Determining placement solutions on a cluttered surface

van Vroonhoven, Max L.M.

Award date:
2023

4WC10 - Bachelor End Project - Control System Technology

Determining placement solutions on a cluttered surface

Quartile 1-2 2022-2023

Full Name	Student ID
Max van Vroonhoven	1460269

Project supervisor:
dr.ir. M.J.G. van de Molengraft

Project coach:
ir. P. van Dooren

Eindhoven, January 19, 2023

Contents

1 Introduction

The following report presents a method to gather the information necessary for a robot to place an object on an unknown table using its own sensors, without additional human input. The purpose of this project is to improve the quality of life of people who have become more dependent on external assistance for seemingly simple and mundane tasks, such as placing a can of soda on their table. As the world population ages, people have become more dependent on this external assistance and look towards robotics for a solution.

The robot that will be used for the project is a Toyota Human Support Robot further referred to as HERO. For the purpose of the project, the most important features of HERO are the gripper and the depth camera on top of its head. The gripper located at the end of HERO's 5 DOF arm will be used for object placement. For data extraction, the depth camera is used, which outputs RGBD images that can be processed further. First, in section 2, a literature study is performed to explore the techniques that can be used in the scope of the project. Second, in section 3, a method is devised using the techniques that have been analyzed. Third, section 4, describes testing the method in an open source simulator, gazebo, that accurately represents HERO's workings in addition to testing the method in real scenarios to find the limitations and problems with the method. Fourth, in section 5, the findings are discussed, and a conclusion regarding the method is formed in section 6.

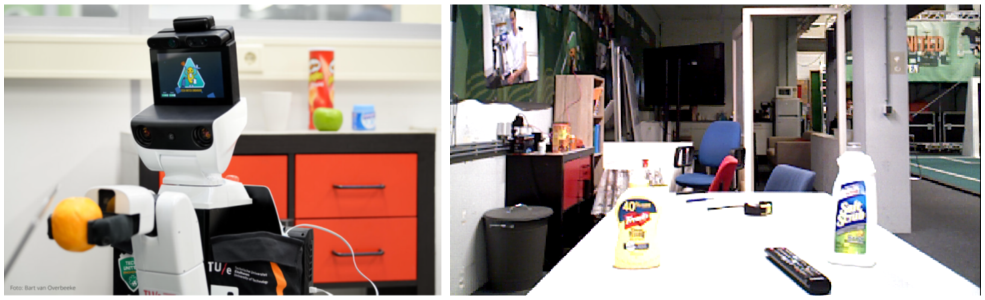


Figure 1: TechUnited's Toyota Human Support Robot called HERO next to a cluttered table surface

1.1 Features and limitations

For HERO to be able to place an object on top of an unknown table, it is of the essence that HERO can obtain more information about the table. This is required to find the free-space on the table where it can perform this action. To determine the free-space, the variations that are contained in the unknown aspect of the table are defined. There are numerous ways in which a table can vary, but there are a few essential characteristics that hold the most importance. Firstly, the geometry of the table must be considered. In the context of the project, the most important aspect of the table's geometry is the geometry of the table sheet and its height in relation to HERO. Secondly, the amount of objects and their respective shape on top of the table is an important variable to take into account. The space that these objects take up on the table sheet is considered occupied space. Thirdly, the amount of information that can be obtained from the table varies. For example, HERO might not be able to examine certain sections of the table sheet due to sensor limitations or an obstructed view. The latter is the case when objects obstruct the view between HERO and the table sheet. Therefore, sections that are not examined or can not be examined by HERO is considered unknown space. To continue, the devised method allows HERO to make a distinction between unknown and occupied space on the table sheet, a safety margin is considered around this space, which finally leaves the remaining space as free-space.

To make this distinction, certain assumptions have been made to simplify the problem. Currently, it is assumed that HERO is already positioned within a meter of the table and facing toward it. Furthermore, assumptions have been made regarding the target table and the environment it is located. Firstly, the table is horizontally level, this is assumed for the placement of objects. Secondly, depth information can be gathered on top of the table using the RGBD camera, therefore the devised method excludes highly reflective or transparent table sheets. Thirdly and finally, it is assumed that the table sheet is solid and contains no holes or gaps. This is considered when undefined unknown space is filtered out.

1.2 Method

To model the top of the table, HERO captures a depth image of the target table and its surroundings. The data that corresponds to the floor is removed after which the remaining data is segmented into point cloud clusters that feature a horizontal planar structure using parallel plane model segmentation. The most dominant cluster is separated. This cluster contains the table sheet, this is due to the removal of the floor and the assumption that HERO is already positioned close to the table. Furthermore, the distinction between placement options and occupied/unknown space is made. The occupied space, consisting of objects that are on top of the table, is defined by filtering out the data that is above the dominant plane. Once the objects are defined it is possible to describe the unknown, occluded, space that is invisible from HERO's perspective due to the objects. This can be done by using direction vectors between HERO's camera position and the position of the points that correspond to the objects. These vectors are then combined with the known height of the table, resulting in a solvable parametric equation. Furthermore, to describe additional unknown space, the space that does not belong to the table is defined. This is done by using direction vectors between the RGBD camera and the data that is below the table sheet, resulting in another solvable parametric equation. Space not belonging to the table sheet is described to find the edge of the table sheet and keep an appropriate distance from these edges when placing an object. In addition, the boundaries of HERO's field of view are defined, similarly to the table edges this information is required to keep an appropriate distance. To continue, placement options are filtered by firstly, constraining the free-space to be a preferred distance away from HERO. Secondly, a safety margin around the occupied and unknown is considered based on HERO's placement error and the size of the object that will be placed, this step will also remove the reading errors on the table sheet's surface. Finally, from the remaining placement options a placement solution is selected, this point is in the center of HERO's view if possible, and is otherwise as close to the center as possible.

2 Literature

The following section aims to explore existing techniques that may be of use within the scope of the project. Furthermore, since this is a continuation project within the TechUnited department, previously performed research will be analyzed and a conclusion will be drawn whether the techniques that are presented in this research can be continued with. To continue, HERO is equipped with a camera that outputs RGBD images. These are images where, besides the standard RGB color information, each pixel also contains an additional depth value. For this reason, methods using point cloud data will be the main focus of the literature research. Considering, point clouds allow HERO to interpret and process the depth information from its surroundings. Besides point cloud processing techniques, image processing techniques are required to visualize the point cloud data and to process the data further.

2.1 Segmentation

In order to locate and evaluate specific regions within point cloud data, the point cloud data will need to be divided into smaller sections first. These sections are called clusters and the process of creating clusters is called segmentation. There are numerous segmentation techniques, that are mostly dependent on the homogeneous properties of the region that the user wishes to evaluate. Within the scope of the project, it is desired to evaluate the top of the table. Therefore, the segmentation techniques that will be investigated are focused on finding a horizontal plane in space.

2.1.1 Plane model segmentation

The first segmentation technique that will be explored is plane model segmentation, which is a Model-Fitting-Based Method [2]. This segmentation technique attempts to find a homogeneous set of points that support a planar model inside a given data set. To determine whether a point is a part of the planar model a 'distance threshold' must be specified. This threshold determines how close a point must be to the model in order to be considered an inlier. The Random Sample Consensus (RANSAC) method is used as the robust estimator of choice. This choice is motivated by RANSAC's simplicity, other estimators build further upon RANSAC and add additional concepts for more specific purposes. RANSAC is an iterative method to estimate parameters of a mathematical model from a data set that contains both inliers and outliers [1]. The algorithm generates solutions using the least amount of data points required to make an appropriate fit of the required model.

Since plane model segmentation is a method that focuses on fitting a given model, it is essential that the given model is appropriate for the task at hand. In the scope of the project, the segmentation target is the table sheet. Since this is a parallel plane the described segmentation technique seems to be an appropriate choice. However, further analysis into other segmentation methods is performed to determine whether there are alternative techniques.

2.1.2 Euclidean clustering

Secondly, euclidean clustering is another technique that is able to cluster homogeneous point cloud data. Euclidean clustering makes use of a 3D grid subdivision of the space using fixed-width boxes. Points are clustered based on the distance between their neighboring points. The technique requires a maximum distance between points and the expected range for the cluster size. This approach is very fast to build and can be exceptionally useful where a volumetric representation of space is required. Euclidean clustering is a feature-based clustering method [2] meaning that this technique primarily uses the geometric structure or spatial-distribution features of points clouds to cluster them. As a result, this may lead to issues in the scope of the project. Since these features are not predictable in the dynamic environment for the use case.

2.1.3 Region growing segmentation

Thirdly, another technique that may be of use is region-growing segmentation, this segmentation type makes use of a Region-Growing-Based Method [2]. This segmentation technique selects a point, a 'seed', and for every seed point, the algorithm finds its neighbouring points [6]. The neighbouring points are then tested. Firstly, every neighbouring point is tested for the angle between its normal and the normal of the current seed point. If the angle is lower than a predefined threshold value then the point is added to the current region. Secondly, every neighbouring point is tested for a curvature threshold, if the curvature is less than the threshold value then the point is added to the set of seeds. Finally, the seed point that has been analyzed is removed from the set of seeds. As a result, once the set of seeds becomes empty it means that the algorithm has completed the growth of a region.

It can be concluded that this segmentation technique relies on a correct selection of the initial seed point. Therefore, considering the purpose of the segmentation technique in the scope of the project it becomes challenging to consistently select an appropriate initial seed point.

2.2 Alternative data processing

After the cluster containing the table has been found, it is unclear how the distinction between free-, occupied- and unknown space can be made. To accomplish this the point cloud data has be to processed further using other techniques that are not necessarily limited to cluster segmentation techniques.

2.2.1 Tabletop Object Detector

The tabletop object detector is a package that 'performs object segmentation and simple object recognition for constrained scenes' [7]. First, the table sheet must be found in the scene using another type of segmentation. Once the table sheet is defined, the points above this plane are considered objects. A clustering algorithm is used to identify individual objects, these individual objects are considered separate clusters. For each object cluster, an iterative fitting technique is used to fit the clustered object to an existing mesh in a database of models. If the fit is appropriate, the database will id the cluster. To perform these steps the method relies on the following assumptions: First, The objects are resting on a table, which is the dominant plane. Second, there is a minimum distance between two objects, this is a specified threshold value. Third and final, the method can only deal with 2 degrees of freedom. Namely, translation along the X and Y axes, where the Z-axis is assumed to be the surface normal of the table sheet. Therefore, to detect an object it must be rotationally symmetric and must be orientated in an expected direction.

