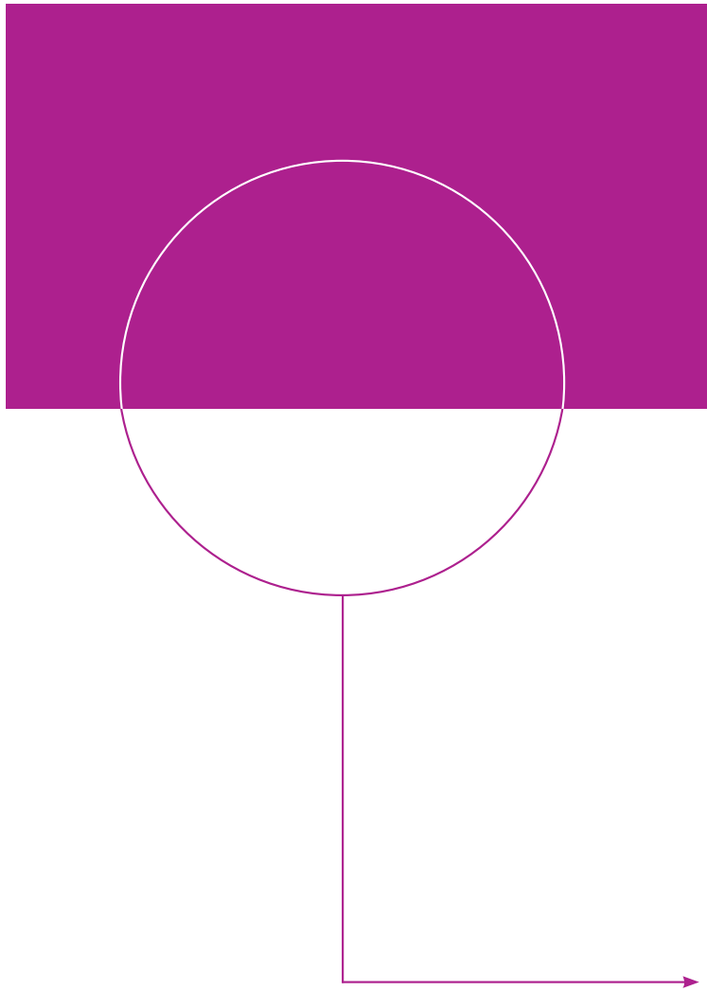


Master's thesis

2019

Vilde Benoni Gjærum

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

Vilde Benoni Gjærum

Autonomous navigation along power lines using monocular camera

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Autonomous navigation along power lines using monocular camera

Cybernetics and robotics
Submission date: June 2019
Supervisor: Rune Storvold

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

This report addresses the problem of autonomously detecting and following power lines in rural areas using a UAV as a step towards achieving autonomous inspection of power lines, power masts, and their components. Such operations are important to utility companies, and the lack of it can potentially be economically devastating for the company. First, a literature review of relevant computer vision techniques and of relevant previous work is conducted. Two new methods for detecting the power lines are proposed in this report, namely AMA RANSAC and AMA RANSAC + HTIVS. AMA RANSAC detects the power lines using computer vision techniques such as the Canny edge detector, the progressive probabilistic Hough line transform, and the RANSAC algorithm combined with apriori knowledge about power lines. AMA RANSAC + HTIVS combines AMA RANSAC with the voting scheme originally proposed in the project thesis this report is a continuation of. This report demonstrates that the power lines position, orientation, and distance from the UAV can be determined using only a monocular camera. Additionally, a simple power mast detector is proposed. Finally, it is shown how to extract a trajectory from the detected power lines in the image that the UAV can navigate by.

Abstrakt

Norwegian translation of the abstract

Denne rapporten tar for seg problemet med autonom deteksjon og navigasjon langs kraftlinjer i landlige omgivelser ved bruk av UAV, som et steg mot å oppnå fullkommen autonomitet i arbeidet med inspeksjon av kraftlinjer, kraftmaster og deres komponenter. Slik inspeksjon er av særdeles viktighet for energiselskapene, da mangelen på denne kan føre til ødeleggende økonomiske konsekvenser for selskapet. Rapporten begynner med en gjen-

nomgang av relevant litteratur på datasynteknikker og tidligere arbeid gjort på området autonom kraftlinjeinspeksjon. I denne rapporten blir to nye algoritmer for deteksjon av kraftlinjer kalt AMA RANSAC og AMA RANSAC + HTIVS presentert. AMA RANSAC algoritmen detekterer kraftlinjene ved å ta i bruk kjente datasynteknikker som Canny kantdetektor, *the progressive probabilistic Hough line transform* og RANSAC kombinert med kunnskapen vi allerede har om kraftlinjers utforming. AMA RANSAC + HTIVS algoritmen kombinerer AMA RANSAC med stemmegivnings-systemet som ble foreslått i prosjektoppgaven denne masteroppgaven bygger videre på. I sin helhet demonstrerer denne rapporten at kraftlinjenes posisjon, orientasjon og distanse fra UAV kan fastslåes med et monokulært kamera. Avslutningsvis demonstreres det hvordan en bane UAVen kan navigere etter kan kalkuleres ut i fra resultatene til AMA RANSAC og AMA RANSAC + HTIVS algoritmene.

Preface

This thesis was conducted during the Spring of 2019, and it is the final step of the M.Sc degree in Cybernetics and Robotics at the Norwegian University of Science and Technology(Trondheim, Norway). This master's thesis is a continuation of the project thesis, [12], written during the Autumn of 2018. Section 1.1, Chapter 2, Section 3.6, and Section 5.2 is based upon the work done in [12].

I would like to express my gratitude to my supervisor Rune Storvold for help and guidance throughout the thesis. Additionally, I want to thank Stian Solbø from NORCE research institute for advice on the technical work, Anastasios Lekkas from NTNU for all the help with the scientific writing. Lastly, I want to thank Karl Ylvisaker for all the support and motivation throughout this thesis.

The work was done on a computer with an Intel(R) Core(TM) i7-8700 processor with 3.20 GHz, and 32 GB RAM running Windows 10. A 5-minute long aerial video from a monocular camera of power lines and power masts was provided by NORCE research institute. Stereo data was unfortunately not available. All code is written in Python, and the two libraries OpenCV and numpy is frequently used.

All relevant source code mentioned in this report can be found at the authors personal github repository, reachable at the following web address:

- `https://github.com/vildursn/Master-thesis`

Additionally, a video of the five different methods evaluated in Section 3.9 can be found on the following web address:

- `https://youtu.be/IQcG-re02Dg`

Vilde Benoni Gjærum

Trondheim, June 3, 2019

Contents

List of Tables

List of Figures

Abbreviations

UAV	=	Unmanned Aerial Vehicle
fps	=	frames per second
RGB	=	Red Green Blue
pdf	=	probability density function
pmf	=	Probability mass function
RANSAC	=	Random Sample Consensus
DBSCAN	=	Density-Based Spatial Clustering of Applications with Noise
PPHLT	=	Progressive Probabilistic Hough Line Transform
AMA RANSAC	=	RANSAC with adaptive maximum number of attempts
HTIVS	=	Hough transform-inspired voting scheme
K-means + HTIVS	=	K-means combined with HTIVS
AMA RANSAC + HTIVS	=	AMA RANSAC combined with HTIVS
GPU	=	Graphics Processing Unit
SO(3)	=	Special Orthogonal Group in R^3
SE(3)	=	Special Euclidean Group in R^3
ToF	=	Time-of-flight
LiDAR	=	Light Detection And Radar
RMS	=	Root Mean Square
IMU	=	Inertial Measurement Unit
DOF	=	Degrees of Freedom
SPMEC	=	Semi Patch Matching based on Epipolar Constraints

Chapter 1

Introduction

In this report, we address the problem of autonomously detecting and navigating along power lines with an unmanned aerial vehicle(UAV) based on optical images. The need for inspection and maintenance of power lines, power masts, and their components warrants a solution to this. In Section 1.1, the motivation for this report and reasoning for using multirotor UAVs, power line-detection based methods, and stereo camera as the data source will be presented. In Section 1.2 the objectives of the thesis are given and in Section 1.3 the approach of the problem is presented. In Section 1.4 the structure of the report is presented.

1.1 Motivation & background

Access to power and electricity is essential all over the world today, including businesses, public areas, and households. Power blackouts can cause enormous economic losses, like property damage and business interruption. Even short and relatively small power blackouts are economically costly and cause big infrastructural problems. Typically, power blackouts are caused by one or several factors such as storms, floods, lightning, aging infrastructures, snow or ice loads, and extreme temperatures[8]. In addition to that, since most power grids are interconnected, a domino effect can be triggered which can cause large-scale blackouts. Not to mention that power is perishable and thus wasted during a

blackout. All things considered, it is clear that preventing blackouts is crucial. Inspection and maintenance of power lines and power masts are one of the main ways to prevent power blackouts. In Norway, the utility company owning the power line infrastructure are held financially responsible for losses businesses and private customers may have as a direct cause of a blackout caused by a failure in their infrastructure. Compensations must be paid to their customers according to a regulation called KILE[2]. KILE costs the utility companies around 800 million NOK per year [1]. This is the only real threat to the survival of the company. Thus, power line and power mast maintenance is crucial to the companies.

An extensive literature review on autonomous power line inspection is given in [18]. Four types of power line inspection tasks were identified, namely mapping and inspection of power line components, vegetation encroachment monitoring, ice detection, and measurement, and disaster monitoring. Monitoring of vegetation encroachment is most often done by inspecting satellite images. This report will focus on navigation between the power masts along the power lines, facilitating for inspection of the lines, masts, and components. The authors in [18] divide the problem into three sub-problems which will be discussed further:

1. Inspection method
2. Data source
3. Data analyzing method

Inspection method

The comparison between inspection methods includes foot patrols, helicopter assisted inspection, automated helicopter assisted inspection, climbing robots and UAVs. UAV inspection excels in areas where traditional methods often fall short. Especially regarding safety, cost, and efficiency. Foot patrols have a very high detection rate but are very time-demanding. Foot patrols also might be infeasible to perform in certain weather conditions or rough terrain. Helicopter assisted inspection is better in rough terrain and is, in general, faster than foot patrols, but it suffers from being less accurate and much more expensive.

The main difference between automated helicopter inspection and regular helicopter inspection is that computer vision techniques are used both when filming and evaluating the films. This makes the process faster. Additionally, foot patrols and helicopter inspections are more dangerous to humans than UAV inspections, while both helicopter inspection methods are riskier in regards to the power lines and power masts in comparison to UAVs. UAV inspections are cheaper than foot patrols and helicopter inspections. Also, UAVs can fly quite close to the power lines and thus getting better images and achieving higher accuracy than helicopters. Fixed-winged UAVs must fly at a higher speed than multirotor UAVs, and hence the multirotor UAVs are able to fly even closer and get higher quality data. In addition, multirotor UAVs have the possibility of hovering over objects that need to be more thoroughly inspected. UAVs are also significantly quieter than helicopters since they can be made much smaller and thus less thrust is needed, which is beneficial when doing inspections close to populated areas or areas where animals live. It is clear that there are several benefits to using multirotor UAVs. The main disadvantage of UAV inspection stems from the technological challenges related to them, such as shorter flight time due to the limited battery capacity and the need for intelligent control systems. Limited battery capacity also constraints the weight of the onboard hardware, which again places processing restrictions on the software. All things considered, a multirotor UAV is preferred. From now on, all mentions of UAV refers to a multirotor UAV. All of the arguments above are summarized in Table 1.1. The methods are rated between one and three, where one is the best and three is the worst, on different aspects of power line and power mast inspection. It should be noted that the ratings are in no way absolute or a result of quantitative evaluation, but rather the author's opinion after doing research and talking to people in the industry. The ratings are meant as a relative review of the different methods, to simplify and summarize the comparison of their strengths and weaknesses.

Data source

There are many different sensors to choose from when instrumenting a system. When it comes to positioning, GPS is the first one that comes to mind. GPS can be used as guidance for navigation along power lines, but not independently. This is due to the power masts not

Inspection Method	Foot patrol	Helicopter	Automated helicopter	Climbing robot	Multirotor UAV	Fixed wing UAV
Human safety	3	2	2	**1**	**1**	**1**
Equipment safety	**1**	3	3	**1**	**1**	**1**
Rough terrain	3	**1**	**1**	**1**	**1**	**1**
Cost	3	3	3	**1**	**1**	**1**
Time	3	2	**1**	3	**1**	**1**
Hovering ability	-	**1**	**1**	-	**1**	3
Power mast navigation	**1**	2	2	2	**1**	3
Detection rate	**1**	2	2	2	**1**	2
Noise	**1**	3	3	**1**	**1**	**1**

Table 1.1: Relative comparison of the different inspection methods.

being mapped accurately enough, and the power lines height between consecutive masts is not mapped at all. Thus, relying on GPS alone becomes infeasible, and an alternative data source is required. Since we want to fly at a certain height over the power lines, we need a sensor that can tell us how far from the power line the UAV is. There are three main types of depth sensors, namely stereo triangulation, time-of-flight(ToF) and structured light. Stereo triangulation, also called passive stereo, is two cameras placed with a known distance, called the baseline, between them. Points observed in both cameras are then matched, and their depth is decided by the use of epipolar geometry which is explained later, in Section 5.4.1[21, Chapter 11]. Both structured light and ToF methods are active sensors. Active sensors estimate depth by emitting energy and measuring how the environment affects it. Structured light emits light in a specific, known structure and uses the deformation of this light pattern to estimate the depth. ToF, such as LiDAR(light detection and ranging), sends out a laser signal and uses the time it takes for the signal to get reflected to calculate the depth. Passive stereo methods mainly capture the depth at the boundaries of objects, whereas the two other methods get more depth information also inside of the boundaries of the objects. Power lines are however very thin, and will not occupy more than a few pixels, so depth values of the boundaries of objects are all that is needed. Furthermore, it is also preferable that the sensors can be used for several applications since this will result in fewer sensor and thus less weight, which is beneficial regarding energy consumption. Since it can be useful to have color information to detect ice or snow on the power lines, rotten or rusted parts on the power masts, or other faults on the components, the preferred choice for the data source is an optical camera. Passive stereo is also more robust against sub-optimal weather conditions than the active methods since both snow and rain reflects the energy emitted. Even though the method proposed in this report only requires a monocular camera, a stereo camera is suggested used as the main data source since it is likely that a stereo camera can negate the need for other sensors during other aspects of the inspection. More specifically, a ZED[1] camera from Stereolabs is suggested used as the data source. It is a complete packaged system which integrates stereo RGB-D cameras with the OpenCV software library. It has range from 0.5 meters up to 20 meters with up to 100 frames per second in high resolution. It is the preferred choice

[1] `https://www.stereolabs.com/zed/`

because it is low-cost, passive, and outputs both a depth map and a color image. It is light, which allows for additional payload, and it has 110 °field of view.

Data analyzing method

The three most common approaches for UAV navigation between consecutive power masts are GPS waypoints-based, power mast detection-based, and power line detection-based. The latter one will be used to calculate a trajectory for the UAV to navigate after since that it is the only one of these alternatives that are able to accurately estimate the distance to the power lines, enabling the UAV to fly at a constant distance. To detect power lines from optical images different computer vision techniques must be used, and the development of such an algorithm is the main contribution of this report. The two new methods developed through the work of this thesis are called AMA RANSAC and AMA RANSAC + HTIVS. The AMA RANSAC algorithm is composed out of well-known methods such as the Canny edge detector, the Hough line transform, and RANSAC combined with apriori knowledge about the power lines. AMA RANSAC + HTIVS is composed out of AMA RANSAC and a modified version of the voting scheme proposed in [12]. The methods developed for this report are light-weight and easily implemented. These methods are presented in Section 3.5, Section 3.8 and Section 5.5. As will be shown, the navigation along the power lines can be achieved using only a monocular camera. Since it only requires a monocular camera it is well suited for light-weight and readily available UAVs.

1.1.1 Previous work

Since the maintenance of the power lines and power masts is paramount to the utility companies, there currently is and have been done a lot of work on achieving autonomous drones capable of conducting the inspections. Both public and private companies work on achieving this. Sevendof[2] and Versor[3] are examples of start-up companies in Norway. Sevendof makes autonomous drones for power line inspection that utility companies can rent,

[2]https://www.sevendof.com/
[3]https://www.versor.io/

while Versor makes software for autonomous drone operations. SINTEF[4] and NORCE[5] are examples of research institutes in Norway also working on autonomous power line inspection. Examples of other prominent companies are Statnett[6], Nordic Unmanned[7], and KVS technologies[8].

This master's thesis is a continuation of the work done in the project thesis, [12], where a literature review of relevant computer vision techniques and on relevant previous work on autonomous navigation and inspection of power lines and power masts was conducted. Additionally, a new method for detecting the power lines in an image was proposed, and estimation of the distance from the UAV to the power lines was suggested done by RANSAC. In [18], an extensive literature review on autonomous vision-based power line inspection was conducted, with a focus on how deep learning can be useful for this application. In [15] a novel method for power line detection from aerial images is proposed. A pulse coupled neural network is used to refine the image before using a knowledge-based line clustering method in Hough space to distinguish between power lines and other detected lines in the image. This method was proved effective on real image data. In [23], power lines were detected from images taken by a UAV by using epipolar constraints and a dense matching method. The root mean square (RMS) was just above 0.2 meters in both estimations of the position and height of the power lines. Some of the methods proposed in previous work were attempted replicated, either in their whole or parts. However, the attempts were unsuccessful in implementation or achieving similar results. In light of this, the focus of the report shifted towards making use of available open source resources, rather than proprietary ones. This thesis continues to address the challenges of autonomous navigation along power lines, as a step towards achieving autonomous power line and power mast inspection, by proposing two new methods to do so using only a monocular camera. As previously mentioned, a stereo camera is still suggested as the main sensor since closer navigation and component-wise inspection around the power

[4] https://www.sintef.no/en/status-inspection-of-power-lines/
[5] https://www.norceresearch.no/
[6] https://www.statnett.no/
[7] http://nordicunmanned.com/
[8] https://kvstech.no/

mast most likely will require more accurate depth information than monocular vision can give.

