Point planning and drift correction based on reliable door detection

Emily S. Mes

Layout: typeset by the author using LaTeX.
Cover illustration: TNO

Point planning and drift correction based on reliable door detection

Navigate SPOT in a search-and-rescue mission

Emily S. Mes
11737220

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisors

Dr. A. Visser
Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Dr. ir. J. Sijs
Department of Defence, Security & Safety
The Netherlands Organization
for applied scientific research (TNO)
Oude Waalsdorperweg 63
2597 AK Den Haag

June 25th, 2021

Acknowledgements

This thesis would not have been possible without the support and guidance of many people, whether or not it was digitally. To start, I would like to thank dr. ir. Joris Sijs, my TNO supervisor, not only for giving me the opportunity to carry out this project, but also for his invaluable guidance and expertise throughout this project. Not to mention the trust when allowing me to work with the SPOT robot freely, and allowing me to make use of the TNO facilities. Despite these unusual times, I was able to visit the TNO location in The Hague once every week, which I am very grateful for. I hope to continue making use of these facilities, since Joris kindly offered to let me continue my research over the summer holidays. Furthermore, I would like to thank dr. Arnoud Visser, my UvA supervisor, for his excellent guidance through each step of the process, including the encouragement he provided. I am grateful for his constructive feedback, the weekly meetings, and the pleasant communication. Finally, I would also like to thank my family and friends for their support throughout my studies, including my boyfriend Duncan Bart for his love and support, not only during this thesis, but also during my entire bachelor studies.

Abstract

TNO, the institution for applied scientific research of the Netherlands, is currently working on a project called SNOW. The objective of this project is to acquire knowledge about the possibilities of autonomous systems in the open world. This is tested with a search-and-rescue mission carried out by the robot SPOT from Boston Dynamics. In this case, the SPOT robot is sent into a house to obtain accurate information on the whereabouts of the people in the house. This application could help the fire brigade immensely, enabling more targeted rescue missions. To move around the house freely, the SPOT robot has the ability to exit a room from every point in that room. The SPOT robot does this by planning points before executing the action to move to those points. It plans one point right before the door and one point right after the door it wants to go through. Currently, these points are planned in a scripted manner: 1 meter before the door and 1 meter after the door. Unfortunately, this is not very robust, because there are situations where the SPOT robot refrains from executing the planned points, for example when there is an obstacle where the point is planned. Therefore, I identify a more flexible way for the SPOT robot to plan the points and therefore the way to exit and move to the next room. Furthermore, the SPOT robot has a tendency to drift, which results in a discrepancy between the SPOT robot's actual and assumptive position. The drift problem that the SPOT robot faces could be solved by performing door detection, since doors are perfect orientation points when the map is known. Once the SPOT robot detects the door frame, the SPOT robot's assumptive position could be corrected to the actual position. Therefore, I aim to perform a door detection approach to eliminate the SPOT robot's drift. In this research, a robustification of a search-and-rescue mission carried out by the SPOT robot is proposed by improving the current point planning and eliminating the drift using door detection.

Contents

CHAPTER 1

Introduction

TNO, the institution for applied scientific research of the Netherlands, is currently working on a project called SNOW. The objective of this project is to acquire knowledge about the possibilities of autonomous systems in the open world. This is tested with a search-and-rescue mission carried out by the robot SPOT from Boston Dynamics (See Figure 1.1). Search-and-rescue research in general has been conducted since the late 1990s, when mobile robot competitions were established after multiple disasters such as an earthquake and a bombing [9]. The interest for developing ground robots to search for trapped victims in disaster areas grew, and those robots were firstly used in the 9/11-disaster in 2001. Since then, teams from all over the world have been working on numerous search-and-rescue challenges and have thereby improved the perception, planning and control of the robots substantially [15]. Furthermore, disaster robots have proven to be useful in search-and-rescue missions after earthquakes, hurricanes, mine collapses, bridge collapses, explosions and flooding [9].

Figure 1.1: SPOT from Boston Dynamics

In this case, the SPOT robot is sent into a house to obtain accurate information on the whereabouts of the people in the house. The SPOT robot has six primitive actions it can perform in the house, of which three tasks have the objective to scan the room for people and potential victims. The first approach is to scan the whole room by rotating the SPOT robot's orientation from the same position. The position of a possible victim can then be estimated, but due to sensor inaccuracies the uncertainty could be fairly significant. The second approach is to scan the area from multiple waypoints in the room. This way, the uncertainty of the location of a possible victim will be lesser, because if the possible victim is detected from different waypoints (and the waypoints are known), the sensor uncertainties can be corrected, as is proposed by Feldman [3]. In this research, an approach is introduced for visual Bayesian classification of recognized objects from multiple viewpoints using spatial correlations in the classifier score. If it is not possible to scan the room (for example due to smoke or darkness), the SPOT robot will use the third option to scan the room: an auditory method. The SPOT robot will reach out to a potential victim by calling out and waiting for a response. Note that the uncertainty in this case is proportional to the size of the room itself, because the SPOT robot cannot determine the location of the person. In principle, it is possible to localize the sound source using a microphone array [17], but the SPOT robot only has access to one microphone, which is why it is not possible to get a direction estimate. Once the room is scanned and a potential victim is seen or heard, the SPOT robot is tasked to approach this person, which is a fourth task. Thereafter, the SPOT robot has a fifth task to identify this person by asking their name. Additionally, the SPOT robot has to find the door and exit from every point in that room. This last task will be the focus of this research.

CHAPTER 2

Problem definition

2.1 Point planning

The SPOT robot enters and exits a room by planning points where it has to go. It plans one point right before the door and one point right after the door it wants to go through. Currently, these points are planned in a scripted manner: 1 meter before the door and 1 meter after the door. A visualization of this action can be seen in Figure 2.1.

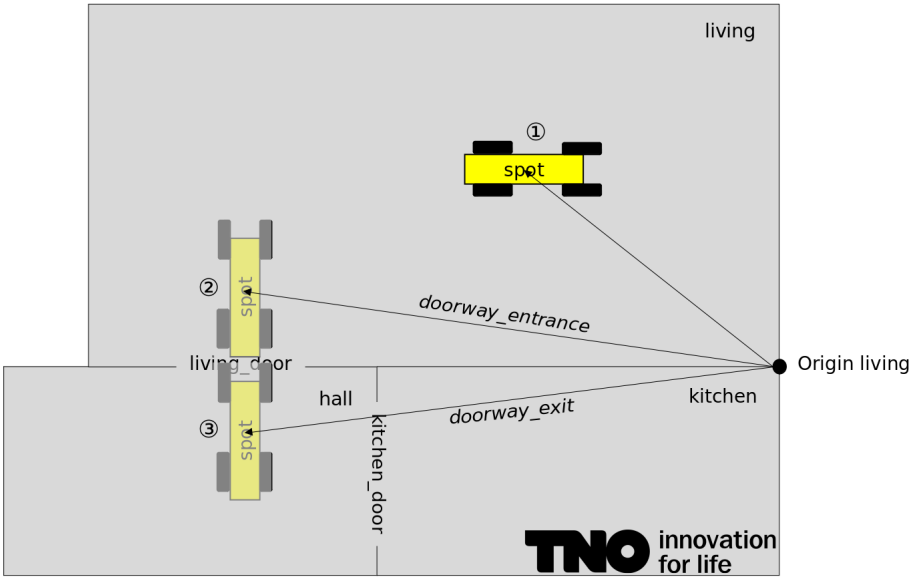
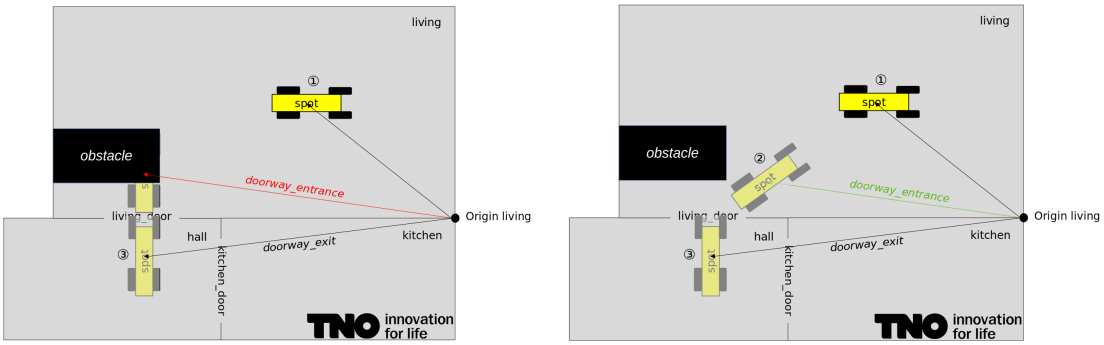


Figure 2.1: A visualization of the `exit_via_door` script. 1 is the initial position of SPOT. This could be every point in the living room. 2 is the point before the door and 3 is the point after the door. These points are planned in a scripted manner.

Unfortunately, this approach is not very flexible, because the SPOT robot may get in trouble for example when there is an obstacle where the SPOT robot wants to plan a point. An example is shown in Figure 2.2. In the figure, there is an obstacle, such as a wall, in front of the door. Therefore, the point before the door can not be planned correctly, which results in the SPOT robot failing to exit the room, which can be seen in Figure 2.2a. For every case specifically, an appropriate solution has to be found. In this case, it may be a solution to place the SPOT robot diagonally, before the door, such that it can exit the room through the small space between the bed and the wall, as seen in Figure 2.2b. Note that there could also be an obstacle after the door. In that case, it is important to adjust the point after the door. In extreme cases,

it could be necessary to change the point before the door, after the obstacle after the door is determined from the point before the door. Additionally, there could be tight hallways or small rooms, where it is easier to move backwards instead of turning and moving forwards. Therefore, in this research the point planning before the door has to be improved.



(a) There is a collision with the obstacle. SPOT fails in exiting the room.

(b) A possible solution to avoid collision with the obstacle.

Figure 2.2: Robustness of the current point planning for the `exit_via_door` script of SPOT.

2.2 Drift correction

Due to the SPOT robot not having a GPS or a compass, it has a tendency to drift, which is an error in the odometry. This results in a discrepancy between the SPOT robot's actual position and the SPOT robot's assumptive position. A slight discrepancy between the SPOT robot's actual and assumptive position in itself is not unworkable, but if a discrepancy results in the SPOT robot thinking it is in one room while it in reality is in another room, it is a significant issue. The drift problem that the SPOT robot faces could be solved by making use of the doors. Indoor doors are important landmarks: they do not move and are therefore perfect orientation points of the robot when the map is known [1]. Once the SPOT robot has correctly determined the point before the door and has positioned itself facing the door, the door frame could be detected using its sensors. This can be done in a model-based or data-based way [12]. When the door frame is extracted from the sensor data, the location of the door relative to the actual position of the SPOT robot is known. Alternatively, the actual location of the SPOT robot relative to the location of the door can be calculated, which could be compared to the SPOT robot's assumptive position to eliminate the drift.

2.3 Research question

This leads to a research question that incorporates both challenges.

> Can the SPOT robot's performance in a search-and-rescue mission be made more flexible by a deterministic point planning approach and applying door detection for drift correction?

CHAPTER 3

Theoretical background

3.1 Navigation

Navigation is commonly regarded as the most challenging competence that is required of an autonomous mobile robot [16]. Robot navigation consists of multiple parts. First of all, the robot has to interpret its sensory data in order to gain relevant information. Secondly, the robot has to localize itself in the current environment. Thirdly, the robot has to know what to do to achieve its given goals. Lastly, the robot has to direct itself to do what it wants itself to do. These four building blocks – *perception, localization, cognition and motion* – are closely intertwined: however good the processing of information is, if the sensory information is not correctly obtained, the robot will have trouble localizing itself in the environment. Alternately, when the robot is successful in all four of those modules, it will succeed in navigating itself safely through an environment. The process of robot navigation can be seen in Figure 3.1.

