ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Bachelor Thesis

Object Detection and Grasp Planning with an Omnidirectional Aerial Manipulator

Spring Term 2021

Supervised by:
Margarita Grinvald
Michel Breyer

Authors:
Martin Inauen
Philippe Brigger

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Object Detection and Grasp Planning with an Omnidirectional Aerial Manipulator

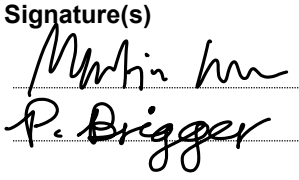
Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Inauen	Martin
Brigger	Philippe

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, August 11th, 2021	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Abstract

Unmanned aerial vehicles have rapidly evolved in recent years. The ability to interact with their environment, known as aerial manipulation, is a particularly new field of drone research. To advance this field of research, the focus project *Griffin* developed a novel, omnidirectional aerial manipulator. Grasping, transporting and placing objects was the main focus of aerial manipulation within the project. The omnidirectional flying platform enables flexible grasping of objects from any pose. The integrated robot arm compensates for the positional error of the flying platform, ensuring a high precision at the end-effector. With this bachelor thesis, the autonomy of the aerial manipulator is increased by implementing an object detection and grasp planning pipeline. With the use of a color camera, the unmanned aerial vehicle can autonomously detect and locate objects of known shape and color in an indoor environment. A depth sensor provides information on the orientation and optimal grasp point of the detected object. With this information, the unmanned aerial vehicle can plan and execute the grasp with an optimal pose. The performance of the pipeline was evaluated with tests in a simulation environment. The tests showed that the platform is able to autonomously detect and grasp objects with the desired accuracy in an optimal flight configuration.

Preface

We would like to express our heartfelt gratitude to Prof. Dr. Roland Siegwart, Margarita Grinvald and Michel Breyer, for their amazing support throughout this thesis. Without their valuable feedback, this thesis never would've been realizable. In particular, we would like to thank Margarita Grinvald and Michel Breyer for supervising us. Their guidance and vast knowledge helped us stay on track and taught us many more things than could be recorded in this thesis. Additionally, we would like to thank ETH Zürich and the Autonomous Systems Lab for providing excellent infrastructure and giving us the chance to pursue our own ideas. Finally, thanks go out to all of the people who proofread the thesis and supported us along the way.

Contents

Symbols

Symbols

d	distance
$\boldsymbol{p}$	point (x,y,z)
f_x,f_y	focal lengths
c_x,c_y	optical center
μ	mean
σ	standard deviation
$\boldsymbol{R}$	rotation matrix
$\boldsymbol{t}$	translational vector
u,v	Image coordinates
x,y,z	x, y and z-coordinate
ϕ, θ, ψ	roll, pitch and yaw angle

Indices

W	World frame
C	Camera frame

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
ASL	Autonomous Systems Lab
EE	End-Effector
CF	Coordinate Frame
FOV	Field of View
DOF	Degrees of Freedom
RGB	Red Green Blue
RGB-D	Red Green Blue Depth
ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle
2D	2-dimensional
3D	3-dimensional

Chapter 1

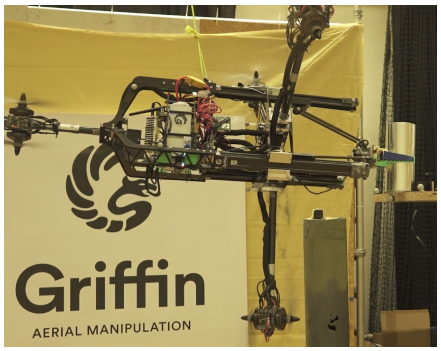
Introduction

1.1 Focus Project Griffin

This bachelor thesis extends the focus project *Griffin* [1]. In the focus project *Griffin*, a novel aerial manipulator concept was developed, built and tested(see Figure 1.1). The design featured an omnidirectional flying platform with an integrated three degrees of freedom (DOF) linear delta manipulator. It was shown that the robot arm can compensate the positional error of the flying platform. Thanks to the omnidirectional flying capabilities, *PrisMAV* could precisely grasp and place cylindrical objects from an arbitrary pose. In particular, the unmanned aerial vehicle (UAV) could grasp an object from the side in what was referred to as the horizontal flight configuration, or from the top in the tricopter flight configuration (see Figure 1.2).

Figure 1.1: *PrisMAV* - The Prismatic Micro Aerial Vehicle. Final prototype of the aerial manipulator developed by the focus project *Griffin*.

(a) Top grasp with *PrisMAV* flying in tricopter flight mode.



(b) Side grasp with *PrisMAV* flying in horizontal flight mode.

Figure 1.2: Aerial manipulation missions in two different flight configurations.

1.2 Motivation

In the scope of the focus project, perception was not considered. That is, to carry out a mission, it was clear beforehand where the target object and drop-off area were located. This significantly simplified a mission, as the object location and drop-off area could directly be sent to *PrisMAV*. Although *PrisMAV* could subsequently automatically execute its mission, a perception algorithm used to detect and locate target objects or obstacles and to plan the grasp would be desirable. Such an algorithm would highly increase the autonomy of the UAV, making it more feasible to imagine using the robot for real-world applications. To increase the UAV's autonomy, this bachelor thesis provides an object detection and grasp planning pipeline specifically designed for *PrisMAV*.

1.3 Goals

Since the focus project *Griffin* designed an omnidirectional aerial manipulator, the pipeline should make use of the UAV's omnidirectional flying capabilities. In particular, the pipeline must be able to decide whether an object should be grasped from the side or from the top. This led to the following thesis goals.

- From RGB images, detect objects of known shape and color
- Locate target object in world coordinates
- Determine the graspability of the target object
- Approach the object with an optimal pose in one of the two flight modes
- Calculate the optimal grasp point
- Execute the grasp

Since the main aim of the thesis is to increase the autonomy of the robot, *PrisMAV* must be capable of running the pipeline autonomously.

Chapter 2

Related Work

Based on the goals for this thesis, relevant literature was consulted and existing solutions to the problem were studied in detail. This built the foundation for the chosen methods of the pipeline.

A broad overview of different methods and techniques that have been used for RGB-D image-based object detection were given in [2]. It summarized and discussed the benefits and limitations of commonly used pipelines. In particular, traditional object detection algorithms were compared to more modern, deep learning methods. The result was that deep learning techniques achieve remarkable performance compared to traditional methods but require large datasets for training. In [3], a full pipeline for finding, picking up and relocating objects was presented. To run the pipeline, a finite state machine was designed. As the colors and shapes of the objects to be detected were known, a blob detector was used to detect the objects. Additionally, an algorithm for finding the 3D pose of an object from the 2D object pose was explained. A different object detection and grasp planning pipeline was presented in [4]. The object detection pipeline used a convolutional neural network to produce object candidates. An alignment algorithm then calculated the object location by filtering the object from the background. The pipeline then calculated the pose of the object by projecting the point cloud of the detected object onto a model point cloud of the object with an iterative closest point based algorithm. In the grasp planning, first, a set of contact points was generated. From these contact points, grasp candidates were generated and their grasp quality was analyzed. The grasp candidate with the highest score was then used to execute the grasp. As in [3], a state machine was used to run the whole pipeline. In the bachelor thesis [5], an object recognition and visual correction pipeline for a block stacking application was developed. The object recognition pipeline detected and estimated the pose of object candidates from a 3D point cloud. This was done by segmenting the ground plane with a random sample consensus based algorithm, and then clustering remaining points into block candidates. Each block candidate was separately analyzed. The grasp planning was done by calculating the midpoint of a target block.

Consulting the relevant literature showed that there has already been a lot of research conducted in the field of object detection and grasp planning. At the same time, it also outlined the fact that there haven't yet been many object detection and grasp planning pipelines constructed for omnidirectional aerial manipulators. The lack of widespread pipelines for omnidirectional aerial manipulators added to the motivation to pursue this thesis which should serve its purpose of propelling the research in this field.

Chapter 3

Methods

The goal of this thesis was to develop an autonomous object detection and grasp planning pipeline for an omnidirectional aerial manipulator. This chapter focuses on explaining the methods that were used to achieve this goal.

The pipeline is broken down into several steps which can be seen in Figure 3.1. In the object detection step, target objects are detected using the RGB image. The UAV then estimates the position of the object using the depth information provided by the depth sensor. After approaching the object, the grasp of the object is planned by taking the placement of the object in the environment into account, and calculating the object centroid.

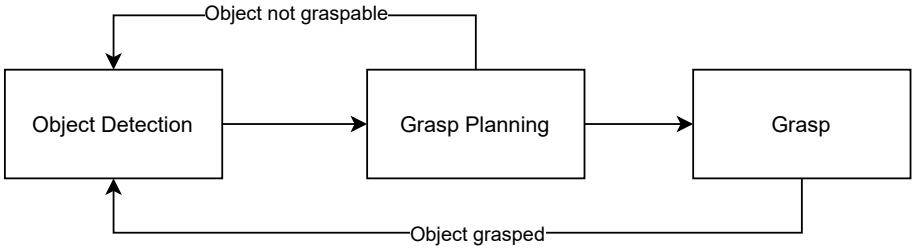


Figure 3.1: Schematic of the presented pipeline.

3.1 Object Detection

The goal of the object detection step, illustrated in Figure 3.2, is to detect red, cylindrical objects in the given environment and to locate them. During this step, the UAV is following a scan routine to search for the objects. After locating the object, the object detection step ends and the grasp planning step starts. During the whole pipeline, *PrisMAV* keeps track of the already processed objects and skipps them if they are re-detected.

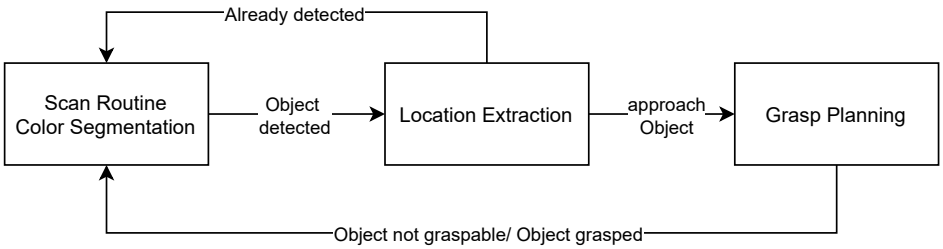
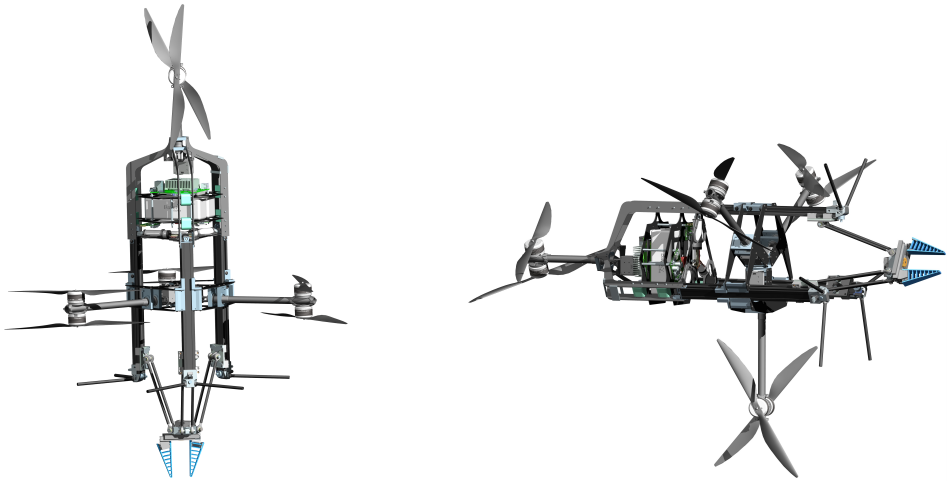


Figure 3.2: The object detection pipeline in detail.

