NTNU – Trondheim
Norwegian University of
Science and Technology

Autonomous Wind Blade Inspection

Hans Erik Heggem

June 2017

MASTER'S THESIS

Department of Engineering Cybernetics

Norwegian University of Science and Technology

Supervisor: Rune Storvold

Abstract

This report addresses autonomous inspection of wind blades using a UAV by proposing to use computer vision combined with structured light to accurately segment and detect the wind blade edges. It will be proven that this method is invariant to both rotation and scale, while it is computationally efficient for use on real-time applications. The blade edges will be detected according to the segmented area of the blade, and identified as Hough transformed lines to easily develop a manouvering scheme which intends to follow the blade edges from root to tip, while maximizing the view of the blade without loosing view of the respective edges. It will also be shown that the method detects the blade tip, regardless of rotation and scale.

Furthermore, the structured light will be utilized to efficiently compute feature point matches from a stereo vision system. These matches will be used to conduct 3D reconstruction of the respective feature points using 2D triangulation. This will enable transformation of coordinates between the image frame and camera frame, including a real-time estimate of the distance between the blade and drone.

Additionally, all relevant source code mentioned in this report may be found open-source at the authors personal github repository, reachable at following web address.

- https://github.com/hansehe/Wind-Blade-Inspection

Preface

This thesis was conducted during the spring of 2017, and concludes the final step of my studies towards a M.Sc. degree in Engineering Cybernetics and Robotics at the Norwegian University of Science and Technology (Trondheim, Norway).

Special thanks to Professor Rune Storvold for serving as my supervisor, and to Norut Northern Research Institute in Tromsø for providing equipment, hardware and technical assistance.

I would also like to pay my gratitude to Senior Scientist Agnar Sivertsen for challenging my ideas while aiding with helpful advices, and most importantly I would like to pay my sincere gratitude to my better half, Ingvild, and to my family for support and motivation throughout my studies.

Tromsø, 2017-06-09

Hans Erik Heggem

Contents

Chapter 1

Introduction

1.1 Background & Motivation

The global wind power industry is rapidly growing, and will play an essential role in supplying the world with renewable energy. More demand for wind power requires larger wind farms and larger windmills, which both generates the necessity for more cost efficent maintenance solutions. In general, maintenance costs amounts to 1.5% to 2% of the original annual investment of a modern windmill [2], and most of the maintenance costs are due to regular servicing such as conducting a rope descent to inspect exposed components for detecting defects on the wind blades. In recent time, UAV investments are on the rise due to the vast possibilites within surveillance, search and rescue and other fields where a UAV may assist workers to provide better and more cost efficient solutions. Such a solution may assist workers to inspect windmills by automatically inspecting the windmills using a UAV to provide high quality recordings, which later are inspected in depth by the field engineers on the ground. This greatly cuts costs and minimizes the risk of personal injury, and provides the ability for field engineers to focus their time on repairing the windmills.

Some companies, such as Aibotix, offer solutions for autonomous windmill inspection using industrial UAVs with various sensors, including video cameras and thermal imagers. Another inspection technique is to use high-definition cameras based on the ground to conduct a remote inspection of the windmill. This thesis will address autonomous inspection of wind blades using a UAV equipped with video cameras and a laser for accurate manouvering along the wind blades.

1.2 Literature Survey

Research has been done during the recent years on automatic windmill inspection using UAVs, and companies such as the Aerialtronics corporation [3] and the AutoCopter Corporation [1] provide solutions using either a drone or an unmanned helicopter. Typical solutions commonly uses infrared recordings [1] to detect heat signatures on the windmill or a GPS combined with a LiDAR sensor [1] to create a 3D map of the windmill, and both methods are used to manouver the UAV around the windmill. Another approach is to address the issue using computer vision, which gives the benefit of using the cameras for inspection as well as a tool for navigation. A solution for targeting a windmill and manouvering towards it was proposed by Stokkeland et al. [50], who used computer vision techniques such as the Canny edge detection and Hough transform to detect features on the windmill. Moreover, Høglund [17] addressed the issue of autonomous inspection of windmills and buildings using optical flow and Hough transform to locally navigate a UAV. This thesis will continue addressing the issue of windmill inspection by focusing on a manouvering solution along the wind blades using computer vision, including two cameras to provide stereo vision. Moreover, the problem will be treated using structured light, as projected by the laser, to segment the blade with high accuracy, and it will be shown that this method will provide an easy way of manouvering the UAV at a given distance to the blade.

1.3 Objectives

The main objectives of this report are as follows:

1. Literature review on relevant computer vision techniques for processing images to detect and identify features on the wind blade.

2. Define a stereo vision technique suitable for reconstructing 3D feature points.

3. Define a method for segmenting the wind blade by detecting the blade edges.

4. Propose a manouvering plan along the wind blade, and conduct simulations.

1.4 Limitations

The report will do a thorough analysis on how to accurately segment a wind blade from its surroundings and how to conduct 3D reconstruction using stereo vision. Additionally, the methods are implemented so that necessary simulations of the respective methods could be conducted. Moreover, the report includes a manouvering solution for following the edges of a wind blade, where only the key manouvering methods are implemented and simulated. Some of the manouvering methods requires implementation of a Gimbal, GPS or sonar which could not be done during the limited time this thesis was conducted.

1.5 Approach

The objectives will be addressed by first reviewing relevant computer vision techniques, including techniques about edge processing, line detection, feature detection and stereopsis. These techniques will constitute the basis for developing methods to solve the respective objectives. The first objective will be met with a solution for detecting the projected feature points of the structured light, whereas the respective feature points will be used to conduct 3D reconstruction and wind blade segmentation. Finally, the wind blade segmentation will be used to accurately detect the blade edges, where the coordinates of the blade edges will constitute the basis for the manouvering plan.

1.6 Structure of the Report

This report continues the work of Heggem [16] who presents a theoretical solution on how to manouver a UAV along a wind blade, so much of the literature review and manouvering solutions are based upon this work. Moreover, the report is organized as follows. Chapter 2 reviews relevant computer vision theories, including feature detection and the concept of stereopsis. Moreover, this chapter will discuss and conduct tests to define a proper method to detect and match feature points of the structured light, and continue with 3D reconstruction of the respective feature points to verify the feasibility of using stereopsis to measure distance to a wind blade. Chapter 3 will review design and dimensions of a conventional wind blade, before presenting a method for detecting the blade edges in chapter 4 and proposing a manouvering plan with simulations in chapter 5. Additionally, the kalman filter will be reviewed in chapter 6 to present it as a suitable tool for fault recovery. Finally, chapter 7 presents the program flow of the application used to simulate blade manouvering, including review of relevant hardware.

Chapter 2

Literature Review of Computer Vision

2.1 Camera Systems

2.1.1 Depth of Field & the Thin Lens Model

A simplified imaging device consist of an optical system, a sensor and an aperture. The optical system is formely known as the lens and focuses light rays coming from one scene point to converge at a point on the sensor, while the aperture controls the amount of light entering through the lens. Moreover, the sensor detects the incoming light rays through photoreceptors such as a CMOS sensor or a CCD sensor. Together, these devices constitute the depth of field which refers to the focus point at where scene objects needs to be to remain in focus. Figure 2.1 illustrates the thin lens model where all rays originating from a scene point P intersects at exactly the same point p on the sensor, whereas f is the focal length, O is the lens center and F is the left and right focal point.

Moreover, by using the thin lens model we can define the basic and fundamental equations for a thin lens as follows in equation (2.1) and (2.2), respectively.

$$\frac{Y}{\hat{Z}} = \frac{y}{\hat{z}} \tag{2.1}$$

$$\frac{1}{\hat{Z}} + \frac{1}{\hat{z}} = \frac{1}{f} \tag{2.2}$$

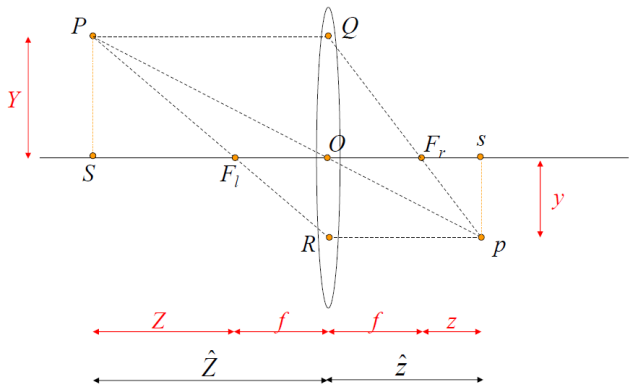


Figure 2.1: The thin lens model [22].

The basic equation states that the height in an image is proportional to the height of the object, while the fundamental equation states that an object point maps to the corresponding image point. Additionally, from the thin lens model it is understood that the depth of field depends on the focal length of the lens and the aperture size, which is illustrated in figure 2.2 where several rays intersect at the same area b which causes a blurred image.

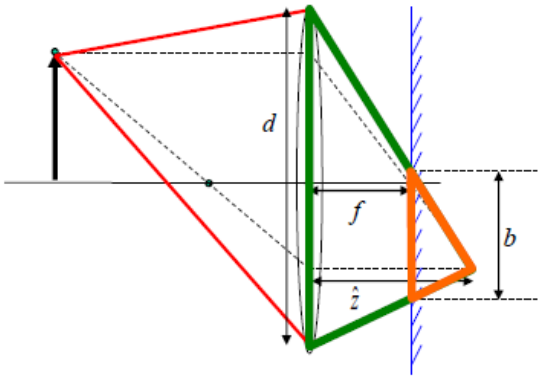


Figure 2.2: Depth of field [22].

The aperture d may be related to the blurred area b by geometry which is stated by equation (2.3). Equation (2.4) rearranges equation (2.3) using the fundamental equation for thin lenses to show that the focus depends solely on the aperture and focal length.

$$\frac{\hat{z}}{d} = \frac{\hat{z} - f}{b} \tag{2.3}$$

$$
\begin{aligned}
b &= \frac{d(\hat{z}-f)}{\hat{z}} \\
&= d\left(1-\frac{1}{\hat{z}}\right) \\
&= d\left(1-f\left(\frac{1}{f}-\frac{1}{\hat{Z}}\right)\right) \\
&= d\left(1+\frac{f}{\hat{Z}}\right) \\
&= \frac{df}{\hat{Z}}
\end{aligned}
\tag{2.4}
$$

Additionally, the field of view according to the focal length and sensor size is estimated by considering the basic equation (2.1) of a thin lens model, and approximating that the focal length is equal to the image distance $\hat{z}$, when considering that the working distance $\hat{Z}$ is considerably larger than the image distance. The field of view is then defined by the following equation (2.5).

$$
Y = \hat{Z}\frac{y}{f} \tag{2.5}
$$

Moreover, the fan angle according to the field of view is illustrated in figure 2.3, and may be found using equation (2.6).

$$
\varphi = 2\arctan\left(\frac{Y}{\hat{Z}}\right) \tag{2.6}
$$

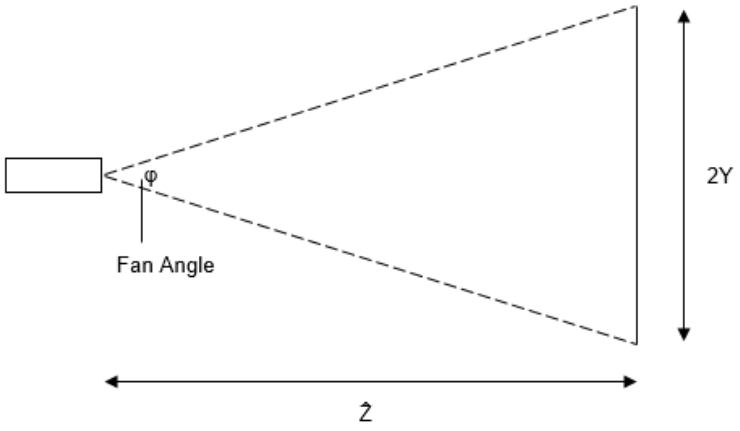


Figure 2.3: Fan angle.

2.2 Image Manipulation

Image manipulation is the first step towards image processing. This involves reducing the image size for lighter computation and filtration of Gaussian noise for the purpose of removing unwanted noise, before moving on with an image processing method such as detecting edges.

2.2.1 Gaussian Filtering

Noise in an image is usually considered Gaussian and may cause disruptions when the gradient estimation of the gray-level pixels are computed. A linear spatial filter, such as a Gaussian filter, is a common technique to smooth and remove Gaussian noise in an image. The technique involves estimating a constant matrix, called a kernel, that is convolved with the original image to compute a filtered image. The kernel expresses the weight of each neighboring pixel. As an example, to compute a simple average filter, also called a box filter, then the kernel values will be given equal average weight and the resulting pixel will be the average estimate of its neighboring pixels.

In general, a linear spatial filtered image I_A is computed by convolving the original image I with a constant kernel matrix A defined by equation (2.7) [51, p. 55] .

$$I_A(i,j) = I * A = \sum_{h=-\frac{M}{2}}^{\frac{m}{2}} \sum_{k=-\frac{M}{2}}^{\frac{m}{2}} A(h,k)I(i-h,j-k) \tag{2.7}$$

The convolution of the original image I and kernel A is easily computed in the frequency domain by the convolution theorem:

$$\mathscr{F}\{I * A\} = \mathscr{F}\{I\}\mathscr{F}\{A\} \tag{2.8}$$

The filtered image is finally computed by estimating the inverse Fourier transform from the result of equation (2.8).

The Gaussian filter is a linear spatial filter, where the kernel is estimated based on a 2D normal density function. The density function gives lower weight to pixels further away from the center pixel, and will therefore be more able to preserve edges, compared to a simple box filter.

The Gaussian kernel distribution in 2D has the form:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.9}$$

The width of the kernel, usually of odd number, and σ, also called the standard deviation of the kernel distribution, must be decided upon. A high standard deviation of the Gaussian σ increases the weight of pixels closer to the central pixel in the kernel. Therefore, the σ should be of higher value to preserve edges in a highly Gaussian noisy picture. There are no fixed solution on how to estimate these values, but the width w and the standard deviation of the Gaussian σ may follow a standard relation, as defined by equation (2.10) [51, p. 59].

$$\sigma_w = w/5 \tag{2.10}$$

2.2.2 Gaussian Pyramid Downsampling

Modern cameras usually produce large images of high resolution that increases computation cost when processing the images, and an image of high resolution may contain too many details that will bee seen as noise when processing edge detection on the image.

The Gaussian pyramid method is used to downsample an image g_0 by reducing spatial density and resolution by convolving image g_0 with a symmetric weighted Gaussian kernel $w(m,n)$ and removing every even-numbered row and column. The resulting downsampled image g_1 will be half the size of the original image, with proportionally lower resolution. Eventually, the Gaussian pyramid generates a set of images, $g_0, g_1, ..g_n$, with image g_n having the lowest spatial density and resolution.

The Gaussian pyramid process is presented by equation (2.11) [43] , and computed by convolution.

$$g_n(i,j) = \sum_{n-2}^{2} \sum_{m-2}^{2} (w(m,n) g_{n-1}(2i+m, 2j+n) \tag{2.11}$$

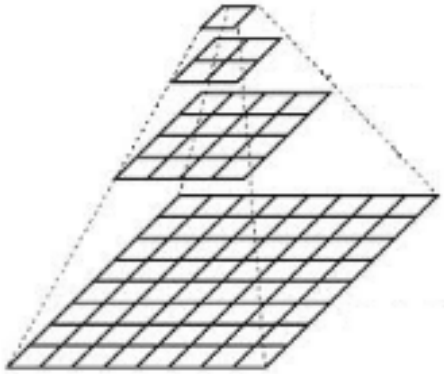


Figure 2.4: Gaussian pyramid [40]

Each level is of half the resolution computed by the level below.