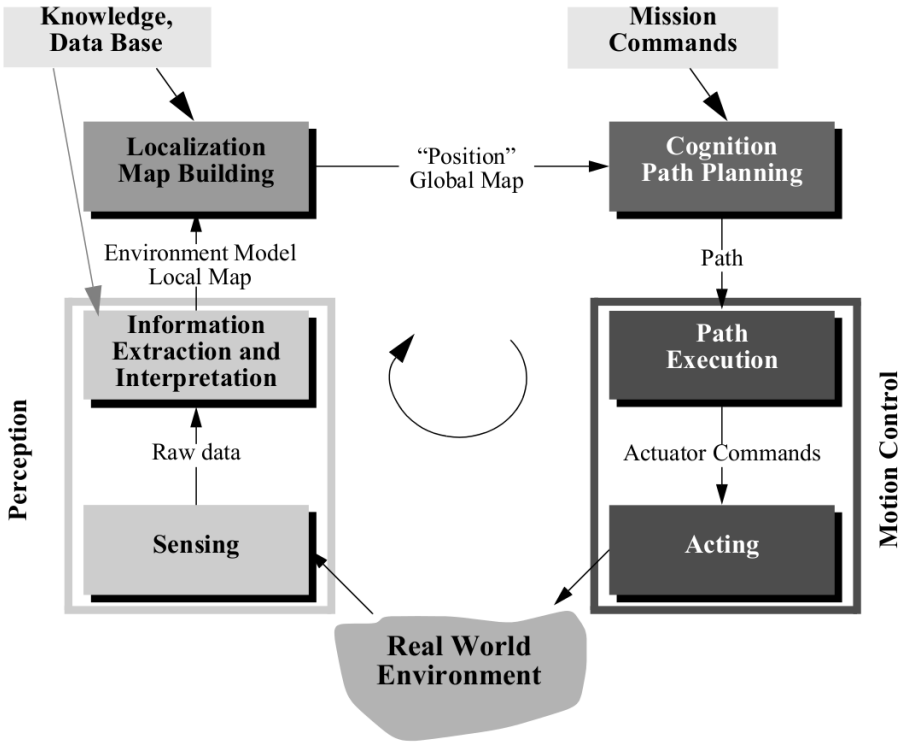


Figure 3.1: An abstract control scheme for mobile robot systems. Courtesy from [16].

The operations of this study are similar to this process. The SPOT robot must firstly position itself in front of the door it wants to go through (motion). Then, it has to use its exteroceptive sensors to obtain a point cloud of the entire door frame and the connecting wall and detect the door from the obtained point cloud (perception). Thereafter, the SPOT robot can localize itself in relation to the determined door frame and possibly correct its position (localization). Next, it can calculate its trajectory to and through the door (cognition) and perform this action (motion), which means that the cycle is complete.

3.2 Statistical Outlier Removal

In this section, the algorithm of statistical outlier removal will be discussed. This algorithm is used in the pre-processing step of the door detection part of the research. Since the focus of the method will mostly be the approach itself, the algorithm is explained here in more depth. Note that the statistical outlier removal algorithm is performed on a three-dimensional point cloud. As the name suggests, the purpose of this algorithm is to remove outlier points in a three-dimensional space. What exactly is classified as an outlier is determined by the parametrization. The parameter k is the number of neighbors that is considered for each point, and the parameter α is a standard deviation multiplier. For every point p_i in the given point cloud the average distance d_i from point p_i to the k closest neighbors is calculated. The mean μ_d and the standard deviation σ_d of all the distances d is determined, after which the threshold T is computed in the following way:

$$T = \mu_d + \alpha \cdot \sigma_d$$

Finally, another loop over all the points p_i is required to eliminate the points that do not satisfy the condition $d_i > T$.

An example of the statistical outlier removal filter is shown in Figure 3.2. Here $k = 50$ and $\alpha = 1.0$.

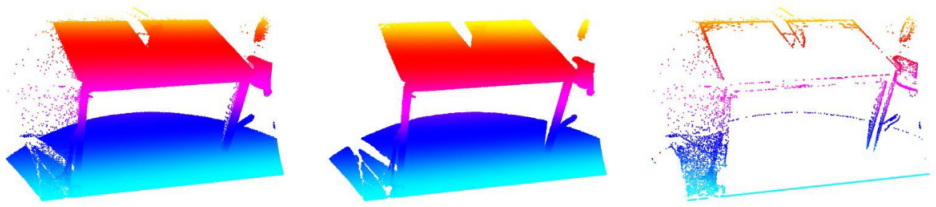


Figure 3.2: Left: the raw point cloud, middle: the (resultant) inliers, right: the (filtered) outliers. Courtesy from [13].

3.3 Plane Segmentation using RANSAC

The segmentation of point clouds is the process of classifying every point of a point cloud. A common approach is to group the points on the same plane together, which is called plane segmentation [5]. In other words, all the points in the point clouds that support the plane model are classified to be inliers, whereas the rest of the points are considered to be outliers. This plane model is calculated using RANSAC, which is an iterative method to estimate parameters of a mathematical model from an observed data set and therefore functions as a robust estimator to fit the model to only inliers. RANSAC randomly selects 3 non-collinear points and calculates the best possible model, which is determined by the overall number of inliers. Note that RANSAC requires a distance threshold value, which determines how close a point must be to the plane to be classified as an inlier. Lastly, the plane segmentation algorithm returns the estimated plane parameters a, b, c, d, which are the parameter of the general equation of a plane:

$$a \cdot x + b \cdot y + c \cdot z + d = 0$$

An example of plane segmentation with a RANSAC model is shown in Figure 3.3. Here the distance threshold value is set to 1 centimeter.

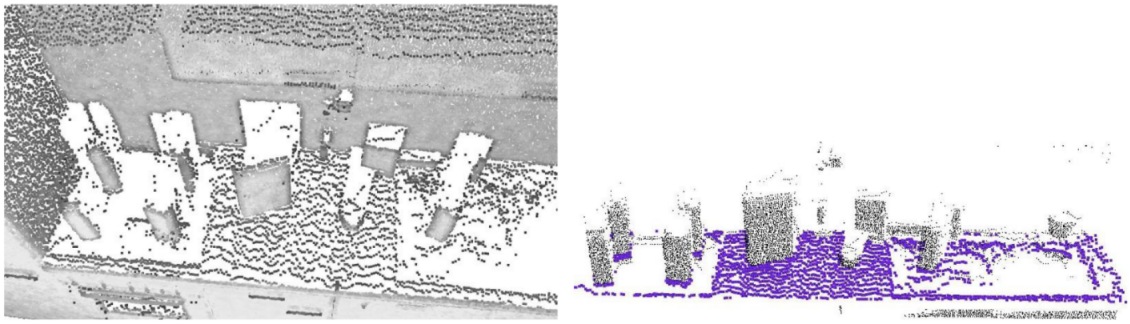


Figure 3.3: Left: the raw point cloud, right: the segmented plane (colored in purple). Courtesy from [13].

3.4 Voxel Grid Downsampling

In this section, the downsampling of a point cloud, which reduces the number of points, is discussed. This can be done using a voxel grid filter. A voxel grid can internally be visualized as three-dimensional pixels. So each voxel, i.e. 3D box, will approximate all the points present in that voxel with their centroid. Figure 3.4 visualizes the downsampling using a voxel grid. The left image (a) is the original point cloud with chosen voxel size the size of the red subpoint cloud. The center image (b) is one voxel with all points in the voxel in blue. Lastly, the right image (c) demonstrates the downsampling of all blue points to one red centroid. Note that the complete downsampled point cloud consists of all (c) point clouds constructed together.

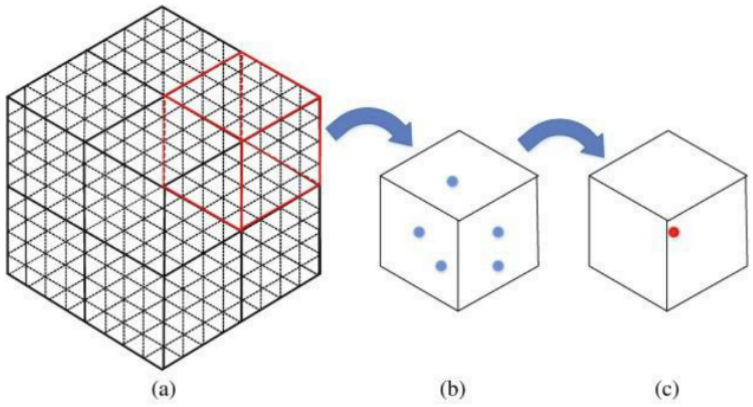
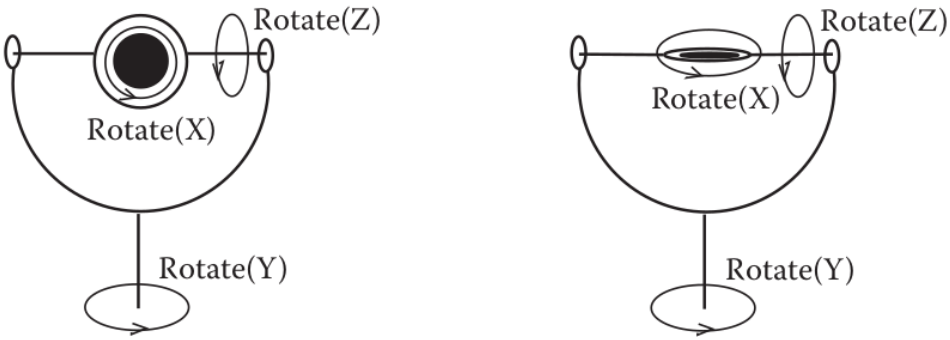


Figure 3.4: Visualization of downsampling the point cloud using a voxel grid.

3.5 Quaternions

A common problem that arises in the field of computer graphics and robotics is gimbal lock. This occurs when there are three angles of rotation, of which one of the three angles is accidentally aligned with another during rotation [8]. This reduces the degree of freedom by one, since a object "lost" one axis to rotate. This problem is shown in Figure 3.5, where gimbal lock occurs when a turn of $\frac{1}{2}\pi$ radians is made around the Z axis, resulting in accidentally aligning the X and Y rotations. Therefore, the object has lost one degree of freedom. Note that the term "lock" is misleading, because, while one degree of freedom is lost, it is still possible to rotate around the "locked" axis.



(a) Original configuration

(b) Gimbal lock configuration

Figure 3.5: The occurrence of gimbal lock. Both X and Y rotations are now performed around the same axis. Courtesy of [8].

Quaternions solve the problem of gimbal lock. Quaternions are four-dimensional vectors that extend the complex numbers and are mostly used to determine the relative orientation in three dimensions [4]. Shortly put, quaternions prevent singularity in the origin.

CHAPTER 4

Related work

In this chapter the related work of multiple challenges of this research will be discussed. Since the primary focus of this research will be the door detection from a point cloud, the theoretical framework regarding door detection will be reviewed at length, whereafter some studies regarding exploration will be examined as well.

4.1 Door detection

While this research consists of multiple challenges, its primary focus will be door detection from a point cloud. Therefore, the theoretical framework regarding door detection will be discussed in this section. Since door detection is a crucial part of indoor autonomous robotics, there is plenty of research done on this matter.

4.1.1 Obtaining a point cloud

Detecting doors by utilizing an obtained point cloud can be done in various ways. In fact, point clouds can be obtained in various ways; depth cameras [20], stereo imagery [18], laser scanners and integration with a color camera [11] all fall within the possibilities of obtaining a point cloud. The state-of-the-art technology even make it possible to obtain a point cloud from a single panorama image [21]. First, the depth and instance segmentation is predicted with a CNN from the panorama input. Then, the 3D scene layout is recovered from the estimated layout depth, whereafter the object shapes can be completed to reconstruct the full scene point cloud. However, this pipeline is out of scope of this research, so the focus will be on point clouds obtained by a range sensor or depth camera.

4.1.2 Door detection without a priori model learning

Yuan *et al* propose a way to extract the wall planes from the point clouds, from which the door frame can be detected using a 3D scanning algorithm [20]. Since the values from the depth camera are much higher for pixels in the door frame (for non-closed doors), the object of interest, the largest black color portion, is easily detected. Thereafter, this object of interest is scanned from its center pixels and iterates upwards and sideways until a white pixel is registered, which is then determined to be a corner coordinates. When all four corner coordinates are detected, the ratio of the door width and the door length clarifies if the object of interest is indeed a door. It is also possible to detect half-closed or entirely closed doors. In case of a half-open door, the doors opening angle can be calculated using the ratio of the door width and the door length. For this approach it is necessary that the robot is perfectly in front of the door that has to be detected. The advantage of this approach is that only the sensed range data is required, while other studies that will be discussed also need a RGB camera, including segmentation and classification, for the door detection. Using only sensed range data means that the algorithm will not be sensitive to lightning.

Another research that shares the approach of classifying doors as "gaps" in the point cloud of the wall plane, proposes a new approach using features such as edges, textures and regions with geometry information that is obtained from range data to make a reliable 3D indoor scene representation [18]. Since doors are detected as regions on the wall without sensed data, only open doors can be detected. This research firstly uses the color information to identify the wall planes, which are calculated using iteratively reweighted least squares robust linear regression. Thereafter, the stereo depth image data is processed by converting it into a dense cloud to estimate the surface from which a segmentation can be done to detect depth discontinuities. Following the generation of a 3D scene that fits the surfaces and detects the room boundaries. From this 3D room reconstruction the doorways can be detected by clustering depth pixels that do not support a concave room structure. While this approach is very robust, the room modeling seems to be out of the scope of this research.