1.2 Objectives

The main objectives of this thesis are as follows:

- Literature review on relevant computer vision techniques for processing images to detect the presence and orientation of power lines.
- Literature review on relevant previous work.
- Developing a method for detecting the presence and orientation of power lines.
- Developing a method for extracting a trajectory from the detected power lines.

1.3 Approach

The method presented was developed on a machine with an Intel(R) Core(TM) i7-8700 processor with 3.20 GHz, and 32 GB RAM running Windows 10. All methods developed during this thesis can be found on the authors GitHub account[9], and it is written in python. Python is a very popular high-level programming language. It is chosen because of its simplicity and that it has broad compatibility with existing software packages, such as numpy and OpenCV which is used in this thesis. Python also allows GPU accelerated programming with CUDA, which the ZED camera uses. Python can be installed on Windows, Mac OS, Linux, Android, and iOS. The methods are developed and tested on a video of power lines and power masts taken from a UAV. The video has 30 fps, 1920x1080 resolution and is 5 minutes long. The video is from Nordmarka in Nordland, Norway. The results of five of the methods run on this dataset is uploaded and can be watched on YouTube[10].

The first objective is addressed in Chapter 2, where relevant computer vision techniques

[9] https://github.com/vildursn/Master-thesis
[10] https://youtu.be/IQcG-re02Dg

are reviewed. The second objective is addressed in Chapter 2 and Chapter 3, where methods from previous work are discussed. Several methods for detecting power lines in an RGB image is presented and evaluated in Chapter 3. Chapter 5 presents how to extract a trajectory based on the detected power lines. This trajectory is the output the UAV will use to navigate after and is the last step of achieving autonomous navigation along the power lines. Additionally, a simple power mast detector is presented in Chapter 4.

1.4 Structure of the report

The work done during this thesis is a continuation of the work done in [12], which concerns a theoretical approach to the problem of autonomous navigation along power lines. More specifically, Section 1.1, Chapter 2, Section 3.6, and Section 5.2 is based upon the work done in that project thesis. Furthermore, the report is structured as follows. In Chapter 2, the theory about different computer vision techniques is presented. In Chapter 3, how to detect power lines in an image is discussed and evaluated. How to detect power masts is presented in Chapter 4. Chapter 5 presents how to extract the depth values of the power lines and how to transform the detected power lines to a trajectory that the UAV can follow. Lastly, the recommendation for further work, summary, and conclusion of the report are given in Chapter 6.

Chapter 2

Theoretical background

This chapter is based upon the work done in[12, Chapter 2]. In this chapter, relevant computer vision techniques will be discussed. Image preprocessing is a fundamental and essential part of achieving good results when working with computer vision, and will be discussed in Section 2.1. In Section 2.2, an edge detection method will be presented. A method for refining the detected edges into detected lines is presented in Section 2.3. Section 2.4 contains a discussion on the potential benefits to be gained from combining information from the color image with depth information. Finally, the chapter is summarized with concluding remarks in Section 2.5.

2.1 Image preprocessing

Cameras are made to serve the human eye, but this does not mean that it serves the machine eye in the same way; thus we need to process an image to make it as presentable as possible for the computer. Image preprocessing is the preparations we do to make the information in the image as clear as possible before we start making use it. As such, the choice of preprocessing methods significantly affects the performance of an algorithm - both in terms of results and run-time. In this section, we will go through some of the most common ways to do this.

2.1.1 Image downsampling

The dataset used in this thesis has frames with 1920x1080 pixels as resolution, and 30 frames per second(fps). Even though higher resolution brings more information, it can also be impractical in the sense that it requires substantially more memory and processing time. Also, not all information is useful. Image downsampling, also called decimation, can be used to reduce the frame size[21, Chapter 3.5]. It could be tempting to just remove some pixels, but this can cause a problem called *aliasing*. To illustrate this, we use the example of an image depicting a chessboard. Reducing the image size by simply removing every other pixel, brings the risk of only removing only the black squares and the resulting downsampled image being all white, or opposite. The Gaussian pyramid method can be used for downsampling while avoiding aliasing.

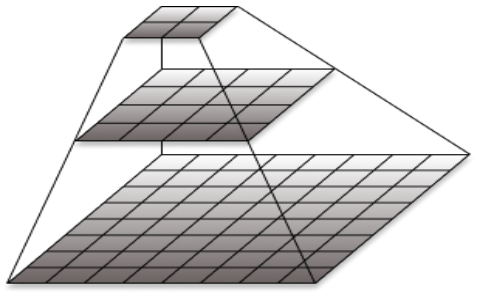


Figure 2.1: Downsampling pyramid.

In Gaussian pyramids, we alternate between blurring and downsampling. The blurring preserves the information by making sure it is not all contained in a single pixel that might be cut during the downsampling. OpenCV's implementation uses a Gaussian filter for blurring between each layer before removing every even-numbered row and column[4]. The downsampling is illustrated in Figure 2.1, and the blurring happens between each layer. If downsampling is needed, but the power lines are blurred out by the Gaussian filter, a Bilateral filter can be tried instead. Both the Gaussian filter and the Bilateral filter is further explained in Section 2.1.2.

2.1.2 Noise removal

All electrical devices experience noise to some degree, and digital cameras are no exception. In general, noise is more prominent in the high-frequency domain. One of the most common ways to detect edges in an image is to make use of the gradients of the image, such as in the Sobel edge filter which is explained in 2.2.1. Edge detection methods that use gradients are typically sensitive to noise, and can in the worst case actually end up amplifying the noise. To avoid this, it is important to perform noise removal on the image before attempting to detect edges. Image filtering enables the application of different types of effects to the images, such as blurring and noise removal. Image filtering is done by either a linear or a non-linear filter. Linear filtering is done by convoluting a kernel matrix with the image pixels, as expressed in Equation 2.1[21, Chapter 3.2].

$$I_{filtered}(i,j) = \sum_k \sum_l I(i+k, j+l)h(k,l) \tag{2.1}$$

The parameters *i* and *j* represents the coordinate of the pixel being changed. The variable $I_{filtered}(i,j)$ represents the pixels value after the filter has been applied, and I(i,j) represents the pixels original value. The function h(k,l) represents the weighting the neighboring pixels have on each other. In other words, the function h(k,l) represents the elements of the kernel matrix. According to [5], the most commonly used filters are: normalized box filter, the Gaussian filter, the median filter, and the bilateral filter. These filters will now be explained and discussed.

Normalized box filter

$$h(k,l) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.2}$$

The normalized box filter is by far the simplest one. A pixel gets the average intensity value of some neighborhood. The bigger the neighborhood the more blurred the image gets. This is a linear filter. The 3x3 normalized box filter kernel is presented in Equation 2.2

Median filter

The median filter replaces each pixels value with the median value of some neighborhood around it, as shown in Equation 2.3[21, Chapter 3.3]. This filter suppresses *salt-and-pepper noise*, which sometimes occurs in digital images. The median filter is a non-linear filter.

$$I_{filtered}(i,j) = med[I(i+k_i, j+k_j)] \quad \forall k_i, k_j \in [0, N] \tag{2.3}$$

The parameter N determines the size of the neighborhood.

Gaussian filter

The Gaussian filter is a linear filter and uses Equation 2.4 for weighting the neighboring pixels.

$$h(x,y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}} \tag{2.4}$$

The parameter A is a constant. The two variables μ_x and μ_y represents the position of the center pixel to be changed, and thus the closest pixels are weighted the most. The weighting decreases as the distance from the neighboring pixel and the center pixel increases. The parameter σ decides how much the neighboring pixels will be weighted, the higher the value, the more the neighbors are weighted, and hence the image is blurred more.

Bilateral filter

The bilateral filter addresses the problem that sometimes noise removing filters also removes the edges that we are interested in detecting[21, Chapter 3.3]. The bilateral filter is similar to the Gaussian filter in the way that the center pixel is weighted the heaviest and that this weight decreases further away from the center. It is, however, different from the Gaussian filter in the way that it also weighs pixels based on the gradient of the image in that location. That is, the bilateral filter blurs less if the local intensity gradient is high. This intensity variation can both be in term of brightness and depth-values, so this could be a way of combining the two images as will be discussed in Section 2.4. The bilateral

filter uses Equation 2.5 and Equation 2.6.

$$I_{filtered}(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)} \tag{2.5}$$

$$w(i,j,k,l) = \exp\left(\frac{-(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\| I(i,j) - I(k,l) \|^2}{2\sigma_g^2}\right) \tag{2.6}$$

The variable $I_{filtered}(i,j)$ in Equation 2.5 represents the value of the pixel with coordinate (i,j) after the filter has been applied, and $I(i,j)$ represents the pixels original value. The variables k and l represents how far from the center pixel the neighboring pixels are. The function w(i,j,k,l) is the weighting function, and it is shown in Equation 2.6. The parameter σ_d defines the weight of neighboring pixels based on their distance to the center pixel, while the parameter σ_g defines how much the neighboring pixels should be weighted based on the local gradient.

Comparison of filters

The normalized box filter and the Gaussian filter is both linear and thus faster to compute than both the median and the bilateral filter which are both non-linear. Even though the Gaussian filter preserves edges better than the normalized box filter, it still blurs the edges which is unfortunate since they contain valuable information useful for detecting lines. The median filter should only be used if a lot of salt-and-pepper noise is expected. Non-linear filters take longer to compute than linear filters, but this increased run-time might be worth it if it leads to better performance.

As we will see in Section 2.2 and Section 2.3, the algorithms following the preprocessing methods are many times more computationally expensive compared to the filters presented here. If the noise removal performs well, it can lead to fewer irrelevant edges and lines detected going into a much more computationally expensive algorithm which thus can improve the run-time of the algorithms coming after the preprocessing. In this way, using a non-linear filter with better performance can bring down the total run-time.

2.1.3 Image histogram

Histograms can be used to represent information about the image not easily seen with the naked eye. Histograms keep count of how many pixels that have a certain intensity of color, or gray if grayscaled. Histograms thus hold information about the images' brightness, contrast, and intensity distribution. Figure 2.2 shows an image containing a power mast and power lines, and the corresponding histograms for each color channel. If the histogram is normalized it can be seen as an image *probability density function*(pdf)[14]. If the whole range of intensities is not used, details in the image can be hard to see and detect. Histogram stretching and equalization, as will be presented here, can make the power lines more prominent in the image.. When many of the pixels have similar intensities, the image

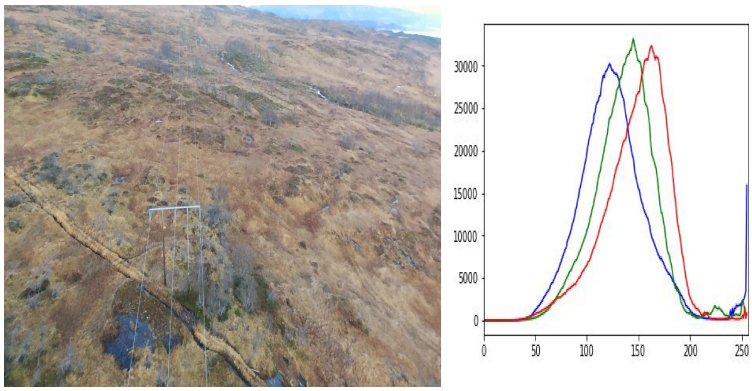


Figure 2.2: Image histogram

has low contrast. Redistribution of the pixel values can be done by histogram stretching or histogram equalization. Histogram stretching is illustrated in Figure 2.3. Stretching the histogram makes the image use the whole range of available intensity values. This is done using Equation 2.7.

$$I_{new}(i,j) = (I(i,j) - min(I))\frac{255}{max(I) - min(I)} \tag{2.7}$$

The parameter I(i,j) is the intensity value of the pixel in x-position i and y-position j. It is here assumed that there are 256 different values that the color channels can have, and min(I) and max(I) represents the minimum and maximum intensity values that is in the

image. Histogram equalization is a little more complex method than histogram stretch-

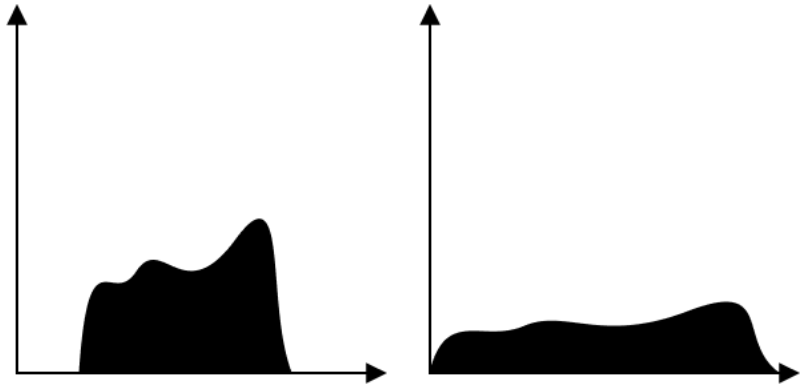


Figure 2.3: Histogram stretching

ing, which also aims to make the image use the whole intensity spectrum. Histogram equalization is done by the three following steps:

1. Normalize the histogram, so it becomes a *probability mass function*(pmf) by diving by the histogram's integral.

2. Calculate the cumulative distribution function for the pmf.

3. Normalize the cumulative distribution function. The maximum value must be 255.

2.1.4 Background removal

Background removal aims to remove or at least drastically blur parts of the image where there is no useful information. By doing this, we might be able to improve both performance and run-time. The background in images of power lines is typically nature and rural areas such as mountains, shorelines, forests or highlands, which also is the case for the dataset used in this thesis. Cities and inhabited areas can also be seen in the background, but since most of the intra-city power gird is beneath the ground, inspections of that part of the power grid must be conducted by other means than by UAV. Therefore, power line inspection in cities and inhabited areas will not be a primary concern in this

report. In this section conventional background removal methods will be presented, followed by how color channel manipulation can be used for this application.
Background removal consists of three main steps[20]:

- **Background model initialization -** In this part a model of the background is made. There are many ways to do this, for example, statistically or with neural networks.
- **Foreground detection -** The foreground of the scene is detected by subtracting the background model from the current frame.
- **Background model maintenance -** The background is not likely to stay the same over time, so the background model made at the initialization step should be updated as time goes by and new frames are seen. This update happens with respect to a learning rate based on how quickly the background is expected to change.

In [20], 29 background subtraction methods on 9 real videos and 20 synthetically made videos was conducted. The paper concluded with 5 methods they found being superior to the others, but even these methods had trouble with the most challenging situations such as a drastic change in brightness due to sunrise or weather changes such as fog or heavy rain. Background removal or background subtraction is most often used to find newly appearing or moving objects on a mostly static background, often for surveillance systems. This is also the case for the 5 methods chosen as the best ones in [20]. There is a significant difference between a static background that only changes by, for example, weather or lighting conditions, and a situation like the one this thesis aims to address, where it is primarily the background itself that is continuously changing. Therefore, none of the conventional background removal methods will be used. Additionally, most background removal algorithms aim to produce a binary map telling if a pixel is either a part of the background or the foreground. Such a binary map is not necessarily useful for this work, since it rather needs to identify important regions in the image.

Color channel manipulation

It would be useful to remove all parts of the background that does not bring any useful information about the power lines' presence or orientation in the image. During summer,

the background will, primarily be made up of the colors green, brown and grey. Because of this, it might be beneficial to increase the contrast in the blue color channel, considering that the power lines have high values in the blue color channel compared to the background. Changing of the contrast and brightness is done by the following Equation 2.8.

$$f(x) = \alpha x + \beta \tag{2.8}$$

The variable x in Equation 2.8 is the R, G or B value of pixel x and the function f(x) is the new R, G or B value of pixel x. If the parameter α is higher than 1, the contrast will be increased. If α is between 0 and 1 the contrast decreases. Higher values of the parameter β make the image brighter, while negative values of β make the image darker.
It is also common to look for edges in grayscaled images, but it is important to remember that information can get lost in the translation. This will happen, for example, when one red pixel with values (255,0,0) and a green pixel with values (0,255,0) is transformed to the same gray shade (255). Pixels like these are said to have the same *hue*.

We know that the lines we are trying to detect will take up only a small fraction of the image, and thus it will not noticeably contribute when calculating the average background color. Background and foreground separation could therefore be implemented by simple thresholding based on how different each pixel is from this average background color. This thresholding method will not suffice to accurately detect the power lines on its own, but it might contribute to preprocessing by enhancing important information in the image. This is because, thresholding is not robust against sudden changes in lighting conditions, shadows or just generally changing backgrounds.

2.2 Edge detection

Edge detection is a fundamental part of image processing. There are three different characteristics in an image to look at for finding edges, namely brightness, color, and texture. Which one gives the most information depends on the given task. In [21, Chapter 4.2] an edge is defined as a location of rapid intensity variation. This can be illustrated by using a topographic map as an example. In a topographic map, the edges would be where the

terrain abruptly changes the height, which in essence means where the gradient is large. Equation 2.9 shows the Jacobian, which is a common mathematical way of finding the slope of an image surface. Its two elements represent the gradient in the x- and y-direction.

$$\mathbf{J}(\mathbf{x}) = \nabla I(\mathbf{x}) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})(\mathbf{x}) \tag{2.9}$$

One can detect edges in both the color image and the depth image acquired by a stereo camera. In the RGB image, we can distinguish the power lines from the background because they have distinctively different colors. In the depth image we can find the power lines by that they will have lower depth value than their surroundings since there usually is nothing else as high, close to the power lines. The depth image can be represented as a grayscale image, and thus conventional edge detectors can be used.