3.1.1 Scan Routine

The scan routine marks the first step of the pipeline. If in the current scan window no objects are detected or all of the objects have been analyzed, *PrisMAV* flies to the next scan window. Figure 3.4 illustrates how the scan window is adapted. To fly from one point to another, generating a trajectory is necessary. The trajectory generator plans a smooth trajectory, as proposed by [6], from the current position of the UAV to the next scan point. The scan routine is carried out in the tricopter flight mode as seen in Figure 3.3a, and the altitude of the UAV remains constant from one scan point to another. It is assumed that there are no obstacles at this altitude preventing *PrisMAV* from following this scan routine, making an object avoidance algorithm unnecessary.



(a) Tricopter flight configuration (b) Horizontal flight configuration

Figure 3.3: Two main flight modes of *PrisMAV*.

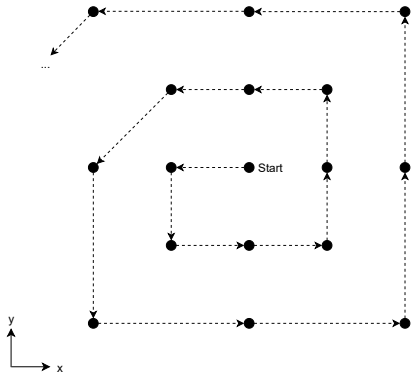
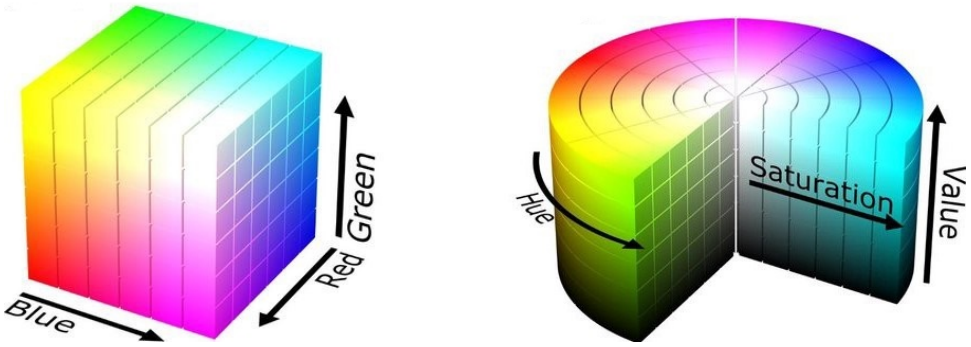


Figure 3.4: The proposed scan routine sketched in the xy-plane where each dot marks a new scan window. During the scan routine, the UAV remains at a constant height (z-coordinate).

3.1.2 Color Segmentation

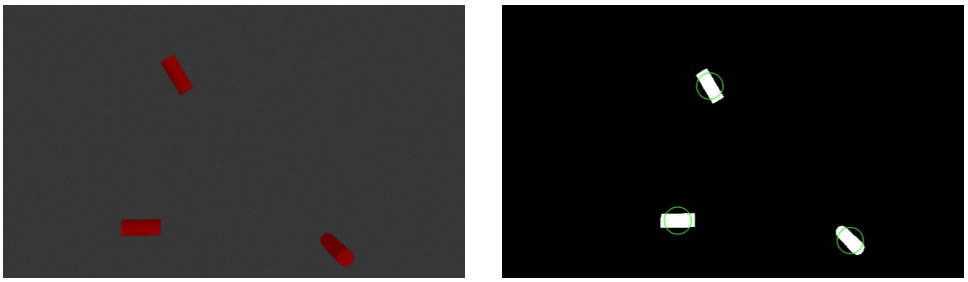
The aim of the color segmentation step is to detect objects of a given color. The output of the detector is also used for the 3D location of the object. In a first step, the RGB image is converted to the HSV color space (Figure 3.5 shows the two color spaces). The HSV color space is used because this color space better represents how people relate to colors than the RGB color space does. As shown in [7], HSV is superior for color segmentation in real world applications when comparing the mean square error and signal-to-noise ratio.



(a) RGB color space (b) HSV color space

Figure 3.5: Different color spaces used in the color segmentation step [8].

After converting the image to HSV color space, each pixel value is evaluated based on the given HSV thresholds. These thresholds must be defined in advance by the user and have to be changed in different lighting conditions. In the scope of this thesis, the objects to be grasped are cylindrical and red (see fig. 3.6a), and the lighting conditions within the simulation environment remain the same. Therefore, a static image of the object can be used to find adequate HSV threshold values. Applying the thresholding on the color image results in a binary image, where all the red pixels appear white, while the other pixels are made black (see fig. 3.6b). Connecting white pixels are grouped together to form blobs and blobs closer together than a minimal distance threshold are merged together. A detected blob is marked with a green circle in the binary image.



(a) Objects in RGB image.

(b) Objects in masked image. Detected blobs marked with green circle.

Figure 3.6: Color segmentation step. Binary mask with detected blobs extracted from the RGB image.

After this step, the blobs are still filtered by size, to remove blobs too small to represent an object. The final output is a binary image and the number as well as location of detected blobs saved in a keypoint vector, which can later be used for computing the 3D location of the target object.

3.1.3 Object Location

The aim of this step is to locate the object in 3D space from the RGB-D image. The required inputs are the mask from the color segmentation, as well as the depth image, along with the intrinsic and extrinsic calibration parameters of the cameras. By overlaying the depth image with the mask, only the relevant area of the depth image is analyzed to calculate the 3D object position. Therefore, it has to be ensured that only one blob appears in the scan window. Otherwise, the calculated position would be the average location of all the present objects and not of one specific object only. To avoid this from happening, *PrisMAV* approaches one specific blob before continuing with the location computation if multiple blobs are detected.

Tranformations

In order to transform a pixel with coordinates u and v from the depth image to a point $\boldsymbol{p_W}$ in the world frame, the point $\boldsymbol{p_C}$ in the camera frame has to be calculated and then transformed to the world frame. The conversion from the depth image to a point $\boldsymbol{p_C}$ in the camera frame can be expressed as follows.

$$\begin{aligned} z_C &= depth\,(u, v) \\ x_C &= \frac{u - c_x}{f_x} z_C \\ y_C &= \frac{v - c_y}{f_y} z_C \end{aligned} \tag{3.1}$$

f_x and f_y are the focal lengths, c_x and c_y are the optical center of the camera image and z_C, x_C and y_C are the 3D point coordinates $\boldsymbol{p_C}$ of the pixel expressed in the camera frame.
This point $\boldsymbol{p_C}$ in the camera frame can then be transformed to a point $\boldsymbol{p_W}$ expressed in the world frame using the transformation matrix $\boldsymbol{T_{W \leftarrow C}}$ from the camera to the world frame as follows.

$$\begin{pmatrix} \boldsymbol{p_W} \\ 1 \end{pmatrix} = \boldsymbol{T_{W \leftarrow C}} \cdot \begin{pmatrix} \boldsymbol{p_C} \\ 1 \end{pmatrix} = \begin{bmatrix} \boldsymbol{R_{W \leftarrow C}} & \boldsymbol{t_{W \leftarrow C}} \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} \boldsymbol{p_C} \\ 1 \end{pmatrix} \tag{3.2}$$

$\boldsymbol{R_{W\leftarrow C}}$ is the rotation matrix from the camera frame to the world frame and $\boldsymbol{t_{W\leftarrow C}}$ is the translation from the camera frame to the world frame.

Blob Center

To make sure only one blob appears in the overlayed depth image, the size of the keypoint vector containing the detected blobs from the color segmentation step is checked. If there is more than one blob detected, *PrisMAV* approaches a specific blob by using Equation (3.1) and Equation (3.2) to transform the blob center position to a point in the world frame.
The camera height is reduced to twenty centimeters above the object, while the image is approximately centered around the x- and y-position of the object.

Initial Centroid

The aforementioned 3D location ensures that only one object is in the FOV of the depth sensor. To make the grasp planning more accurate, a second 3D location is made by calculating the 3D centroid of the object (see Figure 3.7). The depth image is overlayed with the binary mask from the color segmentation step. This depth blob is then transformed to a point cloud in the world frame using the transformation in Equation (3.1). The centroid position of a point cloud with points $\boldsymbol{p_i}$ is then calculated as follows.

$$\boldsymbol{p_{centroid}} = \frac{1}{n}\sum_{i=0}^{n}\boldsymbol{p_i} \tag{3.3}$$

This point $\boldsymbol{p_{centroid}}$ is then transformed to the world frame using Equation (3.2). To end the object detection step, *PrisMAV* approaches the centroid location.

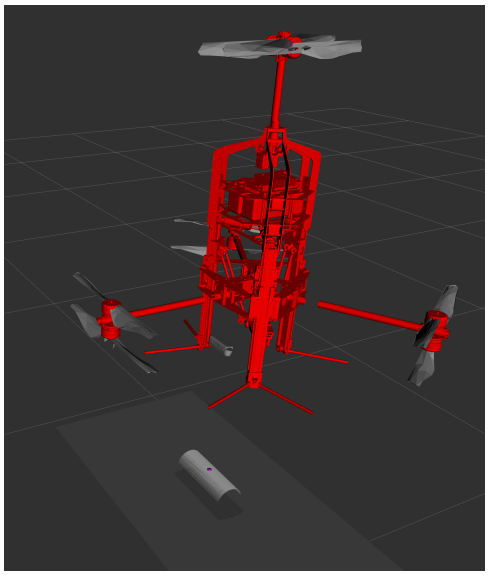
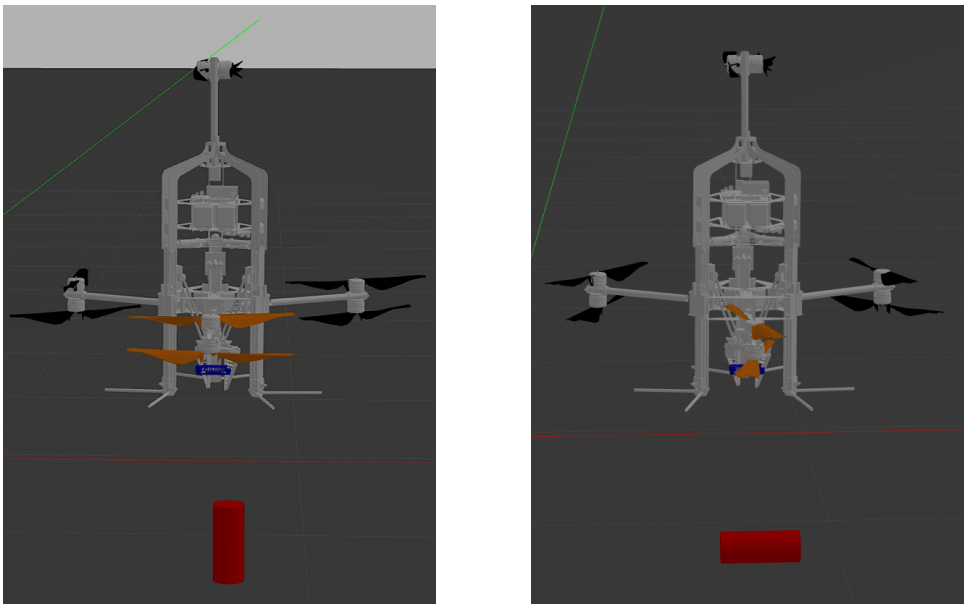


Figure 3.7: Object centroid marked as pink point in 3D point cloud.