The following research integrates the information regarding both the geometry (XYZ coordinates) and color obtained from a 3D laser scanner and a color camera [11]. This is unique, because in contrast to the other researches, this technique is developed in a 6D space framework. This approach detects open doors based on detection of rectangular data holes in the wall planes and closed doors based on detection of the actual wall area and the subsequent processing of the rectual areas that do not correspond to a wall.

4.1.3 Door detection using a priori model learning

The next research registers candidate door openings from the point cloud using Aggregate Channel Features (ACF) and makes use of a trained model to filter out candidates that are not doors, such as mirrors or windows [6]. The door detector is trained with ACF on the candidate region(s) of RGB images that are taken at the same time as the point cloud data. Before the candidate door openings can be detected, the dominant wall planes have to be extracted. This is done using the RANSAC algorithm, which filters out outliers and is therefore capable of finding the dominant wall plane. A drawback of using RANSAC is that the iterations that have to be done can increase extremely when there are multiple dominant wall planes, for example two perpendicular wall planes. This drawback is prevented by running a plane estimation for every slice, to such an extent that there are not two dominant wall planes in the same slice.

4.1.4 Door detection in application

Door detection could also be useful for assistive shared-control wheelchairs [14] [2]. The following studies both assume that the doors meet the standard of the American Disability Act (ADA). Rusu *et al* propose an approach to detect a closed door by an estimation of the surface normal and fit the plane model to every candidate door using RMSAC, whereafter the two major door edges could be estimated. Eventually the best candidate door will be found using a scoring system based on geometric values. Once the closed door is detected, the door handle could be distinguished by estimating the surface curvatures, projecting the handle candidates on the door plane and fitting the best horizontal line using RMSAC. This research inspired another research that applies similar algorithms, but with a focus on open doors [2]. This research firstly finds the dominant wall using RANSAC, calculates the normal extract nearby point and projects that into the plan. Then the gaps are found by scanning a stripe on the wall based on neighbor points. Lastly, all gaps are checked if they meet the ADA standards.

4.2 Exploration

Once the door is detected, the SPOT robot has to ensure that the point after the door is also safe to go. For the task to distinguish safe regions from observation regions that are not guaranteed to be safe, Visser *et al* propose an approach that uses a ray-casting technique to generate two occupancy grids at the same time [19]. One short range occupancy grid that makes a conservative estimate of the safe region and one long range occupancy grid for safe regions which are probably safe but can not be guaranteed to be safe due to potential obstacles. From these two occupancy grids the safe regions and observation regions can be determined. Furthermore, exploration

frontiers, which are interesting locations on a map where the exploration can be continued, can be identified as part in the contour of the safe regions but are not a wall. Lastly, all exploration frontiers are checked to be concave or convex, and concave frontiers are left out, to decrease the number of false positives.

CHAPTER 5

Approach: point planning

In this chapter, the approach of solving the point planning problem described in Section 2.1 is discussed. The deterministic approach consists of multiple steps: calculating the free space, determining the relevant poses and lastly applying an error function to decide which point should be planned before the door.

5.1 Data description

The data received to conduct this research are separate lists with x-coordinates and y-coordinates of corner points of the rooms at the TNO location, which is called the villa. The villa consists of the hall, the living room, the kitchen, the parents' bedroom and the daughter's bedroom. Every room and every door in the villa, is represented with these two lists of x-coordinates and y-coordinates. These lists can be transformed to Shapely Polygon objects, which facilitates visualizing the rooms. Shapely[1] is a Python package for manipulation and analysis of planar geometric objects. A visualization of the data set used for this research can be seen in Figure 5.1. Note that walls are colored in blue and doors are colored in red. Furthermore, the angle definition is shown in the bottom left corner.

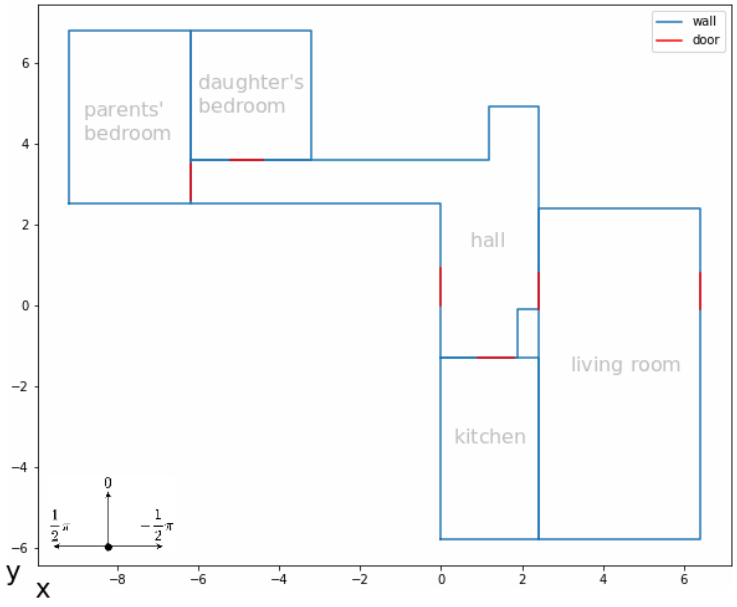


Figure 5.1: The villa of the TNO location

[1] https://pypi.org/project/Shapely/

5.2 Free space

The first step of the point planning approach is the calculation of the free space. The free space is a square that determines whether it is possible for the SPOT robot to execute the currently planned point, or whether it is necessary to find and plan an alternative pose before the door. As indicated in Figure 5.1, there is only one door in the hallway with enough free space for the SPOT robot. The calculation of the free space square is discussed in this section.

The SPOT robot has a length of 110 centimeters and a width of 50 centimeters. The theorem of Pythagoras states that the diagonal of the SPOT robot is $\sqrt{110^2 + 50^2} \approx 121$ centimeters. Ensuring an additional safety buffer for the SPOT robot, the turning radius is approximated to be 125 centimeters. Consequently, a square space of 125 centimeters by 125 centimeters has to be free around the planned point. Since it is known that the planned point is exactly 1 meter before the door, the door coordinates can be used to determine the free space. This free space will be a Polygon with four vectors $(\vec{P_a}, \vec{P_b}, \vec{P_c}, \vec{P_d})$ and is calculated in the following way.
Since the coordinates of the door $(\vec{d_1}, \vec{d_2})$ are given, a door vector $\vec{d}$ can be determined. The door vector is the vector from $\vec{d_1}$ to $\vec{d_2}$.

$$\vec{d} = \vec{d_2} - \vec{d_1}$$

The points of the free space polygon that are connected to the same wall as the door, are an extension of the door vector $\vec{d}$. Therefore, the door vector $\vec{d}$ can be scaled with scaling factors α and β. α scales the door width $\|\vec{d}\|$ to the distance between the point of the wanted free space and the door point $\vec{d_1}$. β scales the door width $\|\vec{d}\|$ to the distance between the wanted free space on the other side of the door and the same door point $\vec{d_1}$.

$$\begin{aligned}
\vec{P_a} &= \vec{d_1} - \alpha \cdot \vec{d} \\
\vec{P_b} &= \vec{d_1} + \beta \cdot \vec{d} \\
\alpha &= \frac{125 - \|\vec{d}\|}{2 \cdot \|\vec{d}\|} \\
\beta &= \alpha + 1
\end{aligned}$$

The remaining two points of the free space polygon can be calculated by rotating the door vector $\vec{d}$ with $-\frac{1}{2}\pi$ radians. The rotated door vector $\vec{d'}$ is then used to calculate $\vec{P_c}$ and $\vec{P_d}$ from $\vec{P_a}$ and $\vec{P_b}$ respectively. Again, a scaling factor γ is needed to scale the door width $\|\vec{d'}\|$ to the desired length of the free space polygon, which is 125 centimeters.

$$\begin{aligned}
R(\theta) &= \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \\
R(-\frac{1}{2}\pi) &= \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \\
\vec{P_c} &= \vec{P_b} + \gamma \cdot (R(-\frac{1}{2}\pi) \cdot \vec{d}) \\
\vec{P_d} &= \vec{P_a} + \gamma \cdot (R(-\frac{1}{2}\pi) \cdot \vec{d}) \\
\gamma &= \frac{125}{\|\vec{d'}\|}
\end{aligned}$$

A visualization of the calculation of the free space polygon can be seen in Figure 5.2. Note that $\vec{d}$ is the door the SPOT robot has to go through, and that the final free space polygon consists of points $\vec{P_a}$, $\vec{P_b}$, $\vec{P_c}$ and $\vec{P_d}$.

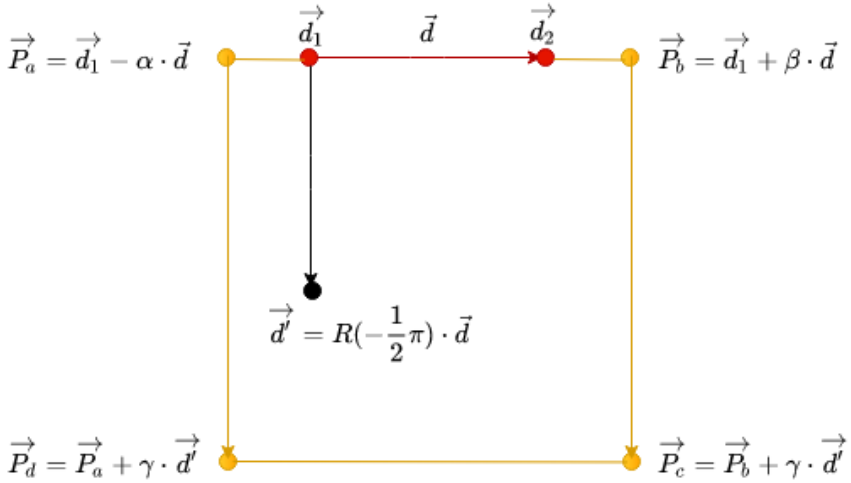


Figure 5.2: A visualization of the calculation of the free space polygon. Note that the door is colored in red, and that the orange square is the calculated free space that consists of point $\vec{P_a}$, $\vec{P_b}$, $\vec{P_c}$ and $\vec{P_d}$.

Once the free space polygon is established, impossible planned points can be identified by calculating the intersection of the free space and the room. If the area of the free space polygon is equal to the area of the intersection, the free space polygon is not discontinued by the room polygon. Therefore, the conclusion that the currently planned point is possible can be drawn. Alternatively, if the area of the free space polygon is larger than the area of the intersection of the free space and the room, an alternative pose has to be determined.

5.3 Relevant poses

In this section, the next step for calculating the best alternative pose is demonstrated, which is establishing relevant poses A pose is a tuple consisting of a two dimensional position and an orientation (x, y, θ). To find the best pose before a door, all possible combinations of predefined positions and predefined angles are evaluated. Note that the focus of this study is on a deterministic approach, which is why the positions and angles are predefined. The predefined positions are obtained by yielding a dodecagonal buffer around the center of the door. A dodecagon was chosen to ensure that there were enough, but not too many poses before the door. The radius of this dodecagon is half the length of the desired free space. This ensures that the positions place the SPOT robot in the center of the free space, such that the SPOT robot has enough space to move and turn. From this dodecagon, five points are located outside the current room and two points are positioned on the wall. Consequently, these points can be discarded, which leaves five relevant points before the door. Furthermore, to guarantee enough space for SPOT to move, positions that have a distance to the closest wall smaller than 0.3 meters are excluded as well. This approach is visualized in Figure 5.3. Note that the door is colored in red, and note the dodecagonal buffer around the center of the door $\vec{d}$ and the five relevant points before the door in blue. The grey points on the dodecagon are discarded as relevant points.

The selected angles range from 0 to π radians, with $\frac{1}{4}\pi$ intervals. This interval was chosen to ensure that there were enough, but not too many poses before the door. These angles have to be rotated depending on the angle of the door. For example, a door perpendicular to the x-axis requires different predefined angles to the door than a door perpendicular to the y-axis. Once all angles are ensured to point towards the current door, all possible combinations of the positions and angles are evaluated. All possible combinations are visualized in Figure 5.3.