2.2.1 Canny edge detector

The Canny Edge detector was developed in 1986[9]. Even though it is old, it is also known as the optimal edge detector[3]. The focus point of the Canny edge detector is to have a low error rate, good localization, and minimal response. This means it focuses on only detecting actual edges and having accurate localization of them, and additionally only detecting each edge once. The Canny edge detector consists of four steps:

1. Noise filtering
2. Finding the gradients of the image
3. Non-maximum suppression
4. Hysteresis

Not all software packages do the noise removal, but a 5x5 Gaussian filter is commonly applied. The Gaussian filter is the same as explained in Section 2.1.2.

Sobel edge filter

In OpenCV's implementation of the Canny edge detector, the linear Sobel edge filter is used to calculate an approximation of the gradients in the x- and y-direction of the image.

It convolves a kernel over the image in the same way as the linear filters presented in Section 2.1.2. Additionally, it calculates the magnitude of the gradient for each pixel. It is only an approximation since it does this calculation pixel-wise and not continuously. The kernels that are used are as defined in Equation 2.10 and Equation 2.11. Equation 2.10 is used for calculating gradients in the horizontal direction, while Equation 2.11 is used for calculating gradients in the vertical direction.

$$h(k,l) = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \tag{2.10}$$

$$h(k,l) = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \tag{2.11}$$

The method uses Equation 2.12 to calculate the magnitude of the gradient.

$$G = |G_x| + |G_y| \tag{2.12}$$

The variable G_x is the magnitude of the gradient in the x-direction, and G_y is the magnitude of the gradient in the y-direction. The variable G represents the total magnitude of the gradient at the pixel. How this may be used to merge the information from the color image and the depth image will be discussed in Section 2.4.

Non-maximum suppression

The Sobel edge filter does not find the edges accurately enough. By removing noisy edges and filtering out everything except the core of the edges, a much clearer edge map is obtained. This is done by non-maximum suppression. It looks at each pixel and compares it with its neighbors with a gradient direction perpendicular to the edge. If the pixel has the largest gradient it is kept as a part of the edge; otherwise, it is discarded.

Hysteresis

Hysteresis is the final step of the Canny edge detector. This step has two parameters that can be adjusted to make the detector work better for a specific task, namely the upper and the lower threshold. If the gradient magnitude of an edge is over the upper threshold, the edge is accepted as an edge. On the other hand, if it is lower than the lower threshold it is disregarded. If the gradient magnitude is in between the lower and the upper threshold, it is accepted only if one of its neighbors have gradient magnitude higher than the upper threshold. This is illustrated in Figure 2.4.

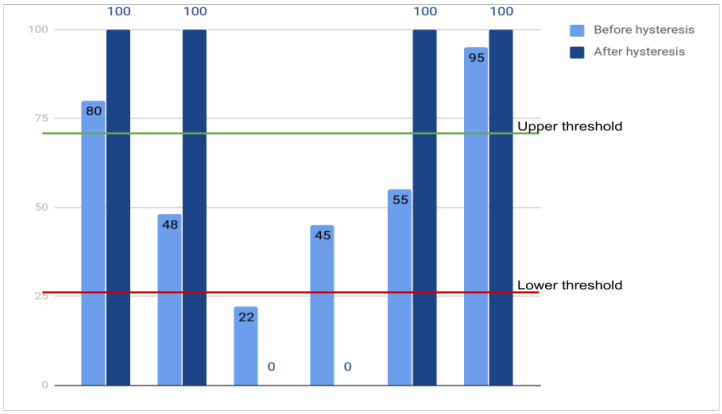


Figure 2.4: Hysteresis

In [16] some of the most common edge detection techniques were studied and compared. The authors concluded that even though the Canny edge detector is computationally heavier than gradient- and Laplacian-based algorithms, it outperforms them in almost all tasks.

2.3 Line detection

Even though edges contain a lot of useful information, what we are really looking for is lines. In this chapter, different line detection methods will be presented and discussed. In Section 2.3.1 the standard Hough line transform is discussed, and in Section 2.3.2 a

probabilistic Hough line transform is presented. RANSAC is presented in Section 2.3.3 and clustering and segmentation methods are presented in Section 2.3.4.

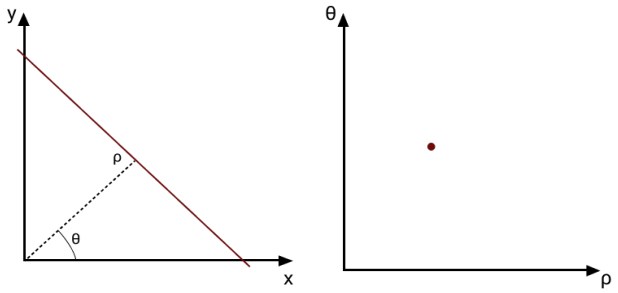
2.3.1 The standard Hough line transform

The Hough line transform identifies straight lines in a binary image[21, Chapter 4.3]. We know that lines easily can be represented with two parameters.

$$y = ax + b \tag{2.13}$$

Cartesian coordinates, as shown in Equation 2.13, is the most common representation for lines. The problem with cartesian coordinates is that it cannot represent vertical lines. Therefore, polar coordinates need to be utilized instead. Polar coordinates use the parameters ρ and θ. The parameter ρ is the length of the line going through the origin and intersecting the represented line perpendicularly, while θ is the angle between the line with length ρ and the x-axis. Equation 2.14 shows how the two coordinate system correlates.

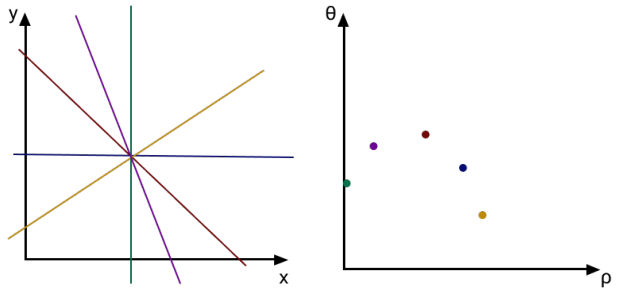
$$\rho = xcos(\theta) + ysin(\theta) \tag{2.14}$$



a) Cartesian space b) Hough space

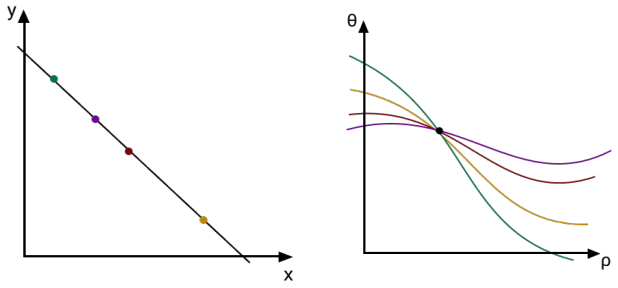
Figure 2.5: Line in image represented in Cartesian and Hough space

The Hough line transform uses the Hough space, which instead of having y and x on the axis, have the polar parameters ρ and θ, on its axes. A straight line in the cartesian space corresponds to one single point in the Hough space, as can be seen in Figure 2.5. If a set of lines that goes through one specific point in the Cartesian space is mapped



a) Cartesian space b) Hough space

Figure 2.6: Point in image in represented Cartesian and Hough space



a) Cartesian space b) Hough space

Figure 2.7: Points on line represented in Cartesian and Hough space

to the Hough space, the points will follow a sinusoidal pattern. This is shown in Figure 2.6. If all the infinitely many lines going through this point in the Cartesian space were mapped to the Hough space, a continuous sinusoid would appear. If a set of points in the Cartesian space, all laying on one straight line, were mapped to the Hough space, they would have different sinusoids that would meet in exactly one point. This is illustrated in Figure 2.7. This is where the Hough line transforms voting system comes in. The Hough line transform takes a binary image as its input. If a pixel contains an edge, it has value 1, if not it has value 0. Since a point in the Hough space corresponds to a straight line in the image space, each intersection can be counted as a vote for that particular line placement and orientation. The points in the Hough space that get more votes than a certain threshold is accepted as a detected line. Lowering this threshold will make the Hough line transform find more lines, increasing it will result in less detected lines.

There exist different types of voting systems for different versions of the Hough trans-

form:

- **One-to-many:** Each edge can vote for all plausible lines passing through it.
- **One-to-one:** Each edge can only vote for one plausible line passing through it.
- **One-to-some:** Each edge can vote for a pre-decided number of plausible lines passing through it.

2.3.2 The probabilistic Hough line transform

The Hough line transform can be slow, and thus not ideal for real-time applications. Luckily, there exist several probabilistic versions of it. The simplest version of a probabilistic Hough line transform randomly chooses a subset of edges which it performs a full Hough line transform on. OpenCV's implementation of a probabilistic Hough line transform is however based on [17] and is called the progressive probabilistic Hough line transform(PPHLT). It remedies the fact that if you have a fixed threshold for how many votes must be cast for a line to be accepted, long lines will need a smaller fraction of its points to be chosen compared to a shorter line. This gives shorter lines, which in our case is the less interesting ones, a higher probability of being accepted. The adaptive threshold that is utilized counteracts this effect by essentially checking if the votes for a line could be due to random noise. If that is not the case, the line is accepted. After a line is accepted, all points supporting that line retract their votes. Additionally, all remaining lines that have not yet voted and supports this accepted line also retract their votes. The algorithm has a one-to-one voting system and is an *anytime-algorithm*, which means that it can be stopped at any time and still yield useful results. The algorithm stops when there are no more votes left to cast. This is its only stopping criteria. This does not mean that all points have voted and a full Hough transform has been performed since many points have been removed before they got to vote.

2.3.3 RANSAC

RANSAC, short for Random Sample Consensus can be used detecting lines[21, Chapter 4.3]. It chooses two random edges and uses the line that intersects both of them as a

hypothesis. It then checks how many of the other edges support this hypothesis. The edges supporting the hypothesis will lie within some error bound ϵ, and are called inliers. The edges that do not fall inside this error bound are called outliers, as shown in Figure 2.8. If enough other edges support the hypothesis, this line is accepted. It is a relatively fast and robust method, which is resistant to noise since it handles outliers. It does, however, need some apriori parameters and it is only able to estimate one model at a time[14]. This is not a problem since the power lines can be represented by a single model. The white points in

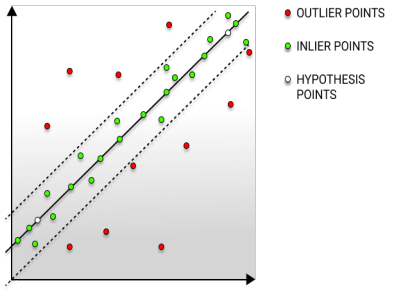


Figure 2.8: Illustration of RANSAC-based line detection.
The hypothesis points shows what two points the hypothesis line is extracted from. The inliers support this hypothesis since they are inside the error bound, while the outlier points are considered to be noise.

Figure 2.8 are the two randomly chosen points, and the line is the hypothesis made based on these two points. The green points are inliers, while the red points are outliers. The dotted line represents the area which is within the error bound ϵ.

2.3.4 Segmentation and clustering

The detected lines are often shorter than the line they belong to in the image. Therefore, it is desirable to connect the smaller line segments into one longer line which better represents the power lines. This can be done by either region growing, where one lets the lines grow together to one line, or by segmentation and clustering methods. In this section clustering and region growing methods are discussed. The common denominator for nearly all clustering methods are the following steps:

1. Start with choosing one or several starting points. They can be chosen manually, randomly or by a rule.

2. The region or cluster grows until it meets some stop criteria.

K-means clustering

K-means clustering is a popularly used clustering method due to its simplicity, but also because of its good results in many applications[14]. The steps of the method are as follows:

1. Pick K random points as the cluster centers. This initial selection can be made more effective by using, for example, statistical rules.

2. Assign all data points in the set to one cluster based on which cluster center it is closest to.

3. New cluster centers are calculated from the average position of the points belonging to each cluster.

4. Repeat step 2 and 3 until the cluster centers do not change anymore.

One of the disadvantages with K-means is that the numbers of clusters need to be known apriori. That does not have to be a problem since the number of power lines is known, but it would be advantageous if the method could be used on several types of power line- and power masts structures. K-means assumes circular shapes, but the power lines are in no way circular. This can be solved by doing the clustering in the Hough space. In [15] a knowledge-based line clustering method in Hough space was used to distinguish between power lines and other detected lines in the image. The major disadvantage with K-means is that it lacks robustness as it is sensitive to noise, since all points are weighted equally, and noise is expected.

DBSCAN

DBSCAN is short for Density-Based Spatial Clustering of Applications with Noise[11]. That the method is density based means that it puts points that are close enough together in the same clusters. What close means depends on what metric is used. DBSCAN differs from K-means in that while K-means puts a point in its closest cluster, DBSCAN only puts

a point in a cluster if they are close enough. DBSCAN labels points as either an outlier or a core point, and it is this outlier-label that makes it robust towards noise[14]. The steps of DBSCAN is as follows:

1. Choose a random point.

2. Label the point. If it is a core point, check all the unlabeled points within reach and label them. This is how the cluster grows. If it is an outlier, choose another random point and label it. If this new random point is a core point, a new cluster is found.

3. Continue until there are no more unlabeled points.

Even though DBSCAN does not need to know how many clusters there are beforehand, there still are some parameters that need to be set. A point is a core point if there are at least P number of points within a radius ϵ from the point, so both P and ϵ must be decided based upon prior knowledge about the usage of the method. It makes an assumption on the density within a cluster, and this needs to be tuned to achieve the best results.

2.4 Combining RGB images and depth images

As mentioned in Section 1.1, stereo cameras can produce RGB-D images. Both the color part and the depth part of the image contains important information about the power lines' placement and orientation in the image. In this section, we will discuss at what time it is best to combine the information these two images hold. More specifically, we will discuss combining them at three different points of the proposed algorithm:

- **Joint bilateral filter** - Combining them from the beginning, during blurring.

- **Joint edge filtering** - Combining them after edge detection, before line detection.

- **Joint line filtering** - Combining them after line detection, before separating power lines from other lines.

Joint bilateral filter

One way to combine the RGB image and the depth image is by using a joint bilateral filter. It works almost exactly the same as the bilateral filter explained in Section 2.1.2, except it also takes the gradients of the depth image into account. That means if there is a sudden change in the RGB image or/and the depth image this area is blurred less. For this application, it could be smart to blur the RGB image less in regions where the image gradient is high for both the RGB and the depth image since the edges from the power lines are present at the same place in both images, but other detected edges, from for example roads or rivers, will only be present in the RGB image.

Joint edge filtering

Another strategy would be to look at the edges found by the Sobel edge filter and try to filter out the edges only found in either the RGB image or the depth image. This can be done by performing Sobel on both the RGB image and the depth image and sum their magnitude, as shown in Equation 2.15. The Canny edge detector can then be tuned to choose edges with either high magnitude in one of the images, or moderately and above high magnitude in both images.

$$\begin{aligned} G_{RGB} &= \mid G_{X_{RGB}} \mid + \mid G_{Y_{RGB}} \mid \\ G_D &= \mid G_{X_D} \mid + \mid G_{Y_D} \mid \\ G &= \mid G_{RGB} \mid + \mid G_D \mid \end{aligned} \tag{2.15}$$

The parameters G_{RGB} and G_D represents the magnitude in RGB and depth values of the image, and G_X and G_Y represents the gradient of the image in x- and y-direction.

Joint line filtering

It is also possible to merge the information from the depth image and the color image by looking at the lines detected by the Hough line transform, recall Section 2.3.1 and Section 2.3.2. The power lines should be detected in both of these images, and lines that are only found in one of these images are most likely not from the power lines.

One aspect that needs to be considered is that if the power lines have snow or ice on them during the winter, the image gradient might not be so high in the RGB image, and thus the power lines edges in the RGB image can be smoothed out. The joint bilateral filter is slower than most other denoising methods and blurring filters. On the other hand, the earlier the info gathered from the RGB and the depth image can be combined, the earlier irrelevant information can be eliminated. This would mean that the processes coming after this combining procedure would only have to be run once instead of twice. Since both the Canny edge detector and especially the Hough transform are relatively computationally expensive, combining them as early as possible could actually bring the overall processing time down.

2.5 Concluding remarks

In Section 2.1, image downsampling and methods for noise removal were presented. More specifically, the Gaussian downsampling pyramid for resizing of the image and different filters for blurring and noise removal was presented. Most cameras give images with high resolution, which can be computationally demanding to process. If downsizing can be done without losing information regarding the power lines position, it should be done since it will lead to lower run-time. Of the four filters presented, the bilateral filter preserves edges better and will thus preserve the power lines better. Instead of using a Gaussian downsampling pyramid, a bilateral downsampling pyramid could be tried, such as suggested in Section 2.1.1. If the power lines do not have strong edges and therefore is hard to locate, histogram stretching or equalization can be performed to increase the contrast in the color image. This will, however, make all edges in the image more prominent, not only the edges belonging to the power lines. Background removal is, in general, a challenging task, and especially for this application where the background is continuously changing. It would be interesting to look into if some of the color channels hold more information than the other two, and then either blur the other channels more or enhance contrast in the one holding most information. It is important to note that this may be specific for the dataset used in this report and thus it might not work ideally under other conditions, such as winter

images. In Section 2.2, the Canny edge detector was presented, and it will be used for edge detection later in this report. The PPHLT from [17] will be used to extract lines from the image based on the edges found by the Canny edge detector. The power lines in the image may not be detected as one long line, but rather several shorter lines. It can thus be useful to merge them together to longer lines. This can be done by region growing, segmentation, or clustering, such as with K-means and DBSCAN which were presented in 2.3.4. Lastly, Section 2.4 discusses how the merging of the information from the RGB image and the depth image can be done. Three ways are presented, namely joint bilateral filtering, joint edge filtering, and joint line filtering. The latter causes the PPHLT to have to run twice. Considering PPHLT is a rather time demanding method, one of the two former is probably better suited for this application since it must be real-time runnable.