2.3 Edge Processing

Edge processing is a fundamental tool when it comes to computer vision. In this section we will discuss different edge detection tools, and proceeed on processing the detected edges for the purpose of locating lines and corners on a wind blade.

2.3.1 Edge Detection

The concept of edge detection is to find a sequence of connected edge elements. Connected edge elements, or edgels, are defined as series of connected pixels, where each pixel has a sharp and locally maximum gray-level gradient in one direction with a low gray-level gradient in the perpendicular direction.

An edge detection method usually involves three steps:

1. Noise smoothing
2. Edge enhancement
3. Edge localization

The first step is carried out to suppress as much noise as possible, while maintaining the edges. Image noise is usually modelled by a white Gaussian zero-mean stochastic process [51, p. 53], since the noise is thought to be random, and is usually suppressed by a Gaussian filter.

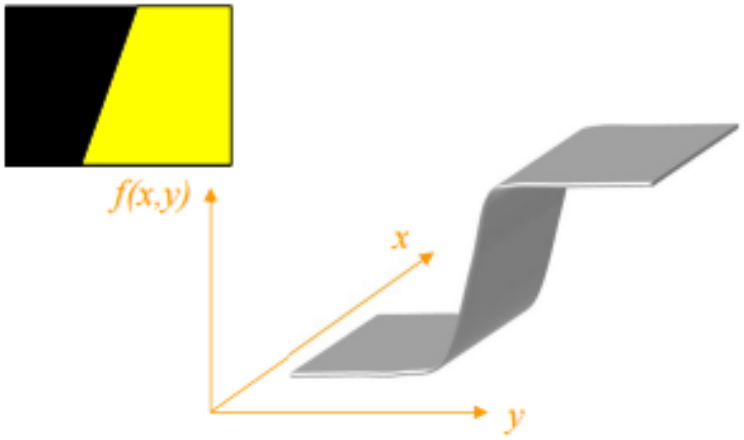


Figure 2.5: Edgel [25]

Sharp, local maximum gray-level gradient in y direction, and low gray-level gradient in x direction.

The second step enhances possible edges by computing the gray-level gradient component and the orientation of the edge normal. There are several methods on how to enhance edges from an image, such as the Sobel filter method or the Gaussian edge filtering method. Edge localization is the final step and involves filtering the resulting edge map by removing edges that are caused by noise. This step is carried out by a non-maximal suppression method that suppresses edged pixels if they are not the local maxima in the gradient direction, and a hysteresis threshold method to remove noisy random edges.

All of the edge detection steps mentioned are carried out by a commonly used Canny edge detection algorithm. The following sections will explain and discuss how to implement a Canny edge detection algorithm to detect edges on a wind blade.

The next step after localizing the edges on a wind blade, will be to carry out an edgel linking method to localize the direction of connected edgels, so that it will be possible to decide upon a direction path to follow along a given wind blade. The Hough transform method for lines is a commonly used edgel linking method, and is used to group straight edges that should be connected, but are disconnected due to noise. Refer to section 2.3.2 for an explanation of the Hough line transform.

Canny Edge Detector

The Canny edge detector is mainly carried out be four steps:

1. Noise smoothing
2. Edge enhancement
3. Non-maximal suppression
4. Hysteresis thresholding

The first step is usually carried out by a Gaussian filter, as mentioned in section 2.2.1, to remove Gaussian noise. The second step involves estimating edge strengths by using an edge enhancement algorithm, such as the Sobel or Gaussian edge filtering method, which are explained in section 2.3.1 and 2.3.1.

The final two steps is carried out to suppress unnecessary non-maximal pixels and remove noisy edges, which results in a clearer and more accurate edgel map.

The non-maximal suppression method helps to reduce broad filtered edges to single-pixel-wide paths so that the edgel map becomes more clear. The algorithm works through each non-zero edge pixel and finds the normal to each edge. If the edge pixel is smaller than one of its neighbors in the normal direction, along the edgel direction, then that means it's not a local maxima and the edge pixel is suppressed to zero.

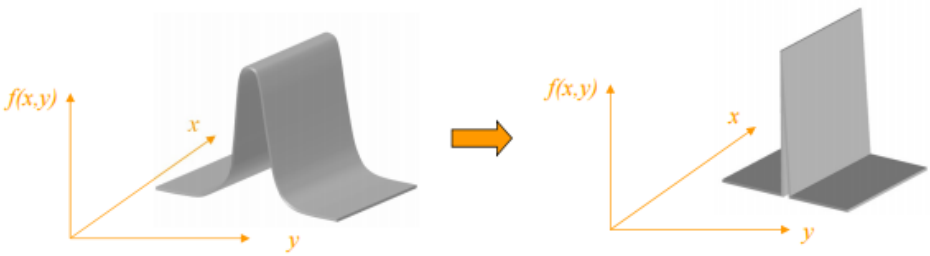


Figure 2.6: Non-maximal suppression [25]

Edgels are thinned by suppressing non-maximal edge points in the gradient direction.

The edgel map at this point will still contain noisy edges caused by Gaussian noise, and the goal is to remove these disturbances without corrupting real edgels. This is solved by introducing hysteresis thresholding, which in contrast to normal thresholding with a single threshold value, has a high and low threshold value. Edge pixels higher than the high threshold are set to one. However, edge pixels with lower magnitude than the high threshold, but still higher than

the low threshold is set to one if the neighboring pixels in the normal direction of the edge gradient have been set to one previously, or it will be suppressed to zero. Edge pixels which are lower than the low threshold value are automatically set to zero.

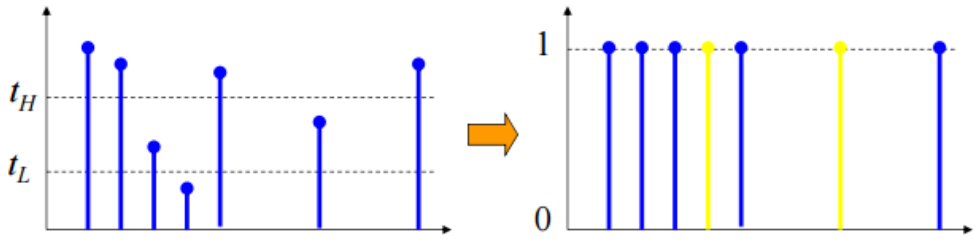


Figure 2.7: Hysteresis thresholding [25]

Blue values are set to one and yellow values are suppressed.

Gaussian Edge Filtering

The Gaussian edge filter is a an approximation of the 1^{st}-order derivatives of a Gaussian. The advantages of using a Gaussian edge filter is that the Gaussian filtration and edge enhancement is computed simultaneously, since a Gaussian weighted kernel in the horizontal and vertical direction is used.

The Gaussian edge filter creates a horizontal $h_x(x,y)$ and vertical $h_y(x,y)$ kernel. The resulting gradient magnitudes, G_x and G_y, in horizontal and vertical direction is found by convolving the input image $f(x,y)$ with the corresponding horizontal and vertical kernels [25].

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.12}$$

$$h_x(x,y) = \frac{\partial h}{\partial x} = -\frac{x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.13}$$

$$h_y(x,y) = \frac{\partial h}{\partial y} = -\frac{y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.14}$$

$$G_x = f * h_x \tag{2.15}$$

$$G_y = f * h_y \tag{2.16}$$

Finally, the resulting edgel map is computed by combining the gradient magnitudes in vertical G_x and horizontal G_y direction to approximate a resulting edgel map. As mentioned, the

Gaussian edge filter enables the possibility to approximate the gradient magnitude in all directions or in a specific angle, as defined by equation (2.17) and (2.18) [25].

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.17}$$

$$G(\theta) = |G_x \cos(\theta) - G_y \sin(\theta)| \tag{2.18}$$

Equation (2.18) is derived by considering a rotation around angle θ to compute a desired angle for the kernel in equation (2.12). The rotated kernel $h_\theta(x', y')$ is found by replacing (x, y) with the rotated values (x', y'), as stated in equation (2.19) and (2.20).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.19}$$

$$\begin{aligned} x' &= x\cos(\theta) - y\sin(\theta) \\ y' &= x\sin(\theta) + y\cos(\theta) \end{aligned} \tag{2.20}$$

The rotated gradient kernels, $h_{\theta_x}(x', y')$ and $h_{\theta_y}(x', y')$ (2.21), is derived by the same method in equation (2.13) and (2.14), and we can see that equation (2.18) is a simplification of computing the same result.

$$\begin{aligned} h_{\theta_x}(x', y') &= -\frac{x\cos(\theta) - y\sin(\theta)}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ h_{\theta_y}(x', y') &= -\frac{x\sin(\theta) + y\cos(\theta)}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{2.21}$$

Sobel Edge Filtering

The Sobel edge filter is a fairly simple edge filtering method. In contrast to the Gaussian edge filter, the horizontal and vertical kernel is fixed with greater weight on pixels closer to the centre

pixel, as defined by equation (2.22) [51, p. 80].

$$h_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.22}$$

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.23}$$

The horizontal G_x and vertical G_y gradient magnitude is approximated by convolving the original image with the corresponding kernels as stated in equation (2.14) and (2.15). The final absolute gradient magnitude is approximated by equation (2.16). However, the Sobel kernel cannot be adjusted to filter edges in a specific angle due to that it is fixed in horizontal and vertical direction.

The absolute gradient magnitude may also be approximated to reduce computation time, as defined by equation (2.24) [25].

$$G = |G_x| + |G_y| \tag{2.24}$$

Concluding Remarks on the Canny Edge Detector

The Canny edge detection function provided by openCV is using the Sobel edge filter method to find the gradient intensity of an image, due to that it is so easily implemented and gives decent results for detecting edges. This function is therefore used in the final solution to compute an edgel map of the image containing the wind blade.

2.3.2 The Hough Transform

The Hough transformation is used to detect complex patterns of points in an edge enhanced binary image, such as lines and curves disconnected due to noise. The edge enhanced image is usually generated by the Canny edge detector. For example, patterns of a line in an image never follows a straight path, but the goal is to present the pattern as a line following the mathematical model for a straight line by equation (2.25) [51, p. 96].

$$a^{\top} x = ax + by + c = 0 \tag{2.25}$$

.

The parameter vector a_0 must be found to represent equation (2.25), but this involves finding the least squared distance to each point following that line pattern, which implies a least squared problem.

The Hough transform solves this pattern detection problem by converting the problem into a simple peak detection problem in parameter space. The parameter space maps all possible lines or curves in the binary image as a single point, simplifying the search process for a connected line or curve.

Standard Hough Line Transform

The first step in the standard hough line transform algorithm is to represent all edge points in $P = \{(x_p, u_p), p = 1, .., M\}$ as a sinusoidal curve in polar parameter space, called the voting stage. A sinusoidal curve is approximated by considering all lines originating from an edge point (x_p, y_p) to any other edge point, where each line is represented by its angle θ and perpendicular distance ρ to origin. The polar parametrization (ρ, θ) of a line is computed by equation (2.26).

$$\rho = x_p \cos(\theta) + y_p \sin(\theta) = \sqrt{x_p^2 + y_p^2} \sin\left(\theta + \tan^{-1}\left(\frac{x_p}{y_p}\right)\right) \tag{2.26}$$

Figure 2.8 shows the polar parametrization and mapping of (ρ, θ) in polar parameter space.

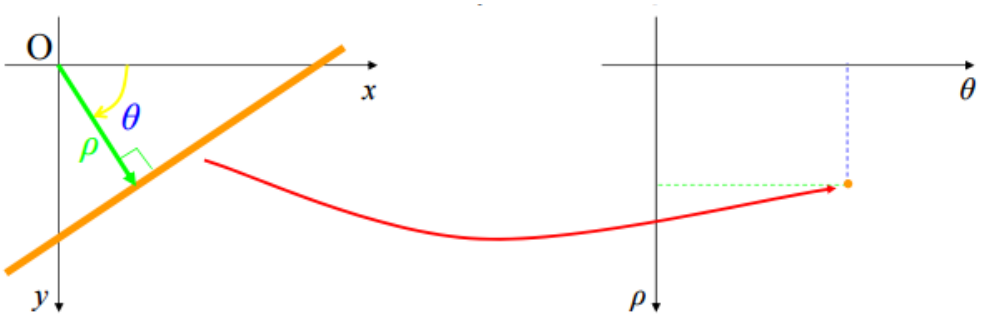


Figure 2.8: Lines in polar space [25]

The resulting polar points of all lines originating from edge point (x_p, y_p) results in a 'one-to-

many' sinusoidal representation of all possible line patterns originating from that edge point, as illustrated in figure 2.9.

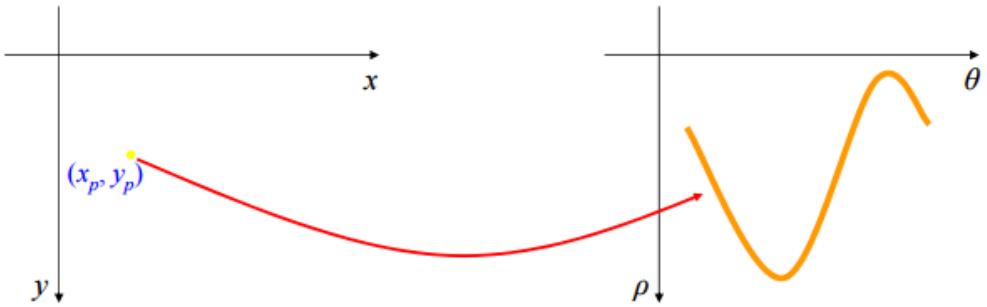


Figure 2.9: Sinusoidal of lines intersecting point (x_p, y_p) [25]

One or more edge points in parameter space will intersect at a common point, called a peak point, if the lines in the original binary image are on the same common line, as illustrated in figure 2.10. A peak point increases depending on how many lines that intersect that point in parameter space.

a b
c d

FIGURE 10.20
Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

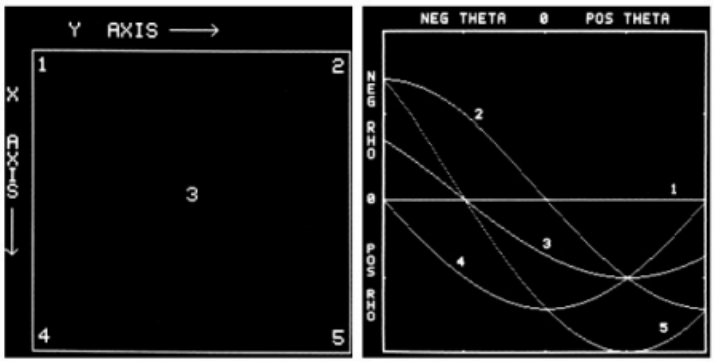


Figure 2.10: Sinusoidal representation of edges [25]

Finally, the Hugh transform searches for a given number of the highest peak points in the parameter space, while lower peaks are ignored to remove small and noisy lines. The resulting peak points are represented as (ρ, θ), and may be drawn as straight lines in Cartesian coordinates by rearranging equation (2.26) to derive equation (2.27).

$$y = -x\cot(\theta) + \rho\csc(\theta) \tag{2.27}$$

Moreover, it should be noted that an implementation of the Hough transform usually con-

siders the origin to be taken at the centre of the image, instead of the (0,0) pixel coordinate at the top left corner of the image. Additionally, the line angle θ is restricted between a range of $0 \leftrightarrow \pi$ or $-\frac{\pi}{2} \leftrightarrow \frac{\pi}{2}$ radians, which means that an implementation must consider a distance parameter ρ that covers both positive and negative values. However, a simple solution to this approach is to rotate the line angle θ by π radians if the distance parameter ρ is negated, while proceeding with the absolute value of the distance parameter as $|\rho|$. This consideration will be used throughout the thesis, and is defined by equation (2.28).

$$\begin{aligned} \theta &= \begin{cases} \theta + \pi, & \text{if } \rho < 0 \\ \theta, & \text{else} \end{cases} \\ \rho &= |\rho| \end{aligned} \tag{2.28}$$

Hough Curve Transform

The hough curve transform algorithm is very similar to the Hough line transformation algorithm since it also includes a voting and search stage. The first step involves finding a suitable curve parametrization, such as the line parametrization in section 2.3.2, as $y = f(x, P)$, where $P = [p_1, ..., p_M]^\top$. In short, the Hough curve transform is processed as in section 2.3.2, but requires additional computation since a curve parametrization is represented in a three-dimensional parameter space. A circle parametrization is presented in equation (2.29). The search stage will look for peak points in the three-dimensional parameter space where radius r and circle parameters (a, b) intersect.

$$r^2 = (x_p - a)^2 + (y_p - b)^2 \tag{2.29}$$

However, this method will only be briefly explained since this method most likely will not be possible to use due to the conditions in this project. Reason is that the targeted wind blade will be in close range, which will make it difficult to detect it as a closed circle. Additionally, it will probably be unnecessary to search for circles on the wind blade.