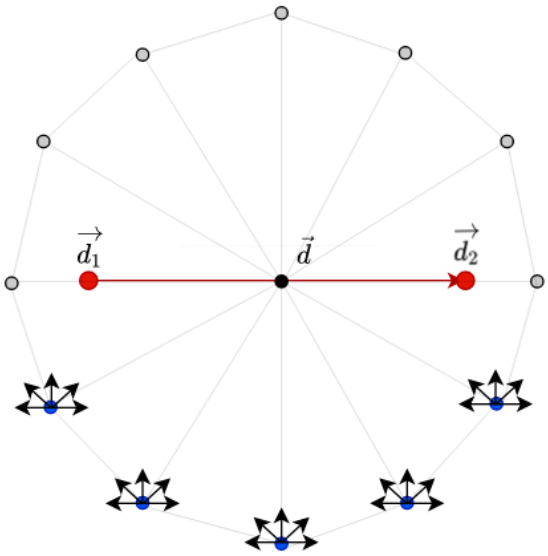


Figure 5.3: A visualization of the calculation of the relevant poses before the door.

5.4 Error function

Once the relevant poses are determined, it is necessary to evaluate and rank those poses, such that the best pose can be passed to the SPOT robot. The ranking procedure is described in this section. When evaluating a pose, two main components are taken into account: the area and orthogonality error. To calculate the area error, an ellipse is drawn around every pose. Consequently, the pose is at the center of the corresponding ellipse. The size of the ellipse is inspired by the size of the SPOT robot, but a small buffer is added. Therefore, the dimensions of every ellipse is 120 centimeters along the angle of the pose, and 60 centimeters perpendicular to the angle of the pose. This leaves enough breathing space for the SPOT robot, which has a length of 110 centimeters and a width of 50 centimeters. Once the ellipse is determined, the intersection between the current room and the ellipse is calculated. If the ellipse does not overlap with the room at all, the area of the intersection will be identical to the area of the ellipse. Alternatively, if the ellipse is cut off by for example a wall of the room, the area of the intersection will be smaller than the area of the ellipse. Therefore, the area error can be calculated by the area of the ellipse minus the area of the intersection.

$$e_{area} = \text{area ellipse} - \text{area intersection}$$

The orthogonality error component is determined by calculating the dot product between the center of the door $\vec{d_c}$ and the pose $\vec{p}$.

$$\vec{a} \cdot \vec{b} = \|a\|\|b\| \cos\theta$$

where θ is the angle between $\vec{a}$ and $\vec{b}$.

From the definition of the dot product, it can be concluded that when the angle between the center of the door and the candidate pose for the SPOT robot is orthogonal ($\theta = \frac{1}{2}\pi$), this results in a dot product of 0, since $\cos\frac{1}{2}\pi = 0$.

Therefore the orthogonality error component can be calculated by the absolute value of the dot product, since the closer the value of the dot product is to 0, the more orthogonal it is.

$$e_{orthogonal} = \mid \vec{d_c} \cdot \vec{p} \mid$$

Combining these components, a decision must be made which of the components is of higher importance. This could be done by assigning a weight to both components in the following form:

$$error = w_1 \cdot e_{area} + w_2 \cdot e_{orthogonal}$$

An alternative is to firstly filter out all poses that do have an area error, and then find the pose with the smallest orthogonal error. In fact, that is exactly what is done in this research, because the main focus is to optimize the area error, since a pose with an optimal angle while the SPOT robot is in the wall is not possible. Therefore, all the poses are firstly ordered on the area error. When two poses happen to have the same area error, the orthogonality error is considered to order those poses correctly. Lastly, when two poses have the same area and orthogonality error, an additional metric is used to choose between those poses, which is the distance between the pose and the SPOT robot's current position.

5.5 Summary

Currently, the points before the door are planned in a scripted manner, but this does not take the environment into account. Therefore, a new approach is proposed to make the point planning before the door more reliable. To summarize, firstly the free space before the door is calculated to determine whether a new pose should be considered. If that is the case, possible points are calculated using a dodecagon around the door. The relevant poses are then obtained by combining those points with the predefined points, whereafter an additional check is made to ensure that the SPOT robot does not collide with a wall. Lastly, an error function is applied to all remaining poses, which sorts the poses from best to worst by firstly the area error, than the angle error and lastly the distance error. Pseudocode of this approach is shown in Algorithm 1.

Algorithm 1: Point planning algorithm

```
Result: List of possible poses from best to worst
door, room, current_position;
freespace ← get_freespace(door);
intersection ← freespace.intersection(room);
if intersection.area ≠ freespace.area then
    positions = get_all_positions(room, door);
    angles = get_all_angles(room, door);
    errors ← {};
    foreach position ∈ positions do
        foreach angle ∈ angles do
            area_error ← get_area_error(position, angle);
            angle_error ← get_angle_error(position, angle;
            distance_error ← get_distance_error(position, current_position);
            errors[position, angle] ← (area_error, angle_error, distance_error);
        end
    end
    best_poses ← sort(errors);
    return best_poses;
end
return NULL;
```

This algorithm will be evaluated in Chapter 7.

CHAPTER 6

Approach: door detection

In this chapter, the approach of solving the drift problem described in Section 2.2 is discussed. The method is to eliminate the drift of the robot by detecting a landmark, in this case a door. When a door is detected, the discrepancy between the SPOT robot's assumptive and actual position can be corrected. The door detection consists of multiple steps. Firstly, the obtained point cloud is pre-processed by applying statistical outlier removal and plane segmentation. Then, the resulting point cloud is downsampled to a voxel grid, which decreases the number of points in the point cloud and makes it easier to perform Euclidean clustering and RANSAC parallel line fitting. The RANSAC parallel line fitting returns two distinguishable lines, which represent the doorframe.

6.1 Data description

The data used to conduct the door detection part of this research are three-dimensional non-coloured point clouds, obtained by the SPOT robot. The SPOT robot has multiple stereo cameras located on his body, of which the depth images are converted into point clouds. The front cameras can be seen in Figure 1.1. A downside of the placement is that all those cameras are pointing slightly downwards, which results in a point cloud which only contains data only up to one meter. A door frame could be detected more easily if its full height was observed, but all data above one meter is out of view.

6.2 Tools

In this section, the tools used to implement the approach of this part of the research are introduced: ROS and PCL. Both tools are well known and well maintained. Furthermore, both tools are open-source and well documented. Figure 6.1 shows the logo of both tools.

(a) The Robot Operating System logo. (b) The Point Cloud Library logo.

Figure 6.1: The logos of both tools.

6.2.1 ROS

The most important tool used in this implementation is the Robot Operating System (ROS), which provides a framework for communicating with multiple processes [10]. ROS consists of

multiple fundamental concepts, namely nodes, topics and messages. A node is a process that performs the computations. Therefore, this is also called the software module. Nodes receive the data to be processed in the form of messages from topics, which are named buses over which nodes exchange messages. And a message is a strictly typed data structure. A node is able to subscribe and publish to multiple topics simultaneously, and there may be multiple concurrent publishers and subscribers to a single topic.

So in this case, there is a topic `/point_cloud` that contains messages of type `sensor_msgs PointCloud` that publishes those messages to a node implemented for this research called `/doordetection_node`. In this node, the required computations take place, whereafter the node publishes the result in a topic `/processed_point_cloud`. Those relations are visualized in a graph shown in Figure 6.2. Note that the rectangles represent topics, and the ellipse represents a node.

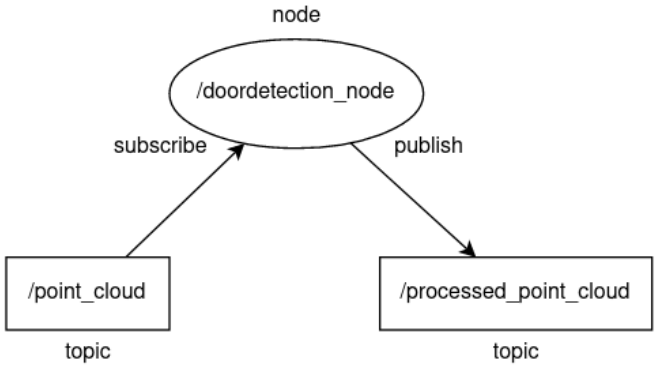


Figure 6.2: Graph of ROS process.

6.2.2 PCL

Another important tool used for this approach is the Point Cloud Library (PCL). PCL is a C++ library for n-dimensional point clouds and three-dimensional geometry processing [13]. The library contains state-of-the-art algorithms for filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. Furthermore, PCL is fully integrated with ROS.

6.3 Data processing

As mentioned before, the data has to be processed in multiple steps, before it can be used to detect doors. In this section, a short overview of these steps will be given. The data processing consists of cropping, statistical outlier removal, plane segmentation, voxel grid downsampling, clustering and RANSAC line fitting. A more detailed explanation of mentioned algorithms is given in Chapter 3.

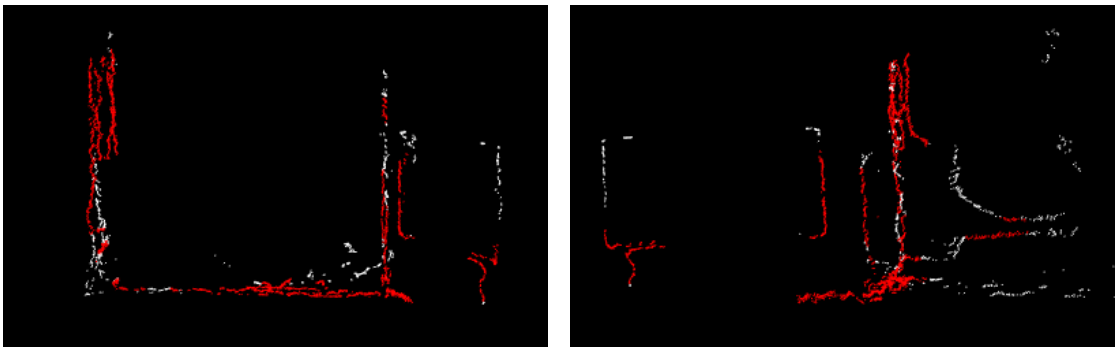
6.3.1 Cropping

First of all, the obtained point cloud covers 360 degrees of the area around the SPOT robot. Consequently, multiple doors may be visible, while only the detection of one door, more specifically the door in front of the SPOT robot, is desired. Therefore, the point cloud is cropped to cover solely the door the SPOT robot wants to go through. Since it is uncertain how exactly the SPOT robot positions itself before the door, the boundaries of the cropped box can not be hard-coded. Therefore, the following information is taken into account when cropping the point cloud: the SPOT robot's current position, the SPOT robot's assumptive coordinates of the left door frame and the SPOT robot's assumptive coordinates of the right door frame. This knowledge can then be captured in 2 quaternions: the minimum point of the cropped point cloud and the maximum point of the cropped point cloud. Note that a wide marge has to be taken around

the SPOT robot's assumptive position, since it is highly likely that the position has drifted. In this research, the necessary quaternions are manually established.

6.3.2 Statistical outlier removal

Once the point cloud consists only of the desired door, outliers can be eliminated by performing statistical outlier removal. This ensures that the cropped point cloud does not contain sensor inaccuracies, which complicates the door detection task. The statistical outlier removal filter removes outliers in a three-dimensional space using statistical analysis techniques. There are two parameters: k and α. k determines the number of neigbors to analyze for each point and α is the standard deviation multiplier, which affects the threshold of the difference between the standard deviation of the mean distance and the relevant point. Therefore, the higher α is, the less outliers are removed. In this study, the values $k = 100$ and $\alpha = 0.1$ are chosen by considering the number of points in the point cloud and the sparseness of the point cloud. Figure 6.3 visualizes this statistical outlier removal step. Note that the white point cloud is the original point cloud after cropping and the red point cloud is the white point cloud after statistical outlier removal. Since the point cloud after statistical outlier removal (red) is plotted over the point cloud before statistical outlier removal (white), the white points can be interpreted as outliers, and likewise the red points can be interpreted as inliers.



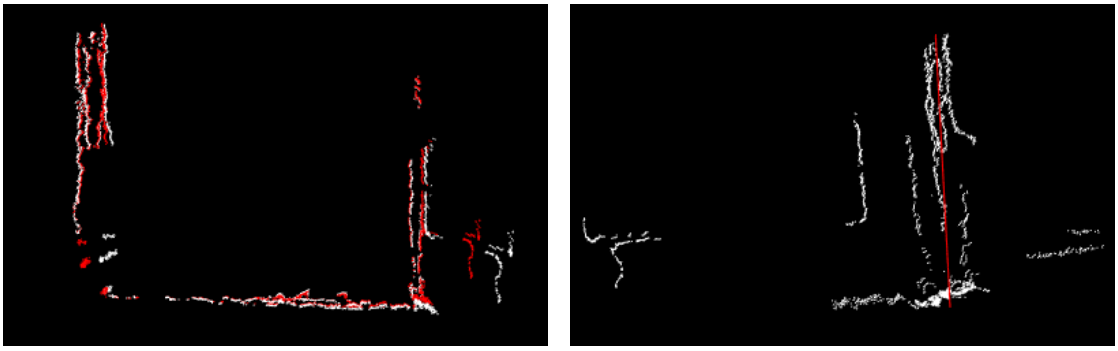
(a) Front view of statistical outlier removal.