Chapter 3

Power line detection

The Hough transform detects lines in an image, but it does not differ between power lines and other similar looking lines, such as edges of a road, fences, and rivers. Other work on the topics of autonomous power line and power mast inspection has been done, as [18] which used deep learning to detect power masts and power masts components in an image. There is reason to believe that deep learning could be applied to detect power lines in an image, but that would require a rather large labeled dataset, which currently does not exist and is not easily obtained either. Therefore, a method that does not rely on a vast amount of labeled data is needed.

In this chapter we will present and discuss different methods that can be used to accurately detect power lines in an image. First, what preprocessing of the images must be done is evaluated in Section 3.1. Some of the methods presented are based on previous work, such as Section 3.2, Section 3.3, and Section 3.7. Additionally, Section 3.6 presents the algorithm originally suggested in [12], the project thesis this report is a continuation of. In Section 3.4, the clustering method DBSCAN is discussed. In Section 3.5, RANSAC and AMA RANSAC is presented. AMA RANSAC is RANSAC combined with apriori knowledge about the power lines. Both versions of RANSAC achieved outstanding accuracy, at the cost of being computationally heavy. In fact, the two methods might turn out to be too slow for real-time usage depending on the UAVs onboard computational resources.

To remedy this, AMA RANSAC and the Hough transform-inspired voting scheme is combined in Section 3.8. The different methods are quantitatively evaluated and compared in Section 3.9. Lastly, Section 3.10 contains concluding remarks about power line detection.

3.1 Image preprocessing

Several image preprocessing methods was presented in Section 2.1. Figure 3.1 shows the three most relevant filters applied to an image of power lines, namely: bilateral filter, Gaussian filter, and normalized box filter. The median filter is not included since it is most effective against salt-and-pepper noise, which the dataset used in this thesis does not have. Smaller filters are favorable since the power lines take up so little space in the image; larger filters would have a high risk of filtering out the power lines altogether. Additionally, as can be seen in Figure 3.1 the bilateral filter preserves the power lines the most and is thus best suited for our application. Even though the bilateral filter has longer run-time than the linear filters, this is negligible in relation to the run-time of the methods presented later in the chapter.

(a) Original image (b) Box normalized filter (c) Gaussian filter (d) Bilateral filter

Figure 3.1: Filter comparison

By inspection of the different color channels, it is clear that the power lines are mainly

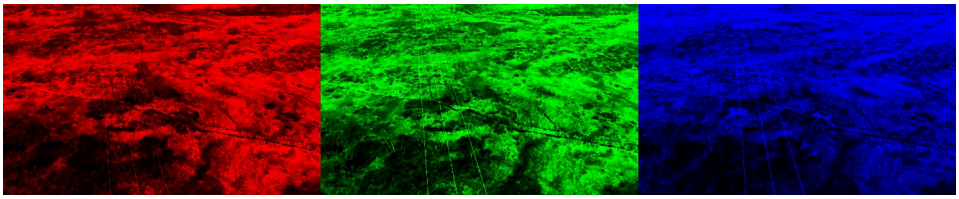


Figure 3.2: Equalized histograms of the color channels.

visible in the green and blue color channel. Figure 3.2 shows this with the color channels histogram equalized. However, this will not be used since we do not know if this is also valid under other conditions such as different landscape or different season. Performing histogram equalization on the images made the power lines clearer in the image, but also led to lines in the background becoming more prominent. This led to more edges and lines to process, but no noticeable improvements in the results. Therefore, no change in the color channels or histogram equalization is not considered going forward in the proposed methods. Additionally, the power lines are not very different in appearance from the average background color, so this will not be used either. Since the power lines are so thin in the images, downsampling could not be performed without having a significant decrease in the performance of the algorithms. The Canny edge detector and the progressive probabilistic Hough line transform showed promising results and is therefore used to find the edges and lines.

3.2 Histogram filtering

In [23], power lines were detected from images taken by a UAV. The root mean square (RMS) error was just above 0.2 meters in both the estimation of the position and height of the power lines. The whole algorithm was not replicated, but the histogram filter used to detect edges was. This filter is based on prior knowledge about the appearance of the power lines and their background, and is intended to only detect edges belonging to the power lines. The power lines are in general brighter and have more homogeneous texture compared to the background. The power lines are quite thin, and in the height that was flown in [23] they only took up 1-3 pixels in width. These characteristics of the power

lines become prominent in an image histogram as shown in Figure 3.3. There is a clear difference between the histograms of power line edges, and the histograms of other edges. Equation 3.1 is the equation used to check whether or not a detected edge belongs to a

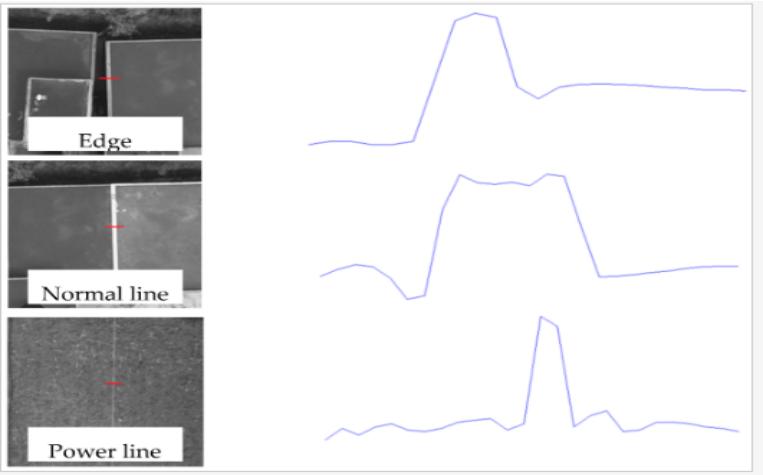


Figure 3.3: Power line histograms.
Figure 6 from [23] showing the gray value histograms along the red line in the images. It should be noted that this is not the same type of image histograms presented in Section 2.1.3, but rather expressing the brightness of pixels along the red line in the images.

power line or not.

$$r = \frac{2g_c - g_l - g_r}{|g_l - g_r| + 1} \tag{3.1}$$

The variable g_c represents the gray value of the center pixel, while the variables g_l and g_r represents the gray value of the pixel to the left and right of the center pixel respectively. It is the value of r that decides whether or not the edge is accepted as a power line edge. This is done by checking if it is above a certain threshold value. Histogram filtering can also be done on a larger window if the image is taken closer to the power line since they would occupy more than just a couple of pixels in the image. This must be adjusted real-time, which makes the method vulnerable if the distance between the UAV and the power lines is either inaccurate or unknown. Additionally, this is based on the appearance of power lines in contrast to the background during summer and might not work as well when the background changes, for example if it is covered by snow. Despite the good results in [23], the histogram filter did not achieve satisfactory results on the dataset used in this

thesis. This could be due to the background or the distance from the power lines being too different. Histogram filtering will not be used in this report.

3.3 Pulse Coupled Neural Network Filter

In [15] a somewhat untraditional power line detection method was proposed. It is untraditional in the sense that most line and edge detecting methods uses a bottom-up approach, while [15] used knowledge about the power lines to implement a top-down approach. The prior knowledge they used was:

1. The power lines brightness is uniform, and they are in general brighter than their background.

2. The power lines approximates a straight line.

3. The power lines are approximately parallel to each other.

The assumption about the power lines being brighter than their background is, as briefly mentioned before, the one that will not hold during winter, and thus the method might not be optimal for this project. Even though the power lines sag, they will be approximate straight when seen from above. The power lines are in essence parallel, but not from the UAVs point of view. They will however point towards approximately the same vanishing point. What differs the pulse coupled neural network(PCNN) from most other edge detectors, for example the Canny edge detector, is that instead of looking for all edges it only looks at edges belonging to the power lines. The PCNN presented in [15] showed very promising results. Even though it is a neural network, it requires no training. It would be interesting to test this method out and check if maybe it can be used on winter images by inverting the image before applying the algorithm. However, replicating from scratch and testing the PCNN is outside the scope of this thesis and methods available in open source libraries such as OpenCV is prioritized.

3.4 DBSCAN

Clustering methods can be used to separate detected lines belonging to power lines and belonging to other lines. DBSCAN is such a clustering method, as explained in Section 2.3.4. Since DBSCAN uses the distance to other points to decide whether or not a point is a core point, the scales of the two parameters ρ, and θ is important. The angle parameter θ typically goes from 0 to 360 degrees, or for this use the scale from 0 to 179 is sufficient, since lines with angle θ equal to any multiple of 180°are parallel, as shown in Figure 3.4.

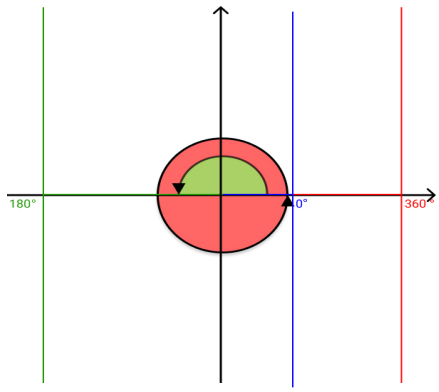


Figure 3.4: Parallel lines in Hough space.
Illustration of how lines with different θ in Hough space still can be parallel. The red line has θ equal 360 degrees, the green line has θ equal 180 degrees, and the blue line has θ equal 0 degrees.

Even though the three lines in Figure 3.4 are parallel, they can have different θ-values from 0°to any multiple of 180°. The length variable ρ, on the other hand, is at maximum equal to the length of the diagonal to the image. This is an edge case not likely to happen since the line would only be barely visible and at the very edge of the image.If the image is 680x1024, the diagonal will be approximately 1229 pixels long. If DBSCAN then uses Euclidean distance to determine whether or not a point is close enough to be part of a cluster, the ρ will dominate the outcome.

a) The original image b) A zoomed in and rotated version of a

Figure 3.5: DBSCAN test images.

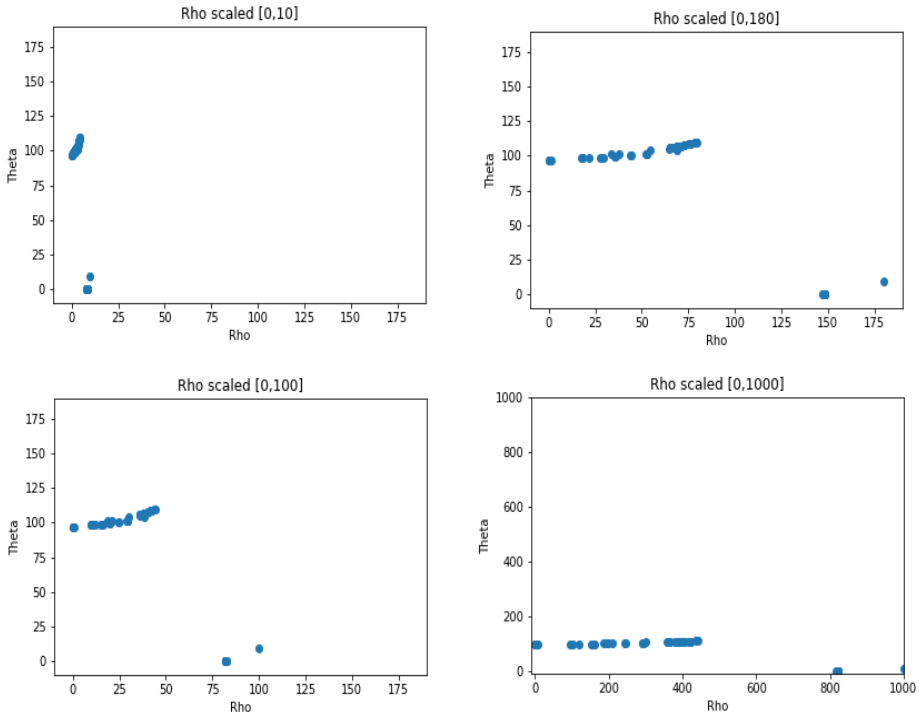


Figure 3.6: Scatter plot of detected lines from image A in Figure 3.5.

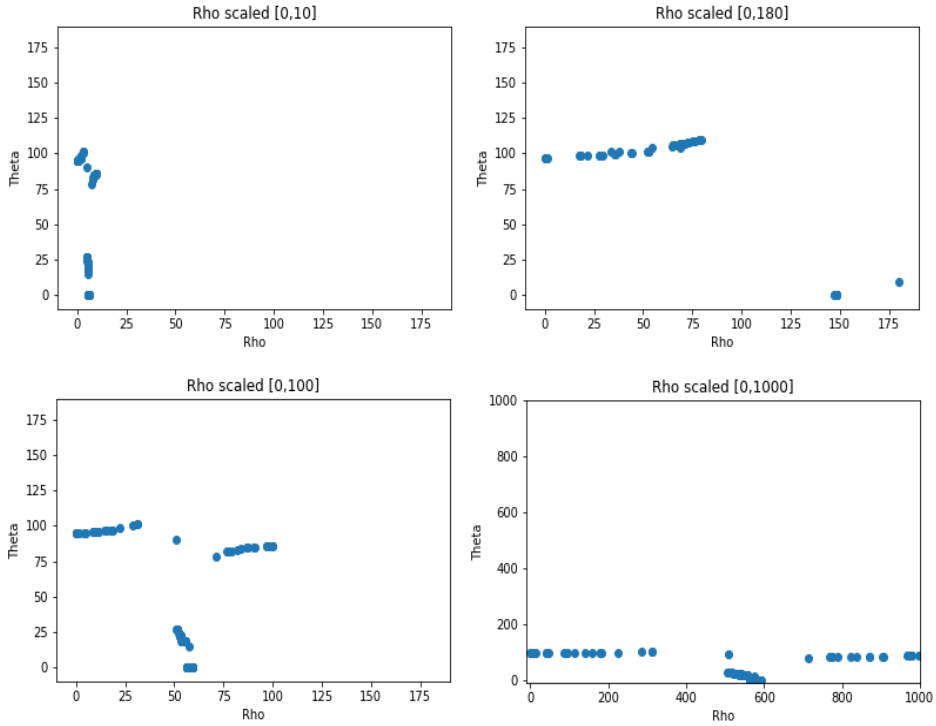


Figure 3.7: Scatter plot of detected lines from image B in Figure 3.5.

Figure 3.6 and Figure 3.7 shows how the detected lines are distributed in the Hough space under different ρ-scales. These two figures are based on the two images in Figure 3.5, where the right image is a slightly zoomed in and rotated version of the left one. Even though it is almost the same image, the density distribution of the found lines is quite different. For example, the plot in Figure 3.7, where the ρ is scaled to be between 0 and 100, displays three clusters with similar density, but the same scaling in Figure 3.6 does not have the same clusters or density. Additionally, in the [0,1000]-scale in Figure 3.7 three different clusters can be seen, but one of them have a significantly higher density than the two others. This indicates that the density of the power lines that we want to be clustered together is not consistent, both between different clusters, between different images and between different scales. Thus, DBSCAN is not well suited for our application.

3.5 RANSAC

RANSAC, as explained in Section 2.3.3, can also be used to detect the power lines. Instead of taking the binary edge image as input, it takes the binary line image as the input. To ensure that the shorter lines do not dominate the longer lines, each pixel at the core of the lines is used as input. This way, the detected lines' voting power is relative to their length. One of the parameters that need to be determined for this method is the maximum number of hypothesises to attempt. When RANSAC tried a maximum of 1000 different hypothesis and used a model of three straight lines, RANSAC detected the power lines with a surprisingly high success rate. A typical result is shown in Figure 3.8.

a) The detected Hough lines

b) The RANSAC output based on the detected Hough lines

Figure 3.8: RANSAC results.

Having a preset number of attempts is one of RANSACs advantages because the run time then only depends on the number of detected lines. In general, the more attempts RANSAC gets, the better the results are. However, the run time increases significantly with the number of hypothesises allowed, so there is a cost-reward-trade off here too. In the average case the correct lines are found within the first 100 attempts, and the remaining attempts then only serve to solidify those results. As such, it is obvious that the optimal number of attempts needs to be identified. However, this is not a static number, since different images require a different number of attempts. When runing on the computer used in this thesis with specifications as described in Section 1.3, running RANSAC with 1000 as the maximum number of attempts took between 1 and 2 seconds for each image frame, which is unacceptable for real-time navigation. It might be real-time runnable on a faster

computer, or on one with a GPU, but this would again require a UAV with a certain size or in a higher price range, to accommodate for the additional hardware. Therefore, in an attempt to lower the run-time by finding the optimal number of attempts for each image, an adaptive parameter for the maximum number of attempts is implemented, as presented in the following algorithm :

Algorithm 1: Adaptive Attempts Power Line RANSAC(AMA RANSAC)

Input: Detected Hough lines, **L**

Output: Detected power lines, **PL**

i = 0

while *the detected power lines,* ***PL****, are not approximate parallel* ***and*** *i <5* **do**

1. **maxAttempts** = 50 + 100*i
2. **PL** = RANSAC(**L**,**maxAttempts**)
3. **i**++

The adaptive max number of attempts is applied so that RANSAC stops running when it has detected the correct hypothesis, or mathematical model, of the power lines. The detected lines are then checked for parallelism, and if they are not approximately parallel, RANSAC will run again with a higher number of maximum attempts. To prevent it from getting stuck, it is only allowed to run 5 times in a row regardless if the detected lines were approximately parallel or not. Figure 3.9 B shows three lines that would not get accepted by the algorithm. To pass as approximately parallel, the lines can not differ with more than 30 °and cannot cross each other anywhere in the image except the uppermost quarter of the image. The drawback of just rerunning RANSAC is that it does not remember what hypothesises it has already tried. A better solution would be to implement RANSAC with adaptive maximum attempts from scratch. Due to time constraints of the project, OpenCV's implementation of RANSAC is used and reran if the detected power lines are not approximately parallel. Even though this is not the optimal implementation, the computational time is significantly lower. Despite this, the algorithm could still not be run in real time on the test computer.

a) Accepted power lines b) Not accepted power lines

Figure 3.9: Example of rejected and accepted power lines.