3.2 Grasp Planning

After finishing the object detection step, *PrisMAV* already has approached the object of interest. The aim of the grasp planning step is to determine whether the

object of interest can be grasped, and if graspable, to calculate the optimal grasp point and pose with which the UAV should execute the grasp (see Figure 3.9). During the focus project, it was seen that grasping a cylindrical object with adaptive fingers works best when the cylinder is lying horizontally.



(a) Not graspable from the top. Potentially graspable from the side.

(b) Graspable from the top.

Figure 3.8: Not graspable vs. graspable object for drone flying in tricopter configuration.

Thus, only cylindrical objects lying horizontally are defined to be graspable for this thesis (see Figure 3.8b). Thanks to *PrisMAV*'s omnidirectional flying capabilities, a vertically standing object (see Figure 3.8a) can, however, potentially be grasped by changing the flight mode of *PrisMAV* to the horizontal flight configuration and approaching the object from the side (see Figure 3.3b). Therefore, aside from calculating the optimal grasp point, an integral part of the grasp planning step is to choose which flight mode optimal to carry out the grasp.

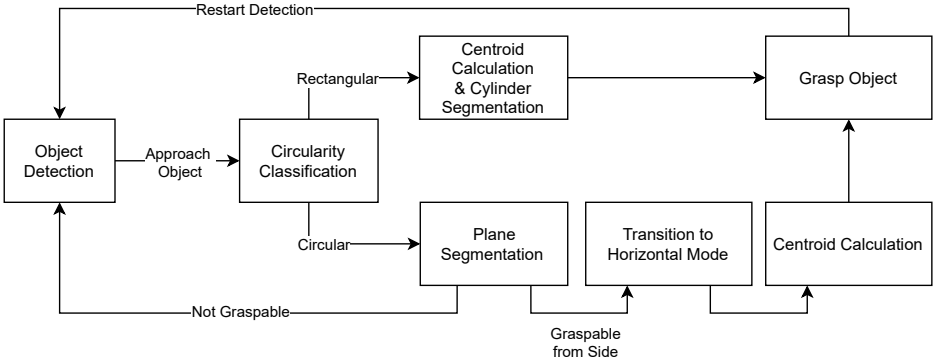


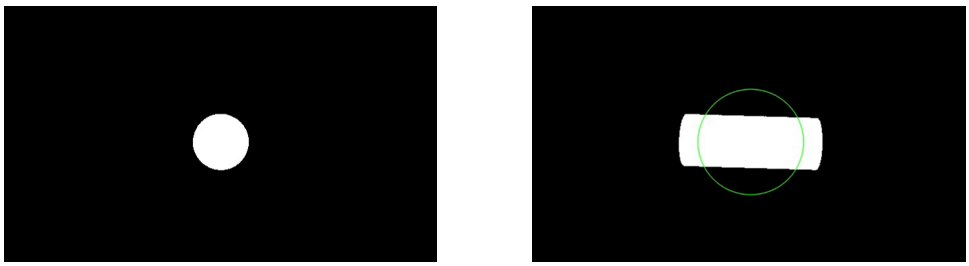
Figure 3.9: The grasp planning pipeline in detail.

3.2.1 Circularity Classification

The first step of the grasp planning is to determine whether an object is graspable. This is done by filtering the blob of the object of interest by circularity. The circularity C of an object is defined as the blob area A divided by the squared blob perimeter P. The ratio ranges from zero, a non-circular object, to one, a perfectly circular object, or, mathematically expressed:

$$C = \frac{4\pi A}{P^2} \ , \ C \ \epsilon \ [0, 1] \tag{3.4}$$

A vertically standing cylinder appears as a circle on the 2D masked image, while a horizontally lying cylinder appears as a rectangle (see Figure 3.10). By setting an appropriate maximum circularity threshold, it can be determined whether an object is graspable in the tricopter flight mode or not. For objects graspable from the top, the grasp planning continues with the cylinder segmentation step which is discussed in Section 3.2.3. The other objects are labeled as potentially graspable and have to be further analyzed to see if they can be grasped from the side.



(a) Not graspable object.

(b) Graspable object marked with green circle.

Figure 3.10: Circularity classification of detected object.

3.2.2 Plane Segmentation

For an object to be graspable from the side, *PrisMAV* needs enough clearance from the floor to be able to fly in the horizontal flight mode. Therefore, an object has to be placed near the edge of an elevated plane (e.g. a shelf) for it to be graspable from the side. If an object is graspable from the side, the best pose for approaching and grasping the object still has to be calculated.

Side Grasp Feasibility

In order to check the feasibility of grasping the object from the horizontal flight configuration, the plane on which the object is located is segmented and the minimal distance from the center of the object to the edge of the plane is calculated.
The plane segmentation is accomplished by thresholding the depth image, which results in a binary mask image. This threshold was chosen according to the flight height above the plane.
From this mask, the contour of the plane can be found by using the algorithm proposed by [9]. The output of this algorithm is a vector $\boldsymbol{C}_{plane}$ containing the u and v values of the plane contour pixels in the binary mask.
The same approach is used to detect the contour of the object. This time, the binary mask is created from the RGB image using the color segmentation approach mentioned in 3.1.2. After this, the contour of the object can be calculated. The contours are drawn as white lines in Figure 3.11.

In order to calculate the center of the object, the moments of the object contour are used. The moments m_{ji} of the object are calculated as follows.

$$m_{ji} = \sum_{uv} u^i * v^j \tag{3.5}$$

In Equation (3.5), u and v incorporate the pixel coordinates. From these moments, the center of the object contour $\bar{u}$ and $\bar{v}$ is found.

$$\bar{u} = \frac{m_{10}}{m_{00}}, \bar{v} = \frac{m_{01}}{m_{00}} \tag{3.6}$$

The center of the object contour is then used to find the minimal distance between the object and the contour of the plane (marked as a green line in Figure 3.11). First, the euclidean distance between the object center and each value in $\boldsymbol{C_{plane}}$ is calculated in the camera frame. As a result, the point on the edge of the plane with the smallest euclidean distance to the center of the object is found. These two points are then transformed to the world frame by using Equation (3.2). After this transformation, the minimal distance in the world frame is obtained and the feasibility of a side grasp can be evaluated. If the minimal distance is below the distance threshold d_{max}, the object is considered to be graspable from the side.

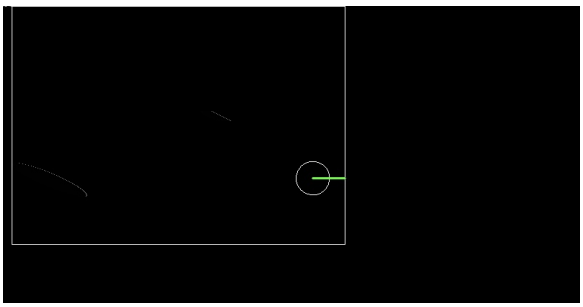
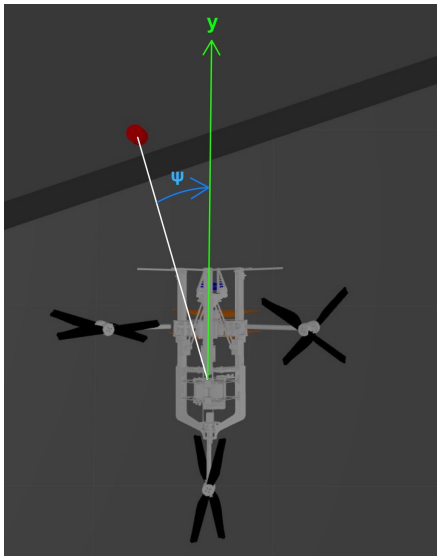


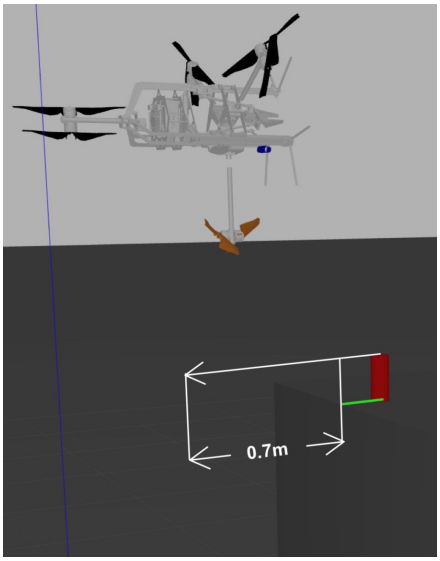
Figure 3.11: Plane segmentation. Contours drawn as white lines. Object is graspable if green line is shorter than the distance threshold d_{max}.

Side Grasp Pose Estimation

Using the object center location and the edge point of the plane with minimal distance to the object as inputs, the side grasp pose of *PrisMAV* is calculated. By constructing a line between these two points, the optimal grasp position for the UAV was set $0.7m$ (drone size + safety distance) away from the plane edge along this line, as well as at the object height, as seen in Figure 3.12b.
The desired orientation of *PrisMAV* is found by calculating the angle between the constructed line and the reference frame as seen in Figure 3.12a. This approach leads to a desired pose in the world frame, which is then sent to *PrisMAV* in incremental steps (see Figure 3.13). Firstly, *PrisMAV* increases it's altitude by $1m$. This allows the UAV to safely transition from the tricopter to the horizontal flight configuration. It was seen that the transition worked best if the UAV first rolls for 90° and subsequently adjusts the yaw angle ψ to face the object. After adapting the yaw angle, *PrisMAV* flies to the object height to conclude the object approach. Now, the UAV is ready to carry out the final grasp planning step described in Section 3.2.4.



(a) Estimated yaw-angle ψ for approaching an object.



(b) Estimated position for approaching an object.

Figure 3.12: Pose estimation to approach an object for the side grasp.