To conclude the technique in its entirety will not suffice because the method uses assumptions that lack in robustness. It is not possible to assume that all the objects that HERO will encounter on a table sheet are cylindrical. However, these assumptions are only needed if an object requires an ID provided by the database. In the scope of the project, this is not a necessity, therefore the method can prove useful to only identify and cluster the individual objects.

2.2.2 You Only Look Once algorithm

The 'You Only Look Once' algorithm, better known as YOLO, is a real-time object detection algorithm. YOLO applies a trained neural network to a single full image. The network divides the image into a grid with multiple cells and predicts bounding boxes for each cell. These bounding boxes contain the center point of an object/person of interest, including a confidence score. The YOLO algorithm is well-known in the computer vision sector due to its exceptional speed, accuracy, and learning capabilities.

The neural network is trained by manually feeding it a large number of images where the bounding box has been provided. Where the bounding box is determined by a height, width, center, and object class.

2.3 Image processing

Once the point cloud data has been processed it can prove useful to visualize the data. To create an appropriate visualization and to further process the image certain techniques can be used. Creating the visualization will be discussed further in section 2.4.2, this subsection aims to solely explore image processing techniques [5].

2.3.1 Dilation

The first mathematical morphological operation that will be discussed is dilation. Mathematical morphology describes the interaction between an input image and a predefined structuring element (SE). While the input images can be of multiple types the operation assumes that there is a foreground and a background. Whereas in binary images the value of 1 represents the foreground and a value of 0 the background. Dilation is an operation that adds pixels to the boundary of objects in an image, in binary this operation changes 0's to 1's if certain requirements are met. The SE will pass over the pixels in the image and if there is an overlap between a target pixel and the SE kernel then the surrounding pixels will be altered. The most notable parameters that can be altered by the user are the number of iterations and the specifications of the SE. The amount of pixels and the shape in which they are altered is dependent on the SE. The size of the

SE is dependent on a kernel size. Furthermore, there are 3 shapes that can be used for the SE. Namely a rectangular shape, a cross shape, and an elliptic shape. Finally, an anchor position can be defined within the element, this position regulates how far the result of the operation is shifted.

2.3.2 Erosion

The second mathematical morphological operation that is investigated is erosion. This operation describes the opposite of the dilation operation. Where dilation will add pixels to an image, erosion will remove pixels instead. Similarly to dilation, erosion requires an input image and a structuring element (SE). The SE passes over the pixels in the image and follows two rules. If there is a full overlap between the structuring element and a target pixel this pixel will not be altered. Else, the pixel is removed in the output image. Parameters that alter this operation are identical to the parameters described in the dilation description.

2.3.3 Opening

This operation is a combination of both morphological operations. Erosion is applied to an image followed by dilation, this operation is most commonly used to remove noise from an image. Considering, once erosion is applied all shapes that are smaller than the structuring element will be removed. However, this also shrinks the main shapes in the image that should be preserved. For this reason, dilation is also applied, dilation will revert the shrinking without adding back noise.

2.3.4 Closing

Additionally, another combination of morphological operations can be made. Here, dilation is applied first followed by erosion. This operation is most often used to remove holes inside the main shapes of an image. These holes could for example be, reading errors in a data set. Once dilation is applied the holes are closed, but the size of the main shape is also increased. Therefore, erosion is followed afterward, similar to the 'Opening' operation. The morphological operation that follows aims to revert the changes made to the main shape which should be preserved from the original image.

2.4 Previous TechUnited research

Within the TechUnited team two previous studies have been completed that are similar to the project as described in section 1. In this sub-section, the information that is provided in these studies will be analyzed. Furthermore, a conclusion is drawn on whether the provided methods can be expanded upon. To do this, the used method is briefly mentioned, including the author's motivation for choosing certain techniques over others. The results of the method are analyzed to determine whether they are functional for the task at hand.

2.4.1 T.M.H. Beurskens' research

T.M.H. Beurskens' report presents a method to generate a table model using multiple viewpoints [4]. Beurskens makes use of HERO's RGBD camera to extract data and uses multiple images from differing viewpoints to create a point cloud. Furthermore, since Beurskens is not only interested in the surface of the table he also wishes to model the legs of the table. To do this, Beurskens segments the point clouds using a combination of Euclidean clustering and RANSAC plane fitting. The cloud is then flattened by removing the height coordinates. To continue, Beurskens generates a 2D concave hull around the cloud. RANSAC is used to fit primitives onto this hull, if the fit is not appropriate a mesh of the concave hull is utilized as a backup. Beurskens describes that the method may not be suitable for more cluttered household scenarios.

Unfortunately, Beurskens' method is not able to produce consistent fits for the table model. Beurskens concluded that this may be due to using multiple viewpoints in a singular point cloud or the used segmentation method. For this reason, Beurskens' method will not be continued with for the remainder of the project.

2.4.2 P. Hundepool's research

P. Hundepool's report presents a method for modeling the top of the table [3]. Similarly to the project at hand, Hundepool makes use of HERO to extract data and test the method. Hundepool captures a depth image of the table, this image is converted into a point cloud. The point cloud is segmented using parallel planar segmentation, using RANSAC as a robustness estimator. The cluster containing the table sheet is extracted manually. It is assumed that the table is horizontally level, therefore to further analyze this cluster Hundepool flattens this cluster on a colored 2D figure. Hundepool describes this 2D figure as a costmap, this figure prints green pixels corresponding to the X- and Y-values of the table-sheet cluster onto a blank canvas. As a result, there is a visual distinction between printed point cloud values and the background canvas. In Hundepool's literature research, it is described that plane model segmentation, section 2.1.1, is the most appropriate segmentation option. He chooses this segmentation method over methods such as euclidean clustering, section 2.1.2, due to the simplicity and robustness. Furthermore, since the project requires the table sheet to be segmented, a horizontal plane will be the resulting cluster. Therefore, Hundepool concludes that it is a logical choice to use a segmentation method that makes a fit based on an expected model.

Hundepool's method contains useful conclusions that can be continued with. For example, the segmentation method that is used in his research leads to satisfying results. Furthermore, the idea to plot the cluster data on a 2D image is a suitable option to further process the data, Figure 2. However, Hundepool was not yet able to create placement options due to a lack of data. Therefore, for the remainder of the project, this can be continued with.

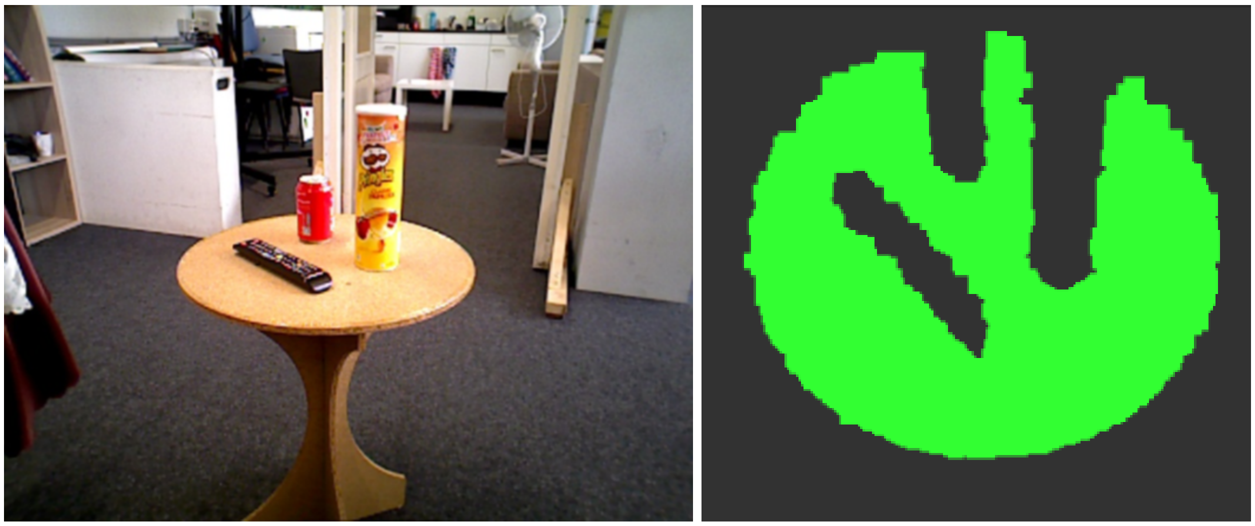


Figure 2: Result of P. Hundepool's method, where a costmap is made that represents a 2D image of the cluttered table-sheet

3 Approach

In this section, the developed method is presented including segments regarding the implementation. Firstly, the data extraction using the RGBD camera is described. Secondly, point cloud data processing is presented, which will go into further detail regarding the cluster segmentation method. Furthermore, additional data is processed to make the distinction between free, occupied, and unknown space. Thereafter image processing techniques are used to create the placement options. Finally, the criteria that determine which placement option is chosen will be described.

Point cloud data processing is realized with the Point Cloud Library (PCL) as a supporting tool. PCL has numerous applications for working with and manipulating point cloud data.

Furthermore, the Open Source Computer Vision Library (OpenCV) is used as a tool to process images. OpenCV is an open-source library containing computer vision and machine learning software.

3.1 Data extraction

For data extraction, the RGBD camera is used, as mentioned the output images of this camera contain depth information, as seen in Figure 3.