Concluding Remarks on the Standard Hough Transform

The Hough transform proves to be robust against noise since noisy edges will contribute randomly to other lines, which most likely will be ignored in the search stage. Furthermore, it proves to handle occlusion well since it handles all edge points independently and searches for intersection points in polar coordinates instead of matching all line possibilities in Cartesian coordinates. In addition, HT detects multiple pattern matches simultaneously when detecting the intersection points in polar coordinates [51, p. 100].

However, the limitations of HT is the rapid increase of computation time during the voting stage according to number of edge points M to process. The voting stage requires $O(M \cdot N_\theta)$ operations, where the polar parameter space is of $N_p \times N_\theta$. M is known to be of much larger size than N_p and N_θ, and the computation time will therefore greatly depend on how many edge points to process. Several approaches on how to reduce computation time of the HT, such as the fast HT, adaptive HT, randomized HT and progessive probabilistic HT, have been developed and proven to be efficient.

Progressive Probabilistic Hough Line Transform

A simple solution to reduce time consumption was proposed by Kiryati et al. [23], which simply picks a random subset m from M edge points, decreasing computation time from $O(M, N_\theta)$ to $O(m, N_\theta)$, where $m < M$. The subset m will give an estimation of all features and noise of all edge points M since it is randomly distributed. Some articles allege that a subset of only 2% was enough to give good results [32], however, a subset of 5% - 15% is a common subset ratio [12].

The progressive probabilistic Hough transform [34] is a development of the probabilistic HT. The method works by randomly selecting a new point for voting. For each line detected, then all remaining points supporting that line will be denied voting rights, thus automatically reducing subset m. Furthermore, the algorithm continuously selects a random point and reducing subset m until each point in M have either voted or been denied voting rights. This drastically reduces computation time, since only a subset $m < M$ is considered, while the rest is denied their voting rights. In addition, this simplifies the use of the probabilistic HT since the subset ratio is automatically detected.

2.3.3 Harris Corner Detector

Corner detection involves finding high gradients in different directions on an edge enhanced image, indicating a corner. The Harris corner detector considers an intensity structure C according to the image gradient E of a local neighborhood Q defined by equation (2.30) [51, p. 82]. $[E_x, E_y]^\top$ is the spatial image gradient on neighborhood Q where $E_x = \frac{\partial E}{\partial x}$ and $E_y = \frac{\partial E}{\partial y}$.

$$C = \begin{bmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{bmatrix} \tag{2.30}$$

This equation is derived from the auto-correlation function, defined by equation (2.31) [8]. Further explanation on how to derive the equation is clearly explained by referring to article [8].

$$c(x,y) = \sum_Q [I(x_i, y_i) - I(x_i + \triangle x, y_i + \triangle y)]^2 = \begin{bmatrix} \triangle x & \triangle y \end{bmatrix} C(x,y) \begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} \tag{2.31}$$

The intensity structure is recognized as a semi-positive definite matrix, hence the eigenvalues of C, λ_1 and λ_2, is non-negative. The eigenvalues of matrix C forms a description of the gradient neighborhood in Q, which can be explained by examining three particular cases when searching for a corner.

- $\lambda_1, \lambda_2 \approx 0$, means that the intensity structure C is close to zero, indicating that the neighborhood Q is uniform and no edge or corner is present.
- $\lambda_1 > \lambda_2$, $\lambda_2 \approx 0$, means that there is a gradient change in one direction, parallel with the eigenvector associated with λ_1, indicating an edge. Either one of the eigenvalues may be big and the other low to indicate an edge.
- $\lambda_1, \lambda_2 > 0$, means that there is two gradients in different directions, parallel with the eigenvector associated with corresponding λ, indicating a corner.

Finally, the Harris corner detector truncates the different cases to a score response, R, as stated in equation (2.31). By intuition, a large score response occurs if both eigenvalues are high which indicates a corner, while small or different eigenvalues results in a low or negative score response. As mentioned, different eigenvalues indicate an edge, while small eigenvalues indicate a uniform neighborhood.

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$
$$(0.04 \leq k \leq 0.06) \text{ [27]} \tag{2.32}$$

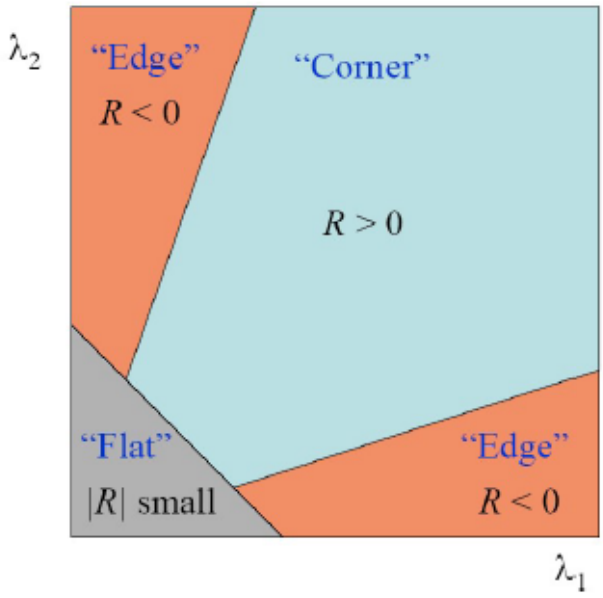
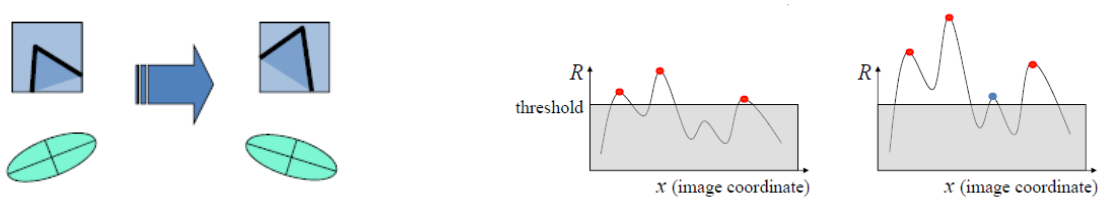


Figure 2.11: Classification of score response R in regard to λ_1 & λ_2 [27].

Properties of the Harris Corner Detector

The properties of the Harris corner detector is defined by its use of eigenvalues, based on the change of intensity for the shift in the neighborhood surrounding a point in the image. This means that the use of eigenvalues offers the property of rotational invariance, which means that an object can be rotated without affecting the response of the Harris corner detector. Furthermore, the use of change in intensity, or derivatives, makes the Harris corner detector invariant to any intensity shift, $I \rightarrow I + b$, and intensity scale, $I \rightarrow aI$, of image I. These invariant properties are illustrated in figure 2.12.



(a) Rotational invariant [27].

(b) Invariant to intensity shift & intensity scale [27].

Figure 2.12: Invariant properties of the Harris corner detector.

However, an obstacle to the Harris corner detector is its non-invariance to image scale. The score response R of the neighborhood surrounding a point is estimated using a kernel of a certain size, which means that the kernel size defines the corner scales to be detected. A small kernel size will be able to detect corners of small scale, while corners of larger scale will remain undetected. On the other hand, a larger kernel size will detect corners of proportionally larger scale. This non-invariance to image scale is illustrated in figure 2.13.

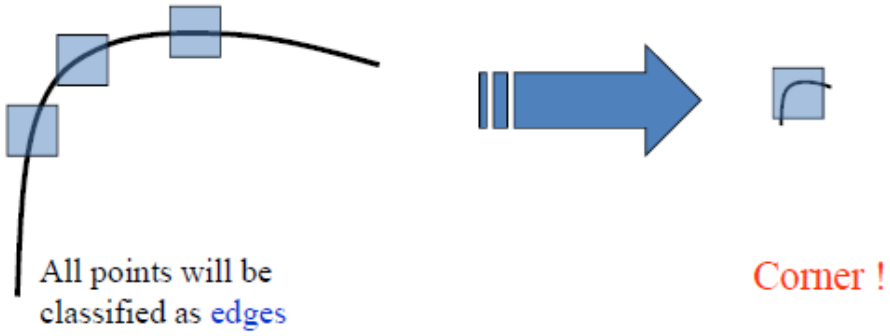


Figure 2.13: Non-invariant property to image scale [27].

2.4 Stereopsis

Stereo vision refers to *the ability to infer information on the 3D structure and distance of a scene from two or more images taken from different viewpoints* [51, p. 140]. Stereopsis refers to depth perception, computed by a stereo vision system, and this section will give an introduction to the aspect of stereo vision.

The stereo vision system must be able to solve two main problems known as the correspon-

dence and reconstruction problem. First approach involves solving the correspondence problem, by correlating items present in both images. However, occlusion is a difficulty since some elements may be present in one image, but not the other, which implies that the stereo vision system must be able to ignore elements that cannot be matched. The second approach involves reconstructing the disparity between corresponding elements into a 3D perception of the real world. Disparity is referred to as the difference in retinal position between corresponding elements in two images, which results in a disparity map when all possible elements in the images are matched. The disparity map is commonly shown as an intensity image, where high intensity implies elements in close range.

2.4.1 Basic Theory of Stereopsis

The change in relative angular displacement of correlated image points across different camera frames is called the parallax, which implies that it is the depth difference. The parallax is dependent on the relative angle between camera views, where a zero difference in relative view angle results in an empty parallax, and hence it will be impossible to reconstruct a 3D map. Therefore, the relative view angle between the camera frames must be different, which is achieved by translating a single camera or use two or more cameras in different locations. The standard stereo vision system of two cameras in parallel will be discussed in this chapter and used in the final solution.

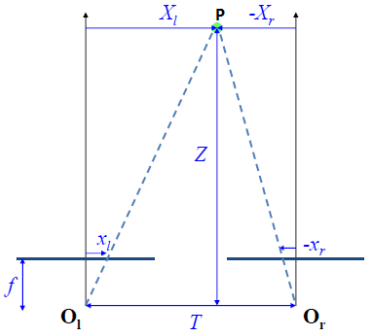


Figure 2.14: Simple 2D triangulation [24]

The basic theories of stereopsis are easier understood by considering a simple 2D triangulation constructed from two 1D pin-hole cameras, as illustrated in figure 2.14 where O_l and O_r are the centers of projection. The motivation is to find the depth Z from fixation point P to

baseline T indicating a triangulation from two known positions in 1D to map the fixation point in 2D. The relation between retinal position x and position of fixation point X is concluded by geometry and stated by equation (2.33) and (2.34) for left and right camera frame, respectively. Equation (2.33) and (2.34) follows the simple relation between known focal length f with retinal position x, and depth Z with fixation point X.

$$\frac{x_l}{f} = \frac{X_l}{Z} \tag{2.33}$$

$$\frac{x_r}{f} = \frac{X_r}{Z} \tag{2.34}$$

Depth Z is found by considering the difference in retinal position between the left and right camera, called the disparity. The disparity d is computed by rearranging equation (2.33) and (2.34) to obtain a relation between the right and left camera frame as stated in equation (2.35). Note that the retinal position and fixation position from the right camera frame is in opposite direction to the left camera frame, hence negated.

$$d = x_l + (-x_r) = f\frac{X_l}{Z} + (-f\frac{X_r}{Z}) = f\frac{X_l - X_r}{Z} \tag{2.35}$$

Moreover, by geometry we see that the baseline T is related to the difference in fixation position between the left and right camera frame as stated in equation (2.36). Finally, we obtain solution (2.37) which relates depth Z to the camera frames.

$$T = X_l + (-X_r) = X_l - X_r \tag{2.36}$$

$$Z = f\frac{T}{d} \tag{2.37}$$

This further proves why distant objects seems to move slower than closer objects, due to that the disparity decreases inversely with the distance to the object. More importantly, this also proves why the depth to objects outside of a measurable fixation point cannot be measured since the disparity will be approximately zero for distant objects leading to an immeasurable distance, as stated by equation (2.38).

$$\lim_{d \to 0} Z \to \infty \tag{2.38}$$

The disparity map is proved to be inversely proportional to the depth map, which yields that a stereo vision system doesn't need to know the focal length and baseline to visualize the difference in depth between objects in the image. Instead, the difference in debt is visualized by computing a disparity map. However, the focal length and baseline must be known to measure the real distance to an object in the image, which is essentially important for measuring the distance between the UAV and the wind blade.

The stereopsis theory represented in this section might give the assumption that stereo vision is a straight forward triangulation of point matches in pixel coordinates, however, the method introduced so far assumes that the triangulated points matched with the same fixation points are known. This leads to the problem of detecting matching points from an image frame to the other related image frame, which might be impossible due to occlusion, distortion and noise. This problem is referred to as the correspondence problem, and there are mainly two approaches to solve this problem which are the appearance based matching method and feature based matching method, which will be discussed in section 2.4.4 and 2.4.5, respectively.

The parameters of a stereo vision system, such as focal lenght, baseline and effective pixel size, are essential to find for calibrating the stereo vision system, and are characterized as intrinsic or extrinsic parameters. The baseline is referred to as a part of the extrinsic parameters in the stereo system, while the focal length and effective pixel size are referred to as parts of the intrinsic parameters. The extrinsic parameters are therefore concluded as the parameters that describe the mapping from the world frame to the camera frame, while the intrinsic parameters characterize the transformation mapping from the camera frame to the image frame [51, p. 144].

2.4.2 Extrinsic & Intrinsic Matrices

A camera projects a 3D world frame into a 2D image frame, hence the goal of stereopsis is to recover the 3D coordinates from a 2D image frame. This transformation mapping involves defining the intrinsic matrix and extrinsic matrix, which relates the transformation mapping from the

world frame to the image frame.

By assuming a simplified pin-hole camera system so that all objects in the image are in focus, means we can compute the fundamental equation of perspective cameras, as stated in equation (2.39).

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z} \tag{2.39}$$

The fundamental equation of perspective cameras relates the 2D mapping to 3D mapping in the camera frame, which is linearized to a 3D transformation mapping in the camera frame as follows in equation (2.40).

$$\begin{bmatrix} kx \\ ky \\ k \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.40}$$

Moreover, any object is measured in the image frame, meaning the camera frame must be transformed to the image frame. This transformation is achieved by considering the effective physical size of a pixel and that the origin of the image frame is in the top left corner of the image, at pixel coordinate (0,0). Furthermore, by considering the conventional right-hand rule for the camera frame, we derive the transformation from the image frame to the camera frame as follows in equation (2.41). The effective physical pixel size is defined as (S_x, S_y), and the pixel coordinates of the image center as (O_x, O_y).

$$x = -(x_{im} - O_x)S_x \qquad y = -(y_{im} - O_y)S_y \tag{2.41}$$

We obtain the transformation matrix from a 2D point in the camera frame to a 2D point in the image frame by rearranging equation (2.41), which yields equation (2.42).