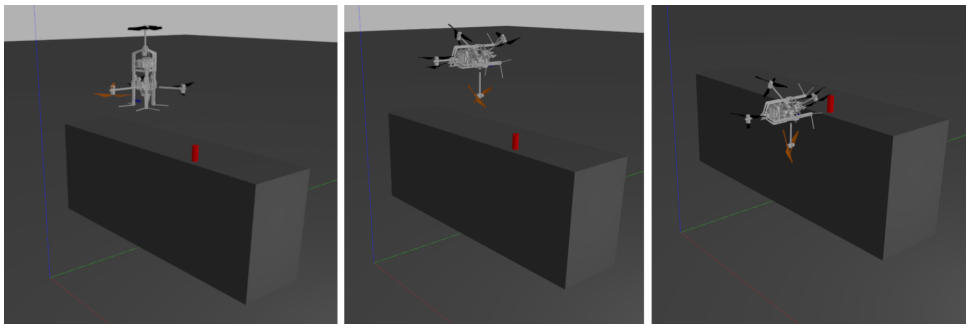


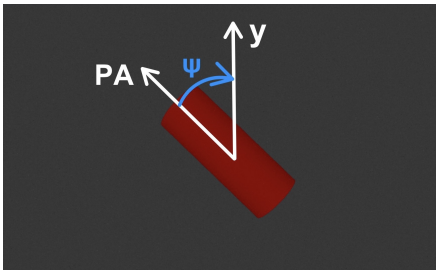
Figure 3.13: *PrisMAV* approaching an object to perform the side grasp.

3.2.3 Cylinder Segmentation

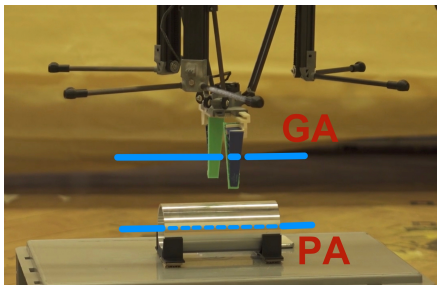
If an object is to be grasped in the tricopter flight mode, *PrisMAV* has to adapt its yaw angle ψ such that the cylinder is aligned with the end-effector (EE) opening. In order to achieve this, the principal axis of the cylinder is calculated from the 3D point cloud of the object. This is done by fitting a cylinder model to the point cloud using a RANSAC (Random sample and consensus) approach as seen in [10]. The outputs of this step are the cylinder model parameters, from which the orientation of the cylinder principal axis is calculated with the following formula.

$$\psi = \arccos\left(\frac{\boldsymbol{PA} \cdot \boldsymbol{y}}{\|\boldsymbol{PA}\|_2 \|\boldsymbol{y}\|_2}\right) \tag{3.7}$$

In Equation (3.7), ψ is the angle between the principal axis $\boldsymbol{PA}$ of the cylinder and $\boldsymbol{y}$ is the camera y-axis as can be seen in Figure 3.14a. $\boldsymbol{PA} \cdot \boldsymbol{y}$ denotes the scalar product between the two vectors while $\|\boldsymbol{PA}\|_2$ and $\|\boldsymbol{y}\|_2$ are the magnitudes of the respective vectors. By yawing by ψ, the EE opening is correctly positioned to grasp an object as the $\boldsymbol{PA}$ is then aligned with the camera y-axis $\boldsymbol{y}$ as shown in Figure 3.14b.



(a) Required yaw angle to align the grasp axis with the cylinder.



(b) Gripper axis aligned with cylinder.

Figure 3.14: Aligning gripper axis GA with principal axis PA of cylinder for top grasp.

3.2.4 Final Location - Grasp Centroid

To conclude the grasp planning, the optimal grasp point still has to be found. The procedure to calculate the grasp point remains the same for an object being grasped from the side as well as top, since at this point *PrisMAV* has already approached the object in a configuration where it can grasp the object. The optimal grasp point is calculated the same way as in Section 3.1.3 by calculating the 3D centroid of the object from the 3D point cloud. In the scope of this thesis, the pipeline ends at this point. In reality, the arm would still move out of the structure and the EE would grasp the object at the grasp point. Due to the lack of a simulation of the manipulator, this step was not carried out in the pipeline.

3.3 Implementation

The object detection and grasp planning algorithms were programmed in C++. Most algorithms were implemented with the help of OpenCV [11] and PCL [10] libraries. To traverse between the different states of the pipeline, a state machine was implemented using SMACH [12] in python. The whole pipeline was run using Robot Operating System (ROS) [13]. An overview of the implemented software structure can be seen in Appendix A.1.
Since two different images are used, namely a depth and an RGB image, a synchronization process between the two different images has to be done in order for the pipeline to work properly. The synchronization method looks at the varying time stamps of the two images and then finds the best match between the two images such that the time stamp difference doesn't exceed a maximum tolerable time difference.

Chapter 4

Testing & Evaluation

In this chapter, the performance and test results of the pipeline outlined in Chapter 3 are presented. All tests were carried out in the simulation environment Gazebo. Each component of the pipeline was tested to see how reliable and precise it is[1]. For this, different simulation worlds were setup where the amount of cylinders varied and either spawned randomly or at a predefined position. The cylinder dimensions remained the same. The cylinder height measured 20 cm while its diameter was 8 cm (like the cylinder used in the focus project). During the tests, the simulation was run using perfect (ground truth) odometry data. Due to the lack of a simulation of the manipulator, the test ended once the optimal grasp point and flight configuration was calculated and approached.

4.1 Testing Results - Object Detection

4.1.1 Scan Routine

The scan routine was tested in an environment with multiple objects at a predefined position. The simulation showed that the scan routine progressed as desired when either no objects were in the scan window or all objects had already been analyzed.

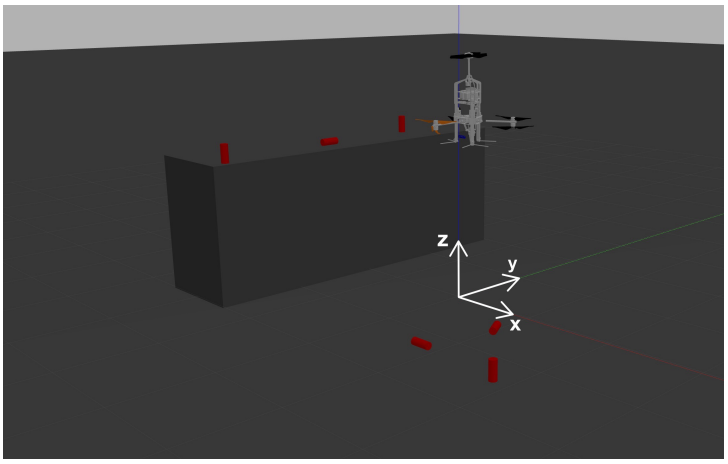


Figure 4.1: Simulation world with multiple objects used to test scan routine.

[1] Video of full pipeline in simulation: `https://www.youtube.com/watch?v=nRRAyTGgqtM`

4.1.2 Color Segmentation

The objects used in the simulation all had the same red color. Precise color threshold values could therefore be chosen by adapting the threshold values until all the blobs appeared in the mask.

4.1.3 Object Location

To evaluate the part of the pipeline concerning the object location, one cylindrical object was spawned at a random position in the simulation world. In 50 tests, the object was spawned laying horizontally on the floor with a random yaw-angle ψ, while in 47 separate tests, the object was spawned standing vertically on a shelf.

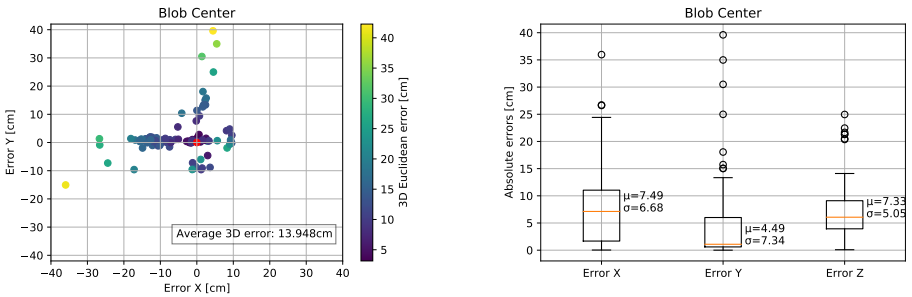
Blob Center

The euclidean error e_{pos} present in the first step of the object location, where the 3D center of the blob was estimated from the 2D RGB image and the depth image, was calculated from the ground truth position $\boldsymbol{p_{gt}}$ and the estimated position $\boldsymbol{p_{est}}$ as follows.

$$e_{pos} = \|\boldsymbol{p_{est}} - \boldsymbol{p_{gt}}\|_2 \tag{4.1}$$

The errors are illustrated in Figure 4.2a. They were plotted on the xy-plane since these two coordinates defined the approach position of the object and thus the FOV of the camera. It is important to note that the data was collected while the UAV was hovering above the object, and the errors must be understood with respect to the coordinate frame (CF) seen in Figure 4.1. During this step, an error in the z-direction was not as critical as an error in the x or y-direction, since the main goal of this step was to get the object centered in the camera image frame. With the color bar, one got an idea of how far the estimated location still differed from the ground truth location of the object.
In Figure 4.2b, the mean and standard deviation of the absolute error of the x, y and z location were plotted as a boxplot, where the absolute error e_i between the estimated coordinate i_{est} and ground truth coordinate i_{gt} was defined as follows.

$$e_i = |i_{est} - i_{gt}| \quad \text{for} \quad i = [x, y, z] \tag{4.2}$$



(a) Location error in xy-plane.

(b) Box plot with absolute errors in x, y and z location.

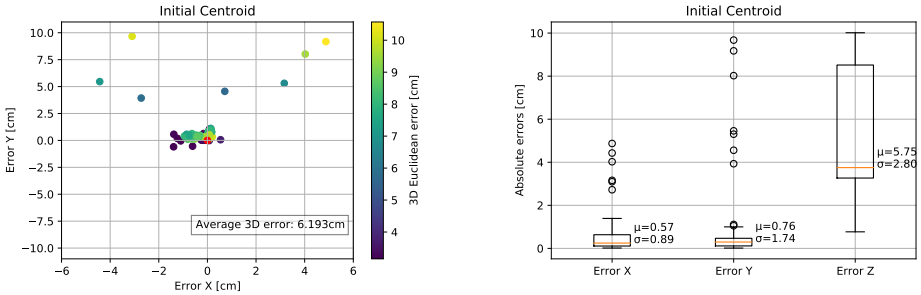
Figure 4.2: Blob center location.

As the plots in fig. 4.2 show, there were still large errors present during the blob center location step. One reason for the errors was that the location calculation often took place when part of the object was cut off in the scan window because it

was located at the very edge of the scan window. Another reason was the timing of the coordinate frame transformation. The transformation was often calculated before the drone had stopped its motion, leading to a false transformation between the camera and world coordinate frame.

Initial Centroid

The precision of the second location step, the initial centroid calculation, is captured in Figure 4.3. As in the blob center step, a boxplot with the mean and standard deviation of the absolute error of the x, y and z location was made, accompanied with a scatter plot. In the scatter plot, the error points were plotted on the xy-plane where the color bar indicates the magnitude of the 3D euclidean error between the initial centroid and the ground truth centroid.



(a) Location error in xy-plane.