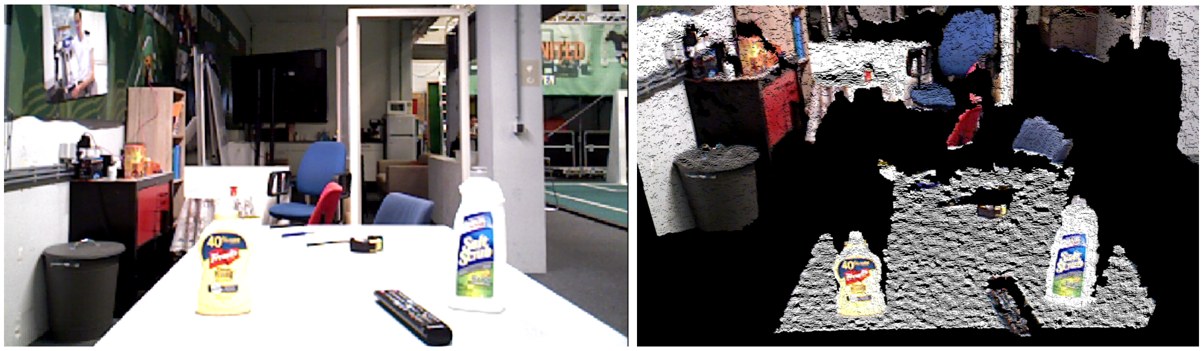


Figure 3: Common RGB image compared to an image including additional depth information, where the black sections in the RGBD image represent space that is not visible from HERO's point of view

The depth information is used to generate point cloud data. This process requires some small steps which translate a pixel location on the lens to a point in space. The RGBD camera returns the distance from each pixel to a plane normal to the camera. Using the known focal distance of the camera, each pixel in the depth image is then translated to a 3D point including a value for each of the 3 color channels. It is possible that for some pixels the depth value is missing, this will result in gaps in the point cloud. This problem occurs when the infrared receiver used for the depth information does not receive the outgoing laser due to large reflections on top of the measured surface. Another possibility is that the laser is absorbed by the material itself. Limiting or preventing these losses of data will be described in section 4. Finally, the point cloud data is saved to a new cloud, this cloud will be referred to as the main cloud because it still contains all the data from its surroundings.

3.2 Segmentation

The segmentation method that is applied is plane segmentation, this choice is motivated by the method's simplicity and adequate results, as seen in P. Hundepool's research [3]. However, before segmentation is applied, the floor is removed from the main cloud using a filter, and the filtered data is added to a new floorless cloud. The conditional removal filter removes the points that have a height value below 2 cm. This is done to limit the number of planar clusters that can be created with the segmentation method.

The specific type of planar segmentation is parallel planar segmentation, using RANSAC as the robust estimator. After segmentation there might be multiple planar components, however, it is assumed that HERO is already positioned in front of the table. As a result, the planar cluster of the table will be the largest, HERO automatically picks the largest cluster based on the number of indices.

Once segmentation is performed a point cloud containing only the table sheet remains. Furthermore, information regarding the properties of this plane is extracted, where one of the important properties is the height of the plane. This table sheet cluster will act as the foundation of the costmap on which more information will be printed, Figure 2. Up until this point P. Hundepool's research is used to act as a foundation for the devised method. However as mentioned in section 2.4.2, placement solutions cannot be made yet due to a lack of data. Therefore the remainder of the section describes how additional information is gathered to allow for the generation of placement solutions.

3.3 Adding additional point cloud data

Besides the table cluster, HERO requires more information before it can determine a valid placement solution. First, the point cloud data of the objects on top of the table is defined. Continuing, the occluded space behind the objects is added based on the object information. Finally, the space that is no longer part of the table sheet is included to represent the edges of the table. Afterward, techniques that no longer required point cloud information will be utilized to complete the costmap.

3.3.1 Objects on table

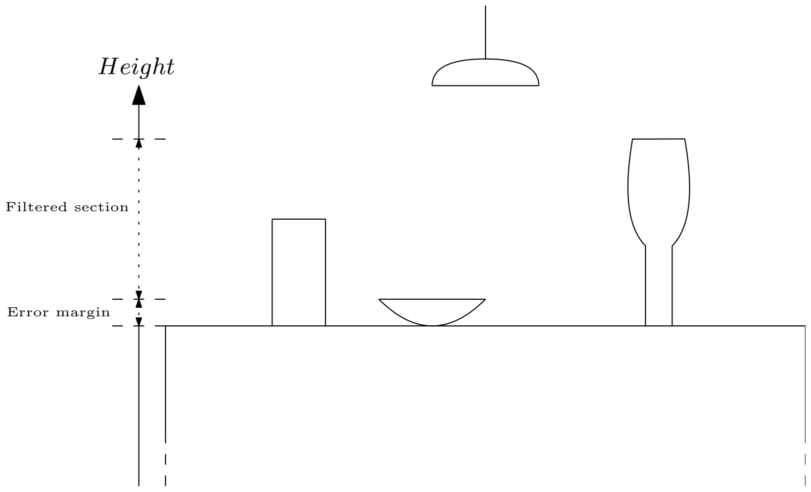


Figure 4: Schematic overview off the object filter

To filter out the point cloud data that represents the objects on the table, a data filter is used. A schematic overview can be seen in Figure 4. The previously extracted table-sheet plane height, summed with an error margin of 5 cm, acts as the minimum required height value for the filtered points. The maximum value for the height is taken as 30 cm, therefore the filter determines that the points that fall within this range are left in the input cloud. The main cloud is used as the input because it still contains all the available data. The remaining points, after filtering, are stored separately in a cloud that will be referred to as the object cloud.

An error margin of 5 cm is chosen after performing tests on table sheets from differing distances, this process is described in section 4. From this, it could be concluded that lower error margins will work if the table sheet is within 50-60 cm of HERO. However, measuring errors appear more frequently beyond this range, resulting in falsely detected objects. A maximum value of 30 cm is determined based on the expected maximum size of an object on a standard household table. A maximum value beyond 30 cm may result in HERO detecting overhanging lamps as objects, and values below 30 cm may result in HERO bumping into an object it did not detect. For example, a flower vase whose diameter has increased between 20-30 cm from the table.

3.3.2 Occluded space

To determine the occluded space a more mathematical approach is used. The method utilizes a vector direction between a point in the object cloud and the position of the camera. This direction vector in combination with the known height, z-value, of the table the x- and y-values of the occluded space can be computed. A graphical representation of the problem at hand is seen in Figure 5.

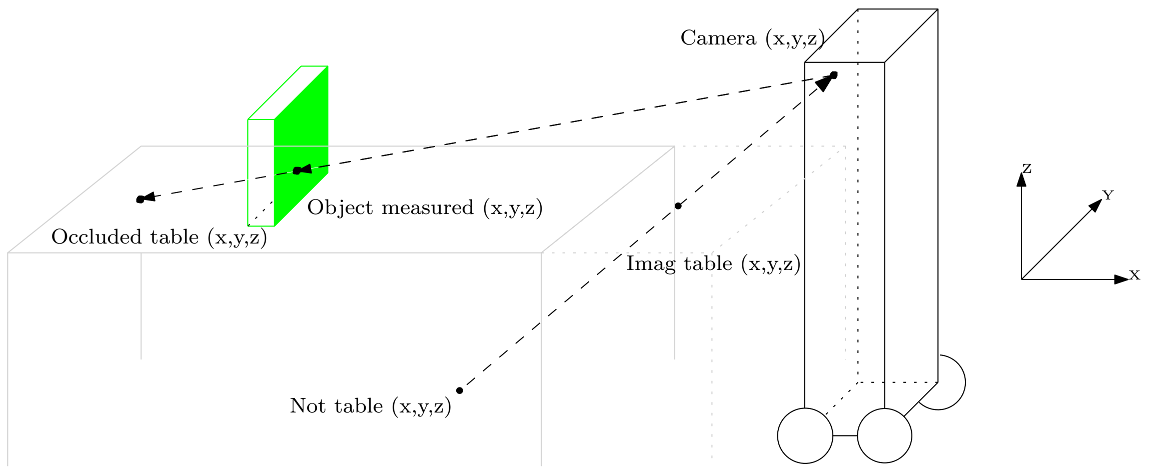


Figure 5: Graphical representation of the occluded space and not table problems

Using the direction vector in combination with the known x-, y- and z-values of a point on an object the following parametric equation can be constructed. Which is solvable with the known value for z-table.

The direction vector for the line that crosses an object is given by the following equation.

$$\overrightarrow{d} = \begin{pmatrix} x_{object} - x_{camera} \\ y_{object} - y_{camera} \\ z_{object} - z_{camera} \end{pmatrix} \tag{1}$$

Combining with the known points of the object:

$$\begin{pmatrix} x_{object} \\ y_{object} \\ z_{object} \end{pmatrix} + \lambda \cdot \overrightarrow{d} = \begin{pmatrix} x_{table} \\ y_{table} \\ z_{table} \end{pmatrix} \tag{2}$$

Solving with z-table and rewriting to define lambda:

$$\lambda = \frac{z_{table} - z_{object}}{z_{object} - z_{camera}} \tag{3}$$

Combining the equations allows the computation of a cloud consisting of points that represent the occluded space.

3.3.3 Not-table

A similar approach can be considered to construct the not table cloud, this problem may also be seen Figure 5. The not table points are filtered out by removing all the points that are equal or larger than the plane height minus the same safety margin as considered in Figure 4.

The direction vector for the line that goes from the Not table points to the camera

$$\overrightarrow{d} = \begin{pmatrix} x_{camera} - x_{notTable} \\ y_{camera} - y_{notTable} \\ z_{camera} - z_{notTable} \end{pmatrix} \tag{4}$$

Using the direction vector in combination with the known x,y, and z of the points that lie below the following parametric equation can be constructed.

$$\begin{pmatrix} x_{notTable} \\ y_{notTable} \\ z_{notTable} \end{pmatrix} + \lambda \cdot \overrightarrow{d} = \begin{pmatrix} x_{ImagTable} \\ y_{ImagTable} \\ z_{ImagTable} \end{pmatrix} \tag{5}$$

Which is solvable with the known value for z_{table}. Continuing, λ can be defined as.

$$\lambda = \frac{z_{ImagTable} - z_{notTable}}{z_{camera} - z_{notTable}} \tag{6}$$

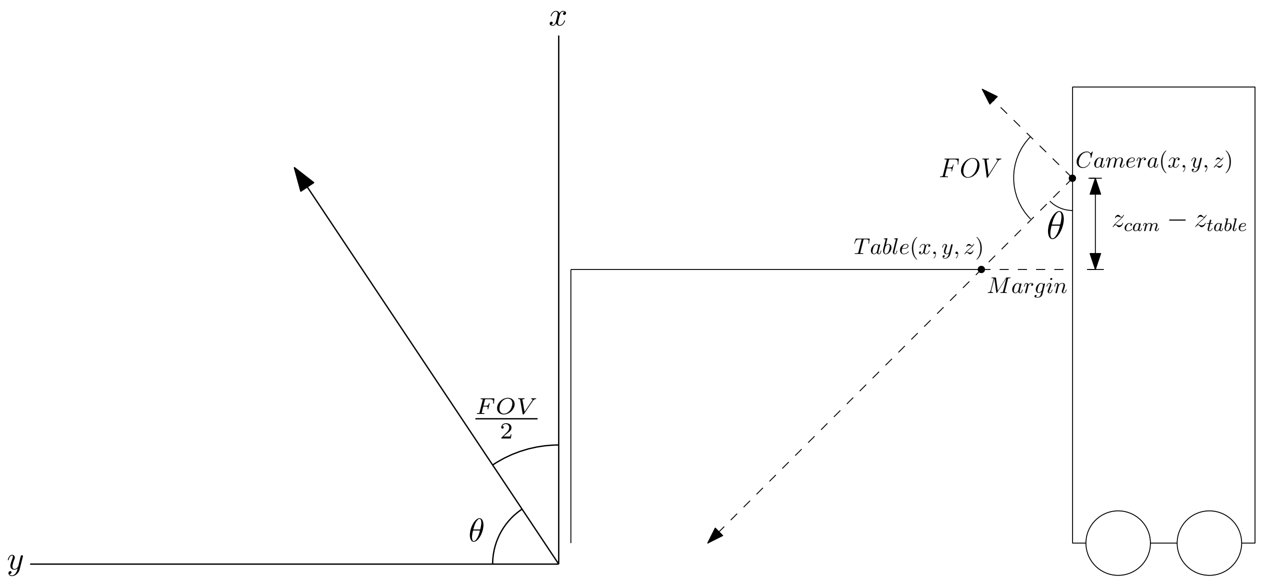
Combining Equations 4,5 and 6 the not-table cloud can be defined.