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{S_x} & 0 & O_x \\ 0 & -\frac{1}{S_y} & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.42}$$

Furhtermore, the intrinsic matrix is derived by combining the perspective projection in the

camera frame from equation (2.40), with the transformation mapping from the camera frame to the image frame given by equation (2.42), which yields the intrinsic matrix as stated by equation (2.43).

$$\begin{bmatrix} kx_{im} \\ ky_{im} \\ k \end{bmatrix} = \begin{bmatrix} -\frac{f}{S_x} & 0 & O_x \\ 0 & -\frac{f}{S_y} & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.43}$$

The transformation mapping from the world frame to the camera frame is defined by the extrinsic parameters, which is defined by a rotation from the world frame to the camera frame followed by a translation, defined by equation (2.44).

$$\mathbf{P}_c = \mathbf{R}_w^c \mathbf{P}_w + \mathbf{T} \tag{2.44}$$

Equation (2.44) is expressed in matrix form by equation (2.45), which constitutes the extrinsic matrix.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.45}$$

Finally, we obtain the projective matrix from the world frame to the image frame by combining the intrinsic matrix with the extrinsic matrix, as derived by equation (2.46).

$$\begin{bmatrix} kx_{im} \\ ky_{im} \\ k \end{bmatrix} = \begin{bmatrix} -\frac{f}{S_x} & 0 & O_x \\ 0 & -\frac{f}{S_y} & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M_{int} M_{ext} \begin{bmatrix} \mathbf{P}_w \\ 1 \end{bmatrix} \tag{2.46}$$

2.4.3 Undistortion

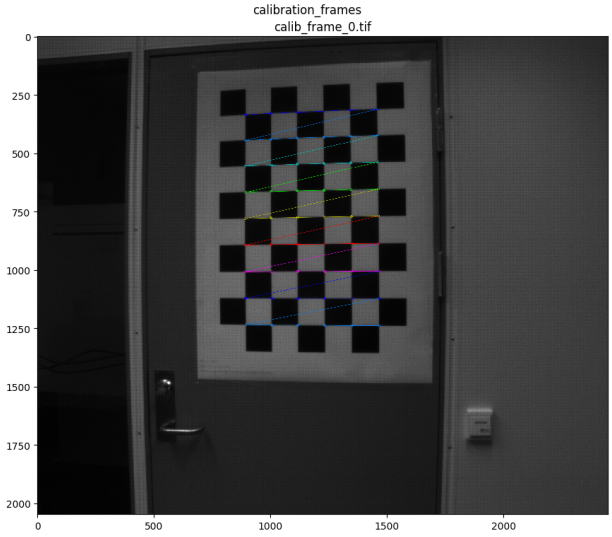
The intrinsic matrix described in the previous chapter 2.4.2 assumes that the mapping from 3D-space to 2D-space is a linear transformation, however, this is only valid if there is no lens distortion. Radial lens distortion is defined as the symmetric distortion caused by imperfections in the curvature of the lens, which causes bending of straight lines from world to image transformations. This effect will drastically reduce the accuracy of a stereo vision system, if it is not accounted for. Undistortion is referred to as determining world to image correspondence by relating sufficiantly many 3D points to its corresponding image point to determine a camera matrix which accounts for radial lens distortion.

Hartley and Zisserman [15] defines a solution by minimizing the geometric error between image point correspondences between a point in the world frame and its corresponding point in the image frame. A solution for computing an initial estimate of the camera matrix is also presented, however, the focal length and sensor size of the cameras are known, so the initial camera matrix will be defined using these parameters. Defining the camera matrix as P, and the corresponding image points in the world frame and camera frame as X_i and x_i, respectively, then Hartley and Zisserman [15] defines the solution of determining a calibrated camera matrix as finding the Maximum Likelihood estimate of the camera matrix P which minimizes the geometric error defined by equation (2.47).

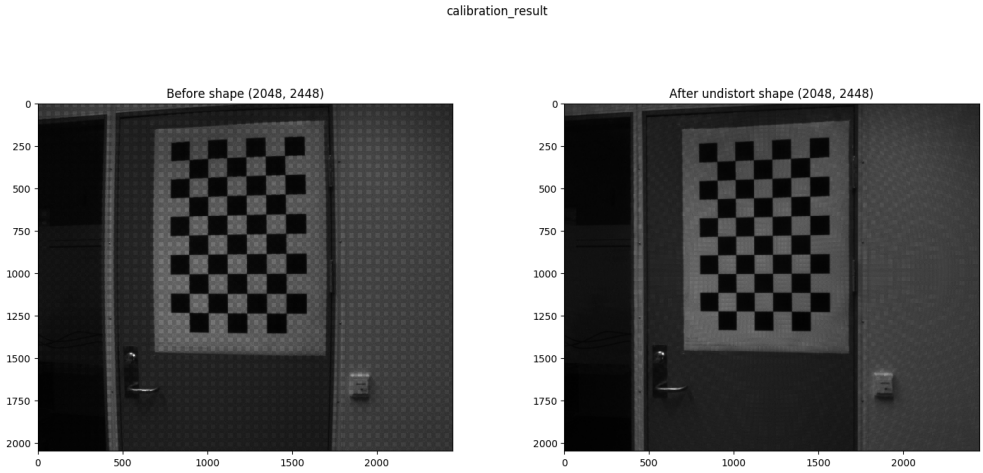
$$\min_P \sum_i d(x_i, PX_i)^2 \tag{2.47}$$

Moreover, Hartley and Zisserman [15] also defines a method for finding point correspondences using a checkerboard pattern which is a known straight line pattern. The steps are defined as computing an edgel map using the Canny edge detection method, then computing straight line fitting to the detected linked edges following an intersection of the computed lines to obtain the checkerboard corners. The final step is to compute an optimized camera matrix using equation (2.47) which contains the distortion parameters. The new camera matrix is then used to compute rectification vectors to be used in a rectification transformation on images taken by the respective camera to compute undistorted transformations between the world frame and the image frame.

The opencv library offers extensive solutions for calibrating a camera using the method mentioned above, which is built into a *CameraCalibration* module to relate distortion coefficients to a specific camera. An example of intersecting lines to detect the checkerboard corners is shown in figure 2.15a, and the result of undistorting the respective image is shown in figure 2.15b.



(a) Detection of checkerboard corners to conduct camera calibration.



(b) Undistortion of the respective image.

Figure 2.15: Camera calibration and undistortion of the respective image.

It should be mentioned that it is necessary to take multiple images of the checkerboard from different angles to achieve good results, since the distortion coefficients needs to be computed for the whole lens. The extrinsic parameters, such as translation and rotational displacement between the cameras, are accounted for when computing the perspective transformation of the stereopsis system in a module called *StereoCalibration*, which merges two modules of the *CameraCalibration* module, representing two perfectly aligned cameras. The rectification vectors for the stereopsis system is computed on the same basis as computing the rectification vectors on a single camera, but accounting for the translational and rotational displacement between the cameras as well.

Additionally, it is only necessary to compute the rectification vectors once, as long as the intrinsic and extrinsic parameters are kept unchanged, since they are invariant to scale, meaning the process may be speeded up by downscaling a frame before undistorting the respective frame.

2.4.4 Appearance Based Matching

A simpel approach to a stereo vision system is to compute a disparity map, recall section 2.4.1, which assumes that the stereo system have prior knowledge of the matching points in the image frames. The appearance based matching method simplifies the correspondence problem by assuming that there is a correlation between corresponding subsets of the image frames, assuming that the pixel points in one image have minor displacement in the other image. The displacement are relative to the baseline, and from equation (2.37) we proved that the disparity is inversely proportional to the distance from the fixation point. However, this also leads to that it will be impossible to find a correlation between image subsets if the displacement is too far apart, which means the appearance based method limits the baseline, thus limits the depth range that can be detected. This is proved by considering equation (2.37) and introducing an estimation error for the measured distance and disparity, δZ and δd, as stated in equation (2.48) [24].

$$\begin{aligned} Z + \delta Z &= f\frac{T}{d + \delta d} \\ f\frac{T}{d} + \delta Z &= f\frac{T}{d + \delta d} \end{aligned} \tag{2.48}$$

Equation (2.48) is derived according to δZ, which results in equation (2.49).

$$\begin{aligned} \delta Z &= -fT\frac{\delta d}{d^2 + d\delta d} \\ d^2 &\gg d\delta d \\ \delta Z &\approx -f\frac{T}{d^2}\delta d \end{aligned} \tag{2.49}$$

By considering the absolute value of the distance error and using equation (2.37) to replace disparity d, as shown in equation (2.50), we find that the distance error is squared proportional to the distance measured. We also note that a longer baseline decreases the distance error.

$$|\delta Z| \approx |f\frac{T}{(f\frac{T}{Z})^2}\delta d| = |\frac{Z^2}{fT}\delta d| \tag{2.50}$$

This proves that a long baseline is beneficial for an accurate stereo vision system, but the appearance based method limits the baseline since the disparity error increases with the displacement of the matching image points. Additionally, the appearance based method assumes that there is a consistent surface reflectance of the elements in the image, called the Lambertian reflectance model. The Lambertian reflectance model assumes that each surface point on an element appears equally bright from any view point.

The appearance based method is carried out by correlating windows of size N within a search region in the image frames I_l and I_r. Instead of directly computing the displacement of the pixel, the method finds the displacement of a neigbouring pixel in the right window with the highest window correlation in the search region. This method is computed for every pixel in the left image $\mathbf{p}_l = [i, j]^\top$, which results in a disparity map representing the differences in depth, recall section 2.4.1. The correlation between the two window patches is defined by equation (2.51) [51, p. 147], where $\mathbf{d} = [d_1, d_2]^\top$ is the displacement in the search region R. Each pixel

in the disparity map $\mathbf{D}$ is computed by the displacement of pixel $\mathbf{p}_l = [i, j]^\top$ in the right image according to the window with the highest correlation, as stated in equation (2.52).

$$c(\mathbf{d}) = \sum_{k=-N}^{N} \sum_{l=-N}^{N} \psi(I_l(i+k, j+l), I_r(i+k-d_1, j+l-d_2)) \tag{2.51}$$

$$\bar{\mathbf{d}} = \arg\max_{d \in R}\{c(\mathbf{d})\} \tag{2.52}$$

The function ψ is the method used to find the window correlation, and the most common method is the sum-of-squared differences method (SSD), also called the block matching method.

Sum of Squares Differences

The sum-of-squares differences (SSD) method is a correlation method of window I in the left image frame and the window patch T in the right image frame, as stated in equation (2.53).

$$\begin{aligned} \psi(I, T) &= \psi(I_l(i+k, j+l), I_r(i+k-d_1, j+l-d_2)) \\ \psi &= (I-T)^2 \\ \arg\min_{d \in R}\{ &\sum_{k=-N}^{N} \sum_{l=-N}^{N} (I-T)^2\} \end{aligned} \tag{2.53}$$

Equation (2.53) shows that the SSD method minimizes the correlation between the window patches, which means that it must be negated to find the maximal correlation. Equation (2.52) is derived using the SSD method which results in equation (2.54).

$$\bar{\mathbf{d}} = \arg\max_{d \in R}\{- \sum_{k=-N}^{N} \sum_{l=-N}^{N} (I-T)^2\} \tag{2.54}$$

This method is an efficient appearance based method to compute a disparity map, and solves the problem with changing intensity values in the image frame which would bias a simple cross-correlation method.

2.4.5 Feature Based Matching

Feature based matching limits the correspondence problem to a set of recognisable feature points, that are invariant to changes in view. These feature points are recognized by feature descriptors in the other image, and matched based on a heuristic technique since multiple points might be recognized as possible matches. The rule of thumb heuristic technique is a simple approach, and works by choosing the matching point with minimilized displacement to the original feature point. Other heuristic methods can be that a fixed pattern must be recognized in the same order for it to be recognized as a matched pattern, which can be useful when recognizing fixed edge patterns. A great deal of the problem is therefore to decide upen a set of feature points to match, which can be circles, curves or any invariant pattern, including fixed edge patterns.

The 3D reconstruction of the matched feature points is more difficult to compute than the appearance based method, since the feature points can have a large difference in view point. However, this means that the cameras can also have large differences in view point and still reconstruct a 3D map. A large baseline leads to large differences in view point, but makes it also possible to reconstruct the depth of feature points further away, recall equation (2.50). The most common approach to reconstruct a 3D map of matched feature points is based on epipolar geometry, and is essential for the understanding of stereopsis.

Epipolar Geometry

Section 2.4.1 introduced the basic theories of stereo vision by simplifying the 3D depth reconstruction to a 2D reconstruction problem. To understand the principles of 3D reconstruction, we must first of all understand the geometry of stereo, known as epipolar geometry.

The basics of epipolar geometry are illustrated in figure 2.16, where the O_L and O_R represents the corresponding left and right center point of projection from two pin-hole cameras with their corresponding image planes, I_l and I_r. Point, $P(X,Y,Z)$, refers to the 3D point to be estimated, which is referred to by vectors $\mathbf{P}_l = [X_l, Y_l, Z_l]^\top$ and $\mathbf{P}_r = [X_r, Y_r, Z_r]^\top$. These vectors are related by the extrinsic parameters, such as the baseline and angle of projection, and are therefore related by a rotation matrix R as stated by equation (2.57). Vectors $\mathbf{p}_l = [x_l, y_l, z_l]^\top$ and

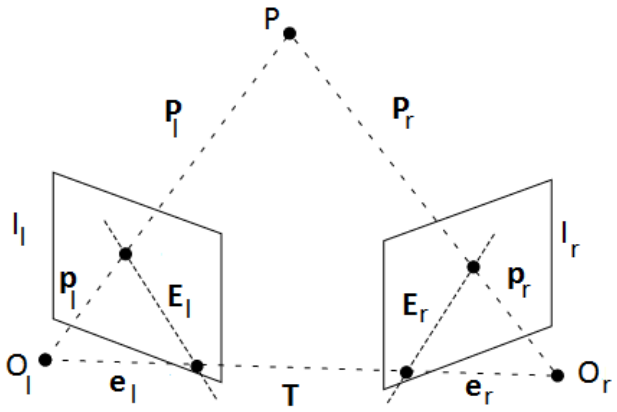


Figure 2.16: Epipolar geometry

$\mathbf{p}_r = [x_r, y_r, z_r]^\top$ refer to the projection of the fixation point onto their respective image plane. The relation between the projection points and the real points in the 3D plane, can be stated by the same geometrical principle as stated by equation (2.33) and (2.34), leading to equation (2.55) and (2.56). The depth parameter z onto the image plane equals the focal length f since the depth value onto the image plane is equal for all projection points.

$$\frac{\mathbf{p}_l}{f} = \frac{\mathbf{P}_l}{Z_l} \tag{2.55}$$

$$\frac{\mathbf{p}_r}{f} = \frac{\mathbf{P}_r}{Z_r} \tag{2.56}$$

$$\mathbf{P}_r = R(\mathbf{P}_l - \mathbf{T}) \tag{2.57}$$

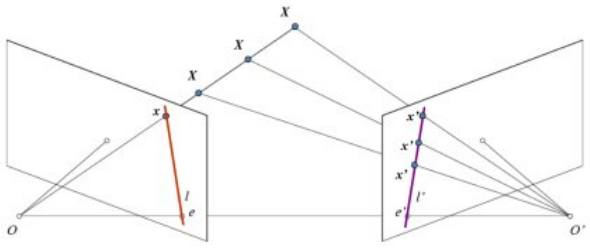


Figure 2.17: Epipolar geometry [38]

The epipole lines, E_l and E_r, are the representations of point P on the image plane, which is the intersection of the epipolar plane (P, O_l, O_r) on the image. This leads to a 1D intersection problem of triangulating point P on the epipolar line in accordance with the opposite epipolar

line known as the epipolar constraint, as illustrated in figure 2.17. This fact can also be used to handle occlusion and noise by rejecting false points that cannot be matched on the epipolar lines. The epipoles, e_l and e_r, are defined as the line intersection on the image planes, thus representing the image of the opposite camera. The centers of projection are usually in parallel, which means that the epipole points are placed in infinity on the epipole lines.