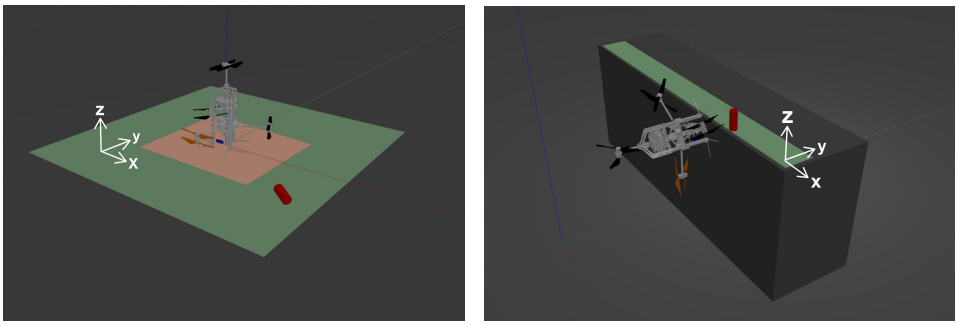
(b) Box plot with absolute errors in x, y and z location.

Figure 4.3: Initial centroid location.

It can be seen that the errors had decreased compared to the previous step. The appearance of outliers in the x and y location errors could be traced back to a bad blob center location approximation. The latter could lead to the object being partially cut-off from the FOV of the camera. Therefore, the centroid calculation also wouldn't consider all of the object's pixels for the centroid calculation and would produce a larger error.

4.2 Testing Results - Grasp Planning

To test the grasp planning, two different simulation worlds were used (see Figure 4.4). One world was made to evaluate the precision of the top grasp. Hence, one cylinder was randomly spawned on the floor. The object was spawned at most 1.2 m and at least 0.8 m away from the spawn position of *PrisMAV* in the x- and y-direction (see Figure 4.4a). The other world was made to test the side grasp. Here, the object was randomly spawned on top of the front half of a shelf (see Figure 4.4b). This allowed to test whether the plane segmentation correctly evaluated if the object is graspable from the side.



(a) Floor object randomly spawned in green area for top grasp.

(b) Shelf object randomly spawned in green area for side grasp.

Figure 4.4: The two different simulation setups used for grasp testing.

4.2.1 Circularity Classification

A good circularity threshold was required to make the circularity classification as robust and reliable as possible. To do this, the circularity value of a horizontally lying cylinder was calculated.

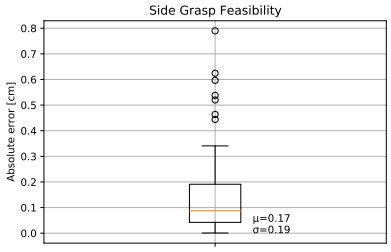
$$C_{hz} = \frac{4\pi \cdot l \cdot d}{(2 \cdot l + 2 \cdot d)^2} = 0.641 \tag{4.3}$$

In eq. (4.3), $l = 20\,\text{cm}$ denotes the height and $d = 8\,\text{cm}$ the diameter of the cylinders used in the simulation. To cover the case where the cylinder wasn't well placed in the image, a safety factor of 1.25 was added and the threshold value was set to 0.8. With this threshold, the pipeline correctly decided whether a target object was graspable.

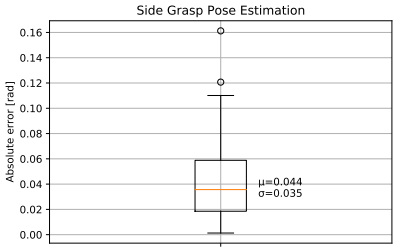
4.2.2 Plane Segmentation

Side Grasp Feasibility and Pose Estimation

Fifty tests with an object on a shelf were recorded to evaluate the plane segmentation step. From these fifty tests, in two tests the pipeline falsely evaluated whether an object was graspable from the side. The mean error and standard deviation of the error measurements are shown in Figure 4.5a. The absolute errors were calculated by subtracting the distance estimate with the ground truth distance between the object center and nearest shelf edge as defined in Equation (4.2). The transition from the tricopter to the horizontal flight configuration was carried out in incremental steps as discussed in Section 3.2.2. The error of the yaw angle adjustment is plotted in Figure 4.5b.



(a) Average error for distance from object center to plane edge used for side grasp feasibility.



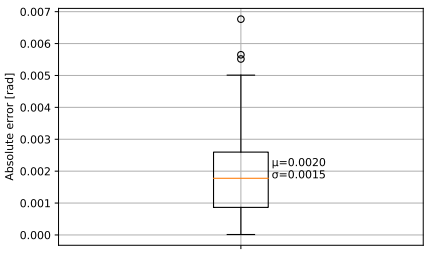
(b) Average error for angle between constructed line and reference frame used for side grasp pose.

Figure 4.5: Box plots of errors in plane segmentation step.

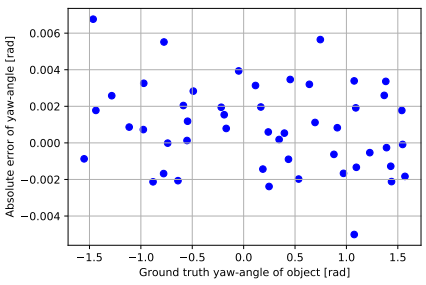
It was seen that the maximum outlier for the distance estimation was 0.8 cm while the grasp pose yaw angle error had a mean of 2.5 °. Since the grasp pose yaw angle error doesn't influence the alignment of the grasp axis and principal cylinder axis, this error won't have a direct impact on the grasp. It could however lead to the structure touching the shelf. Likewise, a distance error could lead to the drone colliding with the shelf as it falsely assumes that a side grasp is possible. To prevent a collision, the distance threshold could be reduced by the maximum outlier.

4.2.3 Cylinder Segmentation

In order to evaluate the cylinder segmentation step of the pipeline, the estimated yaw pose angle was compared to the ground truth yaw pose angle of the object. Because the cylinder was laying on the floor, the roll and pitch angle were assumed to be constant. Therefore, the absolute error of the yaw angle ψ served as a metric to evaluate this step. It is important to note that the orientation values are remapped from $[-\pi, \pi]$ to $[-\frac{\pi}{2}, \frac{\pi}{2}]$, which could be done since the object is assumed to be symmetrical. Figure 4.6a shows the mean and standard deviation of the absolute error of the yaw angle ψ. In Figure 4.6b, it can be seen that the error is independent of the ground truth yaw angle. Additionally, the maximum error of 0.4 ° shouldn't be large enough to misalign the grasp axis from the principal cylinder axis.



(a) Error between estimated and ground truth yaw angle of the object.



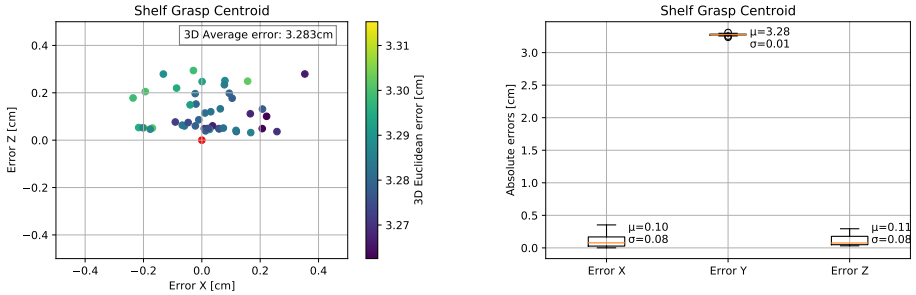
(b) Absolute yaw angle error with different ground truth angles.

Figure 4.6: Yaw angle estimation of the object based on the cylinder segmentation.

4.2.4 Final Location - Grasp Centroid

During the grasp centroid step, *PrisMAV* was either hovering in the tricopter or the horizontal configuration, and thus, the evaluation was done separately. The errors were evaluated the same way as for the initial centroid by subtracting the ground truth object centroid from the calculated grasp centroid. The results of the side grasp are plotted in Figure 4.7 while the top grasp data is illustrated in Figure 4.8. As for the previous two location steps, the mean and standard deviation of the absolute errors were plotted in a box plot, while the planar error coupled with the 3D euclidean error were plotted in a scatter plot.

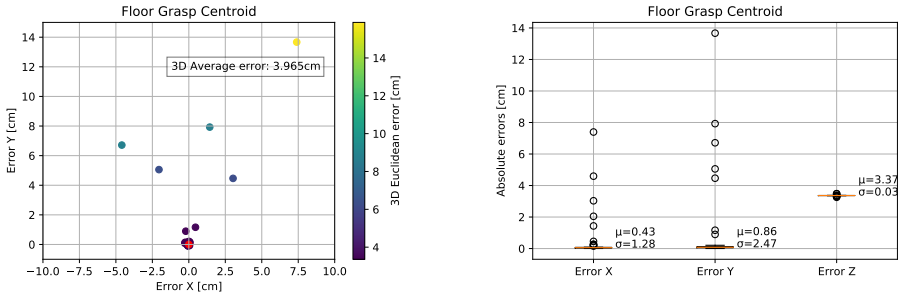
It is important to understand that, for the side grasp, the errors were plotted in the xz-plane as defined in Figure 4.4b. This is because for a side grasp, the x and z coordinates corresponded to the coordinates which the EE must track to have the principal axis of the cylinder aligned with the EE axis. Only if *PrisMAV* was capable of estimating the x and z coordinates of the grasp centroid precisely enough, would the object lie graspable between the adaptive fingers. The y coordinate stood for how close the UAV had to approach the object for a grasp to be possible. Conversely, for the top grasp, the x and y location errors were relevant for the cylinder major axis and EE axis alignment, while the z coordinate conveyed the approach information for a grasp (see Figure 4.4a for CF). As seen in both of the boxplots, one coordinate has a much larger error. This stems from the fact that only half of the cylinder is visible to the depth sensor.



(a) Location error in xz-plane.

(b) Box plot with absolute errors in x, y and z location.

Figure 4.7: Grasp centroid location for object grasped from side.



(a) Location error in xy-plane.

(b) Box plot with absolute errors in x, y and z location.

Figure 4.8: Grasp centroid location for object grasped from top.

As can be seen in Figure 4.7, the grasp centroid errors for the side grasp were significantly smaller than the previous location errors and the standard deviations of the errors were small. This had to do with the fact that during this step, the robot had approached the cylinder optimally. The cylinder thus was nicely centered in the camera image, making it easier to approximate its location. Similar results were found for the top grasp as plotted in Figure 4.8, although there were some significant outliers which need to be addressed to increase the reliability and robustness of the pipeline.

4.3 Performance

An overview of the computational time needed for the different parts of the pipeline is given in Table 4.1. The different times are by no means absolute, and they vary significantly on different computers. They only serve to illustrate the relative computational effort between the different parts of the pipeline. To optimize the pipeline for computational effort, the most promising part of the pipeline would certainly be the cylinder segmentation. In this thesis, the pipeline was not optimized for computational efficiency.

Pipeline Step	**Computational Time [s]**	**Relative Time [%]**
Color Segmentation	0.15	9.3
Centroid Calculation	0.01	0.6
Circularity Classification	0.15	9.3
Cylinder Segmentation	1.3	80.2
Plane Segmentation	0.01	0.6
Overall	**1.62**	**100.0**

Table 4.1: Comparison of the approximate computational time for different parts of the pipeline.