3.6 Hough Transform-inspired Voting Scheme

In [12], an algorithm for differentiating between lines detected from power lines and other lines was proposed. The algorithm below is as stated in [12]:

Algorithm 2: Hough Transform-Inspired Voting Scheme(HTIVS)

Input: Set of detected lines, **L**, in image
Output: ρ and θ of the detected power lines in image
1. Remove all lines with θ that differs more than 45 °from expected power line direction.
2. Remove all lines shorter than $\frac{1}{3}$ of the images height.
3. Let all lines give a weighted vote to their θ and ρ value. The longer the line, the heavier the vote.
4. Choose the θ with the most votes, and up to three ρ values that represent three parallel lines, and return these as the power lines, **PL**.

The algorithm was tested on the dataset used in this thesis, and it was immediately clear that it needed some changes to fit our criteria. Step 1 and 2 diminishes the number of detected lines that do not stem from the power lines. However, an assumption of the UAVs position relative to the power lines is made. Considering the UAV will be following the power lines, this is not a wrong assumption to make, except during initialization or if the power lines abruptly change direction. The power lines only change direction when a power mast is in the image, and even then the change of direction is seldom very abrupt. It is also possible to account for this by detecting the power masts as well as the power lines,

which will be discussed in Chapter 4. The second step of the algorithm assumes that the power lines, in general, cross the whole image. The standard for 22kV and 66kV power lines is that there is a mast approximately every 100 meters. According to [23] the distance between two power masts that carry higher voltages are in general between 300 meters and 500 meters. This means, for both cases, that unless the power mast is in the image, the power lines will run through the whole image following one straight line. On the other hand, if a power mast is actually in the image, the power lines might change direction, thus following two straight lines: one before the mast and another after. Hence, the assumption that the UAV is rotated towards the power lines is not a bad assumption to make. The problem that arose after testing it on a few images from the dataset was that the lines detected by the Hough transform on the power lines are short and thus would be filtered out by the second step. A method for merging similar smaller lines was implemented, but was not robust enough to handle the variety of situations that occur in the dataset. The method looked for close lines with similar angle, but as mentioned in Section 3.4, the lines do not have a constant density in the image, so it was hard finding parameters that worked well. Therefore, the second step must be removed entirely. The third step, the actual voting stage, has run-time $\mathcal{O}(n^2)$, with n being the number of remaining lines after step 1 and 2. The number of remaining lines, **L**, is seldom a large number, so there is no need for a probabilistic approach to the voting stage. The next thing that needs to be considered is how to discretize the voting: mainly determining the size of the voting bins. By trial and error, having around 180 bins showed the most robust results. The method was originally supposed to detect all three power lines, but this did not work. However, it was able to detect the general direction of the power lines. As will be discussed in Section 5.5, if detecting the general direction is enough depends on which method is chosen for determining the distance between the UAV and the power lines, as some of them require that at least two distinct power lines are detected.

For the algorithm to be robust, it must handle the UAV not always having the desired rotation relative to the power lines. Additionally, the lines detected does not always have the same angle as the line they arose from. Therefore, the lines are allowed to differ 45 $^\circ$from the expected angle. This does, however, make it sensitive to other prominent lines

Figure 3.10: Example of a situation where HTIVS struggles

in the background, as can be seen in Figure 3.10. The direction of the upper part of the river and edges that are seen on the horizon is chosen instead of the power lines. This algorithm was fast enough to run in real-time on the computer used in this thesis, but not accurate enough to be used for navigation.

3.7 Knowledge-based line clustering in Hough Space

The same paper that presented PCNN, [15], also used a knowledge-based line clustering method in Hough space to separate power lines from other detected lines in the image. The method can be broken down to three main parts:

1. Identify the different line groups by applying K-means, with K equal to four, as explained in Section 2.3.4, to the lines θ-values.
2. Sum up how many votes from the Hough transform each of the four clusters has.

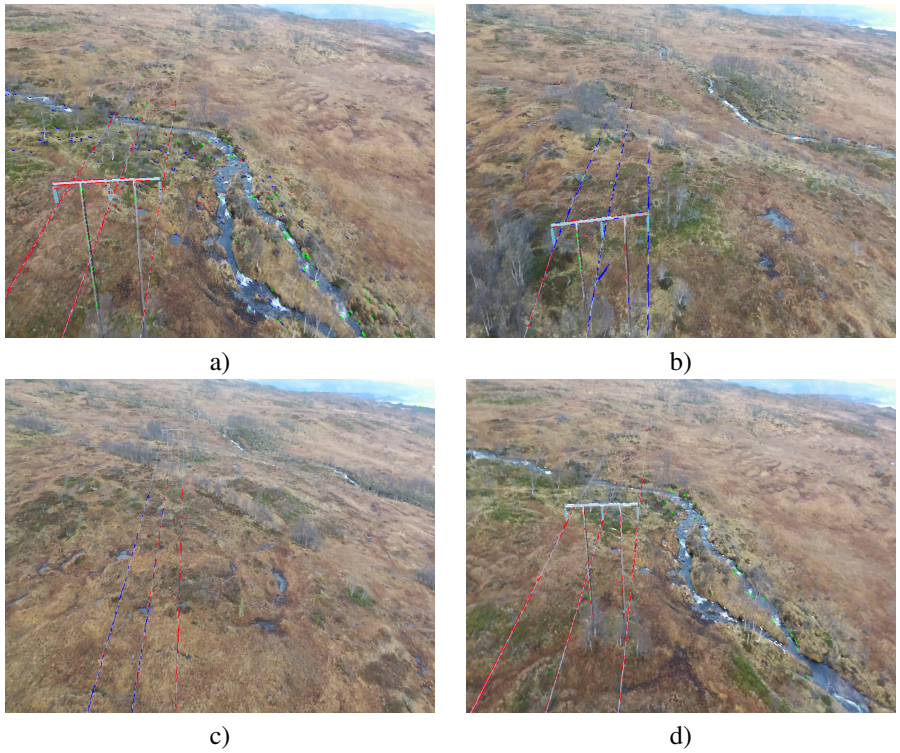
3. Return the cluster with the most votes as the direction of the power lines.

Since the ρ-values are not considered in the first step of the algorithm, approximate parallel lines are clustered together. Here it is important to remember that lines with different θ-values can still be parallel, recall Section 3.4. All θ-values should therefore be between 0 and 180. By using the votes from the Hough transform, lines that are supported by many points get to affect the voting more. This makes the algorithm more robust against lines not coming from the power lines. Extracting the votes from OpenCV's implementation of the PPHLT is not straight forward, and getting these votes would require implementing the Hough transform from scratch. Instead, K-means was running with the length of the lines as a replacement for the Hough votes. Still, it is somewhat unconventional to do cluster analysis on one-dimensional data, and by including ρ-values in the analysis, K-means was able to distinguish better between power lines and other lines. Also, using three clusters instead of four gave cleaner clusters. In Figure 3.7, the lines in the image are clustered in three clusters. The lines are generally clustered into one cluster containing most of the power lines, and two clusters containing all other lines.

Even though one cluster usually contained most of the power lines, it also had a relatively large number of lines not stemming from the power lines. This affected the cluster centers, which made the resulting detected power lines placement and angle inaccurate. Additionally, the power line-cluster was not always the one with the highest number of votes. A condition on detecting if the change in direction was too big in relation to the last frames detected power line was added. If so, the cluster with the second most votes should be tried. However, this did not remedy the inaccuracy. Often, the clusters were not clean enough, meaning they contained lines from both the background and the power lines. In the best case, it could be used as a line filtering method with the Hough transform-inspired Voting Scheme to increase its accuracy, which we will look into in the following section.

Combining K-Means and Hough transform-inspired Voting Scheme

In the algorithm presented below K-Means, AMA RANSAC and the Hough transform-inspired Voting Scheme were combined.
This method yielded more accurate results compared to using the center of the winning



a) b) c) d)

Figure 3.11: K-means clustering results
Clustered on ρ- and θ-values with K = 3

Algorithm 3: Combined K-means and Hough transform-inspired voting scheme(K-Means + HTIVS)

Input: Set of detected lines, **L**, in image
Output: ρ and θ of the 2D-trajectory line to follow
Initialize. Initialize by running RANSAC on all detected lines, **L** with a high number of max attempts, for example, 1000 to find a model of the power lines.
1. Run K-Means with 3 clusters on the power lines.
2. Let all lines within a cluster give a weighted vote based on their length to its cluster. Choose the cluster with the highest number of votes as the power line cluster.
3. Run the Hough transform-inspired Voting Scheme on the lines belonging to this cluster.
4. If the detected line from the voting scheme is not approximate parallel with one of the power lines found in the last power line model, go back to 2 and choose the cluster with second most votes.
5. If none of the clusters is approximate parallel with the last power line model, run AMA RANSAC to update it.

cluster, but was slower due to it having to rerun AMA RANSAC often. Even when rerunning AMA RANSAC often, the results were still mediocre.

3.8 Combining AMA RANSAC and Hough transform-inspired Voting Scheme

AMA RANSAC yielded outstanding results, at the cost of being computationally heavy. This means that if AMA RANSAC is to be used in real-time, the UAV needs to have more powerful computational resources available. This will either increase the cost of the UAV, make the UAV heavier or both - all of which is undesirable, so a computationally lighter method is called for. HTIVS, as explained in Section 3.6, is fast enough for real-time use but is not nearly as accurate as AMA RANSAC. If the background is simple it shows relatively good results, but if the background is more complex and contains disruptive elements such as rivers, roads, or even power masts in the image it struggles. AMA RANSAC and HTIVS will be combined in the hopes of finding their golden mean. In this section, we will go through how they can be combined to obtain good enough results for navigation, fast enough for real-time applications. This will be done for both detecting only one line

which follows the general direction of the power lines and for detecting three distinctive lines. The steps of these two methods are explained in the following section. Both use RANSAC for initialization and AMA RANSAC for updating of the power line model.

Initialization with RANSAC

The algorithm starts with running RANSAC to obtain a good model of the power lines. RANSAC gets 1000 attempts to find the lines initially, which would take way too long to do at every image frame of the flight, but it is no problem at all while hovering during initialization. It is here assumed that the power lines are already in the image. This means that the UAV must be manually flown to a position above the power lines, or that the UAV must have a different method to get there automatically.

Updating the model with Adaptive Max Attempt RANSAC

The model found upon initialization must be updated so that it can be used to filter out the lines coming from edges stemming from something other than the power lines. To do this, AMA RANSAC, as explained in Section 3.5, is utilized. AMA RANSAC is used to prevent RANSAC from running longer than necessary, but still not detecting incorrect lines since we immediately can eliminate most of the incorrectly detected lines based on previous knowledge about the power lines. Updating the power line model every 50th frame regardless if the lines are accepted or not gave higher accuracy.

One line detecting Hough transform-inspired Voting Scheme

The voting scheme used is based upon Algorithm 2 from Section 3.6 with some modifications. Most importantly the output is changed. This version aims to find a line that can be used as a trajectory to follow. It is not necessary to identify each one of the power lines, or any of them at all, as long as we manage to follow them. Another difference is that short lines are not removed. This is because the lines detected in images from the dataset were shorter than originally expected. The region of interest from the first step in Algorithm 4 is found based on the most recent power line model found by RANSAC or

AMA RANSAC. The region of interest is the area between the two outwards power lines extended by 50 pixels. The black lines in Figure 3.12 represent these boundaries. The lines that are marked red, which are filtered out, inside the region of interested are filtered out because they have the wrong angle compared to the most recent power line model. As can be seen in Figure 3.12 some lines on the poles of the power masts are not filtered out, even though they do not belong to the power lines. This is undesirable but does not cause critical errors in the calculated trajectory.

Figure 3.12: Line filtering
Line filtering based on the power line model found by the most recently run RANSAC or AMA RANSAC. The red lines are filtered out, while the white lines remain. The red lines outside the black, vertical boundary lines are filtered out due to their position. The red lines between the two boundary lines are filtered out due to their angle.

Since the power lines are consistent, previous images also hold useful information. Using the power line model found by RANSAC or AMA RANSAC is one way of making use of this information. If Algorithm 4 returns a line that is not approximately parallel to at least one of the three lines detected by the most recent power line model, AMA RANSAC will be rerun to ensure that the model is correct. This increases the average run-time per

Algorithm 4: One Line Detecting HTIVS

Input: Set of detected lines, **L**, in image
Output: ρ and θ of the 2D-trajectory line to follow
1. Remove all lines that are not inside the region of interest.
2. Remove all lines that are not approximate parallel with the last power line model found by RANSAC/AMA RANSAC.
3. Let all lines give a weighted vote to the θ-bin it belongs to based on the length of the line found by the most recent power line model.
4. Each θ-bin has a corresponding weighted averaged ρ. Each line affects this average by the same weight as in step 3.
5. Choose the θ and its corresponding ρ with most votes. Return these as the detected 2D-trajectory line to follow.

image by a little, but ensures robustness.

Three line detecting Hough transform-inspired Voting Scheme

As we will see in Section 5.5, it might be desirable to identify where each of the three lines is. Even though a single line that follows the general direction of the power lines can be used for navigation in the 2D-plane, actually identifying each of the power lines position makes it easier to determine how far away they are from the UAV. In essence, locating each of the power lines in the x,y-plane makes it easier to determine their depth values on the z-axis of the camera. Algorithm 4 can, with only small modifications, find all three lines, as can be seen in Algorithm 5.

Algorithm 5: Three Line Detecting HTIVS

Input: Set of detected lines, **L**, in image
Output: ρ and θ for all three power lines
1. Give each detected line from **L** a label('left', 'middle', 'right' or 'other').
For *each* label except 'other' lines:
3. Let all lines belonging to this label cast a vote to the θ-bin it belongs to weighted by its length. If there are no lines for any of the labels, update the RANSAC model by running RANSAC with an adaptive number of max attempts.
4. Each θ-bin has a corresponding weighted averaged ρ. Each line affects this average by the same weight as in step 3.
5. Choose the θ and its corresponding ρ with the most votes. Return these as the power line corresponding to the label this set of detected lines have.

The labeling is the same as the line filtering presented in Section 3.8, except now it is

done on each line. Each line detected by PPHLT is checked to see if it is close enough, in distance and angle, to the three power lines found by RANSAC or AMA RANSAC. Figure 3.13 shows the results of this labeling. The blue lines are labeled as part of the 'left power line'-label, the green lines are 'middle power line'-label, the red lines are 'right power line'-label and lastly the yellow lines are 'other lines'-label. When the number of frames since the last time the power line was updated approaches the threshold for updating it, more and more of the detected lines will fall into the 'other lines'-label. This is because the model used for labeling gets more inaccurate as it ages. When not enough lines are labeled as something else than other lines, RANSAC is rerun to update the model. This happens more frequently when approaching a power mast, because the lines often slightly change direction right after the power masts. As previously mentioned, the power line model is updated every 50th frame. Even though the same algorithm has to be run three

Figure 3.13: Line labeling.
The blue lines are labelled as the left power line, the green lines are labelled as the middle power line and the red lines are labelled as the right power line.

times, one time for each power line, fewer lines are voting for each time it is run, so the

algorithms average run-time does not increase significantly. It does not always find all three lines either, but it knows which line it has detected. This is an advantage since the trajectory line can be adjusted for this, so that the UAV does not fly in a zig-zag pattern.

3.9 Evaluation and comparison

To further decide which methods are best suited for this application, a quantitative evaluation has been performed. The methods' performance is determined by two factors, their run-time and their accuracy. The five methods compared in this section is:

- **RANSAC** from Section 3.5
- **AMA RANSAC** from Section 3.5
- **HTIVS** from Section 3.6
- **K-means + HTIVS** from Section 3.7
- **AMA RANSAC + HTIVS** from Section 3.8

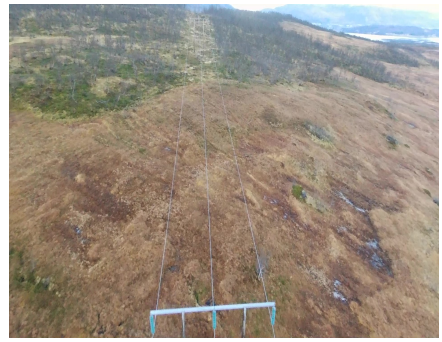
The methods are tested in four different situations, shown in Figure 3.14. Situation A and B both have similar and simple backgrounds but different distance to the power lines. Thus, they can show how the different methods handle different distances and works with relatively ideal backgrounds. Situation C and D are chosen because they have more challenging backgrounds. Therefore, these four test situations can evaluate how robust the different methods are. The accuracy is calculated according to Equation 3.2.

$$ACC = \frac{CD}{CD + ID} 100\% \tag{3.2}$$

ACC represents the accuracy given as a percentage, CD is the number of frames with correct detection of the lines, and ID is the number of frames with incorrect detection of the lines. A correct detection of the power lines for the methods finding three lines, must fulfill the following criteria:

1. A line is detected properly if they do not differ with more than 10 °with the ground truth. The correctness in the bottom half of the image is most important since this is

(a) Far away with basic background.



(b) Close with basic background.

(c) Close with challenging background.

(d) Medium distance with challenging background.

Figure 3.14: The four test situations

what the UAV will first navigate from. A properly detected line for the right power line is inside the green area of Figure 3.15 A.

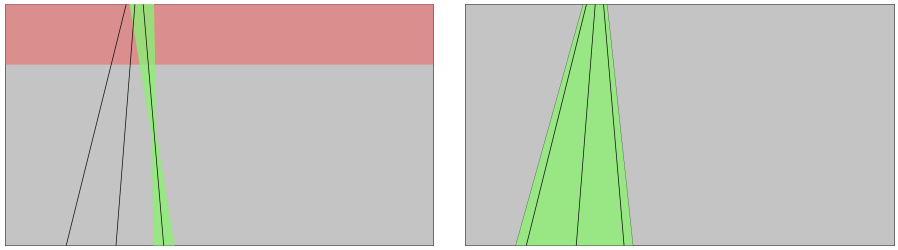
2. Must detect at least two lines properly.