Finally, the obvious problem is to compute the epipolar geometry. The epipolar geometry is represented by the essential matrix E and the fundamental matrix F. These matrices also constitute a 3D reconstruction technique that can be computed without any prior knowledge of the extrinsic or intrinsic parameters.

The Essential Matrix

The essential matrix E defines a link between the epipolar constraint and the extrinsic parameters, which means it describes the location of the other camera relative to the associated camera.

To derive the essential matrix, we start by defining the coplanarity condition of vectors $\mathbf{P}_l$, $\mathbf{T}$ and $(\mathbf{P}_l - \mathbf{T})$, as stated by equation (2.58) [51, p. 153]. Coplanar vectors are defined as vectors parallel to the same plane, or lie on the same plane. The coplanarity condition of 3D vectors is verified if the vectors are linearly dependent, thus satisfying $\mathbf{A} \cdot [\mathbf{B} \times \mathbf{C}] = 0$. Equation (2.58) is derived using equation (2.57), and defines the cross product of equation (2.60) as a multiplication of the skew-symmetrical matrix of $\mathbf{T}$, stated by equation (2.61). This leads to equation (2.62) and (2.63), which shows that the essential matrix E defines the link between the epipolar constraint and the extrinsic parameters. The extrinsic parameters are defined as the relative angle and baseline between the cameras.

$$(\mathbf{P}_l - \mathbf{T})^\top [\mathbf{T} \times \mathbf{P}_l] = 0 \tag{2.58}$$

$$(R^\top \mathbf{P}_r)^\top [\mathbf{T} \times \mathbf{P}_l] = 0 \tag{2.59}$$

$$\mathbf{T} \times \mathbf{P}_l = S(\mathbf{T})\mathbf{P}_l \tag{2.60}$$

$$S(\mathbf{T}) = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{2.61}$$

$$\mathbf{P}_r^\top RS(\mathbf{T})\mathbf{P}_l = 0 \tag{2.62}$$

$$E = RS(\mathbf{T}) \tag{2.63}$$

Furthermore, we continue to derive a solution based on the coordinates in the image plane by rearranging equation (2.55) and (2.56) to derive equation (2.64). This concludes that the essential matrix is the mapping between points and epipolar lines, since the epipolar lines are defined as the projection of the epipolar plane (P, O_l, O_r) on the image. As an example, the epipolar line in the right image plane is represented as the essential matrix multiplied by vector $\mathbf{p}_l$ in the left image plane, stated by equation (2.65).

$$\mathbf{p}_r^\top E\mathbf{p}_l = 0 \tag{2.64}$$

$$\mathbf{E}_r = E\mathbf{p}_l \tag{2.65}$$

In conclusion, the essential matrix offers the possibility to find desired points on the epipolar lines. However, these coordinates are mapped by the extrinsic parameters, as camera coordinates, while the coordinates in the image plane are in pixels, defined by the intrinsic parameters, which means we must find a transformation between intrinsic and extrinsic parameters. This transformation is solved using the fundamental matrix.

The Fundamental Matrix

The fundamental matrix is a representation of the mapping between points and epipolar lines in pixel coordinates, which means it offers the possibility to reconstruct a 3D depth map without any prior knowledge of the stereo vision system. Recall section 2.4.2, the intrinsic parameters is represented by an intrinsic matrix M_{int}, which transforms pixel coordinates $\bar{p}$ to camera coordinates p as stated by equation (2.66).

$$\begin{aligned} \mathbf{p}_r &= M_{int_r}^{-1} \bar{\mathbf{p}}_r \\ \mathbf{p}_l &= M_{int_l}^{-1} \bar{\mathbf{p}}_l \end{aligned} \tag{2.66}$$

From section 2.4.5, we concluded that the essential matrix is defined in terms of camera coordinates. This problem is solved using the intrinsic matrix to transform the camera coordinates in equation (2.62) to pixel coordinates, which constitutes the fundamental matrix F as stated in equation (2.67).

$$\begin{aligned} \mathbf{p}_r^\top E \mathbf{p}_l = \bar{\mathbf{p}}_r^\top F \bar{\mathbf{p}}_l = 0 \\ F = M_{int_r}^{-\top} E M_{int_l}^{-1} \end{aligned} \tag{2.67}$$

Equation (2.67) shows that it is possible to reconstruct the epipolar geometry, and thus reconstruct a 3D depth map without any prior knowledge of the intrinsic or extrinsic parameters. The reason to this is that the fundamental matrix is computed by the intrinsic and extrinsic constant parameters, which means the fundamental matrix can be estimated from point matches in pixel coordinates found in the corresponding images. However, estimating the fundamental matrix without any prior knowledge of the intrinsic and extrinsic parameters comes with a cost of less accuracy. It should therefore be noted that prior knowledge of the stereo system is beneficial.

3D Reconstruction Based on Epipolar Geometry

From section 2.4.5, we introduced the basics of feature based stereo vision and mentioned that matched feature points can be reconstructed in a 3D map using epipolar lines. The first step is therefore to locate the epipolar lines according to the matched feature points. From equation (2.65), we stated that the epipolar line is related to the essential matrix and the opposite point vector, which is rewritten using the fundamental matrix to relate the epipolar lines with pixel coordinates, as stated in equation (2.68).

$$\bar{\mathbf{E}}_r = F\bar{\mathbf{p}}_l \tag{2.68}$$

Furthermore, all the epipolar lines for each feature point lies on the corresponding epipolar line, $\bar{e}_l$ and $\bar{e}_r$, which intersect the image planes. This means we can rewrite equation (2.67) to derive equation (2.69).

$$\begin{aligned} \bar{\mathbf{p}}_r^\top F\bar{\mathbf{e}}_l = 0 \\ \bar{\mathbf{e}}_r^\top F\bar{\mathbf{p}}_l = 0 \end{aligned} \tag{2.69}$$

The epipoles are located knowing that the fundamental matrix is not zero, which means that the epipolar lines are the null space of the fundamental matrix, as stated in equation (2.70).

$$\begin{aligned} F\bar{\mathbf{e}}_l = 0 \\ F^\top\bar{\mathbf{e}}_r = 0 \end{aligned} \tag{2.70}$$

If we know the extrinsic and intrinsic parameters of the stereo system, then the vectors $\mathbf{p}_l$ and $\mathbf{p}_r$ are computed directly using the epipolar lines, and the 3D map can be reconstructed using triangulation. However, the triangulation is not straightforward since the vectors $\mathbf{p}_l$ and $\mathbf{p}_r$ doesn't necessarily intersect, meaning we must find the point of minimum distance between the vectors, which will be the depth point in 3D space. A simple linear triangulation method is discussed in chapter 2.4.6.

2.4.6 Linear Triangulation

A simple linear triangulation method is described by Hartley and Zisserman [15][p. 312], and is defined by combining the measurements of corresponding feature points and their image to world transformation into a form as $\mathbf{AX} = 0$, and solved using a least square method.

The transformation of a point $\mathbf{x}$ in the image frame to the point in the world frame $\mathbf{X}$ is computed using the projection matrix $\mathbf{P}$, defined by equation (2.71).

$$\mathbf{x} = \mathbf{PX} \tag{2.71}$$

The cross product of equation (2.71) results in three equations, whereas two are linearly independent, and is defined as $\mathbf{x} \times (\mathbf{PX})$ and written out in equation (2.72) [15, p. 312] where $\mathbf{P}^{i\top}$ are the rows of the projection matrix $\mathbf{P}$.

$$\begin{aligned} x(\mathbf{P}^{3\top}\mathbf{X}) - (\mathbf{P}^{1\top}\mathbf{X}) &= 0 \\ y(\mathbf{P}^{3\top}\mathbf{X}) - (\mathbf{P}^{2\top}\mathbf{X}) &= 0 \\ x(\mathbf{P}^{2\top}\mathbf{X}) - y(\mathbf{P}^{1\top}\mathbf{X}) &= 0 \end{aligned} \tag{2.72}$$

The linear components of equation (2.72) from the cross product of matched feature points are used to compose the equation of the form $\mathbf{AX} = 0$, and written out in equation (2.73) where $\mathbf{X}$ and $\mathbf{X}'$ are denoted as the matched feature points.

$$\mathbf{A} = \begin{bmatrix} x\mathbf{P}^{3\top} - \mathbf{P}^{1\top} \\ y\mathbf{P}^{3\top} - \mathbf{P}^{2\top} \\ x'\mathbf{P}'^{3\top} - \mathbf{P}'^{1\top} \\ y'\mathbf{P}'^{3\top} - \mathbf{P}'^{2\top} \end{bmatrix} \tag{2.73}$$

Matrix $\mathbf{A}$ gives a total of four equations with four unkowns which means a solution can be found using a least square solution. Hartley and Zisserman [15] solves the least square problem by finding the singular value decomposition of $\mathbf{A}$.

2.5 Structured Light

Traditional cameras provide only 2D images of a complex 3D physical world, which greatly increases the difficulty of recovering an accurate 3D model of real-world objects. Common issues of computer vision is to segment different objects in the image, which in this case is to segment a wind blade from the cluttered background. The Canny edge detection method, mentioned in chapter 2.3.1, is able to extract the blade edges, but the result will be greatly disturbed by edgel features in the background and on the blade. Therefore, segmenting the blade based on the edgels alone will not be feasible. An another option is to use region processing methods, such as the Otsu method [26] or more advanced texture-based region segmentation algorithms such as the iterative K-means clustering method [26]. However, the conditions in the real-world will make it increasingly difficult to segment the blade from either a cluttered earth background or, in some cases, a clear blue sky.

Structured light refers to illumination of known light patterns on the area in focus by the camera, used to simplify the 3D reconstruction of a 3D object from 2D images that lack depth information. Known light patterns greatly reduces the complexity of real-world conditions since a computer vision algorithm has a fixed pattern to search for when reconstructing a 3D object. A simple example of utilizing structured light to reconstruct depth information is to scatter detectable points on a planar surface and compute the depth based on the relationship between the structured light projector and the camera, illustrated by figure 2.18.

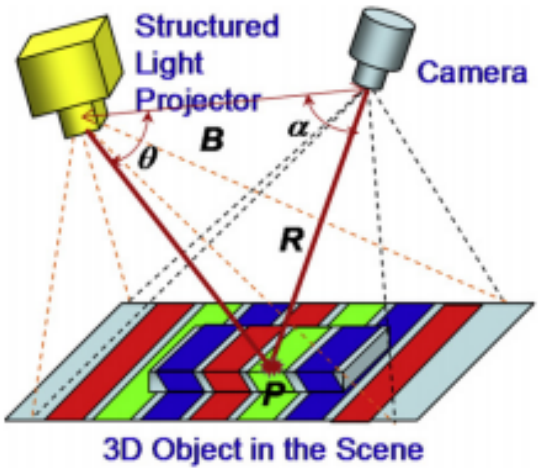


Figure 2.18: Theoretical example of structured light projection [13].

The geometric relationship between the camera and the structured light projector is found by triangulation, much like the concept of stereopsis as mentioned in chapter 2.4.1, and defined by equation (2.74) [13]. The depth from the camera to the illuminated point on the object is defined as R, while the baseline between the structured light projector and the camera is defined as B with projector angle θ and camera angle α.

$$R = B\frac{\sin(\theta)}{\sin(\alpha+\theta)} \tag{2.74}$$

Another benefit of using structured light is that the projected feature points are non-scaling as seen by the camera. The scaling is only affected by the lense distortion of the structured light projector and the physical displacement to the camera, which both are fixed parameters during operation, meaning the feature points are of a fixed scale at any distance.

A major difficulty with structured light is to differentiate the structured light pattern from the real-world objects in the image, and it must be noted that the real-world surface is non-planar which will distort the projected structured light pattern.

This article proposes a very simple solution to the problem of differentiating the structured light pattern from the real-world objects, by simple taking two images of the same surface with and without projecting the structured light pattern. Segmentation of the structured light pattern will simply be the pixel change between the two images, as follows in equation (2.75). A simulation of segmenting the structured light pattern on a wind blade is realized in figure 2.19.

$$\Delta img = |img - img_{sl}| \tag{2.75}$$

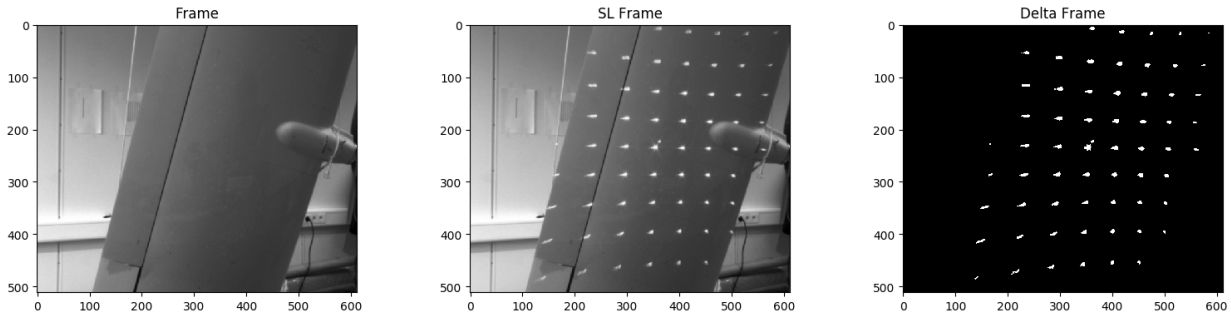


Figure 2.19: Result of equation (2.75) to segment the structured light pattern on a blade.

Another benefit of using structured light is that the segmented area of the structured light pattern effectively shows the area of interest to detect the blade. This assumption is taken since the projected illumination will not be visible on a distant background while inspecting the wind blade, effectively segmenting the area of interest. The structured light pattern therefore works as an efficient basis for computing points of interest to use in a feature based stereopsis method, as well as a basis for segmenting the blade to detect the blade edges.

The structured light pattern was chosen as a dot matrix pattern, as shown in figure 2.19, since grid points are easily detected as blobs using a feature detection method. These blobs are used as matchable feature points between the two cameras in the stereo vision system, and will be used as a basis for segmenting the blade and detecting the blade edges which will be described in detail in chapter 4.

2.5.1 Blob Detection

This chapter reviews different feature detection methods which are usefull for detecting the grid points, commonly known as blobs. Chapter 2.5.2 concludes upon a feature detection method that detects the blobs with high accuracy while keeping the computation delay at a minimum.

Blob Detection

Lindeberg [28] defines a blob as a region associated with at least one local extremum for either a dark or bright blob as a local minimum or maximum within a saddle point. Detecting a blob in a 2D image domain requires an image intensity function that identifies the extremum within the area limited by the saddle point [18]. Template matching is a fast and simple approach for detecting blobs, and works by finding matches in the image using a template. However, real-world conditions limits the use of template matching primarily due to its non-invariance to scale. The majority of blob detection methods have therefore been based on Lindeberg [29] method for analysing structures at different scales, as a scale-space representation. The scale-space representation is derived by applying a Gaussian smoothing kernel at different levels to create a pyramid of different scale-spaces. Blobs at different scales is then found by detecting local extremums in the scale-space at different scales, and matching blobs with linked extremas over

scales to identify the characteristics and position of the blob. Efficient and powerfull feature detection methods have been proposed during the recent years, such as the SIFT Lowe [31], SURF Bay et al. [5] and ORB Rublee et al. [45] method. These methods have proven to be invariant to scale, rotation and translation while keeping computation cost at a minimum and automatically detecting feature points of interest. The Scale Invariant Feature Transform (SIFT) will be desribed in detail to review the theory in which the SURF and ORB method is based upon.

