Real-Time 3D Door Detection and Classification on a Low-Power Device

João Gaspar Ramôa
NOVA LINCS
Universidade da Beira Interior
Covilhã, Portugal
Email: gaspar.gomes{at}ubi.pt

Luís A. Alexandre
NOVA LINCS
Universidade da Beira Interior
Covilhã, Portugal
Email: luis.alexandre{at}ubi.pt

S. Mogo
Universidade da Beira Interior
Covilhã, Portugal
Email: sipmogo{at}gmail.com

***Abstract*—In this paper, we propose two methods for door classification with the goal to help and improve robot navigation in indoor spaces and to be used in other areas and applications since it is not limited to door detection as other related works. Our methods work offline, in low-powered computers as the *Jetson Nano*, in real-time with the ability to differentiate between open, closed and semi-open doors. We use the 3D object classification, *PointNet* and real-time semantic segmentation algorithms *FastFCN* and *FC-HarDNet*. We built a 3D and RGB dataset using a 3D *Realsense* camera D435 with door images in several indoor environments that we make freely available. Both methods are analysed taking into account their accuracy and the speed of the algorithm in a low powered computer. We conclude that it is possible to have a door classification algorithm running in real-time on a low-power device.**

I. Introduction

Every day new mobile robots with better components and software are built for several purposes, from smart vacuum cleaners to intelligent housekeepers that help people with difficulties in their daily tasks.

Door detection and classification are crucial for this type of intelligent systems to safely navigate in indoor spaces. Usually, the task of these systems implies moving between rooms and dealing with doors. It is required to provide the robot with the necessary information about the door so it can safely navigate between rooms without any problem.

Door classification is not restricted to mobile robots and robotics, it can be applied to other problems and areas like helping visually impaired people to safely move between rooms by providing information about the existing doors and their status.

In this paper, we propose one method for door classification that uses only 3D information and another that uses 3D and RGB information. We focus on an approach that works in low-power systems such as the single board computers *Nvidia Jetson Nano* or the *Raspberry Pi*. Our methods work in real-time despite running in low-power systems with weak GPU and are based in the 3D Point cloud classification method *PointNet* [1]. We built a dataset with 3D and RGB images with 3 different classes, open doors, closed doors and semi-open doors, using a 3D Realsense Camera. This dataset was used to train the *PointNet* and to test our developed methods. The methods were also compared in terms of accuracy and speed. We used a single board computer equipped with a 3D Camera and powered by a power-bank. This mobile system was used for testing the speed of our methods. The focus of this work was in the door classification algorithm, without concerning about the rest of the robot hardware.

In short, the contributions of the paper are:

- Two methods, one uses 3D and RGB data and the other uses only 3D data, for door classification, both working in real-time in low-power systems.
- A labelled dataset with RGB and depth images of closed, open and semi-open doors.
- A dataset for semantic segmentation algorithms with annotated doors and door frames.

The remainder of this paper is structured as follows: Section II does an overview of the state-of-the-art. Section III describes the door classification and detection problem. Section IV describes the proposed methods for door classification. Section V describes the dataset built. Section VI describes the experiments and results of our methods. Section VII presents the conclusions and future work.

II. Related Work

There are already a vast number of studies that used door detection and classification for robot navigation tasks as moving between rooms, robotic handle grasping and others. Some have used sonar sensors with visual information, [2], others used only colour and shape information, [3], some have used simple feature extractors, [4], [5] and others have used more modern methods like CNN (Convolutional neural networks), [6] and the use of 3D information, [7], [8], [9] and [10].

Using visual information and ultrasonic sensors to traverse doors was an approach used in [2]. The goal was to traverse an open door with a certain opening angle using a B21 mobile robot equipped with a CCD camera sensor and 24 sonar sensors. The door traverse was divided into two sub-tasks, the door identification and the door crossing. The door identification which was the sub-task of interest for this work, used a vertical *Sobel* filter applied to the grey-scaled image. If there was a column wider than 35 pixels in the filtered image it would mean that the door was in the image. The sonar sensors were used when the robot approached the door at a distance of 1 meter to confirm if it was a door or not.