3.4 Image processing

It is possible that HERO can not see the complete table sheet. This may cause problems at the edges of HERO's image. Furthermore, a safety margin that HERO is required to consider when placing an object must be added.

3.4.1 Camera field of view

For the first problem, the field of view (FOV) of HERO's camera can be added to the bitmap. In this way, HERO is made aware that it cannot place objects at the edges of the image.



(a) Representation of the horizontal FOV component (b) Representation of the vertical FOV component

Figure 6: Adding the camera field of view to set a placement boundary

For the horizontal component of the FOV cone, the relationship between the change for the x- and y-values is used, as represented in Figure 6a. This relationship is the tangent of the angle θ. Where θ is equal to 180 minus the FOV divided by 2. This is iterated over a large number of steps summed with the value in the previous step. Where the first step is the initial position of the camera.

$$\begin{aligned} x(i) &= x(i-1) + n \cdot tan(\theta) \\ y(i) &= y(i-1) \pm n \end{aligned} \tag{7}$$

The vertical aspect of the field of view of HERO's camera can be added by using the camera FOV in combination with the difference in height between the camera and the table, as seen in Figure 6b. This results in a constant x-value, this value is then plotted across all canvas columns.

$$x = x_{camera} + (z_{camera} - z_{table}) \cdot tan(\phi) \tag{8}$$

Where angle ϕ is equal to 180 minus the FOV divided by 2.

3.4.2 Dilation

Once the required information is added to the costmap, the object margin plus an additional error margin have to be considered. When placing an object it is required that the placed object or HERO's gripper does not make contact with any objects on top of the table. It is assumed that the object to be placed is a cylinder, therefore the radius of the placed object determines the distance that needs to be taken into account when placing the object. The reason this assumption is made is to simplify the problem, if the object to be placed where, for example, rectangular. Consequently, the orientation of the placed object in HERO's gripper will determine the required margin. For a cylindrical object, the orientation will not determine the margin, because it remains a constant radius. To continue, there is a difference between the chosen placement location and the location where HERO will place the object. This error has to be accounted for in the margin as well.

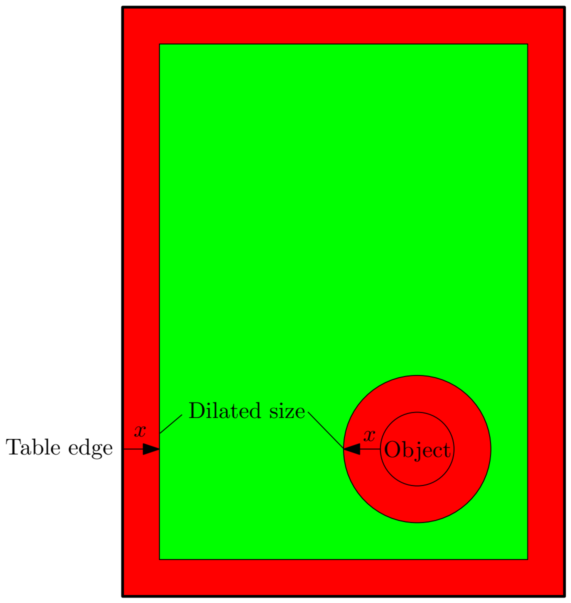


Figure 7: Increased pixel size due to dilation, used to add a safety margin [x]

The image processing technique, dilation, is used to add the margin. Dilation increases the pixel size of the points on the canvas, a schematic representation is given in Figure 7. The increased pixel size and shape are specified by a structuring element. For the required purpose, a circular structuring element shape is chosen with a radius specified by the margin. A circular element is chosen with the motivation that the same margin must be considered in all directions around occupied or unknown space. Therefore another element shape such as a rectangle will not suffice, because this element shape will either leave too much or too little margin in some directions.

3.5 Placement map

Currently, all remaining free space on the canvas is considered a valid placement option. However, HERO is not able to place an object, from its current position, on the other end of the table. Therefore a preferred placement distance is added. After this is specified, the remaining placement options from the canvas are filtered and a point is chosen as a placement solution.

3.5.1 HERO's placement distance

To specify the placement distance, the maximum and minimum distance away from HERO, where it can place an object, is considered. This distance can be seen as a radius because HERO can rotate around its axis to place an object, as seen in the schematic overview in Figure 8. The maximum radius for a placement point is 75 cm away and the minimum placement radius is chosen to be 60 cm. The motivation for these radii comes from asking a member of the TechUnited team that works with HERO.

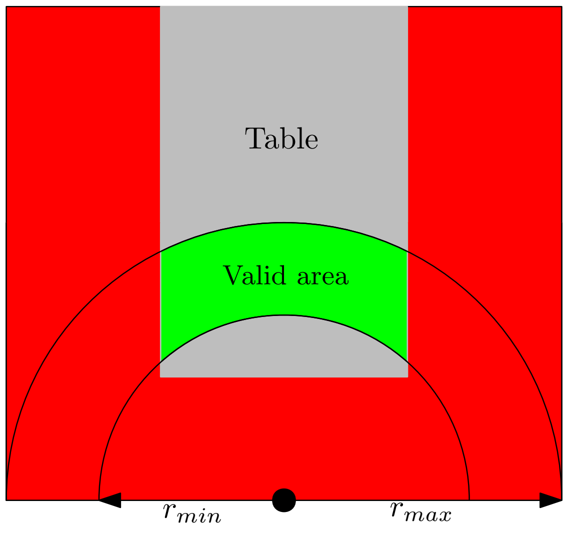


Figure 8: Schematic overview of HERO's preferred placement radius

The preferred placement distance is defined using the properties of the unit circle. The angle, θ, is iterated from 0 to 2π. And the step size, n, is increased to higher values to increase the radius. This is done to remove the points on the costmap that are further away than 75 cm from HERO.

$$\begin{aligned} y &= r_{max} \cdot sin(\theta) + n \cdot sin(\theta) \\ x &= r_{max} \cdot cos(\theta) + n \cdot cos(\theta) \end{aligned} \tag{9}$$

Similarly, the minimum radius is implemented, but here the radius is lowered for each step size, n. As a result, points that are closer than 60 cm away are also removed.

$$\begin{aligned} y &= r_{min} \cdot sin(\theta) - n \cdot sin(\theta) \\ x &= r_{min} \cdot cos(\theta) - n \cdot cos(\theta) \end{aligned} \tag{10}$$

3.5.2 Filtering the placement points

To continue, the remaining placement points are filtered. This is done by setting a condition for the values within the color channels of every pixel on the canvas. For every pixel on the canvas, the value for the RGB channels must be equal to the RGB value of green. If the condition is met the pixel is stored in a vector, and simultaneously printed on a new canvas for visualization purposes.

3.5.3 Determining a placement solution

Finally, once a vector with placement points is made a final solution can be chosen. This is done by choosing the first value in the vector. This is possible, due to the order in which the canvas points are filtered. The filter will first check every column value for a given row, starting from row 0. Which is at the top of the canvas. Therefore, the filter will work downwards, thus the first value that is filtered is either directly in front of HERO at the maximum radius or as close to the middle as possible.

The point that is chosen is considered the placement solution and the X- and Y-coordinates of this point is returned to the user. Furthermore, the point is printed on the placement canvas using a blue colored pixel for visualization purposes. Finally, the Z-coordinate representing the height is added, and a small margin is summed to this value to assure that HERO does not attempt to place the object inside the table.

4 Testing

In this section, the devised method will be tested in order to gain more understanding regarding the implementation and the method's limitations. First, the method is tested in the open-source simulator, gazebo, which is often used by the TechUnited team to initially test new implementations. To continue, the method is tested in real scenarios to find its limitations and problems.

4.1 Simulation testing

The method is tested by following the same steps as described in section 3. First, the data extraction is tested along with the chosen segmentation method. Continuing, the costmap is created and updated with each step to visualize the data processing techniques. Afterward, the costmap is directly altered by using image processing techniques. Finally, the free space is filtered and a point in space is chosen as a placement solution.

4.1.1 Data extraction

The data is extracted from the simulation, and an overview of the situation can be seen in Figure 9. As seen, HERO is located in front of the target table, on this table, a can of soda is resting.