3. The lines detected can only cross each other in the red area, as illustrated in Figure 3.15 A.

For the methods trying to finding a trajectory for the UAV to follow, the following criteria must be fulfilled:

1. The trajectory line must be inside the green area of Figure 3.15 B.

2. The trajectory line must be less than 10 °different from at least one of the power lines.



(a) Three lines (b) One line

Figure 3.15: Evaluation criteria for detected lines.

It should be noted that critical errors and inaccuracies in the detected lines are weighted equally in the calculation of the accuracy. This may be misleading in how well the methods work. Generally, K-means + HTIVS and HTIVS have more critical errors than the other methods. Inaccuracies can lead to the UAV flying in a more zig-zag pattern, whereas critical errors can lead the UAV in the wrong direction and in the worst case make it lose sight of the power lines entirely. Additionally, incorrect detections get more severe if they are consecutive, but the test does not take this into account. K-means + HTIVS and HTIVS only detect one line, while the other methods detect three lines. The methods aiming to detect three lines get an incorrect detection if they only detect one line, even though that line would have passed the criteria for a correct detection of one line. This line could have been used for navigation, and as long as this does not happen in several consecutive frames this would not have been a problem since the UAV and the power lines do not suddenly change height.

The results for situation A is presented in Table 3.1, the results for situation B is presented in Table 3.2, the results for situation C is presented in Table 3.3, and the results for situation D is presented in Table 3.4.

All the methods are tested on the same computer with the specifications from Section 1.3. The run-time of the methods are heavily dependant on the machine they are run on, thus they should only be used for comparing the methods against each other. The relative run-time for the methods may change if programmed for running on a graphics processing unit(GPU).

Method	Avg. run-time	Best run-time	Worst run-time	Accuracy
RANSAC	0.99s	0.818s	1.149s	**100%**
AMA RANSAC	0.147s	0.109s	2.404s	**100%**
HTIVS	**0.0249s**	**0.023s**	**0.03s**	**100%**
K-means + HTIVS	0.069s	0.033s	0.19s	75%
AMA RANSAC + HTIVS	0.04s	0.027s	0.16s	30.9%

Table 3.1: Comparison of data analyzing methods in situation A.

Method	Avg. run-time	Best run-time	Worst run-time	Accuracy
RANSAC	1.322s	1.156s	1.698s	**100%**
AMA RANSAC	0.136s	0.125s	0.155s	**100%**
HTIVS	**0.0278s**	**0.025s**	**0.031s**	83,7%
K-means + HTIVS	0.038s	0.034s	0,044	96,7%
AMA RANSAC + HTIVS	0.0481s	0.029s	0.186s	83.6 %

Table 3.2: Comparison of data analyzing methods in situation B.

Method	Avg. run-time	Best run-time	Worst run-time	Accuracy
RANSAC	2.6s	1.58s	3.125s	62,4%
AMA RANSAC	0.922s	0.155s	6.34s	69%
HTIVS	**0.044s**	**0.036s**	**0.052s**	12,8%
K-means + HTIVS	0.167s	0.044s	5.874s	30,7%
AMA RANSAC + HTIVS	0.369s	0.038s	5.268s	**77%**

Table 3.3: Comparison of data analyzing methods in situation C.

Method	Avg. run-time	Best run-time	Worst run-time	Accuracy
RANSAC	1.2s	0.97s	1.481s	**100%**
AMA RANSAC	0.155s	0.12s	2.571s	**100%**
HTIVS	**0.0296s**	**0.025s**	**0.035s**	**100%**
K-means + HTIVS	0.044s	0.036s	0.181	40,3%
AMA RANSAC + HTIVS	0.045s	0.03s	0.173s	68.8%

Table 3.4: Comparison of data analyzing methods in situation D.

Both RANSAC and AMA RANSAC have 100% accuracy on all situations except in situation c, when there is a river in the background. In situation c, AMA RANSAC actually performs better than RANSAC. This is because AMA RANSAC checks if the detected lines are approximately parallel or not, which RANSAC does not do. RANSAC is considerably slower than the other methods, but has the second best results on accuracy. AMA RANSAC is significantly faster than RANSAC, with the overall best results on the accu-

racy, but is still the second slowest of the methods. K-means + HTIVS and HTIVS only detect one line, while the other methods detect three lines. AMA + HTIVS struggles in situation a, when the UAV is further away from the power lines. This is due to the method needing to label the detected Hough lines, and the labeling method requires the detected lines to be within a certain length for it to label them as power lines; fluctuations in distance greatly affects the labeling method and therefore also AMA RANSAC+ HTIVS. Except for this situation the method shows promising results. As long as not too many incorrect detections are consecutive and it is acceptable for the UAV to sometimes fly in a zig-zag pattern, then AMA RANSAC + HTIVS can be used by the UAV to follow the power lines. HTIVS is the fastest method, but also the least robust method making it unsuited for navigation. The methods are also demonstrated in video-format on YouTube[1]. It should be noted that the algorithms are displayed with 30 fps in the video, so it does not demonstrate the different run-times.

3.10 Concluding remarks

In this chapter, we have addressed the problem of detecting power lines in an image. In Section 3.1 the image preprocessing methods presented in Section 2.1 was tested and further discussed. Out of the four discussed filters in Section 2.1, the bilateral filter preserves the power lines in the image significantly better than both the Gaussian and normalized box filters. The median filter is best for removing salt-and-pepper noise. We also saw that color channel manipulation and histogram stretching or equalization was unnecessary since the Canny edge detector and the progressive probabilistic Hough transform detects line segments on the power lines very well on the original images. The line filtering method using a region of interest can be seen as a type of background removal method, but no other background removal is used. In Section 3.2, histogram filtering around detected lines was discussed as a method for distinguishing between power lines and other detected lines, but since it assumes that the power lines are brighter than their background this method is not robust against changing backgrounds, such as with the changing seasons. Additionally, it needs to know the distance between the power lines and the UAV. Even though very

[1] `https://youtu.be/IQcG-re02Dg`

promising results were shown in the papers for histogram filtering, PCNN, and knowledge-based line clustering, we did not achieve the same good results in this report. It should be noted that the PCNN presented in [15] was not attempted implemented since we prioritized a broad search for fitting algorithms, and thus were unable to implement each from scratch. In Section 3.4, DBSCAN was applied to the lines detected by the Hough line transform in an attempt on clustering the detected lines stemming from the power lines together. The lines detected on the power lines does not have a uniform density so the attempt to use DBSCAN for this was unsuccessful. However, it is possible DBSCAN would work better if paired with a different method for line detection. In Section 3.5, RANSAC was given a binary image where the lines detected by the Hough Transform was outlined. RANSAC produced outstanding results with the drawback of being computationally costly. AMA RANSAC gave even better results than RANSAC originally due to checking if the detected lines were approximately parallel, and with significantly improved run-time. RANSAC turned out to be the cornerstone of all of the combined methods best. In Section 3.6, the algorithm originally proposed in [12], is presented with necessary modifications. This algorithm turned out to be highly efficient, but also vulnerable to line-looking objects in the background such as rivers, roads, and fences. In Section 3.8, a combination of AMA RANSAC and HTIVS was presented. This method gave significantly better and more robust results than HTIVS alone but was also significantly faster than only using RANSAC. If the lines are too far away from the UAV, the accuracy of the methods goes down, so it is dependent on having an accurate method for determining the distance between the UAV and the power lines. Quantitative comparisons between the five methods with the most promising results were shown in Section 3.9. If the UAV has enough computational resources to run AMA RANSAC or the UAV is able to navigate with the method running on only every fifth frame, AMA RANSAC is the recommended method. If not, AMA RANSAC + HTIVS should be used.

Chapter 4

Power mast detection

Up until now, we have primarily focused on inspection of the power lines, and its associated challenges, but in fact, there are way more aspects of the power masts themselves that need to be inspected. The inspection around each power mast should include detecting if insulators are damaged, splints are missing or rusted, or the wires are rusted or worn. Additionally, it should detect if the poles or crossarms are broken, or if the toppads are missing, loose, damaged, or are covered by an object. Toppads are used to protect the

Figure 4.1: Common faults on power masts and their components [18]

wooden poles from rain and snow in Norway [18]. Figure 4.1 shows some of the most common faults on the power masts and its components that must be detected. The actual inspection is outside the scope of the work of this project, but the power masts as a whole need to be detected nevertheless, so as to know when to stop following the power lines and when to start navigation into a position where the inspection of the power masts can start.

4.1 Object detection

Object recognition and object detection are easily confused. However, they are two very different tasks, with very different difficulty level and thus the difference is important. Object recognition aims to answer the question *What is this object*, while object detection aims to answer the question *Where is this* known *object*[21, Chapter 14]. We want to perform object detection on one object. There exists an uncountable number of objects in the world, and making a system that can recognize, differ between and label all of them is very hard. Humans are incredibly talented at recognizing objects in general, even from an early age. Neural networks are currently the only method that is anywhere near human performance if there are many objects in challenging environments[19]. Challenging environments can be changing light conditions, a lot of background clutter or the same type of object that can differ a lot in looks. Neural networks are however not applicable here due to the lack of access to labeled data, as previously mentioned. In this chapter, we will look at methods for detecting if the next power mast is in the image, and locate where in the image it is.

In [21, Chapter 14], the author differs between three main approaches to achieve object detection, namely feature-based methods, template-based methods, and appearance-based methods. The feature-based methods find distinctive parts of the object in the image, and checks if they are placed in a way that makes geometrical sense. The template-based methods are based on making a template of how the object should look, and then comparing the new image, or patches of it, with this template. Based on how similar they are, it can detect if the object is in the image. The choice of similarity measure and model is crucial for the results. Appearance-based methods traverse the image patch by patch

to look for candidates possibly matching the searched for object. Almost all appearance-based methods rely heavily on classifiers trained on data with labeled object-patches and no-object-patches.

Recognition by components

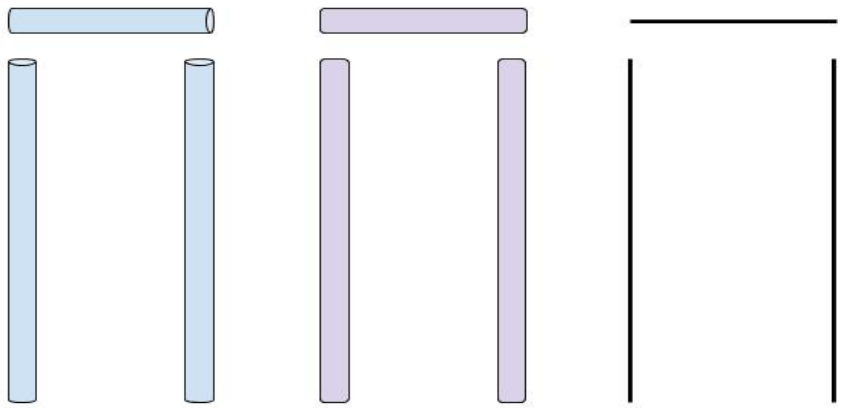


Figure 4.2: The structural components of power masts
Illustration of how to break the power mast down in three components in 3D, 2D, and 1D.

In [19], a method that does object recognition based on recognizing the components of the objects was presented. It says that with only 36 components, called geons, over 30.000 objects can be presented. The objects are encoded with a 4-digit code which states how these geons relate to each other. We will only be trying to detect one object, however, so we do not need this encoding. Recognition by components is a template-based method. We can break down the power masts down to three main components, the two vertical poles which touch the ground and one horizontal pole lying on the two vertical poles. The poles have the shapes of a cylinder. There are however several ways of breaking the power masts down to components, an example of which is illustrated in Figure 4.2.

4.2 Power mast detector

Since the two chosen methods from Section 3.5 and Section 3.8, uses a model of the power lines, we will use this model of the power lines to estimate if there is a power mast in the image. This method presented is rather simple. It is not especially computationally demanding since the lines detected by PPHLT are already labeled as either belonging to the left power line, the middle power line, the right power line or as other lines. We will use this as a basis for finding the lines belonging to the power mast. We will only look at the lines labeled to be other lines since this is where the lines from the power masts will be. Some of the lines from the vertical poles of the power masts will be labeled as belonging to one of the power lines, but that does not actually affect the results of this method, as we will see. The label 'other' is a collective label for all detected lines that does not stem from one of the power lines. This means that random and irrelevant lines from the ground not belonging to either the power masts or the power lines will be here, and we will therefore need to do further sorting of these lines to distinguish those who stem from the power masts and those who do not.

In order to distill out the detected lines stemming from power masts in the other-lines category, the detected lines are checked against the knowledge we already have about power masts, more specifically the horizontal pole of the power masts:

- The lines belonging to the power masts are inside the previously mentioned *region of interest*. That is inside the left and right power line, extended by approximately 1-5% of the image width, relative to the image resolution and frame frequency.
- The lines belonging to the power masts are approximately orthogonal to at least one of the power lines. That is their angle differ by 80-100 °.
- The lines belonging to the power masts crosses all three power lines in visible parts of the image.

If all these criteria are fulfilled, the next thing to do is to decide whether or not enough lines are supporting the same location of the power mast. Since we will be following the power lines, the horizontal pole of the power masts will be approximately parallel to the

x-axis of the image. Thus lines which all belong to this pole of the power masts will have small differences in their y-values. Lines that are closer than a certain distance only with respect to their y-values are called neighbors and can vote for each other. Therefore, the next step of this method is:

- For all lines, let all neighbors (close enough lines) give a weighted vote based on the length of the respective lines.
- Choose the line with most votes as the core line. If it scores higher than a given threshold, it is passed as a core line, and a power mast is detected.

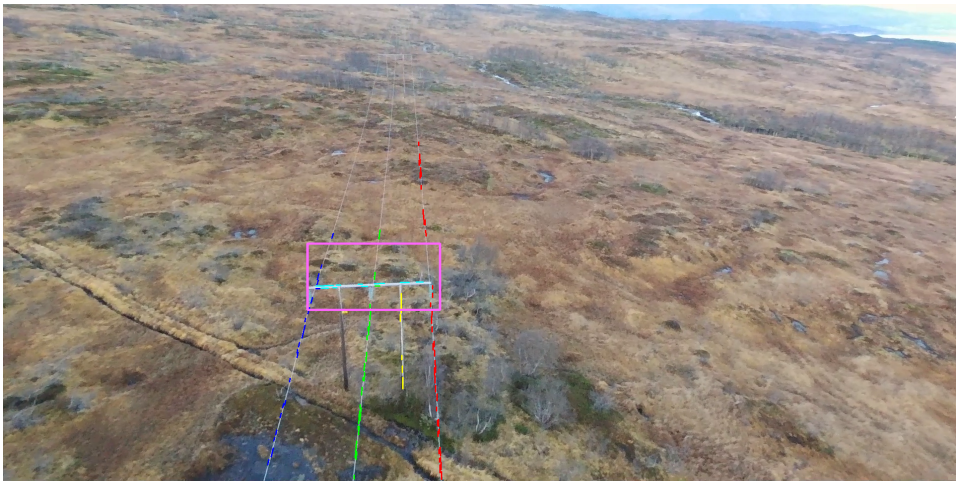


Figure 4.3: Power mast detector result
The dark blue lines are labelled as the left power line, the green lines are labelled as the middle power line, the red lines are labelled as the right power line, the yellow lines are other lines inside the region of interest, and the turquoise lines belongs to the horizontal pole of the power mast. The light blue lines are used to detect the power mast.

This approach has given quite good results, especially in easier situations where no prominent lines other than those from either the power lines or the power masts are present. Figure 4.3 illustrates one of these situations where this was not a problem. The pink box shows the outline of where the horizontal pole of the power mast is expected to be, the light blue lines are lines labeled as power mast lines, and the yellow lines are labeled to belong to other lines.

Even though most of the lines from the background would be filtered out after this, some

situations will be likely to cause more noise than others. For example will a river, fence, road or similar that crosses the power lines approximately orthogonal have a lot of lines that would end up being labeled as power mast lines. This is one of the reasons that this method on its own is in no way enough for navigating around the power mast, but it is rather an indication of where the power masts are located. When a power mast is detected in the image by this method, another method should be called on. This method should do a more thorough check for the power mast, for example by one of the techniques proposed in Section 4.1, and use this for navigation around the power mast and inspecting it. The power mast detector is also demonstrated in video-format[1].

4.3 Concluding remarks

For now, we can see that using a simple power mast detector such as the one presented in Section 4.2 and illustrated in Figure 4.3 is adequate. We will not be performing any navigation or inspection of the power masts, so an indication of when a method doing that should start is good enough. Even though it is simple, it performs well. There are some false positive in certain situations, such as rivers, fences or roads crossing the power lines approximate perpendicularly in the background. This does not happen too often, but the method that is called on positive detections of power masts needs to be able to determine if there really is a power mast present. There are also some false negatives, but mainly when the power masts are still far away. This is due to the same reason as to why HTIVS struggles when the UAV is too far from the power lines since they are based on the same labeling method. However, this can be beneficiary since we only want to detect the masts when they are close enough to start navigating around them anyway. There are very few false negatives when the power mast is close, however, and no occasions of the power masts not being detected at all have been witnessed on the dataset used in this thesis. The method that should be called on by the simple power mast detector, the more robust power mast detector, could potentially be an appearance-based method such as a neural network or by making a template of either the entire power mast or by components such as shown in Figure 4.2. A rather simple method similar to the one used to detect the horizontal pole

[1] https://youtu.be/IQcG-re02Dg

can probably be used to detect the two vertical poles. Then the three main components have been detected and all left to do is to see if they are placed relative to each other in a plausible way. This could be a worthwhile angle for later projects on the subject of autonomous power grid inspection, but it will not be done in this project since the primary focus is on power line detection and following.

Chapter 5

Trajectory for power line following

After detecting and localizing the power lines in the image, these lines must be transformed into a form where they can be used for navigation. From the RGB-images, we only get a 2D-representation of the power lines, and we need them to be in 3D. In this chapter, we will first discuss how to perform coordinate transformations between the relevant coordinate frames in Section 5.1 and in Section 5.2. In Section 5.3 stereopsis is explained, before point matching and epipolar geometry is presented in Section 5.4. In Section 5.5, we will look at how to determine the distance between the UAV and the power lines. Lastly, concluding remarks about finding a trajectory the UAV can use for navigating along the power lines are given in Section 5.6.