(b) Side view of statistical outlier removal.

Figure 6.3: Front and side view of before (white) and after (red) the statistical outlier removal step.

6.3.3 Plane segmentation using RANSAC

Since the cropped and filtered point cloud may still be cluttered, an additional layer to filter the point cloud is applied. The plane segmentation filter examines the point cloud and returns the plane that consists of the most points. Consequently, some sensor inaccuracies are filtered out, since every point in the new point cloud supports the same plane model. The model that was chosen for this algorithm is RANSAC, because of its simplicity, its robustness as estimator and its high accuracy. A shortcoming of using a RANSAC model when segmenting a plane is that RANSAC does not perform well when there are multiple dominant wall planes, for example two perpendicular wall planes. In this case, this is not a problem since the point cloud in question is already cropped to the SPOT robot's assumptive door coordinates, so it is certain that the point cloud only contains one dominant plane. Furthermore, it is necessary to set a distance threshold, which determines when a point is considered to be an inlier. In this research, the distance threshold value was chosen as 10 centimeters, because this value had a good trade-off between simplifying the door frame, while maintaining the door frame to be distinguishable. Figure 6.4 visualizes the plane segmentation step. Note that the white point cloud is the original cloud after cropping and statistical outlier removal and the red point cloud is the white point cloud after plane segmentation using RANSAC. The fact that every point in the plane segmented (red) point cloud supports the same plane model is visible from the side view.



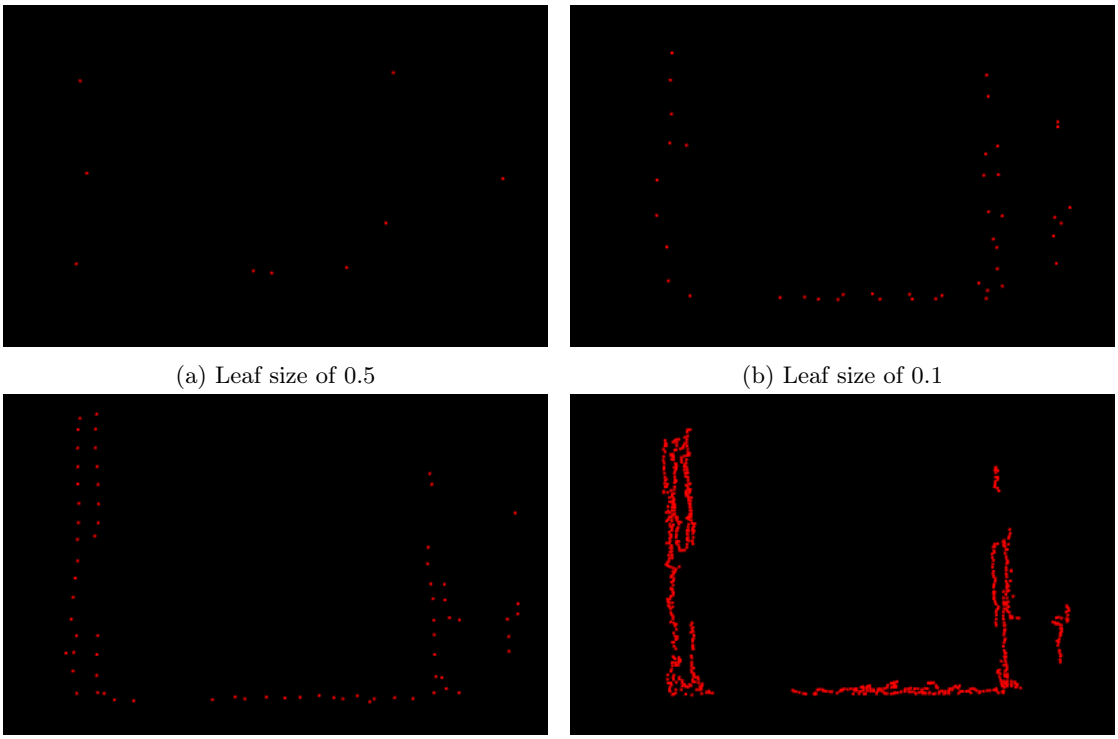
(a) Front view of the plane segmentation.

(b) Side view of the plane segmentation.

Figure 6.4: Front and side view of before (white) and after (red) the plane segmentation step. Note that in the red point cloud every point lies on the same plane.

6.3.4 Voxel grid downsampling

The voxel grid filter creates a grid of voxels in the point cloud. Internally, this can be visualized as 3D pixels. The points inside each voxel are then approximated by their centroid. This leads to downsampling (reduction of the number of points) in the point cloud data. When using a voxel grid filter, a leaf size can be chosen. The leaf size is the size of the voxels. Thus, a larger leaf size means that the voxels are bigger, and that there are less voxels in total. Consequently, since every voxel returns one point, which is the centroid of all the points in that voxel, the downsampled point cloud consists of less points. Alternatively, when the leaf size is really small, it is less of a downsampling, since the "downsampled" point cloud may consist of as many point as before downsampling. The importance of the leaf size is demonstrated in Figure 6.5. Note that the size of the points are slightly increased for visualization purposes.



(a) Leaf size of 0.5

(b) Leaf size of 0.1

(c) Leaf size of 0.05

(d) Leaf size of 0.01

Figure 6.5: Difference leaf sizes when downsampling with voxel grid filter.

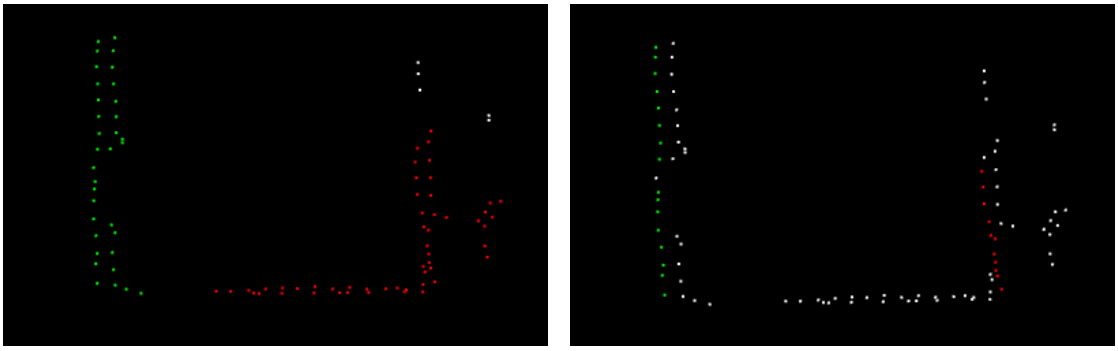
Choosing a leaf size is a trade-off between a simplicifaction of the point cloud, but in such a way that the door frame is still distinguishable. Since it is crucial to be able to detect the door frame after the downsampling, in this research a leaf size of 0.05 was chosen, which resulted in a point cloud shown in Figure 6.5c.

6.3.5 Euclidean clustering

In this section, the Euclidean clustering that is applied to the point cloud will be discussed. In general, Euclidean clustering is an implementation that uses nearest neighbors to cluster points together. When using this cluster algorithm, multiple parameters have to be chosen. First of all, the minClusterSize and the maxClusterSize, that determine the smallest and biggest size of a cluster. For example, in a very noisy and dense point cloud, it may be beneficial to increase both parameters, whereas in a scattered point cloud, the opposite may be useful. Since in this research, the point cloud is downsampled and pre-processed thoroughly, relatively small values were chosen for those parameters (minClusterSize = 5, maxClusterSize = 100). Furthermore, the parameter clusterTolerance needs to be set. This parameter determines the distance that points in the same cluster are allowed to have. So if the value is too small, one object may contain multiple clusters, whereas a value too big leads to multiple objects belonging to the same cluster. A clusterTolerance of 10 centimeters was chosen, since the jambs of the door frame have such width. Lastly, a search method to search the entire point cloud has to be determined. In this research, the choice was made to represent the point cloud in a octree data structure, because this representation is very fast to build. Figure 6.6a shows the two clusters found by the algorithm.

6.3.6 Parallel line using RANSAC

Once the point cloud is clustered into two groups, where both clusters represent one side of the door frame, it is necessary to fit a vertical line to both clusters. Using the door frame information simplifies the task of finding the door frame, since horizontal lines can be filtered out. In this research, a model was used that determines a line parallel with a given axis (here z-axis), within a maximum specified angular deviation. This epsAngle is the maximum allowed difference between the model normal and the given axis, and is specified to be 0.2 radians. The inliers are can extracted using RANSAC, which is shortly described in Section 3.3. The result of fitting lines through the found clusters using RANSAC can be seen in Figure 6.6b. In case multiple clusters, and therefore multiple RANSAC lines are found, an extra check is applied to find the best match between all possible combinations by using the known dependence between both lines, which is that there is approximately 90 centimeters between them.



(a) After Euclidean clustering

(b) After RANSAC parallel line fitting

Figure 6.6: Front view of the point cloud after euclidean clustering (a), and after RANSAC parallel line fitting (b).

6.4 Summary

The SPOT robot has a tendency to drift, which leads to a discrepancy between the SPOT robot's assumptive and actual position. Therefore, an approach is proposed to detect the door and thereby eliminate the drift. To summarize, the door detection algorithm firstly crops the obtained point cloud to the relevant part, which is the door, whereafter statistical outlier removal and plane segmentation is applied to remove outliers. Additionally, the resultant point cloud is downsampled to a voxel grid, whereafter euclidean clustering and RANSAC line fitting is utilized to detect the door frame. Consequently, the door frame in relation to the SPOT robot can be acquired. The last step is for the SPOT robot to correct its position based on this information to correct the drift. This approach will be evaluated in Chapter 8.

CHAPTER 7

Results: point planning

In this chapter, the experiments of the implementation of the aforementioned approach (see Chapter 5) will be discussed. This research was conducted with the data of the villa at the TNO location (see Figure 5.1). Since the hall consists of the most diverse and challenging doors, the results of the point planning in the hall will be examined. The hall is connected to five doors.

In this hallway, the point planning has to be performed on four non-trivial locations, since the free space before those four doors is cut off by the room (see Figure 7.1a). Consequently, an alternative pose before those four doors has to be found. The possible alternative poses are established by predefined angles and predefined points that lie on the dodecagon around the door. Poses too close to the wall are filtered out. The result after the filtering can be seen in Figure 7.1b.

Additionally an ellipse of 1.2 meters by 0.6 meters is drawn around every pose. This ellipse represents the SPOT robot's dimensions. Again, all ellipses, and consequently the corresponding poses, are filtered out if they are discontinued by a wall. The resultant safe poses are shown in Figure 7.1c.

Table 7.1 shows the number of evaluated poses (25 poses for every door), the number of poses after excluding poses with a distance of more than 30 centimeters to the closest wall and the number of poses after excluding poses of which the SPOT robot's body, represented with an ellipse, collides with the wall.

Name of door	Number of all predefined poses	Number of possible poses (after wall filtering)	Number of safe poses (after area error filtering)
hall to parents	25 poses	5 poses	3 poses
hall to living	25 poses	15 poses	8 poses
hall to kitchen	25 poses	15 poses	9 poses
hall to daughter	25 poses	25 poses	14 poses

Table 7.1: Number of filtered poses per door.