In [4], an integrated solution to recognize a door and its knob in an office environment using a humanoid platform is proposed. The goal is for the humanoid to recognize a closed-door and its knob, open the same door and pass through it. To recognize a door they match the features of the input image with the features of a reference image using the STAR Detector [11] as the feature extractor and an on-line randomised tree classifier to match the feature points. If the door is in the scene, the matched feature 3D points are computed and used so that the robot walks towards the door.

The use of colour and shape information can be sufficient for identifying features to efficiently detect doors. The approach in [3] used two neural networks classifiers for recognizing specific components of the door. One was trained for detecting the top, left and the right bar of the door and the other was trained for detecting the corners of the door. A door is detected if at least 3 of these components are detected and have the proper geometric configuration.

In [6], a method is implemented for detecting doors/cabinets and its knobs for robotic grasping using a 3D Kinect camera. It uses a CNN to recognize, detect and segment the region of interest in the image. The CNN used was the *YOLO* Detection System trained with 510 images of doors and 420 of cabinets from the *ImageNet* dataset. After obtaining the *ROI*, the depth information from the 3D camera is used to obtain handle point clouds for robot grasping.

Like the previous approach, in [7], a Kinect sensor is used for door detecting but, this method uses only Depth information. The camera sometimes produces missing points in the depth image, and the algorithm is based in the largest cluster of missing pixels in the depth image. The total number of holes indicates the status of the door, (open or semi-open). The main advantage of this method is that it works with low-resolution depth images.

There are methods developed under a 6D-space framework, like [8], that use both colour (RGB) and geometric information (XYZ) for door detection. For detecting open doors they detect rectangular point cloud data gaps in the wall planes. The detection of closed doors is based in the discontinuities in the colour domain and in the depth dimension. It also does door classification between open and closed doors. The improved version of this algorithm, [10], can even distinguish semi-open doors using the set of points next to the door to calculate the opening angle. Another improvement in [10] was in the dataset, which is larger in size, complexity and variety.

In [9], a method is proposed that uses 3D information for door detection without using a dependent training-set detection algorithm. Initially, the point cloud containing all the scene, including the door, is prepossessed using a voxel-grid filter to reduce its density and its normal vectors are calculated. A region growing algorithm based on the pre-calculated normals is used to separate the door plane from the rest of the point cloud and after that, feature extraction is used to get the edges of the door and the doorknob.

TABLE I
RELATED WORK COMPARISON (DOOR DETECTION).

Method	3D	Closed doors	Open doors	Semi-open doors	Real-time
Monasterio [2]	×	×	✓	×	-
Cicirelli [3]	×	✓	×	✓	×
Kwak [4], Chen, [5]	×	✓	×	×	✓
Llopart [6]	✓	✓	✓	✓	✓
Yuan [7]	✓	×	✓	✓	-
Quintana [8]	✓	✓	✓	×	-
Borgsen [9]	✓	✓	×	×	×
Quintana [10]	✓	✓	✓	✓	-
Ours	✓	✓	✓	✓	✓

To detect doors 3D cameras or sonar sensors are not required, a simple RGB camera can do the job as in [5], focusing on real-time, low-cost and low-power systems. This work used the *Adaboost* algorithm to combine multiple weak classifiers into a strong classifier. The weak classifiers were based in features such as detecting pairs of vertical lines, detecting the concavity between the wall and the doorframe, texture and colour and others. They built a dataset with 309 door RGB images, 100 for training their algorithm and the rest for testing.

Table I summarises the previous approaches and related work to detect and classify doors in indoor spaces, categorising each method studied. Although most of the approaches just do door detection and not classification, as we did in this work, they have the same goal, to provide the robot with the necessary information to move between rooms, and that is the reason why we included them in this paper. The first column states whether the method uses 3D information or not. The following 3 columns states the applicability of the method (closed, open or semi-open doors). The last column focus on whether the method works in real-time or not, based on the experimental results of each method. Four of the methods do not present information regarding their speed and are marked with a "-".