SIFT

The Scale Invariant Feature Transform (SIFT) was first introduced by Lowe [31] who presented a new method for identifying features invariant to image scaling, translation, rotation, and partially invariant to illumination changes and moderate perspective transformations [30]. Due to these properties, the SIFT method has become extensively popular for local feature generation related to object recognition and point matching between different view points of a 3D scene. Prior methods, such as the Harris corner detector, have struggled with non-invariance to scaling and partial occlusion, including other characteristics of a cluttered real-world scene.

The first step of the SIFT method involves identifying scale-invariant features obtained by scale-space filtering and localization of maxima or minima of a difference-of-Gaussian (DoG) function within a difference-of-Gaussian pyramid. The difference-of-Gaussian pyramid is constructed by computing the difference between the adjacent levels in the Guassian pyramid, and the Gaussian pyramid is computed by smoothing and resampling of the image.

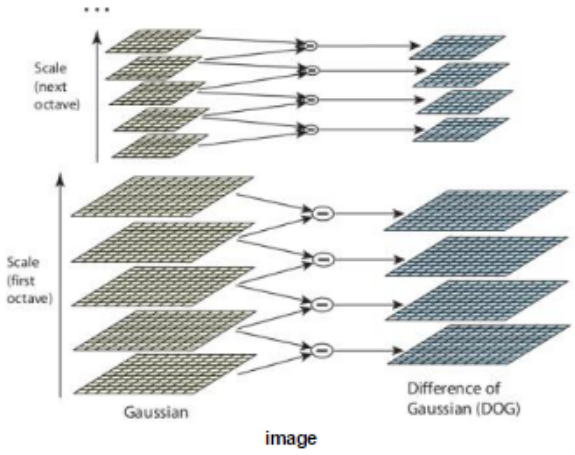


Figure 2.20: DoG pyramid [36].

Moreover, the key points of interest are computed by localization of maxima or minima which are determined by comparing each pixel in the DoG pyramid with eight of its neighbors. If the pixel is a local extrema at the current level, then the closest pixel is calculated at the next level of the pyramid, which is used to repeat the process.

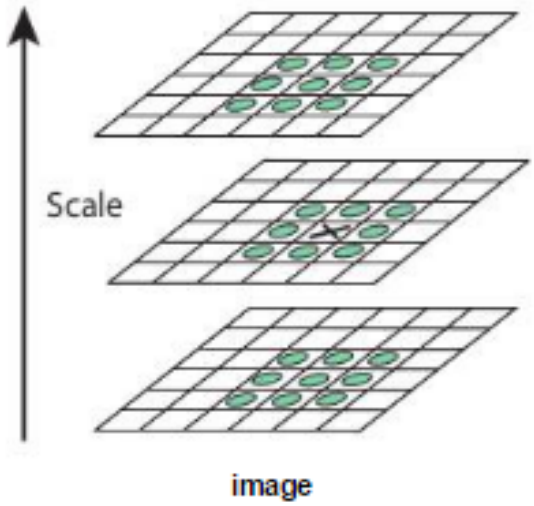


Figure 2.21: Detection of local extremas in different scale levels [36].

Another important process is to suppress interest points along edges since these interest points are less useful for matching. This problem is solved using a concept quite similar to the Harris corner detector by formulating a criterion of the ratio between the eigenvalues of a Hessian matrix computed at the position and scale of the interest point [30] to detect strong interest points.

Next step is to compute image descriptors based on the key interest points invariant to scale and rotation. The scale invariance is obtained by normalizing the local neighborhood of the descriptor according to the scale level of the interest point. The orientation invariance is obtained by accumulating a histogram of the gradient directions of the local neighborhood with the area of the accumulation window proportional to the scale level. The orientation is determined by peak detection in the accumulated orientation histogram, and multiple peaks are detected if the secondary peak is above 80% of the height of the highest peak. In the case of multiple peaks, a new image descriptor is then computed using the detected peaks.

The SIFT descriptor is computed using a rectangular grid centered at the interest point which

is oriented according to the detected peaks in the histogram and scaled proportionally according to the detection scales of the interest point. All of the points in the grid are accumulated into orientation histograms summarizing the contents over the subregions. The length of each arrow is computed using the sum of the gradient magnitudes near that direction within the region. Lowe [31] found that a 4×4 grid with 8 quantized directions usually gives good results, which means that an image descriptor has $4 \times 4 \times 8 = 128$ dimensions for each interest point, and this resulting image descriptor is referred to as the SIFT descriptor.

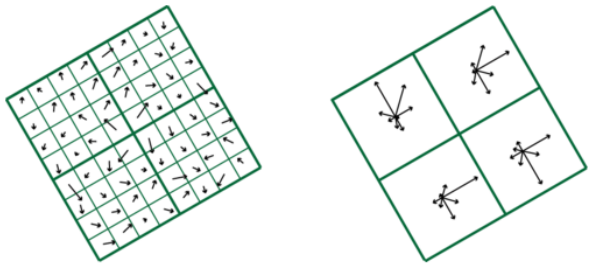


Figure 2.22: Image descriptor computed using a 2×2 grid. Lowe [31] found that a 4×4 grid is normally a good choice [30].

The final step of the SIFT method is to match nearest neighbors of local image descriptors, by finding the point in the other image domain that minimizes the Euclidean distance between the descriptors. However, some matches are very close to one another, so Lowe [31] found that a ratio of closest distance to second closest distance is taken by asserting that the ratio is lower than 0.8.

Simple Blob Detector

The ORB and SURF method are efficient tools for detecting feature points with lower computation cost and matching performance to the SIFT method. However, the structured light pattern is structured as a matrix with uniformly distributed grid points which are easily detected using a very simple blob detector provided by OpenCV [42] and accessed using the SimpleBlobDetector class reference in the OpenCV library. The algorithm does not use scale-space representation to detect interest points, but simplifies the problem by converting the image to multiple binary images with each binary image representing a pixel treshold. The blobs are detected by finding saddle points and their extremas to calculate their centers, which is used to group centers from

several binary images. The groups are then merged to one blob if the distance is closer than a given treshold. Finally, the blob centers, radiuses, sizes and locations are returned as keypoints, as it would have been using either the SIFT, SURF or ORB method. Moreover, the algorithm provides intuitive filtration criterias such as filtration by color, area, circularity, inertia, convexity and minium distance to neighboring blobs which makes this tool very convenient for detecting the grid points. It should also be noted that the algorithm is computationally fast since it does not use scale-space smoothing.

2.5.2 Test Results of Blob Detection

A test was conducted to verify the feasibility of using the simple blob detector, SIFT, SURF or ORB method, and all of the algorithms are found in the opencv or opencv_contrib library. Detection of feature points will be the first step of the computer vision process, so downsampling and undistortion of frames are included in the beginning of this step, and the step is implemented as a $GetPointList(..)$ function in the $BlobDetector$ module. The test was conducted on the blade as seen in figure 2.19 to detect and localize the structured light patterns as blobs. Unfortunately, the SURF algorithm failed to detect any blobs for almost any dataset so it was discharged as a possible solution. However, the results from using the simple blob detector, SIFT and ORB algorithms are shown in figure 2.23, and the computational delay for the respective algorithms are shown in table 2.1. Note that the test was conducted on an Odroid-XU4, as specificed by section 7.3.1, with a frame of height and width of 512×612, and the computational delays are approximations from conducting several similar tests.

Table 2.1: Approximate computational delay for the respective feature detection algorithms.

	≈Delay (seconds)
Simple Blob Detector	0.13
ORB	0.03
SIFT	0.60

By the test, it is shown that the simple blob detector and the SIFT detector performed better than the ORB detector since they localized the feature points with better accuracy. This is easily seen by the centered image in figure 2.23 where multiple blobs are detected on a single blob, and the blob sizes are very inaccurate. The SIFT and simple blob detector both proved

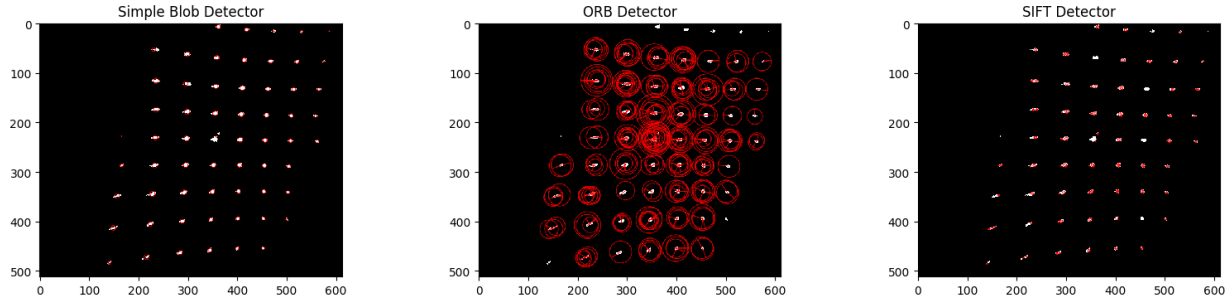


Figure 2.23: Result of using the simple blob detector, SIFT and ORB method to detect feature points. Detected blobs are highlighted as keypoints by the red circles, with radius given by the respective blob size.

to detect the feature points with high accuracy, however, the simple blob detector was more efficient. This was expected since the simple blob detector does not use scale-space smoothing and is therefore non-invariant to scale, whereas the SIFT and ORB descriptor is invariant to scale. Moreover, the filtration criterias of the simple blob detector made it very convenient for detecting these types of feature points, since the scale of the structured light is fixed according to the relative position between the camera and the laser. By conclusion, it was therefore decided to continue with the simple blob detector as the preferred feature detector since it computed accurate keypoints of the respective blobs, and since it was computationally faster than the SIFT algorithm.

2.5.3 Feature Matching

The stereo vision system requires matching correct features between the left and right image. The grid points will provide necessary feature points which are possible to match between the images, however, issues such as occlusion may cause a lack of matching points, or even mismatches. Two different approaches to solving an approximate nearest neighbors search will be discussed to decide upon the most optimal solution.

Fast Approximate Nearest Neighbors Search

Matching feature points are commonly known as solving a nearest-neighbor search problem of some datasets. An optimal nearest-neighbor solution between two sets of points as $P =$

$\{p_0,..,p_n\}$ and $Q = \{q_0,..,q_n\}$ is to do a linear search between the datasets to find the points in P that are nearest to the points in Q, however, this approach will be increasingly time consuming with large datasets. Muja and Lowe [35] mentions that there are no known exact algorithms for providing an optimal nearest-neighbor matching that are faster than linear search, so solutions that provide large speedups are approximate nearest-neighbor solutions. There have been published many solutions to the approximate nearest-neighbor problem with minor loss in accuracy and significant efficiency boosts. Muja and Lowe [35] introduces a solution for approximate nearest-neigbor matching that can be applied to almost any dataset with fully automated parameter selections. The solution was found by considering many known methods for clustering and matching features, and Muja and Lowe [35] found that two algorithms proved to give the best performance, depending on the dataset and desired precision. The first algorithm combines two previously known methods, known as the hierarchical k-means trees search method and the priority search order method, while the second method uses a previously known method called the multiple randomized kd-trees introduced by Silpa-Anan and Hartley [48]. The algorithms are released as public domain in a library called the Fast Library for Approximate Nearest Neighbors (FLANN) and may be accessed using the opencv library.

Block Search & Feature Points

The concept of block matching was introduced in chapter 2.4.4, and describes a solution for matching blocks of pixels in the left frame with correlated blocks in the right frame, and computing the disparity between the blocks with highest correlation. This chapter utilizes that concept and simplifies the nearest-neighbor problem to a search for finding the closest feature points within a given block radius, and finding the optimal match with the lowest disparity.

From section 2.5, it was discussed that the structured light projects feature points of a fixed scale due to the fixed alignment between the laser and the cameras, meaning the scaling distance between the feature points from the same image will be fixed. It is therefore possible to compute a block radius that limits the search to a given scale according to the known scale distance between the feature points. Another benefit of this concept is that the search radius is limited to an area where possible matches should be, efficiently constraining the search for finding mismatches that may be located on the next column in the grid.

The concept first rebuilds the left and right images with known feature point positions and sizes, then a block search for each respective feature point from the left image is computed to find possible matches in the right image. The camera alignments are considered to be translated without any rotation, meaning the correct matches in the right image will be displaced to the left of the feature points from the left image. It is also considered that the images from the left and right image will not give perfectly aligned feature points due to errors such as imperfect camera alignment or distortion, so it is necessary to expand the search radius along the y-axis to some degree, while mainly searching along the x-axis for possible matches. The block search is illustrated in figure 2.24 where the left feature point is matched with the right feature point inside the block radius, seen as a rectangle due to a wider search along the x-axis and a shorter search along the y-axis. This block search is conducted for each feature point in the left image to find an approximation of optimal matches in the right image. Moreover, a simple threshold filter is added to constrain matched feature points to be fairly of the same size. The algorithm is implemented in the module $FeatureStereopsis$ as $PointBlockMatch(..)$ which is summarized in an activity diagram illustrated in figure 2.25. Moreover, the algorithm is considered to perform relatively fast due to extensive use of the numpy library and a fairly low set of feature points. Additionally, only a single tuning parameter needs to be set once, which decreases or increases the search area relative to the fixed scaling distance.

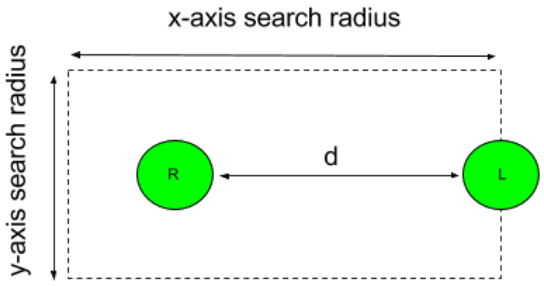


Figure 2.24: Block Search Method

The matched feature points from left and right image are shown in green. Note that the right feature point is shifted to the left according to the disparity distance, d.

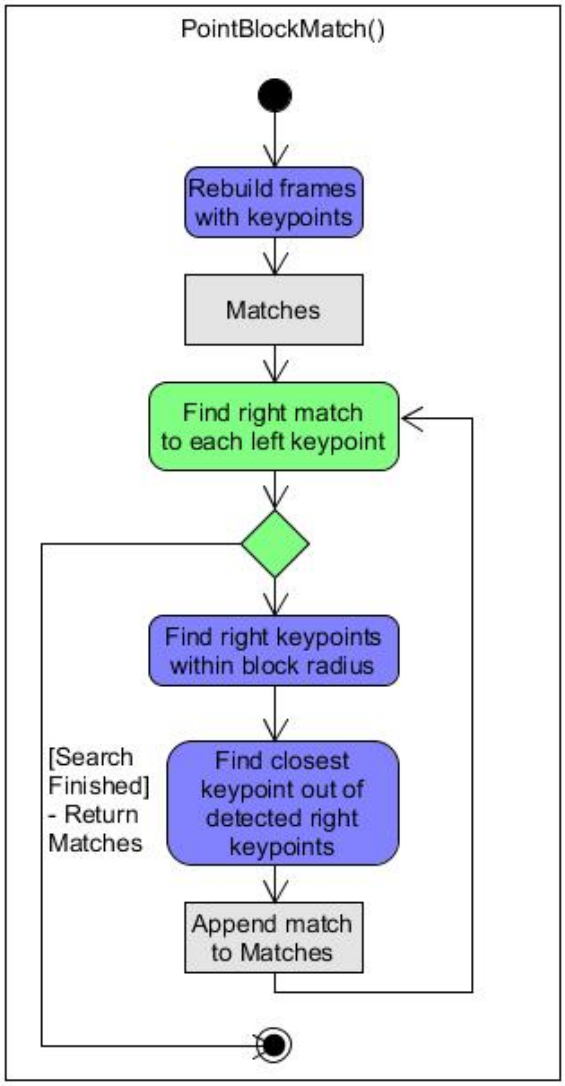


Figure 2.25: Activity diagram of the block search method.