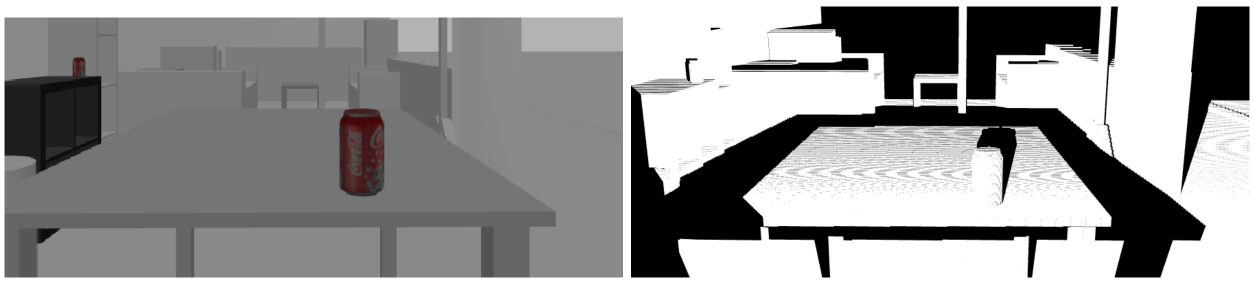


Figure 9: The situation as seen in the RGB image and the depth information that is extracted on the right

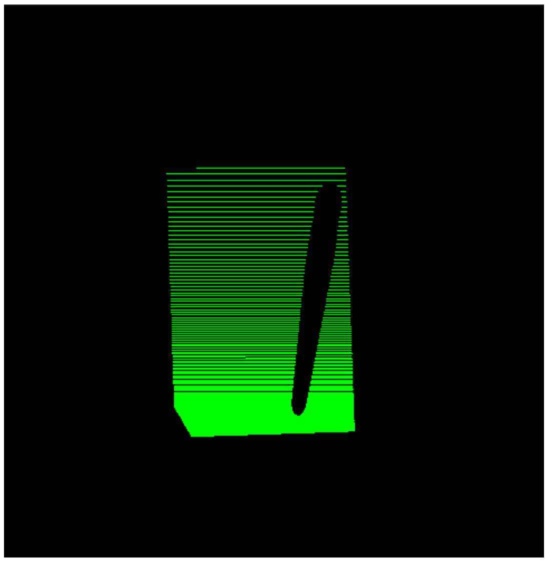
Comparing the simulation data to a real-world scenario such as in Figure 3 it can be concluded that the simulation data is unrealistic. In the sense that the depth information is almost complete and no reading errors are present. This should be kept in mind when analyzing the results of the method when applied to the simulation data.

4.1.2 Segmentation

To continue, parallel planar segmentation is applied and the height of the table is retrieved. Once segmentation is applied, and the correct cluster is selected, the first layer can be added to the costmap. This layer can be analyzed to see whether segmentation was applied properly. As seen in Figure 10a, the cluster of the table sheet is successfully segmented. However, there are some black horizontal lines that become more frequent further away from HERO. These lines are gaps of missing data, that become more frequent when the camera angle on the sheet becomes sharper. Furthermore, the height of the plane is retrieved, in this case, the height of the table sheet is at a height of 77 cm in reference to the ground.

4.1.3 Adding the objects

To be able to identify which space is occupied on the table sheet, the objects will be added to the costmap. To do this a filter is applied on the main cloud, 3.3.1, and the filtered points are stored in a separate cloud. Flattening this cloud, by removing the height values, and plotting the remaining values visualizes the largest diameter of the object on the costmap. The result can be seen in Figure 10b, where the red circle represents the soda can that can be seen in Figure 9.


(a) The segmented table-sheet cluster is flattened on the costmap with green pixels

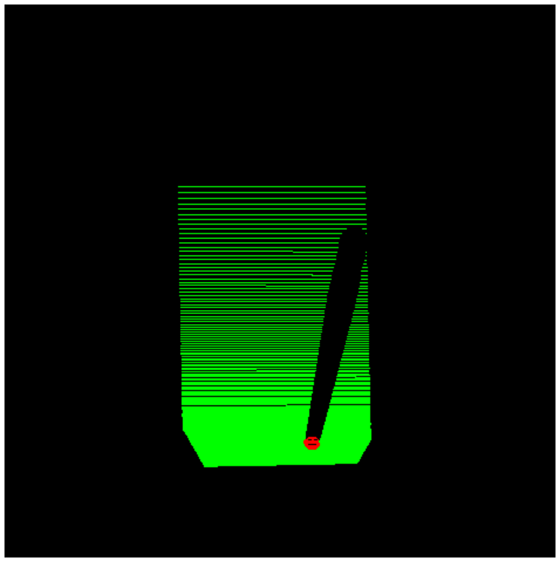
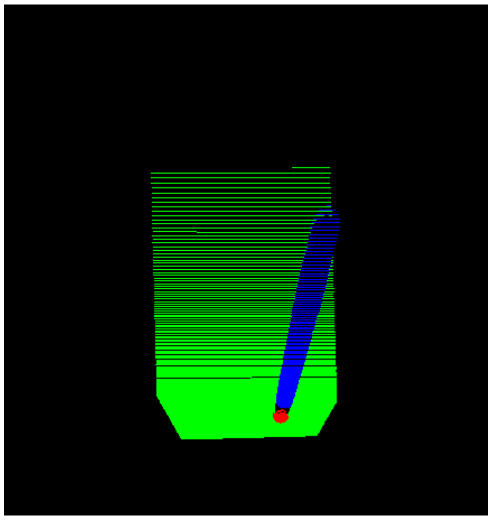

(b) The can of soda is detected and added to the costmap with red pixels

Figure 10: Creating a blank costmap and continuously adding more information

4.1.4 Including occluded space

Using the previously defined object cloud, the unknown space behind the objects is defined. This is accomplished by solving the equations defined in section 3.3.2. Because the simulation data is close to being perfect, it is possible to lower the error margin that is used when filtering the object cloud. While this may not have any visual effect on the object cloud, it greatly influences the occluded space. The error margin used in the filter is required to cut off false height readings of the table plane. However, this margin will also remove the bottom sections of the object, and if these sections are removed it will leave a gap between the occluded space and the objects. This effect can be seen in Figure 11a and Figure 11b, where an error margin of 2 cm is compared to 5 cm.


(a) Occluded space with an error margin of 2 cm

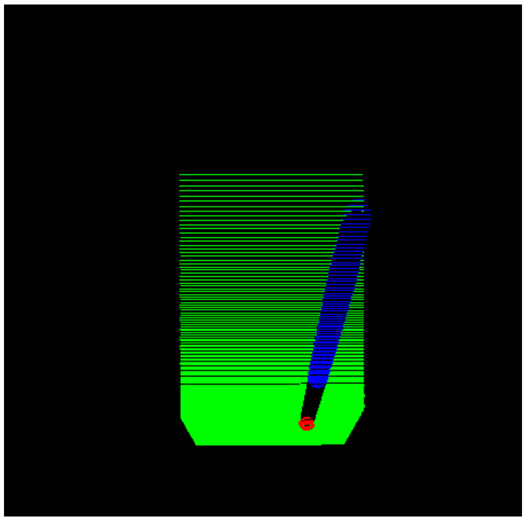
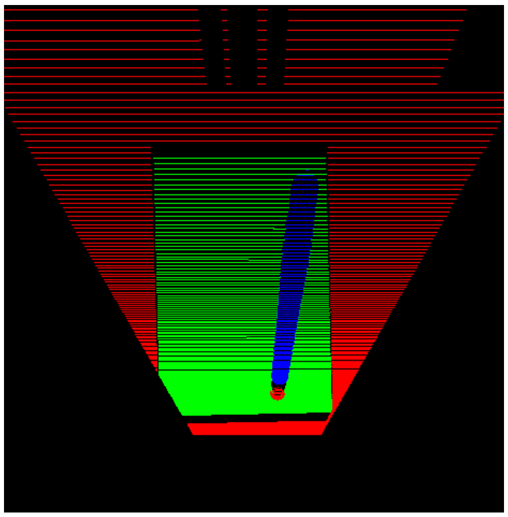

(b) Occluded space with an error margin of 5 cm

Figure 11: Adding the occluded space, in blue, to the costmap and analyzing the influence of the error margin used in the object filter

4.1.5 Adding not table space

To continue, similarly to the occluded space, by solving a set of equations as defined in section 3.3.3, it is possible to add the not table space to find the edges of the table. Additionally, to define the points that are not-table, a filter is used to remove the points that are equal to or larger than the height of the table plane minus another safety margin. As described in the previous section the value for the error margin can be lowered in the simulation. The effect of different size margins can be viewed in Figure 12a and Figure 12b. As seen this change only affects the data near the front edge of the table, this is expected. Because the filter only removes values that are just below the table sheet. On the sides and the back, this will remove no additional points, however, a difference can be expected from the front. This difference is due to the thickness of the table sheet, this can only be viewed from one perspective, as seen in Figure 9. Furthermore, depending on the type of table the entirety of the supporting legs can be seen. And if the error margin is large enough a section of the legs will be removed from the not table space. This explains the missing triangular chunk in Figure 12b. However, the difference between the error margins does not cause a significant difference in results, therefore there should be no issues when using a larger error margin.


(a) Not table space with an error margin of 2 cm

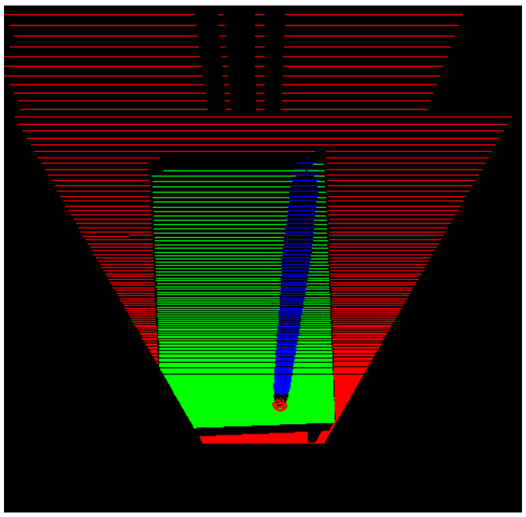

(b) Not table space with an error margin of 5 cm

Figure 12: Adding the not table space, in red, to the costmap and analyzing the influence of the error margin used to define the not table points

4.1.6 Image processing: adding the field of view

Once the point cloud data has been processed, the resulting costmap can be detailed further using image processing techniques. As seen, once the not table costmap is added it is possible to differentiate the edges of the table sheet almost entirely. However as mentioned, it is possible that HERO is not able to see the entire table sheet. As a result, there will be no not table space to differentiate the edge of the table. For this reason, the camera field of view (FOV) is added to the costmap using the implementation as seen in section 3.4.1. The resulting costmap is compared to the frame that was used to extract the data. As seen in Figure 13 the implemented field of view matches perfectly with the not table cloud as required.