5.1 Coordinate transformations

To make use of the power line vectors found in the 2D-image frame, we must first perform a transformation to the 3D-camera frame and then another transformation to the 3D-body frame of the UAV. Hence, two transformations must be done, one 2D-3D transformation and one 3D-3D transformation. To do this, we need to make use of two different types of transformations, namely a Euclidean motion(also called rigid body motion) and a perspective projection. In the coming sections, we will go through both of these, as well as defining the different frames.

Euclidean motion

Euclidean motion defines the family of transformations that preserves the norm and cross product(length and orientation) of any two vectors[21, Chapter 2.1]. It is a transformation that preserves volume. Thus it is also called a rigid body motion.The motion of the origin of the frame is represented by a translation vector, $T \in R^3$, whereas the rotation of the coordinate frame is represented by a rotation matrix, $R \in SO(3)$. The Euclidean motion consists of both a translation, represented by **T**, and a rotation, represented by **R**.

$$SE(3) \equiv \{g = (\mathbf{R}, \mathbf{T}) \mid \mathbf{R} \in \mathbb{R}^3, \mathbf{T} \in \mathbb{R}^3\} \tag{5.1}$$

The Euclidean motion can be represented by the Special Euclidean Group shown in Equation 5.1, and g represents the matrix used to go from camera frame to body frame. Two different transformations, $g_{2,1}$ which transform from the first coordinate frame to the second and $g_{3,2}$ which transforms the second coordinate frame to the third, can be concatenated seamlessly as shown in Equation 5.2[10].

$$\mathbf{X}_3 \;=\; g_{3,2}\,\mathbf{X}_2 \;=\; g_{3,2}\,g_{2,1}\,\mathbf{X}_1 \;=\; g_{3,1}\,\mathbf{X}_1 \tag{5.2}$$

From Equation 5.2 we can easily see that we can find the resulting transformation by using the partial transformations as shown in Equation 5.3.

$$g_{3,1} = g_{3,2}g_{2,1} \tag{5.3}$$

Perspective projection

To make proper use of the information made available by the camera, we must know the camera model. One simple and commonly used mathematical model for the camera is the pinhole camera model, which is illustrated in Figure 5.1. It is based on light from an object only going through a small hole in a barrier between the object and the camera film. This gives a one-to-one mapping between the 3D world and the 2D mapping of it as shown in Figure 5.2[21, Chapter 2.1.5]

The geometry presented in Figure 5.2 is used to transform the image(or pixel) coordi-

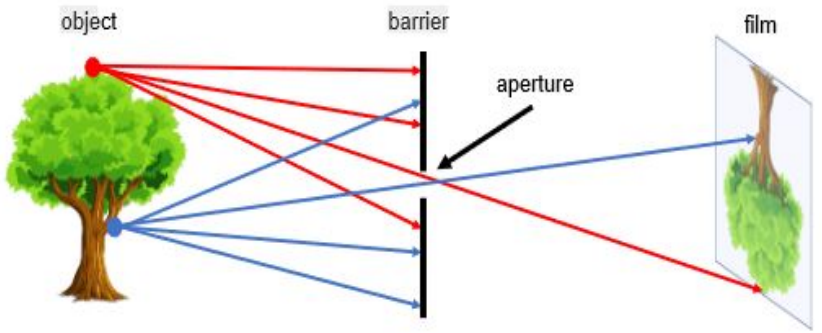


Figure 5.1: The pinhole camera model[13, Figure 1].

nates to the camera frame. The point O_c is the center of the camera, and is the same as the aperture in Figure 5.1. As we can see, pixel coordinates in the image frame often start in the top left corner with positive directions being down and to the right. However, not all image processing tools use this convention, but in such case, a simple inversion must be applied at the end as well.

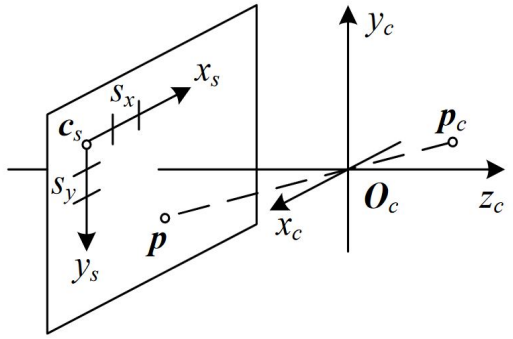


Figure 5.2: Parameters of the pinhole camera model[21, Figure 2.8].

To represent the model of the camera, we use a matrix called the intrinsic matrix, **K**, as shown in Equation 5.4. **K** is found by calibration of the camera. There are many tools for this available, both in OpenCV and the ZED camera software package. The extrinsic matrix is based on the orientation of the camera in space, and together the extrinsic and intrinsic matrices form the camera matrix, **P**, in Equation 5.5.

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & af_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

$$\mathbf{P} = \mathbf{K}g = \mathbf{K}[\mathbf{R} \mid \mathbf{T}] \tag{5.5}$$

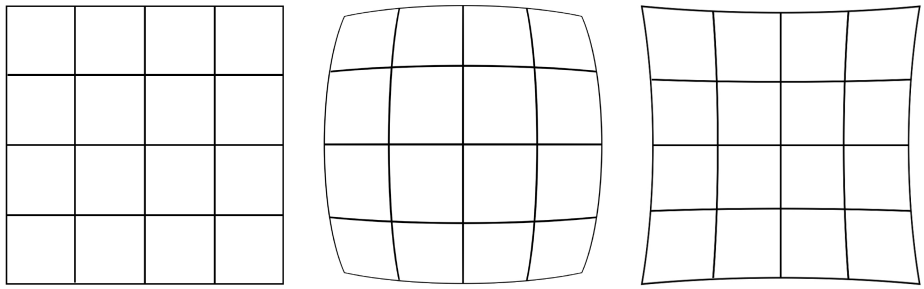
The two parameters S_x and S_y are the pixel size in both directions, they are important when calculating real distances based on distances in the image, as will be discussed in Section 5.5. Usually S_x and S_y are equal.The coordinates (c_x,c_y) marks where the Z_c axis crosses the image plane. Another parameter that is needed to model the pinhole camera is the focal length, **f**, which is the distance between the center of the image and the optical center O_c. If there is any skew between the sensor and the optical axis the parameter **s** must be added to adjust this. The parameter **a** represents the aspect ratio[21, Chapter 2.1.5].

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

The skew parameter and the aspect ratio parameter is however often set to 0 and 1, respectively. This gives us the intrinsic matrix shown in Equation 5.6.

Distortion

The pinhole model assumes an infinitely thin lens, which is not a good approximation in reality. This can cause distortion in the image, the most common one being radial distortion. There are two types of radial distortions shown in Figure 5.3, namely pincushion distortion and barrel distortion[21, Chapter 2.1.6]. The radial distortion appears because the lens has a different thickness at different places and this causes the different parts of the image having different respective focal lengths. If the image frame is also tilted after radial distortion, the image is also tangentially distorted. This can be corrected by using Equation 5.7, Equation 5.8, Equation 5.9 and Equation 5.10 [14]. The first part of Equation 5.9 and Equation 5.10 corrects radial distortion, and the second part corrects tangential distortion.



a) No distortion b) Barrel distortion c) Pincushion distortion

Figure 5.3: Radial distortion

$$x^d = x + \delta_x \tag{5.7}$$

$$y^d = x + \delta_y \tag{5.8}$$

$$\delta_x = x(k_1\, r^2 + k_2\, r^4) + (2\, p_1\, x\, y + p_2(r^2 + 2x^2) \tag{5.9}$$

$$\delta_y = y(k_1\, r^2 + k_2\, r^4) + (2\, p_2\, x\, y + p_1(r^2 + 2y^2) \tag{5.10}$$

The k-parameters must be found by calibration. The r represents how far the pixel is from the center of the image. It is essential for the results that the camera is calibrated properly.

5.2 Coordinate frames

In order to be able to do the coordinate transformations needed to provide the UAV with the trajectory it needs to navigate along the power lines, we must first identify the different coordinate frames and how they relate to each other.

Body frame

The body frame(b-frame) is placed in the geometrical center of the UAV. It follows the standard 6 degrees of freedom(DOF) which represent the free movement in 3D by a rigid-body. This is defined as moving forward/backward along the X-axis, left/right along the Y-axis and up/down along the Z-axis which is referred to as, respectively, surge, sway and heave. Roll is rotating around the X-axis, pitch is rotating around the Y-axis and yaw is rotating around the Z-axis.

Camera frame

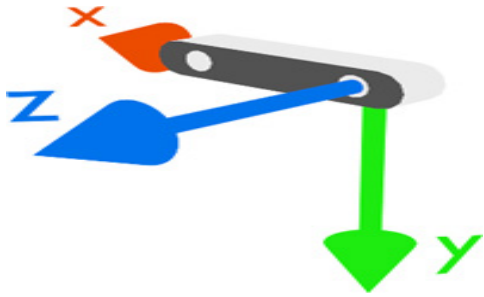


Figure 5.4: Illustration of the ZED camera frame[7].

The camera will be mounted underneath the UAV, tilted slightly downwards. We will assume it is the ZED camera recommended in Section 1. The camera frames origin is placed behind the left lens of the camera, as shown in Figure 5.4. The camera frames X-axis will be parallel with the body frames Y-axis, and the camera frames Y-axis is parallel to the body frames Z-axis. The camera is placed close to the center of the body frame, so the translation between the two frames is negligible within the accuracy needed for power line inspection. Hence only rotation is needed between these two frames.

Image frame

The image frame is the projection of the 3D world onto the 2D image plane. The origin of the image frame is in the top left corner of the image. The y-axis is positive downwards, and the x-axis is positive towards the right. Therefore, the y-axis must be inverted before

it is used for navigation since it does not follow the *right-hand rule*, which the body frame does follow. The transformation between the image frame and the camera frame is done by the camera matrix, **K**, as explained in Section 5.1.

Transformation matrix

The rotation between the camera frame and the body frame consists of one 90 °rotation around the x-axis, one 90 °rotation around the z-axis, and one θ °rotation around the y-axis. The last rotation to correct for is the angle of which the camera is mounted on the UAV body, denoted θ. The Equations 5.11, 5.12, and 5.13 shows the rotation matrices when arbitrarily rotating ϕ degrees around the x-, z- or y-axis respectively.

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix} \tag{5.11}$$

$$\mathbf{R}_z = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

$$\mathbf{R}_y = \begin{bmatrix} cos\phi & 0 & sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{bmatrix} \tag{5.13}$$

Equation 5.15 shows the resulting transformation matrix between the camera frame and the body frame found by equation 5.14.

$$\mathbf{R}_{ib} = \mathbf{R}_x(90)\mathbf{R}_z(90)\mathbf{R}_y(\theta) \tag{5.14}$$

$$\mathbf{R}_{ib} = \begin{bmatrix} 0 & -1 & 0 \\ -sin\theta & 0 & -cos\theta \\ cos\theta & 0 & sin\theta \end{bmatrix} \tag{5.15}$$

Equation 5.16 shows the transformation matrix needed to go from the image frame to

the body frame. It is important not to forget that the image frame has the y-axis pointing downwards, not following the *right-hand rule*, so the y-coordinates still needs to be inverted.

$$\mathbf{F}_b = \mathbf{K}\mathbf{R}_{ib}\mathbf{F}_I \tag{5.16}$$

In Equation 5.16, $\mathbf{F}_b$ represents the body frame and $\mathbf{F}_I$ represents the image frame. If the UAV needs the trajectory vector in world coordinates, this must be done based on the position given by a GPS and the rotation given by a compass or inertial measurement unit(IMU).

5.3 Stereopsis

Stereopsis is how the depth of a point is perceived by looking at it from two different viewpoints, in essence, binocular vision. It is based on the *parallax*, which is the change in the apparent position of a landmark based on the line of sight between the viewpoint and the landmark. Humans do this effortlessly, but in order to apply it to our camera model, we need to break it down mathematically. It is done by dividing the plane into triangles with the known points and unknown landmarks as the vertices and then using trigonometry to calculate the depth. This is illustrated in Figure 5.5. Binocular vision can be obtained by

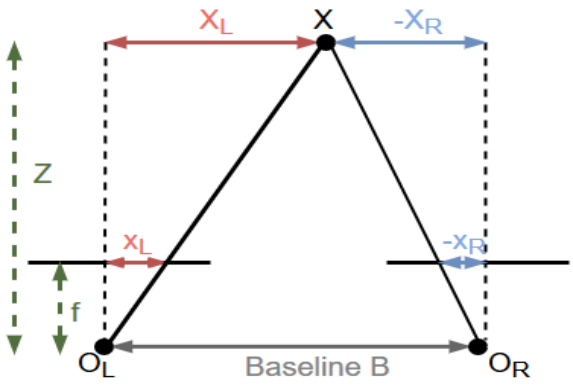


Figure 5.5: Stereopsis

either having two lenses in known relative position to each other or by translating a single camera, taking multiple images. The former is used in this report, and the ZED camera

has known relative position of the two lenses, so the baseline B is known. The landmark X is the point we want to estimate the depth of, whereas x_L and x_R are the corresponding pixel coordinates of X in the two image planes. The center of projections for the two lenses is represented by O_L and O_R. Some known distances form *similar triangles* with the unknown depth, which gives Equation 5.17 and Equation 5.18.

$$\frac{x_L}{f} = \frac{X_L}{Z} \tag{5.17}$$

$$\frac{x_R}{f} = \frac{X_R}{Z} \tag{5.18}$$

By using Equation 5.17 and Equation 5.18 the disparity , the difference of the position of the point in the two images, can be calculated as shown in Equation 5.19.

$$d = x_L + (-x_R) = f\frac{X_L}{Z} - f\frac{X_R}{Z} = f\frac{X_L - X_R}{Z} \tag{5.19}$$

Knowing the definition of the baseline B, shown in Equation 5.20, we see that Equation 5.19 can be simplified. Combining Equation 5.19 and Equation 5.20 yields Equation 5.21.

$$B = X_L + (-X_R) = X_L - X_R \tag{5.20}$$

$$Z = f\frac{B}{d} \tag{5.21}$$

Equation 5.21 shows how the depth of a point can be calculated by knowing the focal length f of the cameras, the baseline B between the lenses, and the landmarks projection to the two image planes. The disparity map is inversely proportional to the depth map, and thus it can be used to show the relative difference of depth in the image without knowing the value of the baseline and the focal length[21, Chapter 11.1]. The disparity map can, however, not be used to find the real depth of objects in the image since it does not know the scale.

Length of baseline

The baseline directly affects the accuracy of the stereo camera's depth vision. Let the estimation error for the depth and disparity estimates be δZ and δd. Equation 5.22 is the same as 5.21, except with error estimates.

$$Z + \delta Z = f\frac{B}{d + \delta d} \tag{5.22}$$

$$\delta Z = -f\frac{\delta d}{d(d + \delta d)}B \tag{5.23}$$

$$d^2 \gg d\delta d \tag{5.24}$$

$$\delta Z \approx -f\frac{B}{d^2}\delta d \tag{5.25}$$

$$\mid \delta Z \mid = \mid f\frac{B}{d^2}\delta d = \mid \frac{Z^2}{fB}\delta d \mid \tag{5.26}$$

Equation 5.23, Equation 5.24, and Equation 5.25 shows the steps needed going from Equation 5.22 to Equation 5.26. From Equation 5.26 it is clear that having a large baseline gives more accurate depth estimates.

5.4 Point matching

Section 5.3 might give the impression that determining the depth of objects in the image is a simple triangulation, but that is only true if the point correspondence in the image is already known. Finding corresponding points in two images, however, is challenging. In this section, the two main methods for matching, intensity-based and feature-based, is presented.

Intensity-based matching

Intensity-based matching assumes that there is only a small change from one frame to the next, which leads to points in the image having a small displacement between the frames. This is a dense matching method since nearly every pixel is accounted for. This requires a

great deal of computing power at each frame, which in turn means that the baseline should be small for this approach to work. Restrictions on the baseline, on the other hand, affects the accuracy and range of the camera, as shown in Equation 5.22. Additionally, intensity-based matching assumes a *lambertian reflectance model*. This means that every point has the same intensity no matter what point of view it is seen from, which is almost never the case.

Feature-based matching

The feature-based matching method reduces the numbers of matches needed to be done compared to intensity-based matching, by not trying to match every pixel. Instead, it finds points with specific characteristics, called feature points, and matches them together. Several points may have similar characteristics, and then several matchings can be plausible. Using a heuristic method together with epipolar geometry is the most common way of deciding which feature points corresponds to each other by assuming minimal displacement is most plausible. Feature-based methods are not as dense as the intensity-based methods. This becomes clearer around objects with few feature points, such as white walls, where feature-based methods struggle with finding the depth. SIFT and ORB are two well-known feature descriptors. While SIFT is the most known, ORB is by far the one being used the most, since SIFT is a licensed method.

Intensity-based matching methods are less accurate and computationally more expensive than feature-based methods[14]. Additionally, intensity-based methods limit the baseline and therefore also the accuracy of the depth. These things considered, a feature-based method is suggested used. Feature-based methods make use of epipolar geometry for more accurate and faster matching of the feature points, which will be explained in Section 5.4.1.