The block matching method utilizes the benefit of a fixed scaling distance between the grid points of the structured light, so a calibration method to approximate the scaling distance between the feature points is implemented. This calibration method is much similar to the block matching method, but searches for the closest neighbor for each respective feature point in a single frame, thus approximating a standard scaling distance by calculating the average closest neighbor distance of all closest neighbor distances. Additionally, drastic neighbor mismatches is removed by filtrating neighbor distances outside of the standard deviation. The calibration method is implemented in the module $BlobScaleDetector$ as $CalibrateStandardScaling(..)$.

2.5.4 Test Results of Feature Matching

A test was conducted to verify wether to use the FLANN library or the block matching method for feature points. First of all, it is necessary to calibrate the stereopsis system as mentioned in section 2.4.3, and this was conducted by placing the cameras in front of a checkerboard as seen in figure 2.26, and running the implemented calibration session. It should be noted that several calibration sessions should be commenced until distortion is properly adjusted for, since a miscalibrated stereopsis system will lead to inaccurate 3D reconstruction.

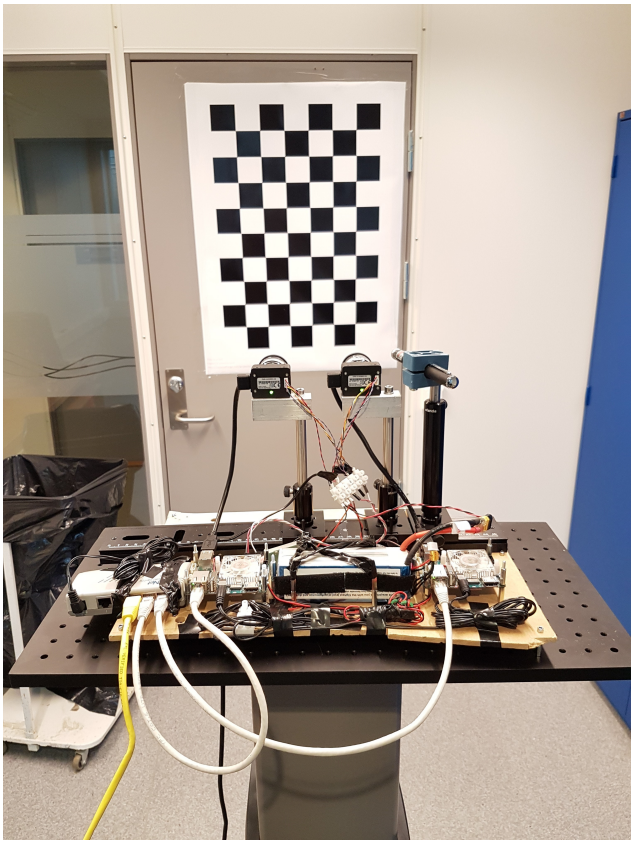


Figure 2.26: Calibration of the stereopsis system.

The test was conducted by targeting the stereopsis system at a wall as seen in figure 2.27. Moreover, during the rest of this thesis, all tests were conducted with a baseline of 5 cm and the frames were downsampled from a height and width of 2048×2448 to a shape of 512×614, using the gaussian pyramid downsampling method as discussed in section 2.2.2.

Figure 2.27: Structured light on wall, as seen from the left and right camera.

The benefit of using the FLANN library is that it is easily implemented, and the following test was commenced to verify the feasibility of using this method. The library requires a training set to build a binary tree for multi-dimension vectors, better referred to as the feature descriptors, and a query set to find the approximate nearest neighbors to the training set, so the left feature points was set as the training set while the right feature points was set as the query set. However, the result was shown to be very inaccurate, as seen in figure 2.28, although the computational delay was close to negligible (≈ 0.004 seconds).

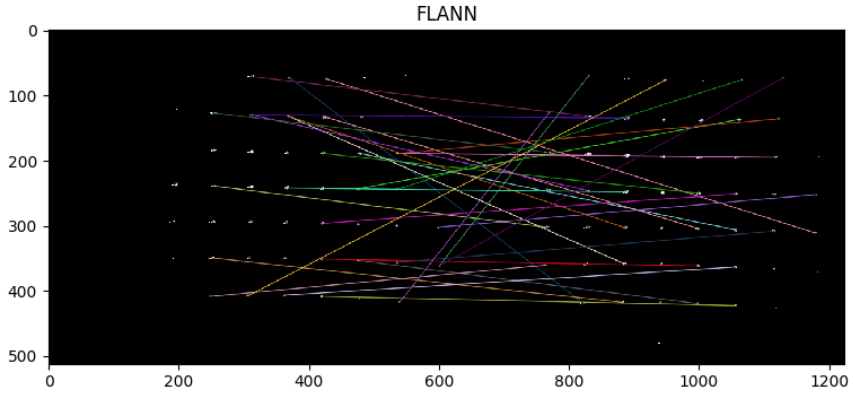


Figure 2.28: Result of using the FLANN library for feature point matching.

Lines between respective matches are drawn, whereas correct matches would be seen as horizontally straight lines due to the the horizontal alignment of the cameras.

The FLANN library failed matching correct feature points mainly since it computes matches using feature descriptors, which in this case is not feasible since the grid point descriptors are too similar, so this method was quickly abolished.

Testing the block matching method was conducted using the same data as when testing the FLANN library, and the result is shown in figure 2.29.

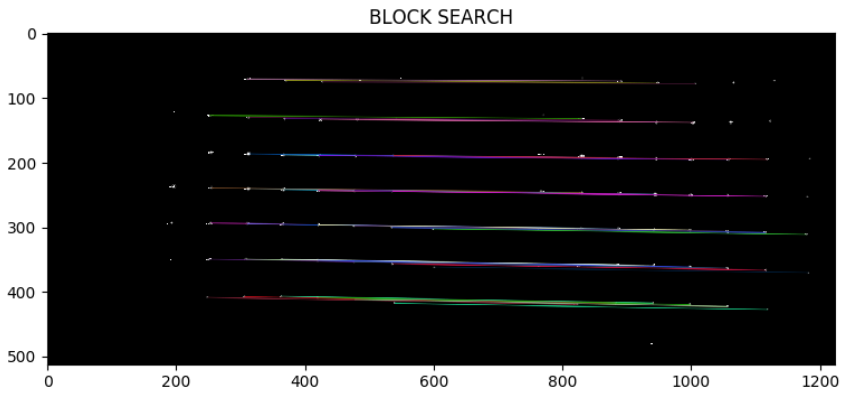


Figure 2.29: Result of using the block matching method for feature point matching.

Lines between respective matches are drawn, whereas correct matches are seen as horizontally straight lines.

First of all, this method was considerably slower than the FLANN library, which was expected since the FLANN library is heavily focused on high efficiency for large data sets, but the block matching method was estimated to spend a maximum of ≈ 0.03 seconds on computing matches and corresponding 3D points due to an extensive use of the numpy library and a fairly low set of feature points. It is also seen that the method computes a very accurate set of matches, so it is easy to conclude that this method is the best option compared to using the FLANN library in these conditions.

2.5.5 Test Results of 3D Reconstruction

3D reconstruction of matched feature points may be conducted using the linear triangulation method by Hartley and Zisserman [15], as presented in section 2.4.6, or by simple 2D triangulation defined by equation (2.37) in section 2.4.1, and both of the methods are found in the $StereoVision$ module as $TriangulatePoints(..)$ and $Compute3DPointsFromDisparity(..)$, respectively. The tests were conducted at a distance of 125 cm and 164 cm to the wall shown in figure 2.27, and the results are shown in tabel 2.2 and 2.3. It should be noted that both of the algorithms are implemented so that they return the 3D points in metric coordinates, which is computed by implementing the theory revised in section 2.4.2. The average distance vector along the Z-axis of all 3D points is presented to verify the accuracy to the real distance, and the standard deviation informs wether the 3D points are approximately uniform along the Z-axis,

which they should be when estimating the distance to a flat surface.

Table 2.2: Estimating a distance of 125 cm to a flat wall.

Distance: 125 cm	Mean Dist. (cm)	Dist. Error (cm)	STD Dist. (cm)
Linear Triangulation	168.08	-43.08	12.55
2D Triangulation	122.70	2.30	6.55

Table 2.3: Estimating a distance of 164 cm to a flat wall.

Distance: 164 cm	Mean Dist. (cm)	Dist. Error (cm)	STD Dist. (cm)
Linear Triangulation	114.70	49.30	10.48
2D Triangulation	181.62	-17.62	14.60

Moreover, more tests were conducted to verify the accuracy of conducting 3D reconstruction and it was found that the simple 2D triangulation method outperforms the linear triangulation method, which is also proven by table 2.2 and 2.3. The test results of using the simple 2D triangulation to reconstruct 3D points are illustrated in figure 2.30, which shows the relation of increasing inaccuracy in response to increased distance to the object.

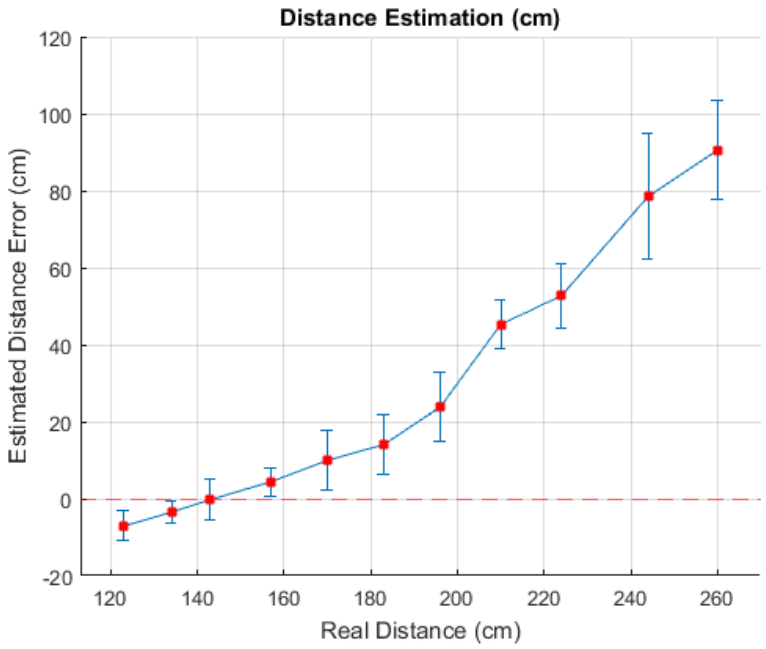


Figure 2.30: Plot showing error of estimated distance relative to the real distance.

The distances were estimated using the simple 2D triangulation method at the real distances marked by the red points. Each point includes an error bar to show the standard deviation of estimated 3D points at the respective distance.

2.5.6 Concluding Remarks on 3D Reconstruction

It was easy to conclude that the FLANN library was not a feasible solution for computing matches, since the descriptors of the feature points were too similar. The solution therefore was to conduct a block matching method solely on finding the closest matching feature points, which was shown to provide good results. As mentioned, the block matching method will work on any distance since the feature points are scale-invariant due to the fixed relative position between the laser and the camera. The simplicity of this method also makes it easy to implement, and it is proven that the algorithm is relatively fast due to an extensive use of the numpy library and a fairly small set of keypoints. Moreover, a possible upgrade may be to give weight to the descriptors of the feature points when finding the optimal match, where only a simple difference in size threshold of the matching keypoints is a part of the current solution. However, a consequence to this upgrade is that the simple blob detector algorithm only computes keypoints, and not descriptors of the respective feature points since it does not use scale-space smoothing, meaning either the SIFT or ORB method would have to be used.

3D reconstruction is easily computed when the feature point matches are found, however, from table 2.2 and 2.3 it is seen that the simple 2D triangulation method outperforms the linear triangulation method. Hartley and Zisserman [15] mentions that the linear triangulation is a non-optimal solution, although it is easily implemented, especially for 3D reconstruction from several views of the feature point. On the other hand, the simple 2D triangulation considers that the cameras are perfectly aligned, in which case it is an optimal solution, which most likely is the reason why the simple 2D triangulation outperforms the linear triangulation method. However, Hartley and Zisserman [15] also proposes an optimal triangulation solution [15][p. 315] that finds the global minimum of a geometric error cost function using a non-iterative algorithm, but it was decided not to implement this method due to its complexity and since the simple 2D triangulation provides notably good results due to the horizontal alignment of the cameras.

It is proven that the accuracy of the stereo vision system decreases with an increasing distance to the respective object, and this will pose a problem for accurately manouvering a UAV at a fixed distance of more then two to three meters with the given baseline of five centimeters. However, the stereo vision system is more flexible on increasing the baseline since feature point matching is used to reconstruct the respective feature points, meaning the baseline may be increased to give better accuracy to the stereo vision system. On the other hand, the baseline is also constrained by the fixed scaling distance between the laser projected feature points since matching feature points will cross if the disparity is higher than the fixed scaling distance. This issue will occur if the baseline is too large and the object is too close, so there should be a tradeoff when selecting the baseline to prevent this occurence while maximizing accuracy.

Camera calibration was found to be the most difficult obstacle, since the 3D reconstruction considers a perfectly undistorted set of feature points. It was found that the stereo vision system had to be recalibrated until an approximately correct set of 3D points were estimated. Additionally, some of the estimated 3D points may be incorrectly estimated either due to a mismatch or inaccurate camera calibration, so simple filtration that removes 3D points outside of the standard deviation along the depth axis of all 3D points were implemented. This type of filtration is considered as a simple solution when it is considered that the main objective is to measure the distance to a relatively flat blade.

Chapter 3

Wind Blade Properties

The dominating wind turbine design follows the horizontal axis wind turbine (HAWT) design, which is conventionally known as a turbine with the shaft mounted horizontally parallel to the ground, as illustrated in figure 3.1. This chapter will introduce basic concepts of the HAWT blade design to review basic characteristics of a modern wind blade, and will briefly review the materials that the wind blades are made of.

Figure 3.1: HAWT designed wind turbine.
(https://www.poweredbymothernature.com/what-is-wind-energy/)

3.1 HAWT Wind Blade Design

A conventional wind blade design follows the concept of creating a lift that pulls the blade upwards. This lift is created by a curved blade design which forces the air sliding along the upper side of the wing to move faster than the air sliding on the lower surface, thus creating an upward pulling force. Figure 3.2 represents a cross section of a typical modern wind blade.

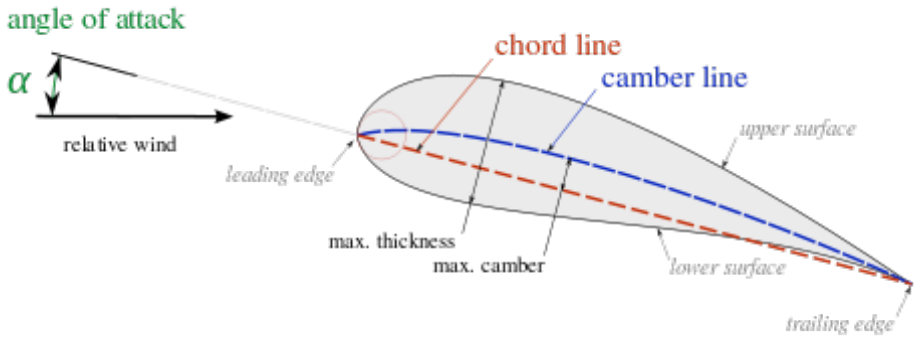


Figure 3.2: Cross section of a wind blade.
(https://en.wikipedia.org/wiki/Blade_solidity)

The curvature of the blade is described by the camber line which characterizes the asymmetry between the top and bottom surface of the blade. Moreover, the angle of attack characterizes the angle of the blade relative to the general direction of the airflow towards the leading side. A steep angle of attack will cause the airflow on the upper side of the blade to whirl which leads to a sudden pressure increase on the upper side of the blade, meaning the lift effect suddenly disappears. This phenonemon is known as stall, and the geometry of the wind blades are designed to minimize the risk of stall. The wind blades are also twisted from root to tip due to that the wind flows increasingly faster along the blade closer to the tip, since the rotational speed is proportionally faster at the tip than closer to the root. Modern wind turbines take advantage of the stall phenomenon to regulate the rotational speed of the wind turbine, and is achieved by simply pitching the blade according to the wind direction [7].