III. Problem Definition

Mobile robots nowadays are used for multiple tasks and purposes in several indoor environments as security guard robots, tour guide robots, vacuum cleaners and others. Usually, in these environments, the robot has to navigate safely between rooms and the biggest obstacles are the doors. The mobile system normally must be able to detect the door in the scene to move to another room. In more complex situations, the robot has not only to detect the door but also has to classify it to decide its next move. Door detection is used in situations where the door is always closed or always open. Door classification is useful in hard situations where the door can be open, closed or semi-open. We decide to work with door classification because it can be used by the robot to solve more complex tasks.

In this work, we focus only in the door classification using computer vision algorithms and methods without concerning

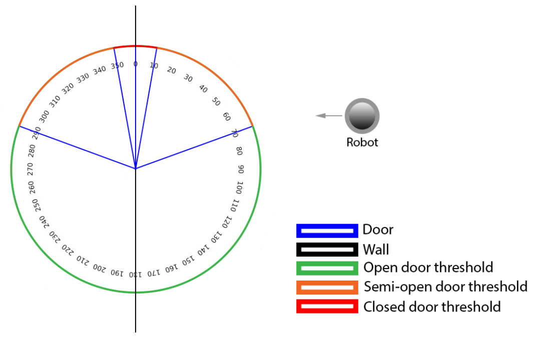

Fig. 1. Opening angles thresholds for closed, semi-open and open doors.

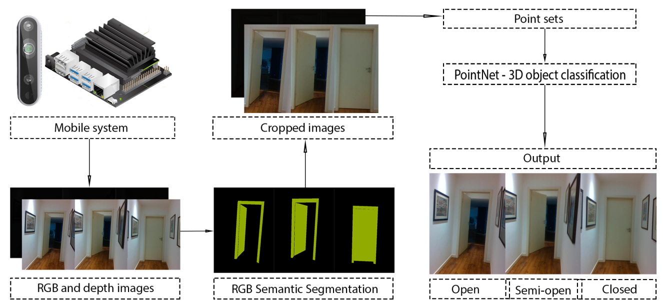

Fig. 2. Algorithm of Method *A* (2D semantic segmentation and 3D object classification).

the after processes and the action that the robot will take according to the opening of the door. We propose that if the door is classified as closed, the robot must call a human for help. If it is open the robot can simply go through it and if it is semi-open, the robot can either get around it or try to open it simply by gently pushing it.

The door can be classified as open, closed and semi-open depending on the door opening angle (angle between the door and the wall where the door is inserted). Doors with opening angles between 0 and 10 degrees are closed, with opening angles between 10 and 70 degrees are considered semi-open and with opening angles higher than 70 degrees are considered open. We also take into account the case of doors with negative angles. This classification is done in the same way as the previous one but with the corresponding negative angles. Figure 1 treats the door opening angles thresholds from a perspective seen from above.

Although the door opening angle is the most important classification factor it is not the only one. We also have in consideration the position of the viewer in relation to the door as a classification factor. For example, the door has an opening angle of 75 degrees but the position of the robot does not allow it to walk forward and go through the door without the need to get around it. In this case, we considered the door as semi-open, because the robot must get around it to go through it. We decide to do this approach of the problem with the objective of applying this work to other areas.

IV. Proposed Method

We propose two different methods for door classification. The first, discussed below in section IV-A, uses the combination of semantic segmentation algorithms with the 3D object classification method. The second, presented in section IV-B, uses only the 3D object classification method.

A. Method A - 2D Semantic Segmentation and 3D Object Classification

For this method, we use both RGB and depth information for door classification, using both of our datasets.