Chapter 5

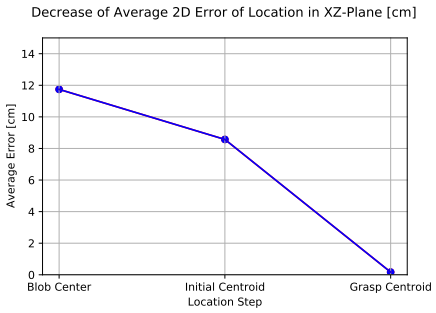
Discussion

In this chapter, the results of 4 are analyzed and improvements are explored. Furthermore, the necessary steps to bring the pipeline from simulation to the real world are discussed.

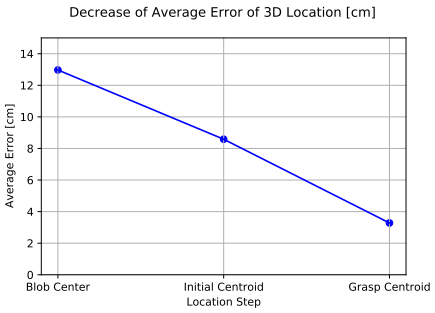
5.1 Error Discussion

5.1.1 Location Errors

As was seen in the many plots in Chapter 4, constant errors were present in the location steps. It could be seen that the error decreased with each subsequent step in the pipeline. This justified the usage of more than one location approximation. During the side grasp, the grasp centroid calculation step was the most important, as it significantly decreased both the 2D and 3D location error (see Figure 5.1a and Figure 5.1b). Alternatively, the initial centroid calculation step was most important for the top grasp, as the location error after this step was only slightly larger than after the final step (see Figure 5.2a and Figure 5.2b).



(a) 2D error development for side grasp with increase in location step.

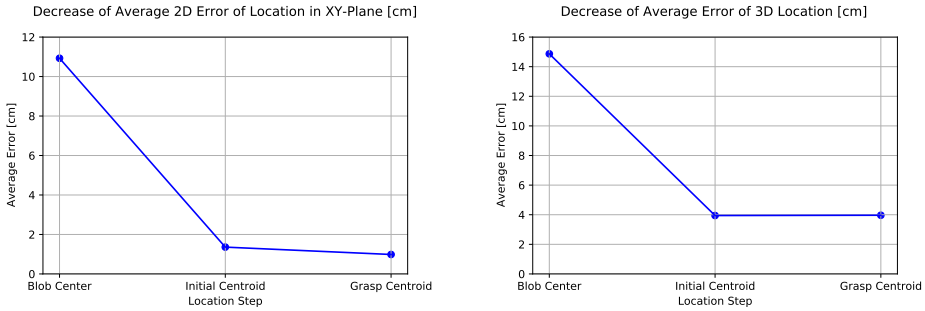


(b) 3D error development for side grasp with increase in location step.

Figure 5.1: Euclidean error development for side grasp.

Side Grasp	Blob Center [cm]	Initial Centroid [cm]	Grasp Centroid [cm]
xz error	11.74	8.57	0.17
3D error	12.97	8.58	3.28

Table 5.1: 3D and xz euclidean error values for side grasp.



(a) 2D error development for top grasp with increase in location step.

(b) 3D error development for top grasp with increase in location step.

Figure 5.2: Euclidean error development for top grasp.

Top Grasp	Blob Center [cm]	Initial Centroid [cm]	Grasp Centroid [cm]
xy error	10.93	1.36	0.99
3D error	14.87	3.95	3.97

Table 5.2: 3D and xy euclidean error values for top grasp.

The errors tabulated in Table 5.1 and Table 5.2 could be explained twofold.

On one hand, the object was generally located at the edge of the image frame when it first appeared during the scan routine. As a result, one half of the object was more visible than the other half of the object. Thus, the pipeline assumed the object was located closer than it actually was, because more pixels from the front half of the object entered into the location calculation. With every subsequent step, the object gradually got centered in the image frame, and the amount of pixels entering the location calculation more evenly came from both object halves. This decrease in the 2D location error is visible in Figure 5.1a as well as Figure 5.2a.

On the other hand, normally only half of the object was visible to the depth sensor. As a consequence, the grasp height was estimated too high. With grasp height, the distance that the EE must extend out of the structure to grasp the object is understood. Therefore, for a top grasp, the z location of the object defines the grasp height, while for a side grasp, the y coordinate of the object defines the grasp height (see Figure 4.4 for an illustration).

For the top grasp, the initial centroid calculation significantly decreased the 3D location error (see Figure 5.2b and Table 5.2), because the object was located better in the image frame than for the blob center calculation. The grasp centroid calculation did not change the error significantly, since the depth image frame was similar to the frame used in the previous step, where the object already was roughly centered in the frame.

For the side grasp, the grasp centroid calculation error was most affected by the 3D location error due to the UAV transitioning from one flight mode to another (see Figure 5.1b and Table 5.1). The depth sensor only saw half of the object for the grasp height calculation during the grasp centroid calculation whereas the 2D error of the blob center and initial centroid were comparatively large because during those steps only the top base was in the depth image frame.

To decrease the location errors, the grasp height of the grasp centroid could be adapted by the constant error which was recorded during the tests. The side grasp centroid height could be reduced by 3.28 cm underlined by the fact that there is a very small variance in the y location error of the box plot (see Figure 4.7b), while

the top grasp centroid height could be adapted by 3.32 cm for the analogous reason that the z location error of the box plot has a small variance (see Figure 4.8b). The relevant location coordinate for the grasp height differed due to the different flight modes with respect to the relevant CFs (see Figure 4.4 for CFs).

Grasp Type	With Offset [cm]	Without Offset [cm]
Side	3.28	0.17
Top	3.97	0.99

Table 5.3: 3D euclidean grasp centroid error values for side and top grasp with and without subtracted static offset observed in box plots Figure 4.7b and Figure 4.8b.

As Table 5.3 shows, by subtracting the static grasp height error, the 3D euclidean error of both the top and side grasp centroid error are less than one centimeter. This error now stems from a planar offset between the estimate and ground truth point and not from a false grasp height estimate, which can be seen from the similarity between the updated 3D errors and the original 2D euclidean errors tabulated in Table 5.1 and Table 5.2. In other words, the EE opening width must be on average at least twice the error 0.17 cm respectively 0.99 cm wider than the cylinder for a side and top grasp to succeed. To conclude, the opening width of the EE should be roughly 2 cm wider than the cylinder to compensate for the pipeline error.

The 2D location error could be reduced by calculating the final grasp centroid from a higher altitude. Especially for the top grasp, the UAV approached the object very closely after the color segmentation step. As a result, the object was sometimes not entirely in the image frame during the centroid calculations, which in return lead to a shifted centroid estimation and larger 2D euclidean errors.

Furthermore, the 2D and 3D blob center location error could be eliminated by adapting the scan routine. Instead of approximating the 3D location of a blob and approaching this location to only have one blob in the scan window, the blobs appearing in a certain scan window could be clustered. Like this, the centroid of only one cluster could be calculated and used as an initial approach point. Basing the initial approach point on the 3D point cloud of a blob cluster instead of the 2D RGB image should produce a smaller error, since less approximations are made to estimate the location. In return, the subsequent initial- and grasp centroid calculations would produce better results because the UAV approach point after the first step would be more accurate.

5.1.2 Angle Errors

The pose estimation of the cylinder was done with different methods. If the object was laying on the floor, the angle was estimated using the cylinder segmentation as explained in Section 3.2.3. The mean cylinder segmentation error of 0.002 rad, which is equivalent to 0.12 °, was acceptable (see Figure 4.6a for the plot). However, the relative computational effort of 80.2 % for this operation was rather high (see Table 4.1 for the relative computational time).

To increase the computational efficiency of the pipeline, the yaw angle ψ of the object could certainly be estimated in better ways. The axis of the cylinder could, for instance, be calculated by using the contour of the object and extracting the principal axis of this contour. Assuming that the UAV is hovering above the object, one could calculate the yaw angle ψ of the object in the world frame by comparing the orientation of the principal axis to the y axis of the world frame. The grasp axis is, namely, aligned with the world y axis at take-off. Thus, to grasp an object, the grasp axis must be aligned with the principal axis of the cylinder. This can be achieved by yawing by the aforementioned ψ degrees.

If the object was placed on a shelf, the difference between the estimated orientation and optimal orientation of *PrisMAV* was viewed as the angle error. The mean error of 0.044 rad, which is equivalent to 2.52 °, was clearly larger than at the cylinder segmentation, whereas the relative computational effort of 0.6 % was small (see Figure 4.5b for the mean angle error and Table 4.1 for the relative computational time).
To decrease the error magnitude, the plane and the cylinder could be extracted from the point cloud using a RANSAC approach. This could lead to a more accurate estimation of the yaw angle, although an increase in the computational effort would be expected.

5.1.3 Plane Segmentation - Distance Error

The estimation of the distance between the shelf and the object was a crucial part of the pipeline. If the object was falsely assumed to be graspable, this could lead to a crash of *PrisMAV*, since the platform could collide with the shelf. To prevent this from happening, the threshold for the maximal distance between the object and shelf edge could be reduced. Although this would prevent *PrisMAV* from colliding with the shelf, it would also restrict the grasping potential of the UAV. Certain objects which theoretically would be graspable, wouldn't be grasped, because the reduced maximal distance threshold wouldn't allow the grasp anymore.

A better solution would be to use an object avoidance algorithm. A possible implementation would be to only execute a side grasp if a generated trajectory wouldn't come into contact with the shelf model. The shelf model, in turn, could be extracted from the 3D point cloud. The algorithm for the distance estimation would remain the same.

5.2 From Simulation to Real World

To be able to use the pipeline on the real system, several aspects would still need to be considered. Foremost, during the tests in Chapter 4, noisy odometry data was not present. To get a better impression of how the pipeline might perform in the real world, a simulation with noise on the odometry sensor plugin was set up. The noise on the sensor plugin was modelled to replicate the odometry data captured from real life flight tests. The latter showed that the flight controller was capable of hovering at a precision of ± 7cm [1]. As can be seen in the plots (see Figure 5.3), roughly the same error was present in the odometry data during the noisy simulation.

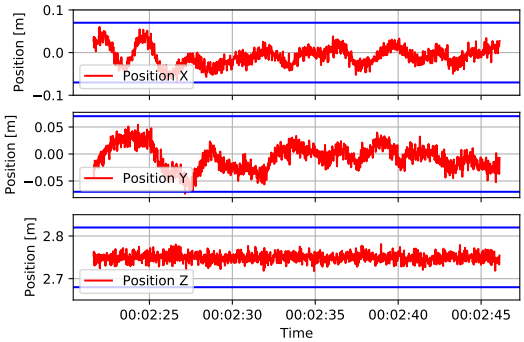


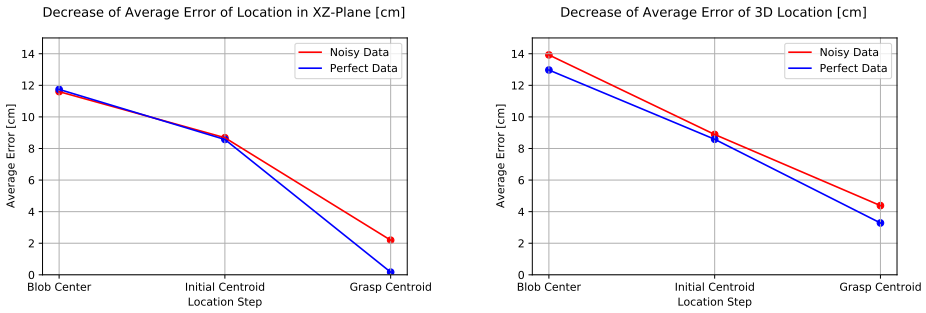
Figure 5.3: Position tracking of *PrisMAV* in simulation with noisy odometry data. Blue lines indicate ± 7cm marks.