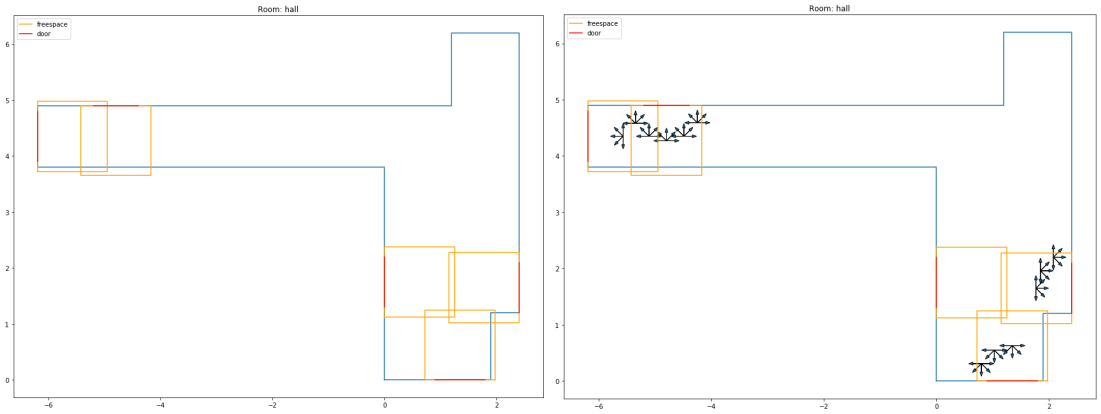
	Door: hall to parents			Door: hall to living		
rank	pose (x, y, θ)	area error	angle error	pose (x, y, θ)	area error	angle error
1	(-5.57, 4.35, $\frac{1}{2}\pi$)	0.0	0.0	(1.77, 1.65, $-\frac{1}{2}\pi$)	0.0	0.0
2	(-5.57, 4.35, $\frac{3}{4}\pi$)	0.0	$\frac{1}{4}\pi$	(1.85, 1.96, $-\frac{3}{4}\pi$)	0.0	0.27215
3	(-5.57, 4.35, $\frac{1}{4}\pi$)	0.0	$\frac{1}{4}\pi$	(2.08, 2.2, π)	0.0	0.52694
4	(-5.57, 4.35, 0)	0.01589	$\frac{1}{2}\pi$	(1.77, 1.65, $-\frac{1}{4}\pi$)	0.0	$\frac{1}{4}\pi$
5	(-5.57, 4.35, π)	0.01589	$\frac{1}{2}\pi$	(1.85, 1.96, π)	0.0	1.05754
6	–	–	–	(1.85, 1.96, $-\frac{1}{4}\pi$)	0.0	1.29865
7	–	–	–	(1.85, 1.96, 0)	0.0	2.08405
8	–	–	–	(2.08, 2.2, 0)	0.0	2.61465
9	–	–	–	(1.77, 1.65, π)	0.00338	$\frac{1}{2}\pi$
10	–	–	–	(1.77, 1.65, 0)	0.00338	$\frac{1}{2}\pi$

	Door: hall to kitchen			Door: hall to daughter		
rank	pose (x, y, θ)	area error	angle error	pose (x, y, θ)	area error	angle error
1	(1.35, 0.63, π)	0.0	0.0	(-4.49, 4.35, $\frac{1}{4}\pi$)	0.0	0.27215
2	(1.03, 0.55, $-\frac{3}{4}\pi$)	0.0	0.25845	(-5.11, 4.35, $-\frac{1}{4}\pi$)	0.0	0.27215
3	(0.8, 0.31, $-\frac{1}{2}\pi$)	0.0	0.51325	(-4.25, 4.59, $\frac{1}{2}\pi$)	0.0	0.51325
4	(0.8, 0.31, $\frac{1}{2}\pi$)	0.0	0.51325	(-5.35, 4.58, $-\frac{1}{2}\pi$)	0.0	0.51325
5	(1.35, 0.63, $-\frac{3}{4}\pi$)	0.0	$\frac{1}{4}\pi$	(-4.49, 4.35, $\frac{1}{2}\pi$)	0.0	1.05754
6	(1.03, 0.55, $-\frac{1}{2}\pi$)	0.0	1.04285	(-5.11, 4.35, $-\frac{1}{2}\pi$)	0.0	1.05754
7	(1.03, 0.55, $\frac{1}{2}\pi$)	0.0	1.04285	(-5.11, 4.35, $\frac{1}{4}\pi$)	0.0	1.29865
8	(1.03, 0.55, $\frac{3}{4}\pi$)	0.0	1.82925	(-4.49, 4.35, $-\frac{1}{4}\pi$)	0.0	1.29865
9	(1.35, 0.63, $\frac{3}{4}\pi$)	0.0	$\frac{3}{4}\pi$	(-4.8, 4.27, $\frac{1}{2}\pi$)	0.0	$\frac{1}{2}\pi$
10	(1.03, 0.55, π)	0.00795	0.52694	(-4.8, 4.27, $-\frac{1}{2}\pi$)	0.0	2.08405

Table 7.2: Top ten poses of each door and its corresponding ranking values. The following applies to the errors: the lower, the better.

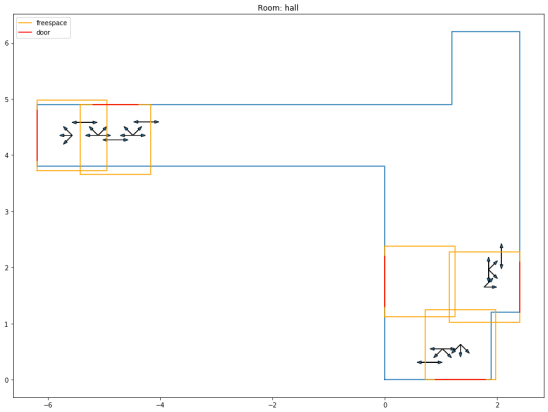
Then the safe poses are ordered by orthogonality to the door, since the new pose is required to face the door, and preferably is orthogonal to the door. Table 7.2 shows the top ten poses for each door, and the corresponding values on which the ranking is made: the area error and the angle error. Note that all poses with an area error larger than 0 are already filtered out, which is why the door from the hall to the parents only shows the top five poses. Furthermore, the number of safe poses shown in Table 7.1 match with the number of doors in Table 7.2 with an area error of 0. Needless to say, the lower the area error and the angle error are respectively, the better.

The best poses can be seen on the map of the hall in Figure 7.1d. For visualization purposes, only the best three poses are shown with the corresponding ellipse. These best positions are the start points of the SPOT robot's internal path-planning to plan a collision free path through the doorway.

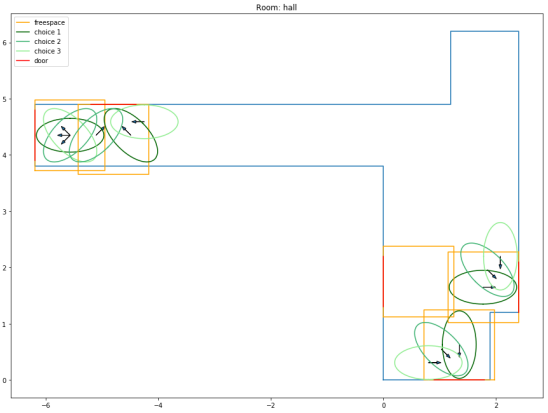


(a) Determined free spaces

(b) Possible alternative poses



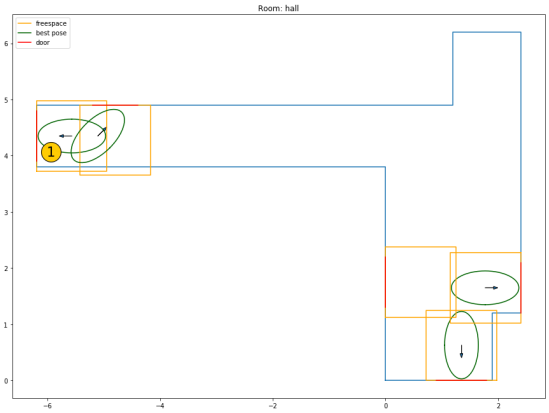
(c) Safe poses



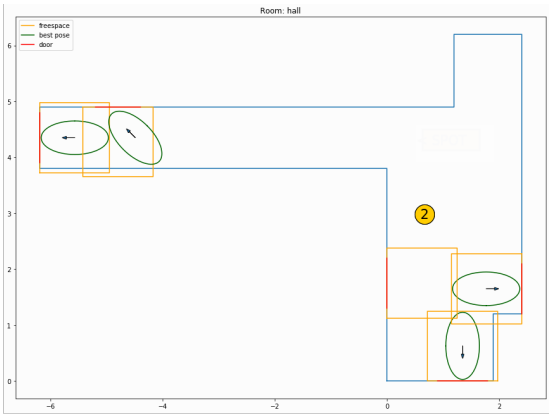
(d) Best three poses

Figure 7.1: Point planning results

Lastly, the planned points also depend on the current position of the SPOT robot. Figure 7.2 shows the best poses of two scenarios where the SPOT robot is positioned differently.



(a) Best poses when the SPOT robot is at the 1.



(b) Best poses when the SPOT robot is at the 2.

Figure 7.2: Two different scenarios regarding the SPOT robot's position.

CHAPTER 8

Results: door detection

In this chapter, the approach discussed in Chapter 6 will be evaluated. This is done with point cloud data from aforementioned villa at the TNO location.

Similar to previous chapter, the results of the door detection of just the hall will be examined, because the hall consists of the most diverse and challenging doors. Consequently, it is a good opportunity to test the best three poses retrieved by the point planning algorithm shown in Figure 7.1d. Therefore, in total twelve point clouds will be tested, which can be categorized in four categories by pose towards the door. The definitions of the categories and the amount of tested point clouds of each category can be seen in Table 8.1.

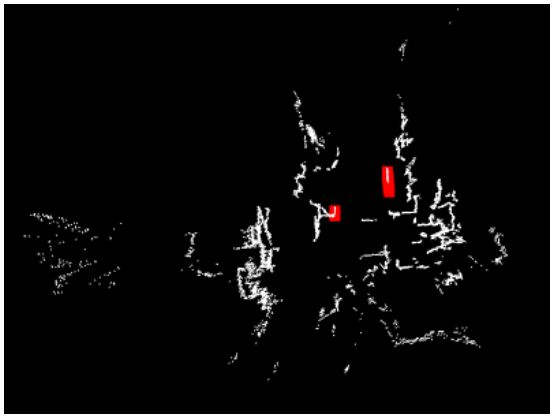
Category	Description	Number of point clouds
A	Perpendicular (angle to door: $\frac{1}{2}\pi$)	3 point clouds
B	Diagonally (angle to door: $\frac{1}{4}\pi$)	4 point clouds
C	Diagonally (angle to door: $1\frac{3}{4}\pi$)	2 point clouds
D	Parallel (angle to door: 0)	3 point clouds

Table 8.1: Categories of this experiment.

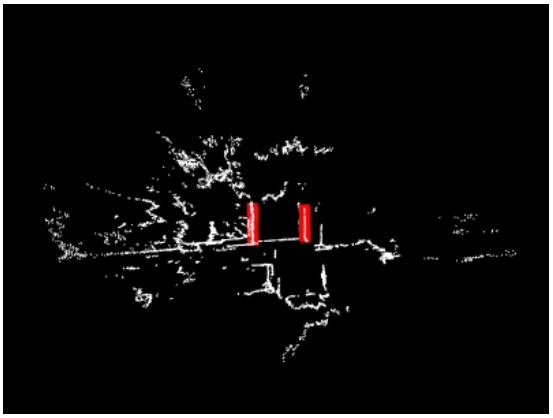
In the following sections, the results of applying the door detection algorithm to the obtained point clouds will be shown with visualizations and discussed per category. The original point cloud as captured by the SPOT robot will be shown in white. When the algorithm detects a door, or thinks it has detected a door, the detected door will be plotted in red. This is plotted over the original white point cloud, but the point size is increased to show it more clearly. When the algorithm could not detect a door in a point cloud, the door jamb that had to be detected is highlighted in purple, which is done manually for visualization purposes.

8.1 Category A

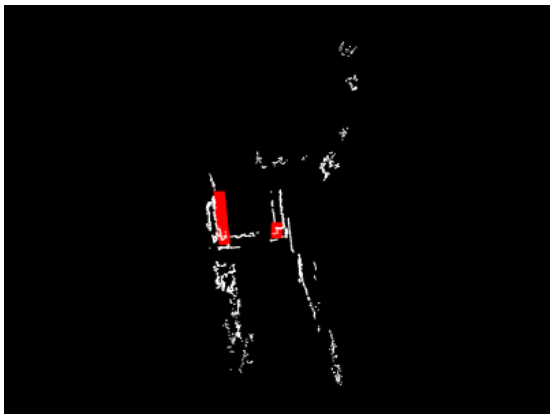
In this category, the point clouds obtained by the SPOT robot will be taken directly before the door. This category also includes point clouds that are obtained when the point planning algorithm is not applied, and therefore the points before the door are planned in a scripted manner, since in that case, the SPOT robot will be facing the door directly as well. The results of applying the door detection pipeline to these point clouds are shown in Figure 8.1. Notice that the point clouds of this category are all returned as the best pose in the point planning algorithm. This is due to the fact that the angle error ensures that poses directly facing the door are preferred.



(a) Point cloud captured in front of the kitchen (first pose)



(b) Point cloud captured in front of the living room (first pose)



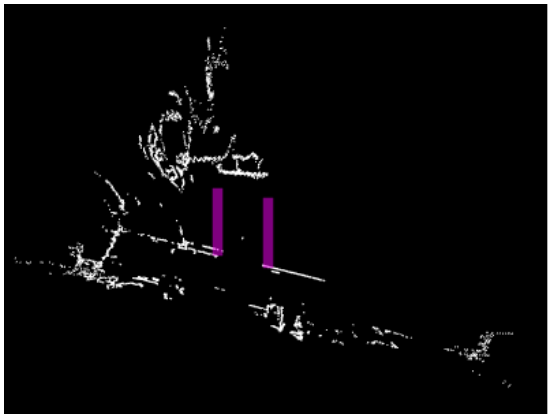
(c) Point cloud captured in front of the parents' bedroom (first pose)

Figure 8.1: Results category A. The door jambs detected by the algorithm are shown in red.