After receiving both RGB and depth frames from the Realsense 3D camera we use a semantic segmentation method and draw a bounding box around the biggest area of pixels of the "door" class resulting from the semantic segmentation. The depth channel is aligned with the RGB channel. The depth image is cropped according to the bounding box of the RGB image, resulting in a depth image with only the door. Using the *Open3D* library [12], we converted the cropped depth image to a grey-scale point cloud. The point cloud goes to the 3D object classification *PointNet*, [1], trained with our dataset for *PointNet*, with 3 classes. The *PointNet* does the inference with the point cloud and returns the result of the classification.

Figure 2 represents the described method.

Regarding the semantic segmentation algorithms we use the *FastFCN Rethinking Dilated Convolution in the Backbone for Semantic Segmentation*, [13] and the *Fully Convolutional HarDNet* which was based in the *HarDNet: A Low Memory Traffic Network*, [14]. The *FastFCN* was used because its test score was in the first three best global ranks for semantic segmentation in the *ADE20K* dataset. We also tried to implement the *EncNet*, [15] which is the network that the *FastFCN* is based on, but the implementation provided could only work in multi-GPU machines. The *ADE20K* dataset is very important for door semantic segmentation since the "door" class is labelled and it has indoor images with doors. If a semantic segmentation method performs well in this dataset it will also perform well in ours. We also used the *FC-HarDNet* because it had the best global rank metric value for real-time semantic segmentation in the *Cityscapes* dataset. We used it because it was faster than the previous method and we were pursuing a real-time door classification method.

As the 3D object classification method, we used the *PointNet*. This method accepts unordered point sets and classifies them according to their 3D shape. We used the provided repository in [1] and changed the default dataset which was the *ShapeNet* to our dataset adjusting the data loader and the number of classes acordingly.

The difference between the methods of this family is in the 2D semantic segmentation algorithm used (*FastFCN* and *FC-HarDNet*). These methods are compared later in section VI.

B. Method B - 3D Object Classification

For this method, we only used the 3D object classification method *PointNet*. Instead of receiving both RGB and depth

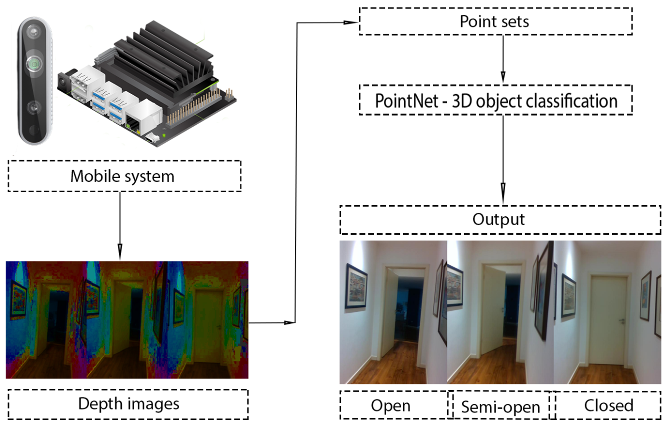

Fig. 3. Algorithm of Method *B* (only 3D object classification).

data, we use only the depth data. The depth data is converted to a point cloud using the *Open3D* library and then converted to a point set. These point sets are the input of the *PointNet*. Unlike the previous methods, *A*, this method uses the entire point cloud without cutting it because we do not have the bounding box of the door. Although the point cloud is bigger, because it is not cropped, the number of points that enter the *PointNet* is the same. This happens because the *PointNet* has a parameter, ”number of points”, which we will denote by K, that defines the number of points of the input point set that will be randomly selected and classified. This method's algorithm is visually represented in Figure 3.

This method is faster than the previous one in terms of frames per second because it does not use semantic segmentation algorithms and uses the same 3D object classification algorithm.

One parameter we can adjust in the method, is if the point cloud undergoes uniform downsampling or not. The K parameter makes *PointNet* randomly select that number of points and if we have a big point set we might get a set that does not represent uniformly the depth data. We use the uniform downsampling algorithm from *Open3D* in the original point cloud, with approximately 300000 points, to get a downsampled version of the same 10 times smaller. The default value of K was 2500 which was too small for our point sets (300000 or 30000). This number was increased to 10000 and could not be further increased because of the small GPU memory of the mobile system.