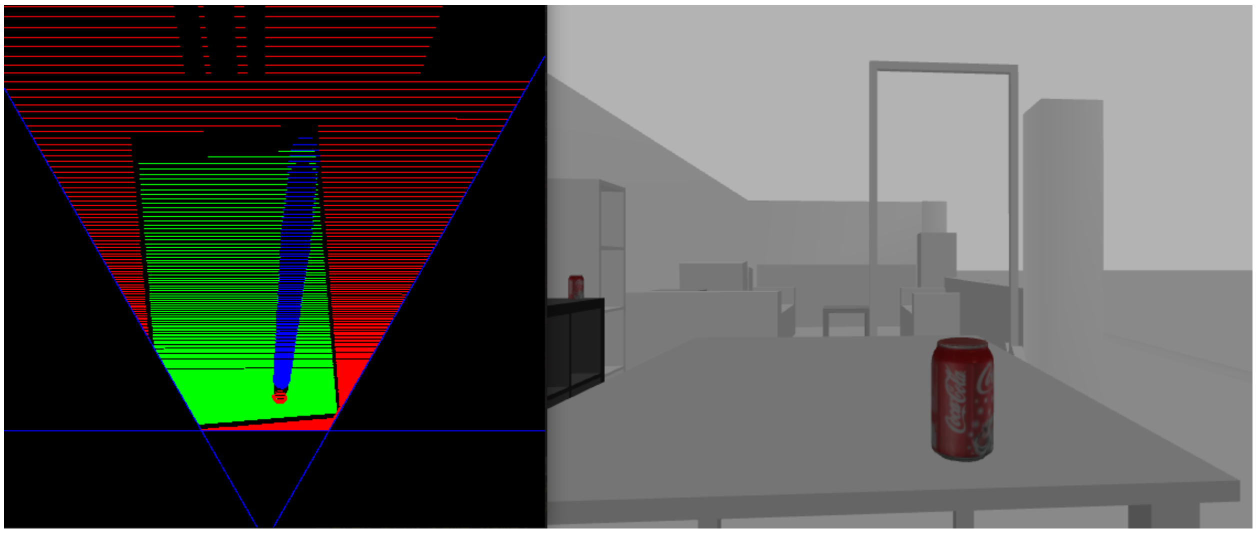


Figure 13: Costmap including the FOV, in blue, compared to the frame that was used to extract the data

4.1.7 Image processing: dilating the costmap

Finally, once all the required information is successfully added to the costmap, a collision margin must be considered when placing an object. This margin is added through the use of the morphological operation, dilation. Dilation increased the size of the colored pixels on the costmap, in this way the green section that is currently considered as green space will shrink. For this test the object that will be placed has a diameter of 10 cm, the placement error is equal to 2 cm, and the thickness of HERO's gripper is around 2 times 2 cm. Therefore the diameter of the structuring element is equal to 16 cm. The result of the morphological operation can be seen in Figure 14.

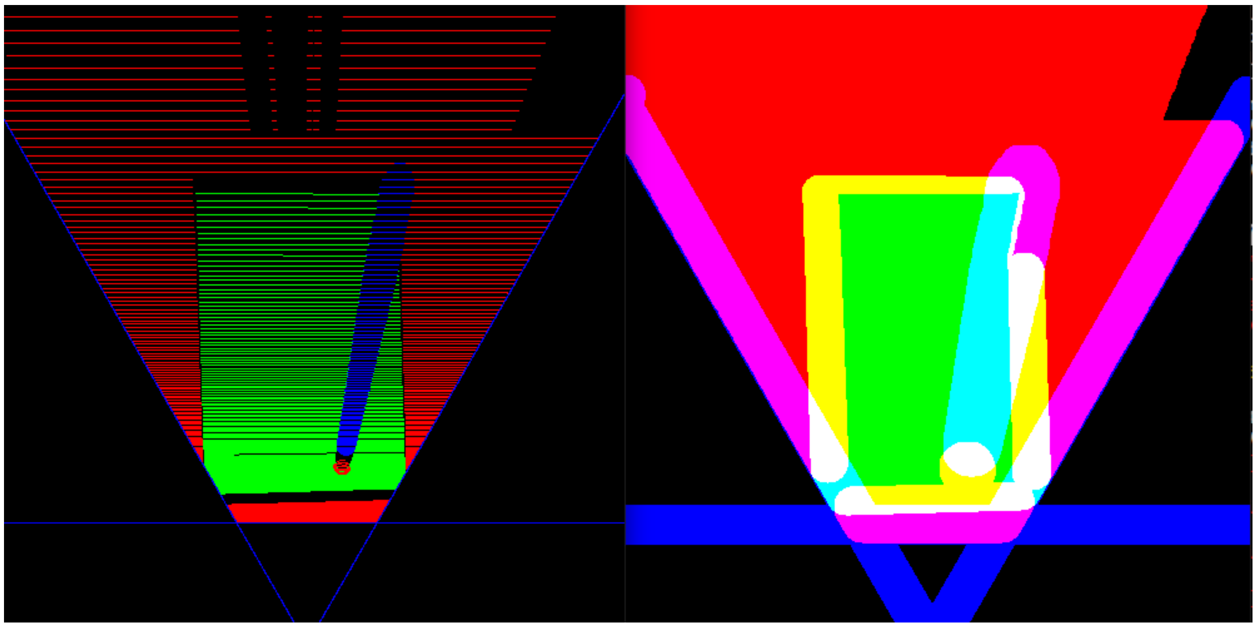


Figure 14: Side-by-side view of the dilated and non-dilated costmap

As seen a strange phenomenon occurred during dilation, the RGB values for the dilated pixels have been combined. For example, the yellow edges are the combination of the red, not table, data and the green, table cluster. There are even combinations where 3 colors have merged, this occurs between the red objects, the green table, and the blue occluded space. This results in a white color, RGB values [255 255 255]. It is not a desired effect, nonetheless, it has no impact on the functionality of the morphological operation. Because as seen, the pixels have successfully grown and the green table sheet has shrunk as a result. Furthermore, the horizontal lines that were lacking in data have been weaved closed.

4.1.8 Creating the placement costmap

To continue, HERO's preferred placement radius is added using the equations as seen in 3.5.1. The 'Valid area' as seen in Figure 8 will remain unaltered. Points that fall outside the maximum radius or inside the minimum radius are altered in color. As a result, only the valid placement options remain perfectly green, RGB [0 255 0], this can be seen in Figure 15. Furthermore, the valid placement options are filtered based on the perfectly green RGB value and are plotted on the new placement option costmap.

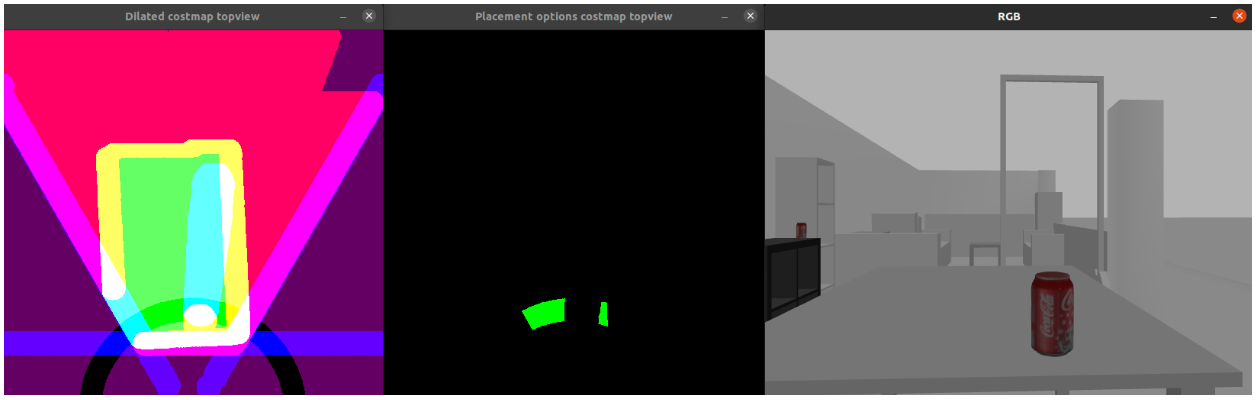


Figure 15: Final costmap on left with filtered points in the middle and the simulation view on the right

Finally, the placement solution is chosen based on the filtering process as mentioned in section 3.5.3. The blue pixel is plotted on the placement canvas and its coordinates are returned. The point that was chosen by the method has the coordinates (X, Y, Z), (0.745, 0.08, 0.79). Which is a little less than 75 cm away from HERO's base frame. Furthermore, the output coordinates are also in reference to HERO's base.

To conclude the simulation experiments, the method delivers good results for a perfect data set. However, during the real scenario tests, it will become clear whether the method is robust enough to deliver acceptable results for realistic data sets.

4.2 Real scenario testing

To verify whether the method delivers acceptable results, HERO will be presented with a cluttered table. At which it will attempt to create a placement solution. HERO is presented with the same table from different camera angles to test the impact on the results. The aspects that are expected to differ in the real tests compared to the simulation are the steps that require the use of the point cloud data, such as segmentation and adding additional information to the costmap. The result of the image processing techniques will largely remain the same because they do not directly use point cloud data.

4.2.1 Scenario 1: Default view angle and height

For the first scenario, HERO's camera will be positioned in the default setup, meaning it is positioned 1 meter off the ground and is looking parallel to the table. This camera position is similar to the one used for the simulation testing. The results from scenario 1 can be seen in Figure 16 and Figure 17. As seen, HERO was not able to fully segment the table sheet from the extracted data. Objects such as cans and fruits are detected minimally resulting in the gaps between the red objects and the blue occluded space. The not-table information looks to be added successfully, as seen in the large red surface in the top right section of the costmap. However, a problem can be seen with the implementation of the field of view, it is assumed that

HERO's camera is pointed perfectly parallel with respect to the table, this is not the case and results in an incorrect representation of the boundaries of HERO's view. The difference between the defined field of view and the actual field of view is visible when looking at the edge of the not table section in the top right of the costmap. Where the error is the small gap between the blue FOV line and the red, not table space.

