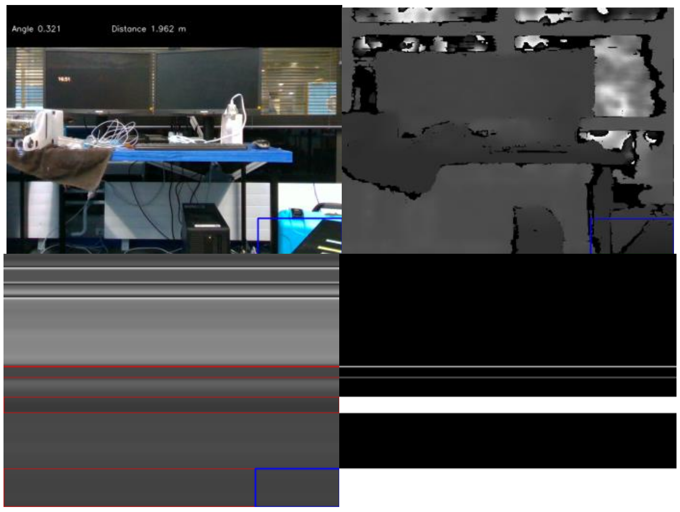

Fig. 13. A false positive where the program has detected the wrong part of the image as the desk, shown by the blue rectangle.

V. Conclusion & Future Work

The results show a clear improvement in detecting a rectangular desk and the largest viable docking location in a variety of environments. The program is independent of lighting conditions, as it only uses the depth frame, making its use more flexible. Issues occur when an object occupies more of the image horizontally than the desk and is situated closer to the camera. The program also does not know how to easily identify when there is not a desk present in the frame, to improve this a convolutional neural network (CNN) could be used as a first pass to first find if a desk is present in the frame. This would increase the initial computational time; however it would not be necessary to run this after a desk has been found. A large amount of training would be required for the CNN to reliably identify a desk in any situation. By polling a handful of wheelchair users across a range of ages, it was found that the assumption that the user would only attempt to use the program when facing a desk was agreed upon unanimously.

The next step would be to implement the calculated distance and angle information into real world speed control on the wheelchair. After this has been achieved, tracking of obstacles after they have moved out of the frame should be implemented. A point cloud map generated using SLAM [18] could be beneficial in tracking past obstacles. A Kalman or particle filter could also be implemented to track the edge of the desk to detect if it has moved an unexpected amount between frames, therefore identifying when the desk has been detected in the wrong location.

References

[1] D. Vanhooydonck *et al.*, ‘Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model’, *Robot. Auton. Syst.*, vol. 58, pp. 963–977, Jan. 2010.

[2] B. M. Faria, L. P. Reis, and N. Lau, ‘A methodology for creating an adapted command language for driving an intelligent wheelchair’, *J. Intell. Robot. Syst.*, vol. 80, no. 3–4, pp. 609–623, Dec. 2015.

[3] ‘Home | Dynamic Controls’. [Online]. Available: https://dynamiccontrols.com/. [Accessed: 07-Jun-2018].

[4] ‘Intel® RealSense^{TM} Depth Camera D435 Product Specifications’, *Intel® ARK (Product Specs)*. [Online]. Available: https://ark.intel.com/products/128255/Intel-RealSense-Depth-Camera-D435. [Accessed: 17-Apr-2018].

[5] D. Muse, C. Weber, and S. Wermter, ‘Robot docking based on omnidirectional vision and reinforcement learning’, *Knowl.-Based Syst.*, vol. 19, no. 5, pp. 324–332, 01 2006.

[6] Lucas Amilton Martins and Richard Green, ‘Automated Recognition of Docking Locations for Electric-Powered Wheelchairs’, University of Canterbury, Computer Vision Lab, 2016.

[7] Elliot Langdon and Richard Green, ‘Recognition of Docking Locations for Electric Wheelchairs’, University of Canterbury, Computer Vision Lab, 2017.

[8] S. Jain and B. Argall, ‘Automated perception of safe docking locations with alignment information for assistive wheelchairs’, in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4997–5002.

[9] M. M. Williamson, R. Murray-Smith, and V. Hansen, ‘Robot docking using mixtures of gaussians’, in *Advances in Neural Information Processing Systems*, 1999, pp. 945–951.

[10] J. Wang, W. Chen, and W. Liao, ‘An Improved Localization and Navigation Method for Intelligent Wheelchair In Narrow and Crowded Environments’, *IFAC Proc. Vol.*, vol. 46, no. 13, pp. 389–394, 2013.

[11] C. Weber, S. Wermter, and A. Zochios, ‘Robot docking with neural vision and reinforcement’, *Knowl.-Based Syst.*, vol. 17, pp. 165–172, Jan. 2004.

[12] 'A docking control method in narrow space for intelligent wheelchair', *2012 IEEE Int. Conf. Mechatron. Autom. Mechatron. Autom. ICMA 2012 Int. Conf. On*, p. 1615, 2012.

[13] 'Vision-based control of a smart wheelchair for the automated transport and retrieval system (ATRS)', *Proc. 2006 IEEE Int. Conf. Robot. Autom. 2006 ICRA 2006 Robot. Autom. 2006 ICRA 2006 Proc. 2006 IEEE Int. Conf. On*, p. 3423, 2006.

[14] Z. (1 Wei 2), W. (1 Chen 2), and J. (1 Wang 2), *3D semantic map-based shared control for smart wheelchair*, vol. 7507 LNAI. 2012.

[15] 'Intel® RealSense^{TM} Camera R200 Product Specifications'. [Online]. Available: https://ark.intel.com/products/92256/Intel-RealSense-Camera-R200. [Accessed: 17-Apr-2018].

[16] 'Intel RealSense R200 poor details in the depth stream · Issue #263 · intel-ros/realsense', *GitHub*. [Online]. Available: https://github.com/intel-ros/realsense/issues/263. [Accessed: 17-Apr-2018].

[17] E. W. Weisstein, 'Green's Theorem'. [Online]. Available: http://mathworld.wolfram.com/GreensTheorem.html. [Accessed: 06-Jun-2018].

[18] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, 'MonoSLAM: Real-Time Single Camera SLAM', *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.