V. Datasets

We built two different datasets, one for the 3D object classification algorithm *PointNet*,[1], and other for the semantics segmentation algorithms, [13] and [14].

For building the datasets, we used a portable system constituted by a *Raspberry Pi 3B+* powered by a power-bank with a 3D Realsense Camera, model D435. This camera has a horizontal viewing angle (86 degrees) higher than the vertical viewing angle (57 degrees). We rotated the camera 90 degrees to switch the angles with the purpose of including all the door area in the image. The camera was placed 135cm above the floor.

TABLE II
Dataset comparison with related work.

DataSet	3D	RGB	Number of samples
Chen [5]	×	✓	309
Llopart [6]	×	✓	510
Quintana [10]	✓	×	35
Ours	✓	✓	**1206**

We captured several images of doors and its surroundings with different textures and sizes. Some images have obstacles that obstruct and hide part of the door such as, chairs, tables, furniture and even persons. The goal was to create a more generic and realistic real-world dataset. We also changed the pose to get different perspectives of the same door. The images captured are from our university, public places and people's houses.

A. Dataset for PointNet

The dataset for *PointNet* is constituted by RGB images and corresponding depth images both with the size 480 x 640 pixels. The depth images are in grey-scale with pixels values between 0 and 255 and we used a depth scale equal to 1/16. The depth in meters is equal to depth scale * pixel value, for example, if the pixel value is equal to 32 it means that that pixel is 2 meters away from the viewer (1/16 * 32 = 2). In total this dataset has 1206 door images, 468 of those are images of closed doors, 588 open doors and 150 semi-open doors. We divided, the set in train, validation and test. For the test and validation set, we used 20 samples of each class giving a total of 60 samples for test and 60 for validation. We used the remaining samples of each class to built the training set with a total of 1086 images.

Table II compares our dataset for PointNet with the datasets built in related works. From table II we can conclude that our dataset has more samples than the other datasets and has RGB and Depth images which none of the related work databases has. Figure 4, represents a few images from our dataset. This dataset is freely available in the following link: [1]

B. Dataset for Semantic Segmentation

We also built a dataset for training the semantic segmentation algorithms which were used for door classification. This dataset used part of the RGB images from the PointNet Dataset(V-A). To built this dataset we used the Computer Vision Annotation Tool (*CVAT*), [16]. This tool allows us to draw polygons in the RGB images that represent one class. Using this mode we drew rectangles around the doors and door frames in each image.

This dataset has 240 grey-scaled images with the size 480 x 640. These images were randomly chosen before annotated. As we are just concerned with the doors and door frames, we only used two classes in this dataset. The pixel value is 1 if it corresponds to a door or door frame, and is 2 if it does not.

[1]https://github.com/gasparramoa/DoorDetect-Class-Dataset

Fig. 4. Sample images from our RGB dataset.

VI. Experiments and Discussion

We compared our methods against each other in real-time scenarios. We did not compare with the methods of the related work because their focus was in door detection while ours is in door classification.

Both the semantic segmentation algorithms as the *PointNet* were trained in a machine with 16GB of RAM memory, a 256GB SDD disk, an AMD Ryzen 7 2700 processor with 16 Threads and a GeForce GTX 1080 ti graphic with 12 GB. The mobile system where the speed tests were made is composed by a *Jetson Nano* in 10 Watt mode with a Realsense 3D camera D435 without fan.

It is important to mention that we did not change any of the algorithms used in our methods as the *PointNet*, *FastFCN* and *FC-HarDNet*. We only changed the data loaders and did the necessary configurations to work with our data sets.

A. Method A

We compared the two aforementioned semantic segmentation algorithms for method *A*. We split our semantic segmentation dataset presented in section V-B, in train and test sets with 200 and 40 samples respectively. We used the pixel accuracy and the intersection over union (*IoU*) as the evaluation metrics. The intersection over union is the area of superposition between the predicted segmentation and the ground truth divided by the area of union of these last two. The pixel accuracy is the percent of pixels in the input image that are classified correctly. We also compared the training and inference time in the aforementioned desktop computer.