In Figure 5.4, the performance of the pipeline with noisy odometry data versus perfect odometry data was plotted. As the euclidean error values in Table 5.4 and Table 5.5 suggest, the pipeline was robust to the odometry noise.

The decrease of the error in the blob center location with the noisy odometry data may seem counterintuitive (see Table 5.4). This could come from the fact that the noisy odometry data comes with a delay. Without noise, the scan routine is stopped as soon as a blob appears at the edge of the image frame. With the noisy odometry data, however, the scan routine is stopped slightly later because the data comes with a delay. This means that without noise, part of the object could potentially not be in the FOV of the camera as the blob is at the edge of the image frame. Conversely, with the noisy odometry data, the entire object should appear in the camera FOV as the scan routine is stopped with a slight delay. Calculating the blob center with the entire object in the image frame yields better results and explains why the blob center location with the noisy odometry data is slightly more accurate.

The most significant error increase with the noisy odometry test was found for the grasp centroid calculation. In particular, the 2D euclidean error increased from 0.17 cm to 2.19 cm. As discussed in Section 5.1.1, this 2D euclidean error is relevant for aligning the grasp axis with the principal cylinder axis. It essentially determines whether a grasp succeeds, since the grasp height could simply be adapted by the constant offset found in the test data (see Appendix A.2).

In other words, the opening width of the EE must on average be twice the error of 2.19 cm greater than the cylinder's diameter for a grasp to be successful (so approximately 4 cm). During the focus project, it was seen that the manipulator of *PrisMAV* could compensate the positional error of the flying platform with a precision of ± 1 cm [1]. Therefore, the opening width of the EE should roughly be 6 cm greater than the cylinder's diameter for a grasp to realistically be successful.



(a) 2D error development for side grasp with increase in location step for noisy vs. perfect odometry data.

(b) 3D error development for side grasp with increase in location step for noisy vs. perfect odometry data.

Figure 5.4: Euclidean error development for side grasp with noisy vs. perfect odometry data.

Data	Blob Center Error	Initial Centroid Error	Grasp Centroid Error
Perfect	11.74cm	8.57cm	0.17cm
Noisy	11.60cm	8.68cm	2.19cm
Δ	-1.2%	+1.3%	+1188.2%

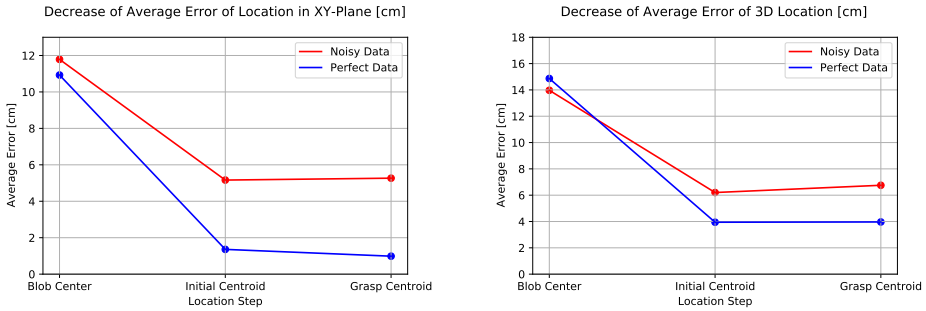
Table 5.4: 2D euclidean error development for side grasp with noisy vs. perfect odometry data.

Data	Blob Center Error	Initial Centroid Error	Grasp Centroid Error
Perfect	12.97cm	8.58cm	3.28cm
Noisy	13.93cm	8.89cm	4.38cm
Δ	+7.4%	+3.6%	+33.5%

Table 5.5: 3D euclidean error development for side grasp with noisy vs. perfect odometry data.

Similar results were found with the top grasp tests. Initially, the errors with the noisy odometry data were much larger than with the perfect data (see Figure 5.5, Table 5.6 and Table 5.7). This was due to significant outliers appearing in the centroid calculations. One way to deal with these outliers would be to add an additional blob detection step after the grasp centroid calculation but before the grasp execution. The blob detection could be used to verify that the blob center is within a certain threshold distance of the image center. If this weren't the case, it would be clear that the UAV miscalculated the grasp centroid, and a grasp wouldn't succeed. This is because the object wouldn't lie in the center of the image frame, and, hence, wouldn't end up in between the adaptive fingers of the EE. Thanks to this additional blob detection step, the grasp planning step could be restarted and a failed grasp could be prevented.

With this in mind, the data for the top grasp was re-plotted without outliers (see Figure 5.6). It was seen that these error values were very similar to the error values found in the side grasp tests (see Table 5.8 and Table 5.9). The grasp centroid error decreased from 5.27 cm with outliers to 2.29 cm without outliers for the 2D case. Therefore, like for the side grasp, it is reasonable to assume that the opening width of the EE should be roughly 6 cm greater than the cylinder's diameter for a grasp to succeed. Additionally, as discussed in Section 5.1.1, the grasp height could be adapted by the constant grasp height offset coming from the fact that the depth sensor only calculates the height using half of the object (see Appendix A.2 for the plots).



(a) 2D error development for top grasp with increase in location step for noisy vs. perfect odometry data.

(b) 3D error development for top grasp with increase in location step for noisy vs. perfect odometry data.

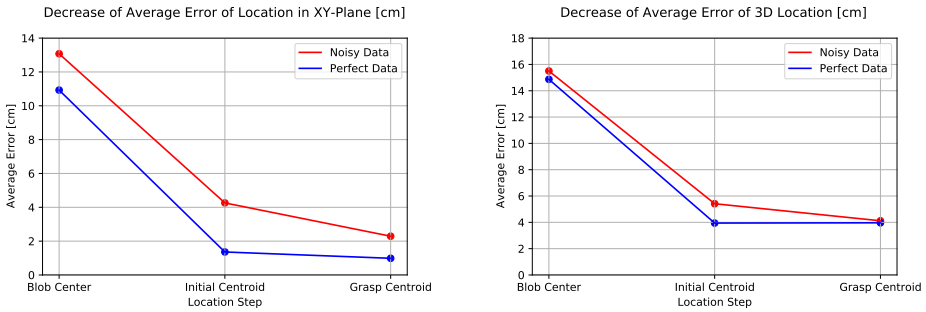
Figure 5.5: Euclidean error development for top grasp with noisy vs. perfect odometry data.

Data	Blob Center Error	Initial Centroid Error	Grasp Centroid Error
Perfect	10.93cm	1.36cm	0.99cm
Noisy	11.79cm	5.16cm	5.27cm
Δ	+7.9%	+279.4%	+432.3%

Table 5.6: 2D euclidean error development for top grasp with noisy vs. perfect odometry data.

Data	Blob Center Error	Initial Centroid Error	Grasp Centroid Error
Perfect	14.87cm	3.95cm	3.97cm
Noisy	13.97cm	6.20cm	6.75cm
Δ	-6.1%	+57.0%	+70.0%

Table 5.7: 3D euclidean error development for top grasp with noisy vs. perfect odometry data.



(a) 2D error development for top grasp with increase in location step for noisy vs. perfect odometry data.

(b) 3D error development for top grasp with increase in location step for noisy vs. perfect odometry data.

Figure 5.6: Euclidean error development for top grasp with noisy vs. perfect odometry data without outliers.

Data	Blob Center Error	Initial Centroid Error	Grasp Centroid Error
Perfect	10.93cm	1.36cm	0.99cm
Noisy	13.08cm	4.26cm	2.29cm
Δ	+19.7%	+213.2%	+131.3%

Table 5.8: 2D euclidean error development for top grasp with noisy vs. perfect odometry data without outliers.

Data	Blob Center Error	Initial Centroid Error	Grasp Centroid Error
Perfect	14.87cm	3.95cm	3.97cm
Noisy	15.50	5.42cm	4.12cm
Δ	+4.2%	+37.2%	+3.8%

Table 5.9: 3D euclidean error development for top grasp with noisy vs. perfect odometry data without outliers.

The computational time and effort of the pipeline is another aspect which would need to be considered in more detail before using the pipeline on the real system. In

real life, namely, this factor plays a very important role. On one hand, flight time is restricted. On the other hand, the computational resources on the robot are limited. Therefore, having a pipeline that runs quickly and efficiently is essential, since this enables many grasps. To decrease the computational time of the pipeline, the software architecture could be improved by simplifying the communication between the state machine and the pipeline node (see Appendix A.1 for a diagram of the software architecture). Also, the cylinder segmentation step could be optimized as discussed in Section 5.1.2.

Furthermore, a real RGB-D camera would first have to be calibrated before being able to use the pipeline for the real system. Especially for the extrinsic calibration, the transformation from the camera to the world frame would need to be extended by first taking the transformation from the camera to the body, and secondly, from the body to the world frame. This additional transformation would increase the error of the camera calibration, since the calibration parameters can never be modelled perfectly.

In real flight tests, alternating lighting conditions would be present. This would make it necessary to adapt the HSV thresholds for the color segmentation. Another challenge of moving to the real system would include placing the sensors on the UAV such that the FOV is not obstructed by the gripper. The sensors should also be placed somewhere where vibrations aren't prevalent. Finally, a real sensor would provide the pipeline with noisy odometry data, which in return would decrease the performance of the pipeline. Therefore, filtering the sensor data before it enters the pipeline would be necessary.

Chapter 6

Conclusion

6.1 Summary

The bachelor thesis *Object Detection and Grasp Planning with an Omnidirectional Aerial Manipulator* aimed to increase the autonomy of *PrisMAV*, the aerial manipulator developed during the focus project *Griffin*. This was done by designing an object detection and grasp planning pipeline. A state machine was implemented to autonomously run the pipeline. The pipeline was tested in simulation.

The pipeline consisted of various steps. It started by detecting an object with the means of a scan routine and color segmentation algorithm. Once detected, the object was located in 3D space and the UAV approached the object. Next, the circularity of the object was analyzed to see whether the object was graspable from the top or the side. For a top grasp, the final pose and grasp point were evaluated and the grasp was executed. A plane segmentation algorithm was used to verify that an object which wasn't graspable from the top was graspable from the side. A side grasp then still required *PrisMAV* to transition into the horizontal flight mode before the final grasp point could be calculated and the object could be grasped.

Simulation tests showed that with each subsequent step in the pipeline, the location error of the object decreased. The 3D error could be reduced by subtracting the constant offset in the height estimation which came from the fact that the depth sensor only saw the front half of the object. Simulation with noisy data showed that the errors increased, but stayed within an acceptable range given that an outlier rejection algorithm is added to the pipeline. The cylinder segmentation step showed good results but had a high computational effort. Conversely, the plane segmentation could still show better results whilst having a low computational effort.