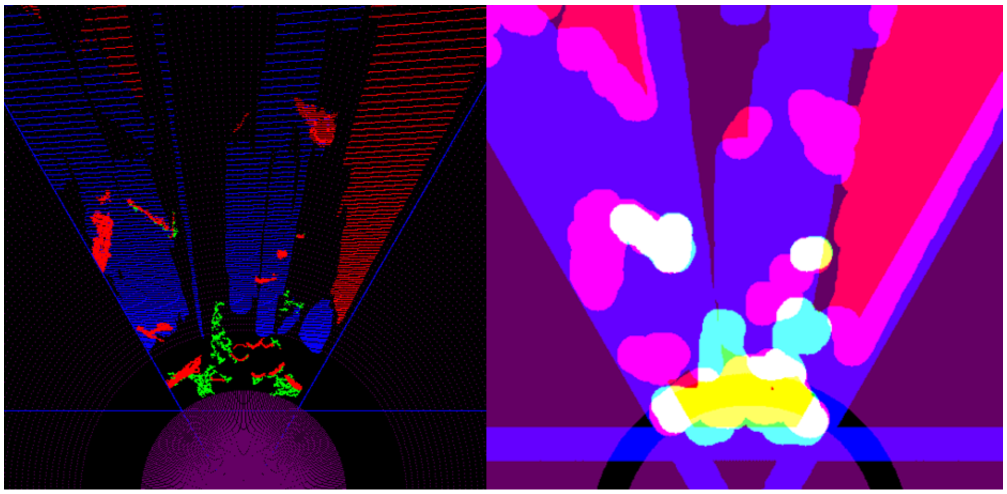


Figure 16: Final costmap for scenario 1 and its dilated version

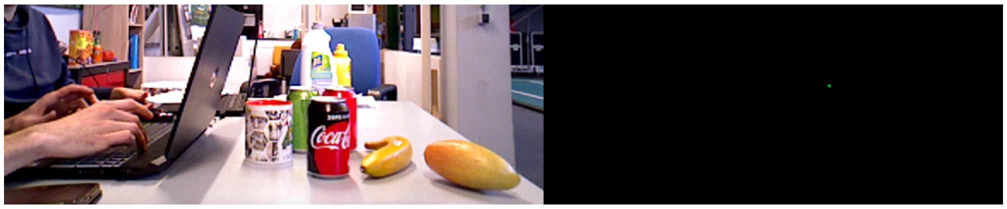


Figure 17: RGB image showing the point of view and the resulting placement options for scenario 1

To investigate the cause for the poorly segmented table sheet, the depth information from the extracted data is investigated. In Figure 18 it can be seen that HERO is not able to obtain enough depth information from the table itself. As a result, large sections of the plane are missing after segmentation, if a planar model is even recognized. This lack of data is most likely caused due to IR receiver not receiving the transmitted laser due to a large number of reflections.

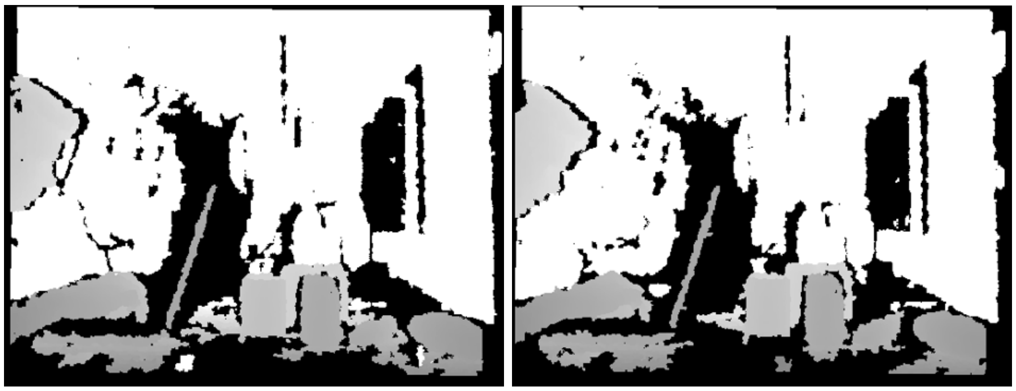


Figure 18: Extracted depth information displayed for 2 images, black refers to unsuccessfully measurements, grey/white sections contain depth information where the intensity is determined by the range

4.2.2 Scenario 2: Camera pointed towards the table at the same height

In the second scenario, HERO's camera will be pitched forward to look directly at the table. As a result, HERO should be able to obtain more information, because the angle between the transmitted laser and the table sheet is reduced. Which should reduce the number of reflections. The result of the change in camera orientation can be seen in Figure 19 and Figure 20.

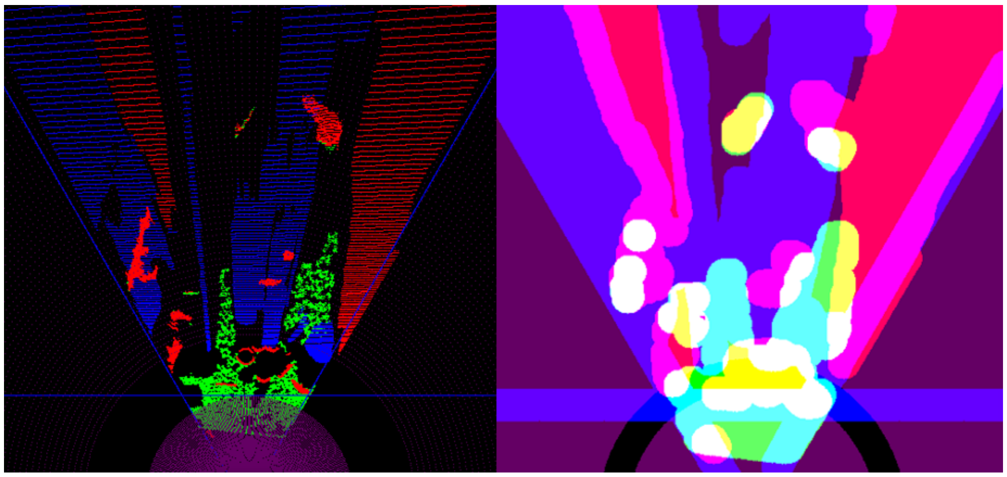


Figure 19: Final costmap for scenario 2 and its dilated version

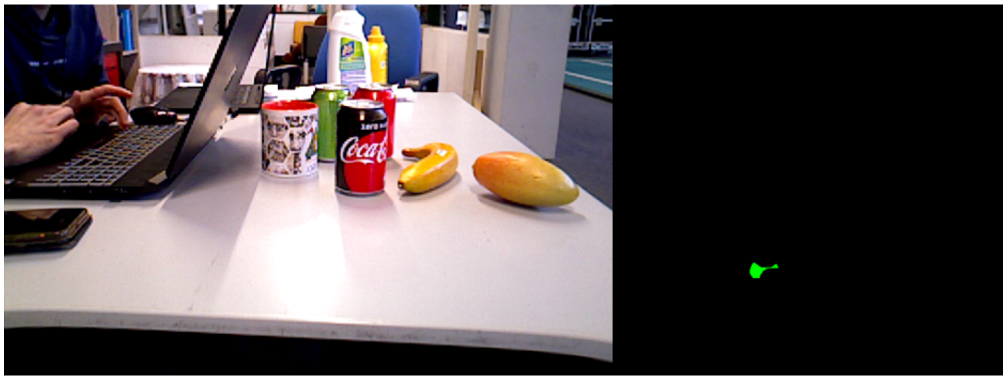


Figure 20: RGB image showing the point of view and the resulting placement options for scenario 2

As expected, this change has a positive effect on the amount of depth information that is registered. The table sheet is segmented properly within a range of 1 meter of HERO, with little to no missing information within 60 cm. To continue, it is noticeable that HERO is not able to detect the laptop's keyboard and the phone. It is to be expected that HERO can not filter these items as objects, because they fall within the error margin as shown in Figure 4. However, as seen in the non-dilated costmap, the laptop screen becomes visible once it rises above the margin. Continuing, the vertical FOV component, Figure 6b, does not take into account the pitching motion of the camera. As a result, the method assumes that HERO is looking parallel to the table and incorrectly defines the edge of the view. Disregarding this mistake it can be seen that HERO is able to create placement options, Figure 20. The method determines that HERO can place an object between the laptop and the mug next to it. However, looking at the RGB image it can be seen that this will most likely not be possible. There are better placement options closer to HERO but due to the range constraint, these points are not filtered as placement options.

4.2.3 Scenario 3: Camera pointed towards the table from an increased height

Finally, for the third scenario, HERO's camera is pitched toward the table as seen in scenario 2. However, now HERO's neck is extended by an additional 20 cm to position the camera higher up. This is done to experiment whether the depth information can be improved further. If this is the case then it might be possible to lower the error margin of the object cloud, which will result in better object detection.

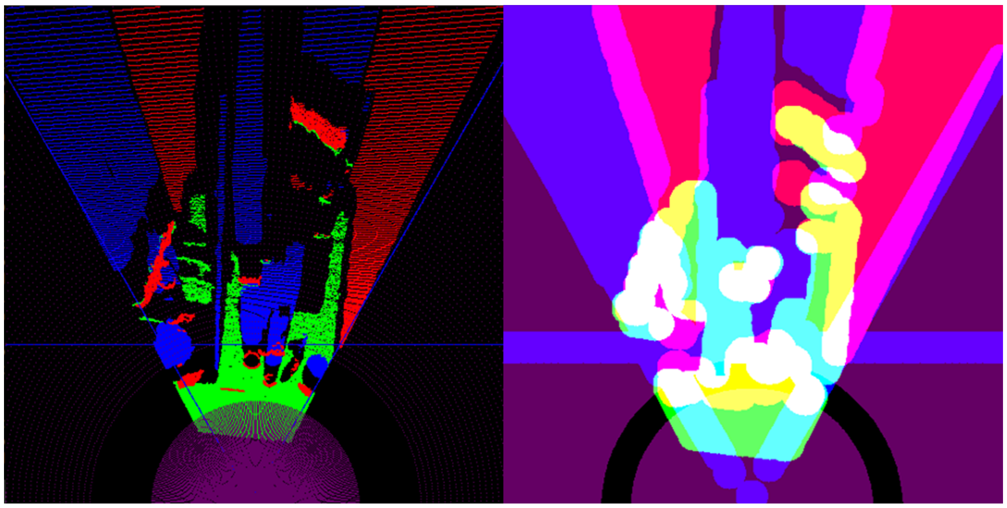


Figure 21: Final costmap for scenario 3 and its dilated version

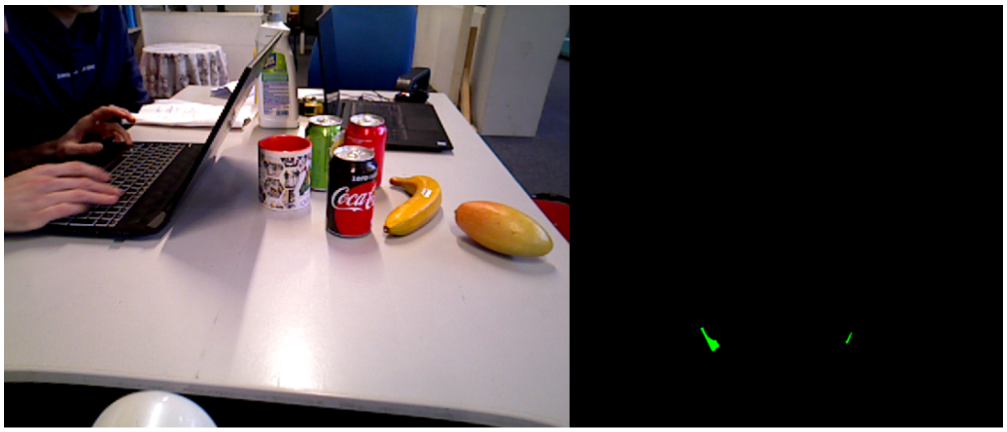


Figure 22: RGB image showing the point of view and the resulting placement options for scenario 3