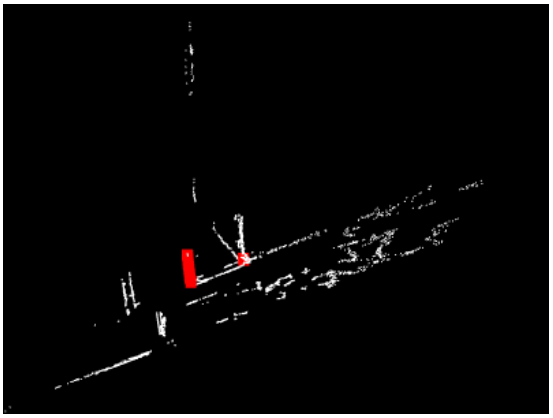
Note that some door jambs are somewhat shorter than other door jambs. In case of the left door jamb seen in Figure 8.1a, the entire door jambs is not captured in the point cloud, since the door jambs is not visible in the original point cloud (in white) as obtained by the SPOT robot. This means that the door detection algorithm performed to the best of its ability given the obtained point cloud. As for the right door jamb in Figure 8.1c, the shortness is probably due to the fact that the area around that door jamb in the point cloud was too cluttered, which resulted in the algorithm to regard the points in the (for people) distinguishable lines as outliers and filter those out in the outlier removal step of the algorithm. However, since for the door correction not the entire door frame coordinates are needed, this is not problem. Consequently, the algorithm was able to detect the door correctly in every case of this category, which yields an accuracy of 100%.

8.2 Category B

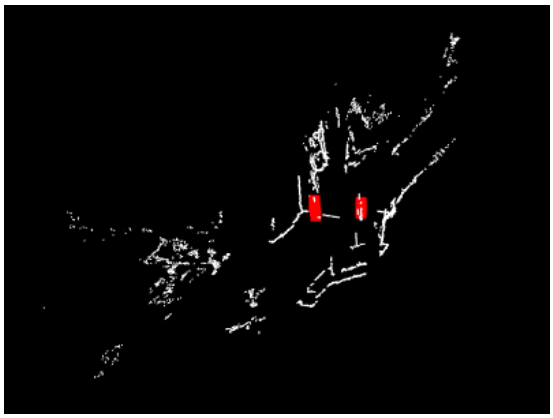
Category B consists of point clouds that are captured when the SPOT robot is positioned diagonally before the door, while facing the door. The angle between the door and the SPOT robot is then $\frac{1}{4}\pi$ radians. In this experiment, there are four point clouds obtained this way. These are shown in Figure 8.2.



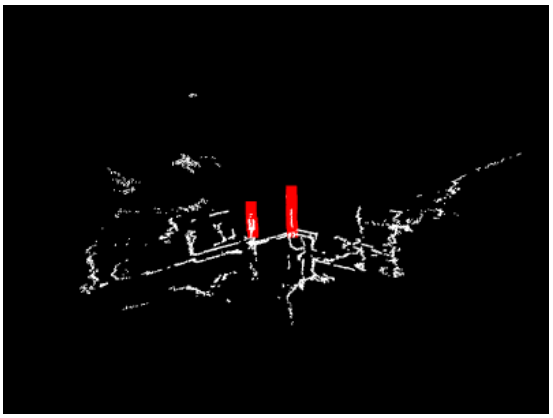
(a) Point cloud captured in front of the daughter's bedroom (first pose)



(b) Point cloud captured in front of the daughter's bedroom (second pose)



(c) Point cloud captured in front of the kitchen (second pose)



(d) Point cloud captured in front of the living room (second pose)

Figure 8.2: Results category B. The door jambs detected by the algorithm are shown in red. The purple lines represent where the door should have been detected (false negatives).

The door detection algorithm performs fairly well for point clouds in this category, since the figure above shows that the door was found in three of four cases. Figure 8.2a shows the point cloud in which the door was not correctly detected, but this is due to the fact that the door frame was not captured at all in the point cloud. Note that the purple lines are manually added to indicate where the door had to be determined; the algorithm did not find it. The rest of the figures show that the door was correctly found. In Figure 8.2b, the right door jamb is again somewhat shorter. This may be due to the fact that the sensor data around the right door jamb is noisy, which leads to the algorithm filtering out a big portion of the line. Nevertheless, the algorithm was able to detect the door. This concludes the experiments of this category with an accuracy of 75%.

8.3 Category C

Category C consists of point clouds that are captured when the SPOT robot is positioned diagonally before the door. However, in contrast with category B, the SPOT robot does not face the door, but is positioned the other way. Therefore, the angle between the door and the SPOT robot is $1\frac{1}{4}\pi$ radians. In this experiment, two point clouds fall in this category. Notice that the point clouds in this category are taken in front of the same door. These are shown in Figure 8.3.

(a) Point cloud captured in front of the parents' bedroom (second pose)

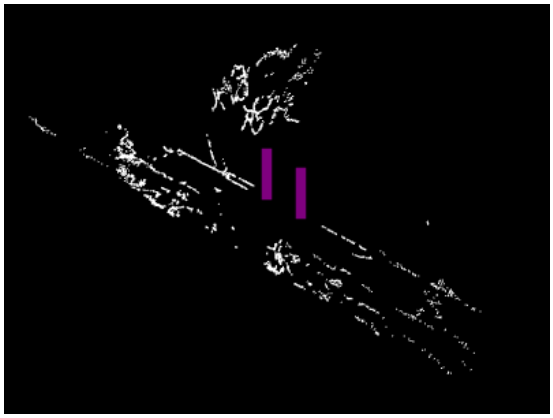
(b) Point cloud captured in front of the parents' bedroom (third pose)

Figure 8.3: Results category C. The door jambs detected by the algorithm are shown in red. The purple lines represent where the door should have been detected (false negatives).

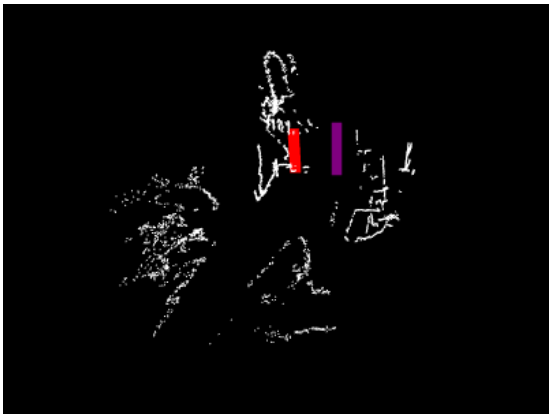
In the experiments of this category, only three of the four door jambs have been found. This gives an accuracy of 75% of detecting the door jambs correctly, but strictly, only one of two doors has been correctly detected, which gives an accuracy of 50%. Note the shorter door jamb in Figure 8.3a, which is a problem addressed earlier, but nevertheless the algorithm was able to detect the door correctly. In Figure 8.3b, the result shows that only one door jamb was detected. The left door jamb (visualized in purple) is not captured correctly in the point cloud, which makes it impossible for the algorithm to detect the door, since the algorithm relies on line fitting of the points in the point cloud.

8.4 Category D

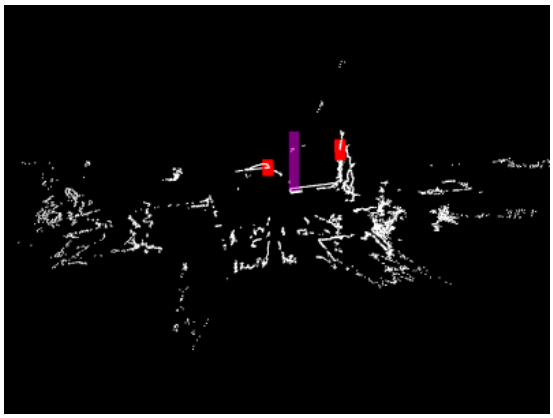
In the last category of point clouds that are experimented with, the point clouds are taken while the SPOT robot is positioned parallel to the door. There are three point clouds obtained this way. Those are shown in Figure 8.4.



(a) Point cloud captured in front of the daughter's bedroom (third pose)



(b) Point cloud captured in front of the kitchen (third pose)



(c) Point cloud captured in front of the living room (third pose)

Figure 8.4: Results category D. The door jambs detected by the algorithm are shown in red. The purple lines represent where the door should have been detected (false negatives).

The door detection algorithm does not perform well with point clouds in this category; it does not meet the expectations. In Figure 8.4a, two purple lines are visible, which represent the door frame that should have been detected. This was not the case, since there were no points in the point cloud that represented the door frame. In other words, a human-being would most likely also be unable to detect the door with given point cloud. This same situation arises in Figure 8.4b and Figure 8.4c as well, where one door jamb is not visible in the point cloud. In both experiments, the algorithm is able to detect just one correct door jamb. In Figure 8.4c, there is also a case of a false positive, which is a door jamb that is detected by the algorithm when there should not be one. This happened with the right door frame. So, the algorithm detected the two red lines as the door frame, but in reality the door frame consists of the purple line and the red line right from the purple line. This false positive most likely occurred due to the window and closet next to the door (See Figure B.1 in Appendix B). To combine the results, two of the six door jambs were detected correctly, which yields an accuracy of approximately 33,33%, but none of the three doors were detected correctly, which results in an accuracy of 0%.

8.5 Statistics

In this chapter, twelve point clouds in four categories were tested to evaluate the door detection algorithm. An overview of the statistics is shown in Table 8.2.

The table shows the door jambs that were correctly detected by the algorithm, and compares this to whether the door frame could be found by a human given the same point cloud. From the

	Found correctly		Not found correctly		
Category	Detectable	Not detectable	Detectable	Not detectable	Total
A	6	0	0	0	6
B	6	0	0	2	8
C	3	0	0	1	6
D	2	0	0	4	6
Total	17	0	0	7	24

Table 8.2: Overview of results. The table shows the number of door jambs and whether they were found correctly by the algorithm, and whether a human would have detected the door jambs given the point cloud.

table we can deduce that overall 17 of the 24 door jambs were detected correctly, which yields an accuracy of approximately 70.80%. However, the door jambs that could not be correctly detected were also not captured well in the point cloud. This means that a human would probably also have difficulties detecting the location of the door. This is due to the camera (placement) of the SPOT robot. Firstly, the SPOT robot is equipped with five stereo cameras, of which depth images are converted into point clouds. This may result in sensitivity to light, whereas a range sensor would not have this problem. Secondly, the cameras are pointing slightly downwards, which means that every point above one meter is not visible. A camera that is able to capture the entire environment is less prone to noise, because a larger part of the door is visible. Thirdly, the SPOT robot seems to have a "blind spot" when the camera is too close to the wall. This results in the door frame not being visible at all in the point cloud, for example in Figure 8.4a. Therefore, the current bottleneck of the door detection algorithm is probably the stereo cameras of the SPOT robot. Nevertheless, the door detection algorithm was able to reach an accuracy of 70.80%. If the accuracy is calculated by the number of doors that were detected correctly (instead of door jambs), the algorithm would reach an accuracy of approximately 58.33% (7 of 12 doors). This is a start, but there is a lot of improvement possible, which will be discussed in the next chapter.

CHAPTER 9

Conclusions

This study set out to make the SPOT robot's performance in a search-and-rescue mission more reliable by applying a point planning approach and correcting the drift based on door detection. The concluding findings will be examined in this chapter.

9.1 Point planning

Currently, the point planning before the door is done in a scripted manner: 1 meter before each door. Problems may arise when the planned point is not accessible to the SPOT robot. Therefore, this study suggests an improved approach to plan a safe point before the door. This deterministic approach takes the map of the environment into account, and finds relevant poses before the door, whereafter unsafe poses are filtered out and the rest is sorted from best to worst. This sorting enables the SPOT robot to potentially try the next best pose, when the best pose is not possible for some reason.

9.1.1 Discussion

The small sample size of the experimented doors did not allow to make any generalizations about the results of the point planning algorithm. Currently, the algorithm is only tested in the villa at the TNO location, where for example every door size is identical. Therefore, to make some generalizations about the performance of the algorithm, this has to be tested more thoroughly. Notwithstanding this limitation, the results suggest that the algorithm is able to find the best poses safely.