To conclude, the thesis goals were achieved. The pipeline was namely able to autonomously detect and locate red, cylindrical objects. Subsequently, the graspability of the target object was determined. This allowed *PrisMAV* to approach the target object with an optimal pose and in the adequate flight mode. Finally, the optimal grasp point was calculated.

6.2 Outlook

For aerial manipulation to gain a foothold in industry and find commercial use, more work and research is necessary. Focus project *Griffin* created a novel aerial manipulator concept. It encompasses an omnidirectional flying platform, to be able to manipulate from any arbitrary pose, and a robotic arm with adaptive fingers

as an end-effector, to compensate for the positional error of the flying platform and flexibly grasp objects. Judging by the tests made in simulation, the proposed object detection and grasp planning pipeline successfully increases the autonomy of the omnidirectional aerial manipulator and enables an autonomous grasp mission from take-off to landing. Nevertheless, the pipeline still would need to be tested in real life on the real system since a simulation never perfectly depicts reality. In particular, the influence of fluctuating light conditions was not present in simulation. It would be interesting to see how the pipeline performs in the real world. It would also show how representative the simulation results were. Furthermore, the pipeline would need to be extended to deal with objects other than red cylinders. The pipeline could be extended to grasp objects of undefined shapes as the adaptive fingers would also allow to grasp objects that aren't cylindrical. It also could be interesting to deal with objects of various colors. Once an application were specified, the pipeline could be redesigned to meet the requirements of the use-case. Other features, such as obstacle avoidance, would also be required before *PrisMAV* could be used for a commercial application.

Bibliography

[1] F. Bühlmann, J. Näf, M. Rubio, D. Gisler, M. Hüsser, N. Ospelt, M. Inauen, and P. Brigger, "Griffin Aerial Manipulation Final Report."

[2] Isaac Ronald Ward, H. Laga, and M. Bennamoun, "RGB-D image-based Object Detection: from Traditional Methods to Deep Learning Techniques," in *RGB-D Image Analysis and Processing*.

[3] R. Bähnemann, M. Pantic, M. Popović, D. Schindler, M. Tranzatto, M. Kamel, M. Grimm, J. Widauer, R. Siegwart, and J. Nieto, "The ETH-MAV Team in the MBZ International Robotics Challenge," Oct. 2018.

[4] P. Ramon-Soria, B. C. Arrue, and A. Ollero, "Grasp Planning and Visual Servoing for an Outdoors Aerial Dual Manipulator," Nov. 2019.

[5] P. Z. David Dubach, "Object Recognition and Visual Correction for a Block Stacking Application."

[6] C. Richter, A. Bry, and N. Roy, "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments," in *Robotics Research*, M. Inaba and P. Corke, Eds. Cham: Springer International Publishing, 2016, vol. 114, pp. 649–666, series Title: Springer Tracts in Advanced Robotics.

[7] D. Bora, A. Gupta, and F. Khan, "Comparing the performance of l*a*b* and hsv color spaces with respect to color image segmentation," 06 2015.

[8] Wikipedia contributors, "Hsl and hsv — Wikipedia, the free encyclopedia," 2004, [Online; accessed 22-March-2010].

[9] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[10] R. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," 05 2011, p. *.

[11] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[12] J. Bohren, "Smach," https://github.com/ros/executive_smach, 2010.

[13] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system."

Appendix A

Appendix

A.1 Software Implementation

The pipeline presented in this report was implemented using several ROS [13] nodes. In Figure A.1, the implemented nodes for running the pipeline can be seen. The *pipeline_node* contains all the algorithms presented in Chapter 3. This node was implemented using C++ and the OpenCV [11] and PCL [10] libraries. The *state_machine* was written in python using the SMACH [12] library. It governed when the pipeline needed to progress from one step to another, and made sure the *pipeline_node* ran the right algorithm at the right moment. The *pipeline_node* sent the relevant data to the *state_machine*, which then sent the desired setpoints to the *flight_controller_node* of *PrisMAV* via the *trajectory_planner*. The red part of Figure A.1 indicates, that the feedback from *PrisMAV* was just simulated sensor data. For the real life application, this red part would have to be exchanged, and the measures discussed in Section 5.2 would need to be considered.

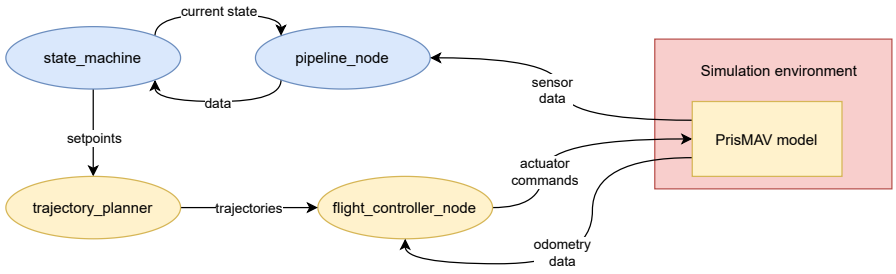
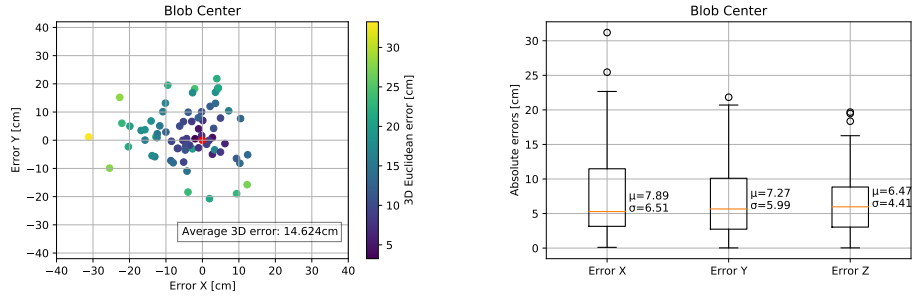


Figure A.1: Schematic of the implemented ROS nodes. The blue nodes represent the nodes implemented for the scope of this thesis, the yellow nodes (and the PrisMAV model) were implemented during the focus project *Griffin*. [1]

A.2 Simulation with Noisy Odometry Data - Plots

The box plots containing the mean and standard deviations of the x, y and z location errors for the tests done in the simulation with noisy odometry data are shown below. Each plot is accompanied with a scatter plot containing either the xy- or xz-plane error information. The scatter plot is extended with a color bar. The color bar indicates the effective 3D euclidean error between the estimated location and

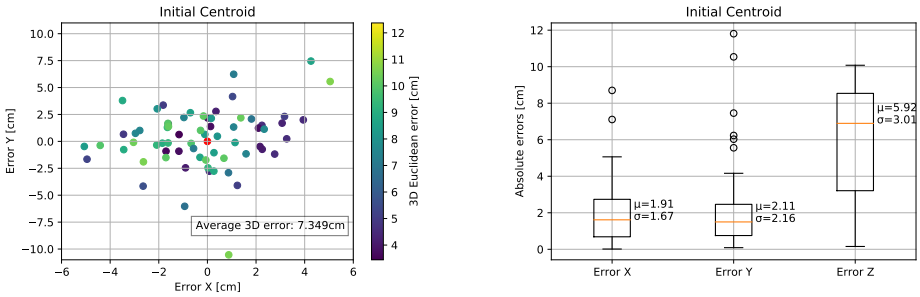
the ground truth location of the object. Like for the tests without noisy odometry data, the blob center and initial centroid plots were done by calculating the locations for objects spawned both on the floor as well as on the shelf. Here, the xy-plane was of interest, since the aim of these two steps was to center the cylinder in the image frame (see Figure A.2 and Figure A.3). For a top grasp, the grasp height had a static error of 3.24 cm with a small standard deviation as seen in Figure A.5b. This static error could therefore be subtracted from the grasp centroid estimate for a grasp. For a side grasp, the grasp height had a mean static error of 3.58 cm, however, with a substantial standard deviation as shown in Figure A.4b. Tests with the real system would show if subtracting the average error would yield a high enough grasp success rate.



(a) Location error in xy-plane.

(b) Box plot with absolute errors in x, y and z location.

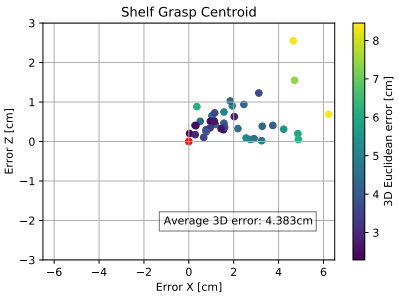
Figure A.2: Blob center location.



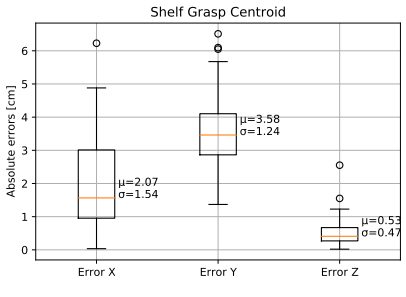
(a) Location error in xy-plane.

(b) Box plot with absolute errors in x, y and z location.

Figure A.3: Initial centroid location.

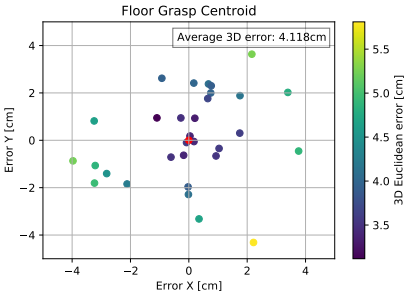


(a) Location error in xz-plane.

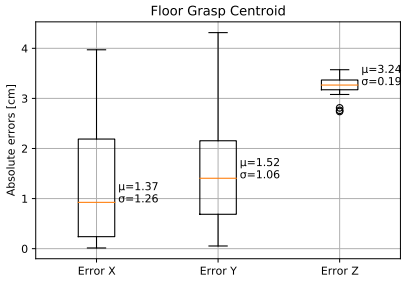


(b) Box plot with absolute errors in x, y and z location.

Figure A.4: Grasp centroid location for side grasp.

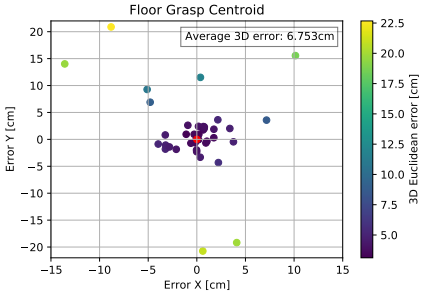


(a) Location error in xy-plane.

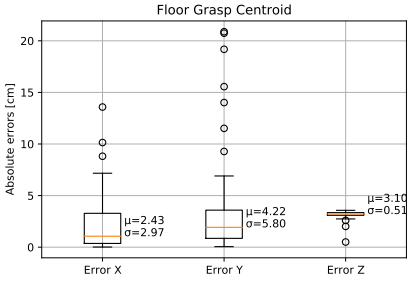


(b) Box plot with absolute errors in x, y and z location.

Figure A.5: Grasp centroid location for top grasp without outliers.



(a) Location error in xy-plane.



(b) Box plot with absolute errors in x, y and z location.

Figure A.6: Grasp centroid location for top grasp with outliers.