As seen in the resulting costmap, there is more depth data available therefore the segmented plane contains fewer holes as compared to the bitmap in Figure 19. Note that the FOV components are still not implemented correctly, the cause of these errors has been presented in the previous scenarios. Furthermore, thin objects such as the laptop, or the banana, are still not properly filtered due to the margin. However, larger objects are now filtered much better. HERO is now able to see the top of objects, as a result, more information can be acquired which leads to a more detailed object cloud. Subsequently, the occluded space that is based on the object cloud is also more precise. Finally, because HERO is standing close to the table it does not filter out the large section of free space towards the front edge of the table. But instead, due to the incorrect implementation of the FOV, it determines valid placement options near the edges of the table.

5 Discussion

In the following section, a reflection regarding the quality of the presented method is presented. Furthermore, recommendations are given concerning improving the implementation of the devised method. Finally, suggestions are proposed to expand the method further.

5.1 Reflection

From the start of the project, the end goal was clear, namely to add enough information to the costmap to allow for a viable placement solution. Fortunately, P. Hundepool provided a solid foundation for the method, which allowed for a clear overview of an expected result. However, it was not always clear how new information should be added, it was difficult to find literature that supported ideas on how to implement certain steps. For this reason, some sections of the implementation took longer than initially expected. As a result, not enough time was spent testing the method in real scenarios. This could have been avoided by prioritizing the more important sections of the method's implementation.

The following sub-sections will go into further detail regarding the quality of the results as seen in section 4.2. Where the results of each step of the method are analyzed. Furthermore, if applicable, solutions to implementation faults are presented and suggestions regarding future work are discussed.

5.2 Data extraction

The amount of data that is extracted is largely dependent on the position of HERO's camera. In scenario 1 it can be concluded that not enough depth information is obtained to properly apply the method. However, this was greatly improved once the camera was tilted as seen in scenario 2. But, the greatest results could be seen in scenario 3, where HERO's neck is extended to position the camera higher above the table. By default, HERO is positioned in the same manner as seen in scenario 1, and manual input is required for HERO to extend his neck and directly look at the table. Therefore, future work could include the implementation of automating this process. Where potentially, with an implementation of the YOLO algorithm, the size and position of the bounding box can determine in which direction HERO should be looking and whether his neck should be extended.

Due to time constraints at the closing stages of the problem, it is not yet clear whether enough data can be generated from a single viewpoint. However, if more depth information is required it might be an option to use multiple viewpoints and combine their information.

5.3 Segmentation

Throughout the project, only parallel planar segmentation has been used as a result of the conclusions as seen in Hundepool's research. Therefore, it is unclear whether other segmentation may give better results. However, for the goal of the project, the segmentation delivers satisfying results. Initially, it was feared that the segmentation method would merge the table-sheet cluster with other parallel clusters located on the table, such as laptops or phones. However, in scenarios 2 and 3 it can be concluded that this is not the case. Where it does not include the laptops and the phone in the table cluster.

5.4 Adding objects

Defining the objects is an important step to define where HERO may not place an object. Using a filter with a margin of 5 cm to the table it can be seen that a lot of objects are filtered, but not all. Given a more ideal data set, the margin can be lowered considerably, as a result, thinner objects can be filtered. In scenario 3, the objects were defined using the same margin of 5 cm. However, considering that the depth information is much more complete than with initial tests, the error margin can be lowered to 3-4 cm and possibly even below that. A suggestion on how to calculate the error margin that should be used: The calculated average height of the table-sheet cluster could be subtracted from the maximum height value found in the cluster. However, since the height value of the cluster varies further away from HERO, this might not always be the most optimal value. But this type of implementation could be considered.

Unfortunately, it will be very difficult to filter out all the objects that are above the table plane. This is unfortunately a limitation of the method. To extend, and possibly improve this aspect of the method, research can be done to add an additional segmentation method that can specifically target objects.

5.5 Occluded space

The implementation of the occluded space gives satisfactory results when the objects are well-defined. With an error margin of 5 cm, there is a large gap between the object and the start of the occluded space compared to an error margin of 2 cm, as seen in Figure 11a and Figure 11b. As mentioned, it could be considered to lower this margin using the appropriate camera position. Another option is to consider this problem in a 2D fashion, the height values are removed from the vector equation. This will leave a direction vector and a starting point instead, if this is plotted then the problem is resolved. However, as a result, it is not possible to determine where the occluded space stops. But this isn't necessarily a problem, because, this will guarantee that HERO will not find a placement solution behind a product.

5.6 Not table

The not-table implementation has given satisfactory results in all the tested scenarios. It clearly shows depicts the edges of the table.

5.7 Field of view

As mentioned, during the real scenario testing it quickly became apparent that this implementation does not function well on HERO. The implementation assumes that the camera is perfectly parallel with respect to the table. However, this is a condition that cannot be met in real scenarios, especially since to increase the amount of depth information the camera has to be tilted. Therefore, a possible solution is given that, due to time constraints, is not yet implemented.

Since the horizontal field of view is a simple linear line through two points, the solution should focus on defining 2 points that the line must pass through. One point that can be defined uses the position of HERO's camera. The other point must fall on the edge of the image.

The vertical field of view can be neglected using the current implementation because a distance constraint is used instead. Therefore HERO will not pick points that are close to him in any regard.

5.8 Dilation

A large amount of time was spent attempting to dilate specific colored pixels on the costmap. This is preferred over dilating all the pixels at once because then different sections can be treated differently. For example, a larger safety margin can be given to objects on the table in comparison to the not-table space. Furthermore, this would also allow the use of other morphological operations, such as opening and closing, to process the bitmap data without altering all the pixels. Unfortunately, after spending a large amount of time performing literature studies on specifically dilating certain colors, it was decided to continue with the less optimal, but usable, result. Therefore, future work could consist of creating an original structering element that alters specific RGB values.

5.9 Placement costmap & Point selection

Throughout testing, it could be seen that HERO is not yet able to consistently select an appropriate placement point. However, this is not directly due to a problem with the method, but more so due to problems with the implementation. For example, in scenario 3 due to the incorrect implementation of the FOV, placement options are given at the edges of the table. If this problem were to be neglected, then HERO would not have been able to obtain a placement option within the preferable placement radius.

To continue, an aspect that is currently missing regarding the final generation of placement points is what HERO should do if it does not obtain a placement option. It might be an option for HERO to drive backward and forwards to possibly find a placement option. Or otherwise, approach the table from a different side if possible. But, this is a problem that can be considered for future work.

Finally, the selection of a placement option is currently chosen to be the furthest point that is closest to the middle. This is a simple criterion that should function well in most cases. But it might be interesting to investigate other criteria, to give a potential user more freedom on where objects will be placed. However, this is not a main priority considering the remaining problems of the method and its implementation.

6 Conclusion

To conclude, the devised method for determining placement solutions on a cluttered surface has been presented in section 3. The goal of the method is, to allow a robot to place an object on an unknown table surface. The robot captures a depth image using an RGBD camera. This image is converted into a point cloud, which is segmented using parallel plane model segmentation to obtain a cluster containing only the table sheet. Objects above the table sheet are filtered and the occluded space behind the objects is defined. Furthermore, the points below the plane are used to define the edges of the table, and together with the field of view of the robot's camera, the placement boundaries can be created. The information is plotted on a 2D colored image called the costmap. Using the image processing technique, dilation, a margin is created for the occupied and unknown space that the robot has to consider when placing an object. In addition, the robot's preferred placement distance is implemented to remove the points that are either too far or too close to the robot. As a result, the sections of the table sheet that remain free-space are considered viable placement options and can be filtered from the costmap. A placement solution is chosen from these options, the final solution is chosen to be the placement option that is the furthest away and is the closest to the center of the image.

The chosen method can lead to a satisfying result as discussed section 5. However, due to implementation mistakes in certain steps of the method, the result is currently lacking. The robot is not yet able to consistently define appropriate placement options, and the method is not fully automated.

Suggestions regarding resolving these mistakes are presented, such as a different implementation of the field of view. Furthermore, recommendations for future work are given, such as automating the process of tilting the robot's camera or adding further segmentation methods to create a better object filter.

References

[1] Richard Honti, Ján Erdélyi, Alojz Kopacik. (2018). *Plane Segmentation from Point Clouds*

[2] Xuming Ge, Jingyuan Zhang, Bo Xu, Hao Shu, Min Chen. (2022). *An Efficient Plane-Segmentation Method for Indoor Point Clouds Based on Countability of Saliency Directions*

[3] P. Hundepool. (2022). *Table modeling for a service robot from a single viewpoint towards object placement*

[4] T.M.H. Beurskens. (2022). *Modelling a table using the Hero robot*

[5] Jesús Balado, Peter van Oosterom, Lucía Díaz-Vilariño, Martijn Meijers. (2020). *Mathematical morphology directly applied to point cloud data*

[6] Point Cloud Library (PCL). *Region growing segmentation.*
Retrieved from: `https://pcl.readthedocs.io/projects/tutorials/en/latest/region_growing_segmentation.html`

[7] Marius Muja, Matei Ciocarlie. *Tabletop Object Detector.*
Retrieved from: `http://wiki.ros.org/tabletop_object_detector`

[8] Case description:
Peter van Dooren. (2022). *Where can I put this?*