9.1.2 Future work

Future research might explore the possibilities of considering obstacles before the door. For example when a garbage can is positioned before the door, the SPOT robot has to find a safe way to enter through the door without moving the garbage can. A few alterations to the current approach can make this possible, since the SPOT robot captures an obstacle map as well. When the obstacle map can be combined with the knowledge of the map of the environment, the SPOT robot will be able to avoid the obstacles. Furthermore, this addition allows for considering moving objects as well, which will be an extremely useful contribution to the current point planning.

9.2 Door detection

In the door detection section of this research, a solution is proposed to correct the drift that the SPOT robot tends to have. This drift correction can be done by detecting a landmark, in this case a door. The door detection approach consists of a proposed pipeline that processes the obtained point cloud by the SPOT robot and consists of cropping, statistical outlier removal, plane segmentation, voxel grid downsampling, Euclidean clustering and line fitting. This pipeline reached an accuracy of 70.80% of detected door jambs, and an accuracy of 58.33% of detected entire door frames.

9.2.1 Discussion

In this study, the results are reported with 12 experimented doors in the villa at the TNO location. Since the study was limited to just 12 doors, it is not possible to make generalizations about the door detection algorithm. Furthermore, it can be interesting to study more different poses before the door when capturing the point cloud, since this has an effect on whether the door frame can be detected correctly.

9.2.2 Future work

The results of the door detection algorithm indicate that currently the primary bottleneck is the quality of the obtained point clouds. The point clouds are slightly pointed downwards, which means that point above approximately 1 meter are not visible. Therefore, this door detection algorithm is not comparable to other discussed algorithms in the field. Furthermore, the SPOT robot seems to have some "blind spots" when capturing the point cloud. These are points where the SPOT robot is not able to capture points at all, which results in visible holes in the point cloud. Future research should be carried out to establish whether the algorithm would perform better with a different sensor, such as the PAL sensor. An alternative is to aggregate multiple point clouds, which may fill those "blind spots" and make the data set less sparse.

Furthermore, a natural progression of this work is to analyze the pipeline intensively by performing an Ablation study, which looks that the contribution of every step in the pipeline individually by skipping them one by one. Such study may provide new insights about the proposed door detection algorithm. Furthermore, the scope of this study was limited in terms of some generalizations, such as the cropping step is currently done manually. However, the SPOT robot allows to consider its currents position relative to the door, which makes automatic cropping possible. This would increase the usability of the door detection algorithm considerably.

9.3 Conclusion

The research question of this thesis is:

> Can the SPOT robot's performance in a search-and-rescue mission be made more flexible by a deterministic point planning approach and applying door detection for drift correction?

To formulate an answer to this research question, the results of both components have to be combined. Firstly, the point planning has definitely improved the flexibility of the SPOT robot's performance, since the points before the door were before this study planned in a scripted manner, which is not flexible at all. Secondly, the concluding findings of the door detection algorithm yielded promising results, but there was a significant difference between the categories (poses before the door). Therefore, the door detection can not yet be applied successfully to every door. However, every door detected prevents accumulated drift, something that is not done in the SNOW project before this study. Consequently, the door detection algorithm is able to, even it does not have an accuracy of 100%, correct the drift when it succeeds, which is an improvement in the SPOT robot's flexibility as well. To conclude, the SPOT robot's performance in a search-and-rescue mission has been made more flexible. While this study is definitely a step in the right

direction by making the performance more flexible, whether the performance can be entirely robustified still remains to be seen in future work. Nonetheless, this thesis has provided a deeper insight into the reliability of the performance, and lays the groundwork for future research into robustification of the SPOT robot's performance in a search-and-rescue mission.

Bibliography

[1] Purificacion Carinena et al. "Landmark detection in mobile robotics using fuzzy temporal rules". In: *IEEE Transactions on Fuzzy Systems* 12.4 (2004), pp. 423–435.

[2] Matthew Derry and Brenna Argall. "Automated doorway detection for assistive shared-control wheelchairs". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 1254–1259.

[3] Yuri Feldman and Vadim Indelman. "Spatially-dependent Bayesian semantic perception under model and localization uncertainty". In: *Auton. Robots* 44.6 (2020), pp. 1091–1119. DOI: 10.1007/s10514-020-09921-0. URL: https://doi.org/10.1007/s10514-020-09921-0.

[4] Ron Goldman. "Understanding quaternions". In: *Graphical models* 73.2 (2011), pp. 21–49.

[5] Richard Honti, Ján Erdélyi, and Alojz Kopáčik. "Plane segmentation from point clouds". In: *Pollack Periodica* 13.2 (2018), pp. 159–171.

[6] Burak Kakillioglu, Koray Ozcan, and Senem Velipasalar. "Doorway detection for autonomous indoor navigation of unmanned vehicles". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 3837–3841.

[7] G Lowe. "Sift-the scale invariant feature transform". In: *Int. J* 2.91-110 (2004), p. 2.

[8] Steve Marschner and Peter Shirley. *Fundamentals of computer graphics*. CRC Press, 2018.

[9] Robin R. Murphy, Satoshi Tadokoro, and Alexander Kleiner. "Disaster Robotics". In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Springer Handbooks. Springer, 2016, pp. 1577–1604. DOI: 10.1007/978-3-319-32552-1_60. URL: https://doi.org/10.1007/978-3-319-32552-1%5C_60.

[10] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[11] Blanca Quintana et al. "Door detection in 3D coloured point clouds of indoor environments". In: *Automation in Construction* 85 (2018), pp. 146–166.

[12] João Ramôa et al. "Real-Time 2D-3D Door Detection and Classification on Jetson Nano". In: (Jan. 2021).

[13] Radu Bogdan Rusu and Steve Cousins. "3d is here: Point cloud library (pcl)". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 1–4.

[14] Radu Bogdan Rusu et al. "Laser-based perception for door and handle identification". In: *2009 International Conference on Advanced Robotics*. IEEE. 2009, pp. 1–8.

[15] Raymond Sheh, Sören Schwertfeger, and Arnoud Visser. "16 Years of RoboCup Rescue". In: *Künstliche Intell.* 30.3-4 (2016), pp. 267–277. DOI: 10.1007/s13218-016-0444-x. URL: https://doi.org/10.1007/s13218-016-0444-x.

[16] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[17] J-M Valin et al. "Robust sound source localization using a microphone array on a mobile robot". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*. Vol. 2. IEEE. 2003, pp. 1228–1233.

[18] Karthik Mahesh Varadarajan and Markus Vincze. "3D room modeling and doorway detection from indoor stereo imagery using feature guided piecewise depth diffusion". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 2758–2765.

[19] Arnoud Visser et al. "Beyond frontier exploration". In: *Robot Soccer World Cup*. Springer. 2007, pp. 113–123.

[20] Ting Han Yuan et al. "An automated 3D scanning algorithm using depth cameras for door detection". In: *2015 International Electronics Symposium (IES)*. IEEE. 2015, pp. 58–61.

[21] Wei Zeng, Sezer Karaoglu, and T. Gevers. "Pano2Scene: 3D Indoor Semantic Scene Reconstruction from a Single Indoor Panorama Image". In: *BMVC*. 2020.

Appendices

APPENDIX A

SIFT

Spot has 5 pairs of stereo cameras that provide black and white images and video. These cameras are located on 5 different places on its body: on its left and right side, on its back and two on its front. The cameras at the front (called frontleft and frontright) ensure that the SPOT robot has a wider view by pointing the frontleft camera slightly to the right direction, and likewise for the frontright camera.

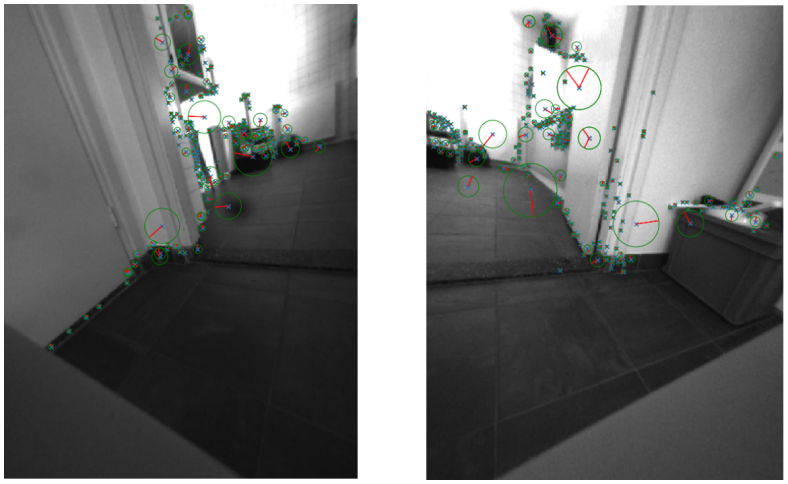
An example of the SPOT robot's front-facing view can be seen in Figure A.1.

Figure A.1: Left: frontright camera, right: frontleft camera

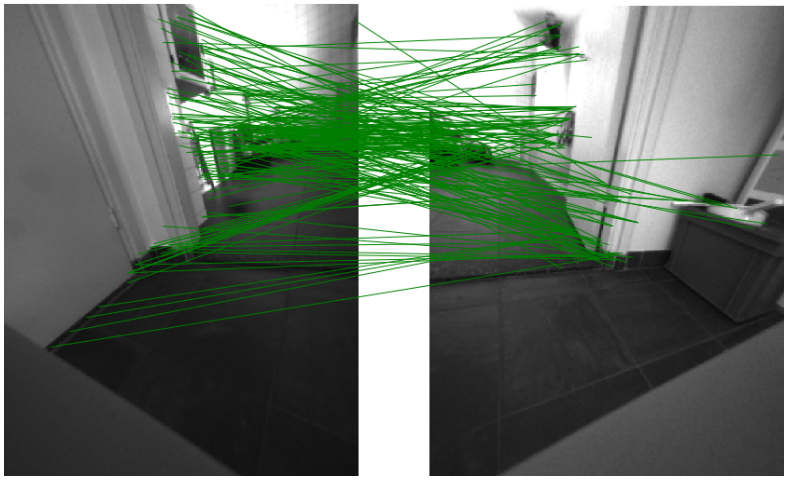
A possible idea was to apply SIFT (Scale-Invariant Feature Transform) to stitch these images together as one image, where the lower part of the door would be entirely visible. The next step of this approach was to apply Hough Transform to detect the door frame. Note that Hough Transform is only possible with images, which is why this approach was not chosen when working with point cloud data.

In order to stitch images together, SIFT matches features that are visible on both images with the following steps: scale space extrema detection, keypoint localization, orientation assignment

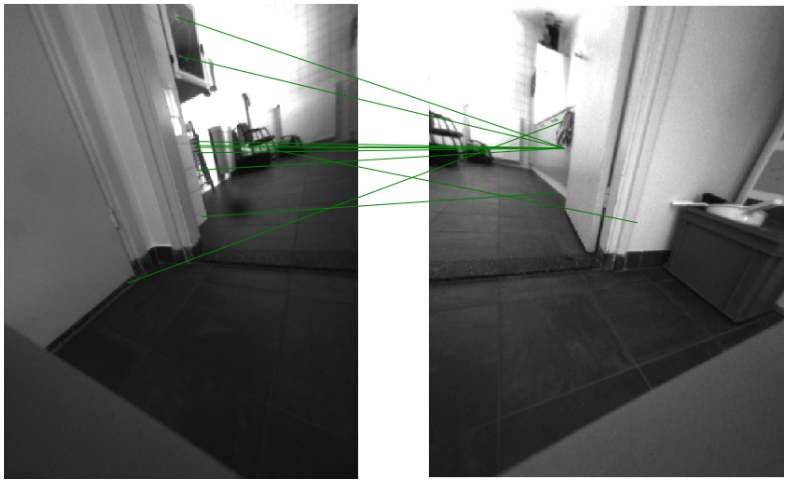
and keypoint descriptor. Then RANSAC can be applied to remove false matches. Since this approach was too insufficient to include in the original research, SIFT will not be explained in more depth. I recommend this classic and field-changing scientific paper about SIFT by Lowe [7].



(a) Extract interesting features using SIFT.



(b) The interesting features of both images are matched to each other.



(c) RANSAC filters out the outliers.

Figure A.2: Application of SIFT to the SPOT robot's front cameras.

A visualization of applying SIFT to afore shown images can be seen in Figure A.2, which demonstrates that the SIFT algorithm performs poorly. There are two possible explanations for that. Firstly, the camera of the SPOT robot does not capture images with high resolution quality. Secondly, since the camera just faces a door, there are not a lot of interesting features that SIFT can utilize to match both images.

APPENDIX B

Situation

Figure B.1: Pose of the SPOT robot when capturing the point cloud shown in Figure 8.4c. Note the vertical line that is visible through the window, which was detected by the SPOT robot as door jamb (false positive), since the point cloud did not capture the left door jamb correctly.