The *FastFCN* algorithm has better results in pixel accuracy and *IoU* in the test set when compared with the *FC-HarDNet* algorithm (III) but, as we mentioned, our focus is in real-time door classification methods. The *FC-HarDNet* is not as good as the *FastFCN* at door segmentation but it has a much

TABLE III
Comparison between using the *FastFCN* and the *FC-HarDNet* algorithms in method A.

Method A with	Test pixel accuracy	IoU	Training time (sec)	Inference time (sec)
FastFCN	0.909	0.808	567	0.515
FC-HarDNet	0.701	0.418	426	0.019

TABLE IV
Evaluation of Method B with the original size point clouds in *PointNet* and using downsampled point clouds.

Point cloud size	Mean validation accuracy	*Jetson Nano* inference time(sec)	Downsampling time(sec)
30k	0.428	0.111	0.386
300k	0.417	0.111	-

smaller inference time (is more than 20 times faster) and more important, it is compatible with the *Jetson Nano*. Taking this into account, we opted to use the *FC-HarDNet* algorithm in method A.

B. Method B

As our focus is in real-time door classification methods, we built a downsampled version of our dataset for *PointNet* using the voxel downsampling tool from the *Open3D*, [12] library. As the *PointNet* randomly selects the K points in the point clouds, the points selected in the downsampled point cloud will better represent the point cloud because the downsampled cloud has fewer points (30000 on average) compared with the original cloud (300000 on average). The goal here was to see if it was worth it to downsample the point clouds taking into account the time it takes to do it and the improvement in validation accuracy compared with the original point clouds.

We trained the *PointNet* during 10 epochs with batch size equal to 20 and K=10000. For each approach, we trained 3 times and used the best validation accuracy value. We used a voxel size, in the voxel downsampling *Open3D* tool, that produced a proportion of 10 to 1 in the downsampled point cloud.

Table IV shows the difference between using the original size and the downsampled point clouds. The mean validation accuracy is a little better in the downsampled point clouds as expected. The *PointNet* is more likely to select points that represent uniformly the point cloud since these have fewer points than the original size ones. The inference time in *Jetson Nano* is the same for both approaches since the number of points selected is the same (K=10000) but with the downsampling time, the downsampled approach is almost 5 times slower than the original one ((0.111+0.386)/0.111 = 4.47). In view of the above and taking into account that our focus is on real-time methods, we opted to use the original size point clouds and discard the downsampling.

C. Method A vs Method B

Method A has semantic segmentation that is not used in method B. Certainly, method B is faster but the addiction of

TABLE V
COMPARISON OF THE METHODS ASSUMING THAT THE SEMANTIC SEGMENTATION MODULE IS RETURNING THE CORRECT OUTPUT.

Method	Mean test accuracy	*Jetson Nano* inference time(sec)	Segmentation time(sec)
A (after segment.)	0.494	0.111	0.131
B	0.433	0.111	-

semantic segmentation removes unnecessary information for the object classification which could lead to better results in terms of accuracy. We compared both methods with respect to speed and test accuracy. We created another version of our 3D dataset with cropped point clouds that represented the output of the semantic segmentation module from the first method. This dataset is exactly equal to the original in terms of sample numbers and the distribution in the test, validation and train set is also the same. We trained the *PointNet* with this new dataset and we compared the results with the original dataset. This way we can compare both methods assuming that the semantic segmentation module returns the correct cropped point cloud.

Analysing the results of table V we came to the conclusion that the addition of semantic segmentation in method A does not pay off the time it takes because of the difference in test accuracy. Method A takes twice as long when compared to method B. It is the semantic segmentation time (0.131 sec) plus the inference time of the *PointNet* (0.111 sec). Although we are removing unnecessary information on the point cloud, we are also removing information about the door surroundings which has an important role to help classifying doors. This is the justification for the small difference in test accuracy between method A and method B.