5.4.1 Epipolar geometry

Epipolar geometry can be used for faster feature matching. Every point on the O_LX-line in Figure 5.6 projects to the same point, x_L, in the left image plane. This is why we cannot

perceive depth from just one camera without knowing the size of an object in the image, and in that way acquire scale. If we want to match the point x_L in the left image to a point in the right image, we can use the O_LX-line to narrow the search area down to just a line in the right image instead of the entire image. This is done by projecting the O_LX-line to the right image plane. This line, L_R in Figure 5.6, is called the epipolar line of x_L. The epipolar constraint states that the corresponding point, x_R, to x_L must lie on x_Ls epipolar line,L_R. The plane formed by the 3D-point X and the two center of projections, O_L and O_R, is called the epipolar plane. The two epipoles, e_L and e_R, are the points where the camera center is projected onto the other image plane[21, Chapter 11.1]. The projection

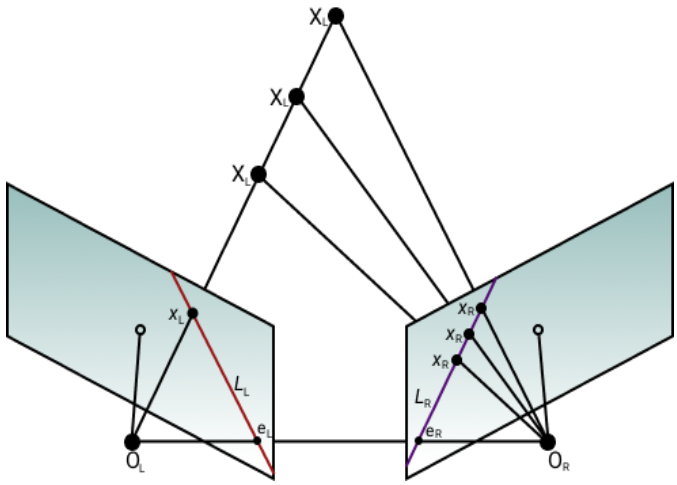


Figure 5.6: Epipolar geometry

of a point in one image to the epipolar line in the other image is done using the essential matrix, denoted E, as shown in Equation 5.27 and Equation 5.28.

$$L = E^T \hat{x} \tag{5.27}$$

$$E = R\hat{T} \tag{5.28}$$

R is the rotation between the left and the right camera lenses, and $\hat{T}$ is the skew-symmetric matrix representation of the translation between the left and the right camera lenses, as stated in 5.29 and 5.30. The same goes for x and $\hat{x}$.

$$T = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T \tag{5.29}$$

$$\hat{T} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{5.30}$$

Since the essential matrix consists of the extrinsic parameters, the epipolar line is actually found relative to the camera coordinate frame. As previously mentioned in Section 5.1, the intrinsic matrix K does the mapping between the camera frame and the image frame, as shown in Equation 5.31.

$$x^c = Kx^i \tag{5.31}$$

The intrinsic matrix is represented by K, whereas x^c and x^i represents the point x in the camera frame and the image frame.

$$x_R^c E x_L^c = 0 \tag{5.32}$$

Combining Equation 5.31 and 5.32 yields

$$x_R^i F x_L^i = 0 \tag{5.33}$$

where the fundamental matrix F is defined as

$$F = K^{-T} E K^{-1} \tag{5.34}$$

The fundamental matrix does the same as the essential matrix but in the image frame's coordinates. It is possible to estimate the fundamental matrix using known point correspondences. This is not as accurate as knowing the intrinsic parameters, so it should only be done if the intrinsic parameters are unknown. Equation 5.32 and Equation 5.33 are actually *coplanarity conditions*, stating that all the points must be on the epipolar plane.

5.5 Finding distance to line

In this section, three methods for determining the distance to the power lines from the UAV is presented. Two of these methods require a stereo camera, whereas the method presented first works on monocular cameras.

Depth estimate from monocular image

As presented in Section 5.3 and Equation 5.21, the focal length, the depth of the power lines, and the distance between two power lines in real-life and in the image forms two similar triangles. Given that we know the true distance between two power lines we can find the distance from the camera to the power lines by geometry. This similar triangle can be expressed by Equation 5.35.

$$Z = f\frac{d}{D} \tag{5.35}$$

This is called *the basic proportionality theorem.* The focal length is represented by f, and the depth is represented by Z, the distance between the power lines in the image is represented by d, and the real-life distance between two power lines is represented by D as shown in Figure 5.7. The distance between two power lines in the image can be calculated by counting pixels and using the parameters S_x and S_y, found during calibration, to compute it in meters. This method can be used for all types of power masts and power lines, but the distance between the lines must be known. Additionally, at least two power lines must be detected. This does not need to be done for every pixel, but rather in constant

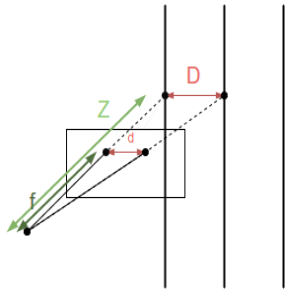


Figure 5.7: Basic proportionality of power lines and their projection to the image plane.

intervals along the line, and then the depth between these points can be assumed following

several straight lines.

ZED depth map

The ZED camera generates a depth value for every pixel in the image[6]. It is visualized by a grayscale image where pixels with value 255 are closest, and pixels with value 0 are furthest away. The depth map is calculated with the left image's image center as the origin of the coordinate frame, recall Figure 5.4. The depth of the power lines can be extracted by either running a simple least squares method along the lines with the depth values as input. Alternatively, RANSAC can be used. The depth map can also be used to find the power lines, as discussed in Section 2.4.

ZED stereo images

[22] proposed a method called SPMEC(Semi Patch Matching based on Epipolar Constraints). It uses PCNN and histogram filtering as presented in Section 3.2 and Section 3.3 to extract the 2D vectors of the power lines, and then SPMEC to find the depth values of the power lines. Figure 5.8 shows how the epipolar lines from points on a power line is projected to the other image. The detected power lines and the projected epipolar line are

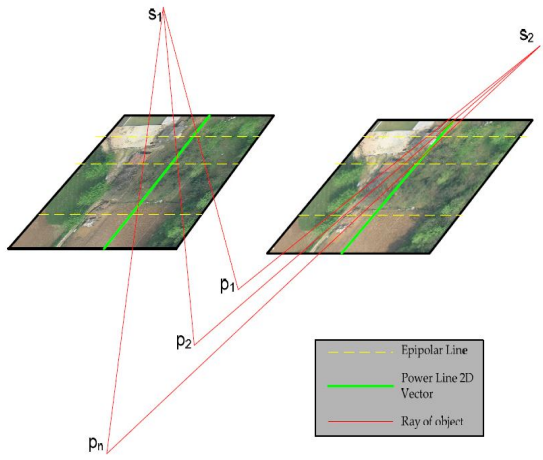


Figure 5.8: Epipolar constraints between power lines[22].

given in Cartesian coordinates, as showed in Equation 5.36.

$$y = ax + b \tag{5.36}$$

Epipolar lines from the left image do not need to be projected for every point along it, but rather periodically along the line. Using epipolar lines simplifies the search for a match along the lines significantly. The cross point between the epipolar line in the right image projected by a point in the left image and the corresponding power line in the right image is done by using Equation 5.37 and Equation 5.38.

$$x = \frac{b_{EL} - b_{PL}}{a_{PL} - a_{EL}} \tag{5.37}$$

$$y = a_{PL/EL}x + b_{PL/EL} \tag{5.38}$$

The slope of the power line and the epipolar line is given by a_{PL} and a_{EL} respectively. The same applies to the y-intercept, b, of Equation 5.37 and Equation 5.38. Even though it is not necessary to find point correspondences on every point along the lines, finding more correspondences makes the system more robust against mismatches. As many point correspondences as is allowed while still running real-time should be done, it is also important to prioritize finding matches in the lower part of the image since this is the closest part of the power line.

5.6 Concluding remarks

In this chapter, it is shown how to go from detected power lines to calculating a trajectory the UAV can navigate by. In Section 5.1 the theory behind coordinate transformations was presented. In Section 5.2 the different coordinate frames relevant to the UAV and how to go from one to the other was presented. Stereopsis was presented in Section 5.3, whereas point matching was presented in Section 5.4. In Section 5.5, three methods for how to derive the distance between the power lines and the UAV is presented. The first and simplest method proposed make use of that the distance between two power lines is static and known, and calculates the depth value of the power lines by using similar triangles. If

this method is used, a stereo camera is not necessary for detecting and navigating along power lines. If the depth map given by the ZED camera is accurate enough along the power lines, the depth values of the power lines can be acquired quite easily by running a least squares method along the lines. The most complex method is the third one, used in [22], which does feature-based matching only on the power lines. The simplest method based on similar triangles is suggested since it is not computationally demanding and enables the possibility of using a monocular camera, which can be beneficial when the weight of the UAV is important. This is an elegant method, which is easy to implement.

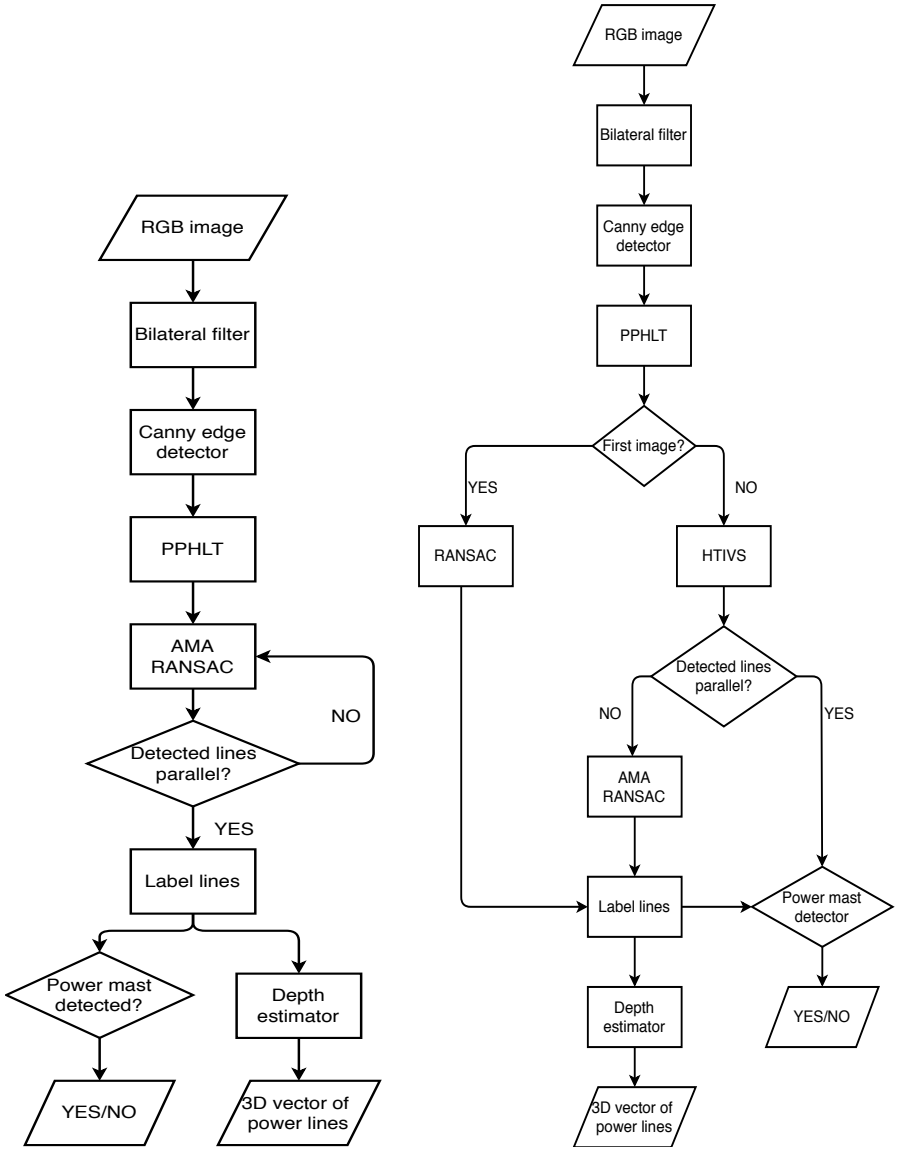
Chapter 6

Summary and conclusions

In this thesis, an elegant and easily implementable method for autonomous power line following suitable for light-weight and off-the-shelves UAVs using only the information from a monocular camera was developed. Unlike most other existing methods, this new method does not require heavy onboard computational resources or involved implementation. Instead, it makes use of a variety of tried-and-true algorithms and mathematical concepts in a way that, to the author's knowledge, has not been done before. UAVs and more specifically multirotor UAVs are well-suited for power line and power mast inspection since they bring the benefits of low cost, the ability of hovering, higher safety, and higher accuracy. The problem of autonomously navigating along power lines can, as we have seen, be divided into three subtasks: power line detection in the image, determining the distance between the power lines and the UAV, and power mast detection.

For detection of power lines in an RGB image, two methods were proposed. AMA RANSAC is based upon the well-known RANSAC algorithm for three lines. To improve the run time it starts with a rather low number of attempts, and checks if the lines detected are approximately parallel. If they are not, RANSAC is run again with a higher number of attempts, before it again checks for parallelity. This method proved to be highly accurate, but too slow for running real-time on the computer used during this thesis, on a 30 fps 1080x1920 p video. The other method suggested is a combination of the AMA RANSAC

and HTIVS. HTIVS was proposed in the project thesis this master is based upon, and it is inspired by the voting scheme used in the Hough line transform. The combination of the two, AMA RANSAC + HTIVS, is more efficient than AMA RANSAC, but not as accurate. If the UAV has computational resources for running AMA RANSAC or can navigate on trajectories from for example every fifth frame, AMA RANSAC should be used. They both detect three lines, which enables detecting the distance from the power lines and to the UAV without needing a stereo camera. For determining the distance between the power lines and the UAV, three methods were discussed. By using that we know the distance between the power lines based on how much voltage they carry, the distance can be calculated by simple triangulation. The power mast detector is also performed on an RGB image, which means that it is up to the power mast inspection method to decide whether or not a stereo camera is necessary. An overview of the two methods proposed is given in Figure 6.1. As is clear from the figure, AMA RANSAC is far easier to implement than AMA RANSAC + HTIVS.



(a) AMA RANSAC (b) AMA RANSAC + HTIVS

Figure 6.1: Overview of the proposed methods.

6.1 Recommendations for future work

This master's thesis was carried out over 5 months of the spring of 2019, and during this time several ideas for improvements and extensions to the suggested methods arose. In this section, the suggestions for future work are given.

Short term

Most importantly, more testing should be done. Live navigation based on the calculated trajectory must be tested. Datasets appropriate for testing the method for determining the distance between power lines and UAV must be gathered, and the method rigorously tested. Additionally, the two proposed methods should be tested in different environments such as other landscapes, other seasons and weather conditions.

Medium-long term

This report proposes a simple power mast detector, but a more refined method should be developed to decide when the UAV should switch to power mast inspection navigation. Additionally, the UAV should perform inspection while following the lines. The UAV should have a dedicated method for detecting faults that can occur on the power lines, such as ice or snow loads and vegetation encroachment.

Long term

Future work in the long term should address a method for navigating around power masts and inspecting each and every one of its components. This, together with the inspection of power lines and further detection of power masts, decides whether or not a monocular camera is sufficient, or if a stereo camera or a monocular camera with another sensor such as ToF is needed. If the UAV is able to navigate along power lines, around power masts and inspecting each and every component by itself, autonomous power line and power mast inspection are achieved.

Bibliography

[1] Kile - kvalitetsjusterte inntektsrammer ved ikke levert energi. https://www.sintef.no/kile-kvalitetsjusterte-inntektsrammer-ved-ikke-lev/. Accessed: 2018-04-28.

[2] Kvalitetsinsentiver-kile. https://www.nve.no/reguleringsmyndigheten-for-energi-rme-marked-og-monopol/okonomisk-regulering-av-nettselskap/reguleringsmodellen/kvalitetsinsentiver-kile/. Accessed: 2018-04-26.

[3] Opencv - canny edge detector. https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html. Accessed: 2018-15-10.

[4] Opencv - gaussian downsampling. https://docs.opencv.org/2.4/doc/tutorials/imgproc/pyramids/pyramids.html. Accessed: 2018-19-11.

[5] Opencv - image smoothing. https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/

gausian_median_blur_bilateral_filter.html. Accessed: 2018-15-10.

[6] Zed - frame. https://www.stereolabs.com/docs/depth-sensing/. Accessed: 2018-06-05.

[7] Zed-frame. https://www.stereolabs.com/docs/positional-tracking/coordinate-frames/. Accessed: 2018-03-12.

[8] M. Bruch, V. Münch, M. Aichinger, M. Kuhn, M. Weymann, and G. Schmid. Power blackout risks. *CRO forum*, pages 3,9,23,26–29, 2011.

[9] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[10] D. Cremers. Lecture slides from computer vision 2: Multiple view geometry, tum, Spring 2018.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.

[12] V. B. Gjærum. Autonomous navigation along power lines, project thesis ttk4550, ntnu. Autumn 2018.

[13] K. Hata and S. Savarese. Course notes for lecture 1 in cs231a: Computer vision, from 3d reconstruction to recognition, winter 2018.

[14] M. Leonardi and S. Haugo. Lecture slides for ttk25 computer vision for control, ntnu, Fall 2018.

[15] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang. Towards automatic power line detection for a uav surveillance system using pulse coupled neural filter and an improved hough transform. *Machine Vision and Applications*, 21(5):677–686, 2009.

[16] A. H. Maini R. Study and comparison of various image detection techniques. 3, n.d.

[17] K. J. Matas J, Galambos C. Robust detection of lines using the progressive probabilistic hough transform. 78, April 2000.

[18] D. Roverso, V. Nhan Nguyen, and R. Jenssen. Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. 99, 07 2018.

[19] G. Sharma, Gupta, and R. Malik. Shape based object recognition in images: A review.

[20] A. Sobral and A. Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21, 2014.

[21] R. Szeliski. *Computer Vision: Algorithms and Applications (Texts in Computer Science)*. Springer, 2010.

[22] Y. Zhang, X. Yuan, W. Li, and S. Chen. Automatic power line inspection using uav images. *Remote Sensing*, 9(8):824, 2017.

[23] L. W. C. S. Zhang Y, Yuan X. Automatic power line inspection using uav images. August 2017.