D. *Comparing with others*

We did not compare our methods with the ones from related works because those works focused door detection, while we did door classification. The *Llopart* method, [6], works in every class of door as it can be seen in table I, but it does not actually do door classification since his method cannot recognize the difference between doors, it just detects them. *Quintana* method, [10], is the only method that does door classification implicitly by differentiating closed doors from open and semi-open doors using the opening angle to differentiate open from semi-open doors. Their dataset (35 point clouds) is not as complete as ours (1206 point clouds) and their method does not work in real-time, although it presented excellent results in their dataset.

VII. CONCLUSION

In this paper, we proposed a new method for door classification to improve robot navigation and to provide it with the information to move between rooms. Our method works in real-time in low powered computers as the *Jetson Nano* from *Nvidia*. The robot does not have to be connected to the internet because our method works offline. We also built a big dataset with RGB images and their respective point clouds and a dataset for semantic segmentation derived from the previous one.

Our work can be used in other areas and applications, as for systems that help visually impaired people navigate in indoor spaces to improve their lifestyle and other systems that use the information of the door class. This work can be compared to future methods using our online published dataset and our accuracy results.

For future work we plan to improve our methods and test other 3D object classification algorithms other than the *PointNet*.

ACKNOWLEDGMENT

This work is supported by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT-Fundação para a Ciência e a Tecnologia, through national funds.

REFERENCES

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016. [Online]. Available: http://arxiv.org/abs/1612.00593

[2] I. Monasterio, E. Lazkano, I. Rano, and B. Sierra, "Learning to traverse doors using visual information," *Mathematics and Computers in Simulation*, vol. 60, pp. 347–356, 09 2002.

[3] G. Cicirelli, T. D'Orazio, and A. Distante, "Target recognition by components for mobile robot navigation," *J. Exp. Theor. Artif. Intell.*, vol. 15, pp. 281–297, 07 2003.

[4] N. Kwak, H. Arisumi, and K. Yokoi, "Visual recognition of a door and its knob for a humanoid robot," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2079–2084.

[5] Zhichao Chen and S. T. Birchfield, "Visual detection of lintel-occluded doors from a single image," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2008, pp. 1–8.

[6] A. Llopart, O. Ravn, and N. A. Andersen, "Door and cabinet recognition using convolutional neural nets and real-time method fusion for handle detection and grasping," in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, April 2017, pp. 144–149.

[7] T. H. Yuan, F. H. Hashim, W. M. D. W. Zaki, and A. B. Huddin, "An automated 3d scanning algorithm using depth cameras for door detection," in *2015 International Electronics Symposium (IES)*, Sep. 2015, pp. 58–61.

[8] B. Quintana, S. A. Prieto, A. Adán, and F. Bosché, "Door detection in 3d colored laser scans for autonomous indoor navigation," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct 2016, pp. 1–8.

[9] S. Meyer Zu Borgsen, M. Schöpfer, L. Ziegler, and S. Wachsmuth, "Automated door detection with a 3d-sensor," in *2014 Canadian Conference on Computer and Robot Vision*, May 2014, pp. 276–282.

[10] B. Quintana Galera, S. Prieto, A. Adan, and F. Bosché, "Door detection in 3d coloured point clouds of indoor environments," *Automation in Construction*, vol. 85, p. 146–166, 01 2018.

[11] Willow Garage Star Detector. [Online]. Available: http://pr.willowgarage.com/wiki/Star-Detector

[12] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[13] H. Wu, J. Zhang, K. Huang, K. Liang, and Y. Yizhou, "Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation," in *arXiv preprint arXiv:1903.11816*, 2019.

[14] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, "Hardnet: A low memory traffic network," *ArXiv*, vol. abs/1909.00948, 2019.

[15] H. Zhang, K. J. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," *CoRR*, vol. abs/1803.08904, 2018. [Online]. Available: http://arxiv.org/abs/1803.08904

[16] Computer Vision Annotation Tool: A Universal Approach to Data Annotation. [Online]. Available: https://github.com/opencv/cvat