Furthermore, modern wind blade designs are unanimously based upon Betz's law about ideal breaking of the wind. Betz' Law states that a maximum of 59% of the kinetic energy in the wind can be converted to mechanical energy using a wind turbine [7], and this theorem is

the basis for different methods for calculating an ideal geometrical form of a HAWT wind blade. A simple method for calculating the chord length at a radial distance from the hub is defined by equation 3.1 [47].

$$C_{opt} = \frac{2\pi r}{n} \frac{8}{9C_L} \frac{U_{wd}}{\lambda V_r} \tag{3.1}$$

where:

C_{opt}	= Optimum chord length
r	= Radius(m)
n	= Number of blades
C_L	= Lift coefficient
V_w	= Local windspeed
U	= Windspeed
U_{wd}	= Design windspeed (m/s)
Ω	= Rotational velocity (rad/s)
$\lambda = \frac{\Omega r}{V_w}$	= Local tip speed ratio
$V_r = \sqrt{W_w^2 + U^2}$	= Local resultant air velocity (m/s)

A typical blade design is presented in figure 3.3. The tip speed is the foremost important parameter, where a higher tip speed demands reduced chord lengths due to wind blade aerodynamics, which means that the chord length is increasingly more narrow closer to the tip.

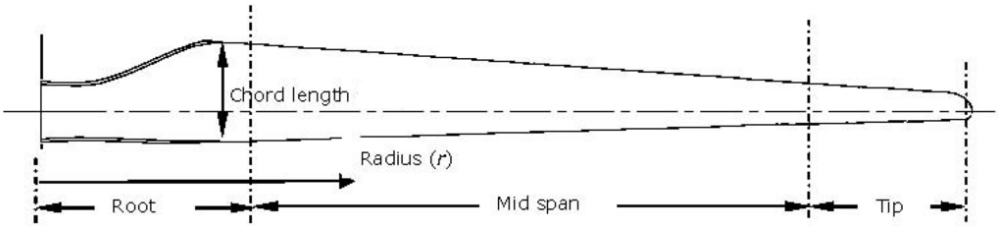


Figure 3.3: Typical HAWT blade design [47].

Moreover, the wind blades are usually made of fiber reinforced composite materials and reinforced with an outer layer made of epoxy prepreg materials of E-glass and carbon fibre. Damages on the wind blade is usually caused by wear and tear on the leading edge which will be visible as small cracks or erosion. More severe types of damages may be caused by falling ice from the wing above, or even lightning.

3.1.1 Wind Blade Dimensions

It is important to review the dimensions for a typical modern wind turbine to get an aspect of which camera specifiations are necessary and to decide upon a proper inspection distance between the UAV drone and the blade surface. Wind blade dimensions for a V80-2MW wind turbine [52] provided by the leading wind turbine provider Vestas is considered and presented in table 3.1.

Table 3.1: V80-2MW Blade Dimensions

Length	39m
Max. chord	3.5m

Chapter 4

Detecting Blade Edges

The manouvering plan which will be presented in chapter 5 requires that the blade edges are localized and identified as straight lines. The Hough line transformation, discussed in section 2.3.2, computes straight lines as (ρ, θ) in an edgel map. However, it will be very difficult and computationally time consuming to differentiate the blade from a cluttered or smooth background, making it very difficult to accurately segment the blade and localize the blade edges. A solution to this problem will be presented in this chapter, which will be shown to drastically reduce computation time while improving accuracy of localizing the blade edges.

Assuming that the structured light presented in section 2.5 is detected as a grid of feature points, then the method may be summarized as follows

1. Structure the feature points as a matrix by computing horizontal and vertical Hough lines along the rows and columns of the grid.

2. Detect correlated feature points along each vertical and horizontal Hough line to compute a set of bounded lines along the respective row or column. The bounded lines are computed by interpolating the respective feature points.

3. Compute the edgel map of the original frame and localize the edgels of interest by starting a search from the start and end position of the bounded lines, as detected in the previous step, to find the most significant edgel along that row or column, efficiently segmenting the blade.

4. Compute four Hough lines along the top, bottom, left and right side of the segmented area using the edgels detected in the previous step, which represents the blade edges.

The final step results in four Houg lines along the top, bottom, left and right side of the segmented blade area, which can be used to compute a heading following the blade, as discussed in chapter 5. A more descriptive explanation with examples of the method is discussed in section 4.1 and 4.2.

4.1 Segmenting the Blade

The method works be segmenting the blade using the detected feature points which are structured as a grid with approximately straight columns and rows. As mentioned in the previous section, the first step of the method is therefore to structure the feature points as a matrix by computing horizontal and vertical Hough lines, which is easily done by limiting the Hough line transformation to $[0, \frac{\pi}{2}]$ radian degrees. Note that the horizontal Hough lines are computed by considering either 0 or π radian degrees. Computation time and the amount of possible Hough lines are drastically reduced since the parameter space is limited to only two different degrees and a fairly low set of feature points, so all possible peak points in the Hough parameter space are considered. A concatenation of the respective Hough lines is then commenced, which is done by sorting the Hough lines according to the respective degrees, which is followed by computing the median ρ of Hough line matches that are considered as representing the same row or column. Finding correct Hough line matches are solved by considering a threshold distance which is scaled by reffering to the fixed scale distance between the feature points and a tuning parameter. The concatenation of Hough lines is repeated until the change in amount of Hough lines stabilizes, meaning there are no more Hough lines to be matched, and finally the resulting matrix of Hough lines is returned. Figure 4.1 shows an example of a resulting Hough line matrix computed using feature points on a blade.

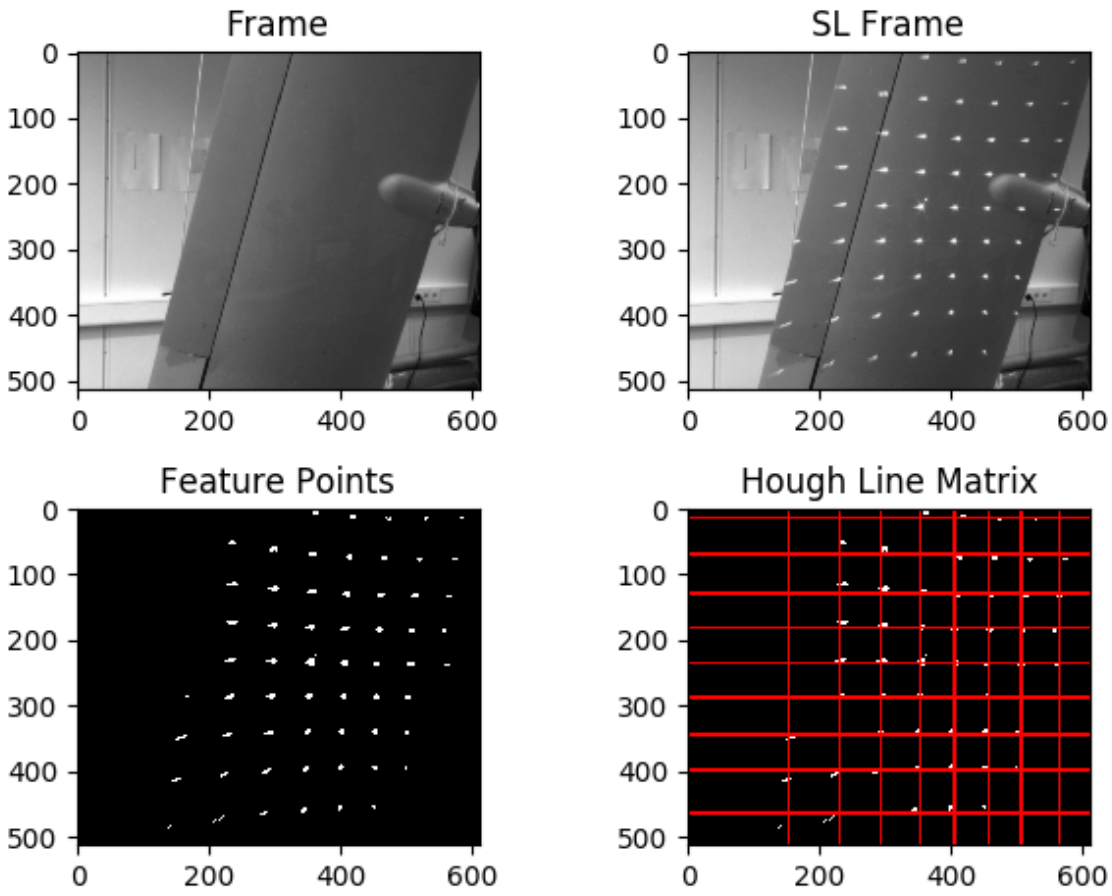


Figure 4.1: Hough line matrix

The frames on the first row shows the original images of the blade with and without structured light. Next row shows the detected feature points and the resulting Hough line matrix, computed using the respective feature points.

The algorithm for computing a Hough line matrix is implemented in the $detectLines$ module as $HoughLinesPointMatrix(..)$.

Step two of the method searches for feature points along each vertical and horizontal Hough line to compute lines limited to the area of the blade. The algorithm considers each vertical and horizontal Hough line in the Hough line matrix to represent a line of correlated feature points, however, the difficulty is to search for the correct feature points along the respective line. As in step one, this is solved by reffering to the fixed scale distance distance between the feature points and a given tuning parameter to create a search area along the respective vertical or horizontal Hough line. Furthermore, the detected feature points along the respective Hough line are then sorted with respect to frame positions, and interpolated to compute a bounded straight line

along those feature points. The bounded straight lines gives an indication of the blade area in the frame, as shown in figure 4.2, which will be used in step three to segment the blade. Step two of the method is implemented in the $detectLines$ module as $FindLineLimits(..)$.

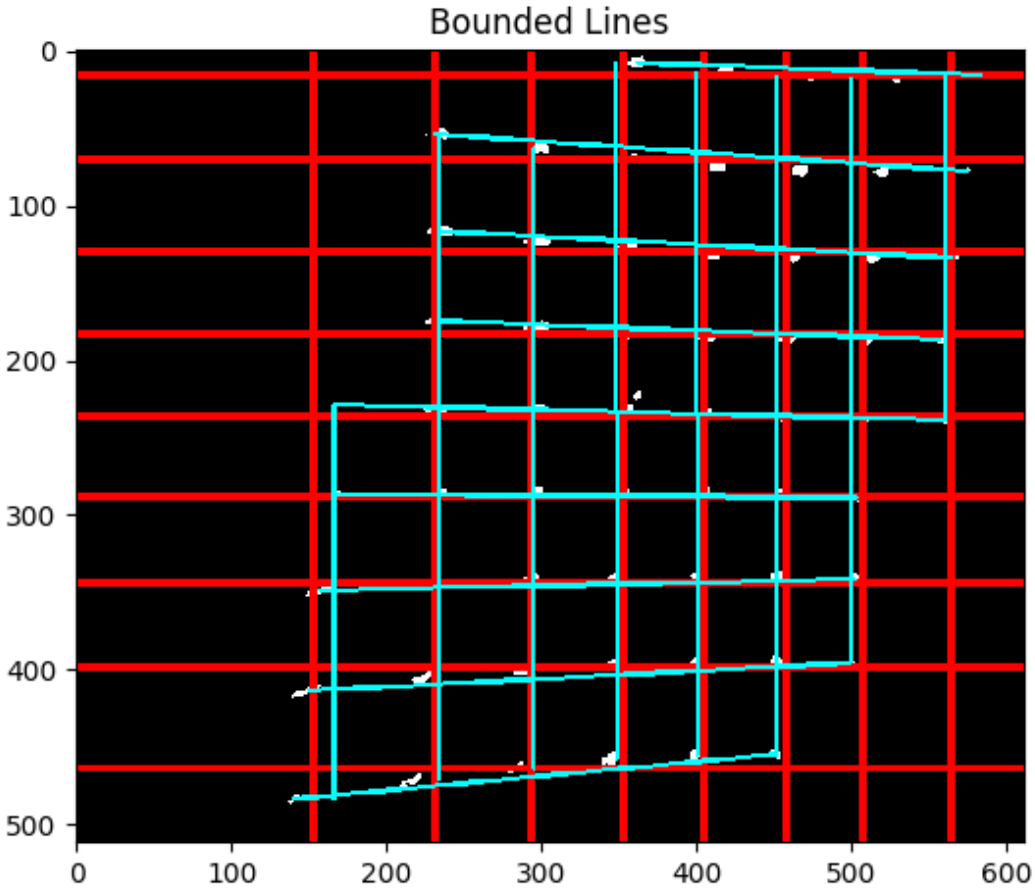


Figure 4.2: Bounded lines

This figure represents step two of the method, and shows the bounded lines as a result of interpolating correlated feature points along the horizontal and vertical Hough lines of the Hough line matrix.

4.2 Detecting Blade Edges as Hough Lines

Step three of the method aims at segmenting the blade with accuracy. The algorithm starts off by computing the edgel map of the original frame without structured light using the Canny edge detection method, as seen by the left image in figure 4.3. Moreover, the motivation is to segment the blade by localizing the blade edgels, so the algorithm considers the bounded lines as an indication on where to start a greedy search for the respective edgels. The greedy search starts at the end point of each bounded line and is conducted by detecting the first edgel in the horizontal or vertical direction, as illustrated by the arrows in figure 4.3.

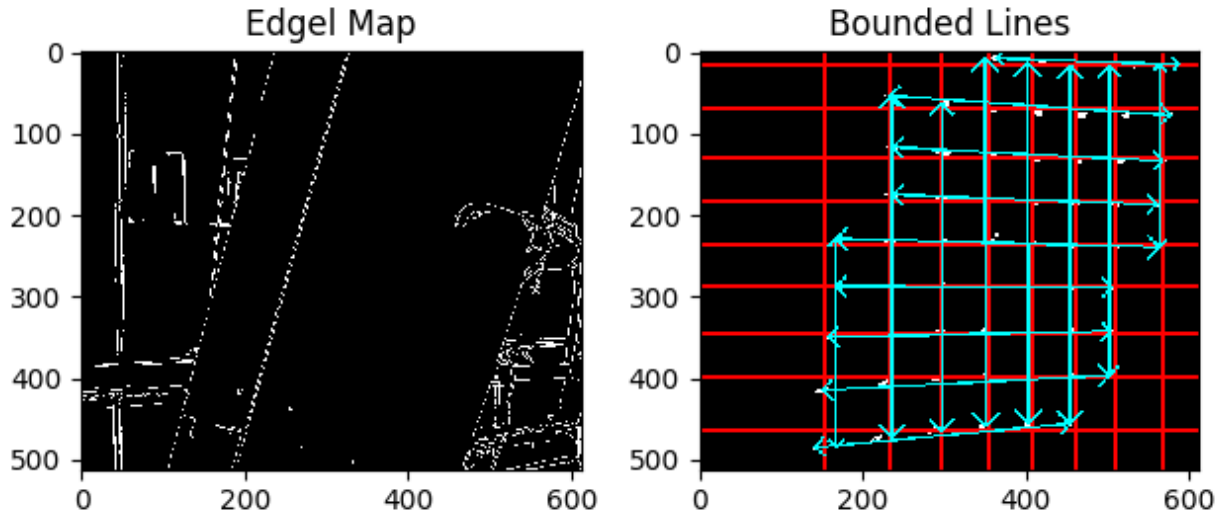


Figure 4.3: Edgel search using bounded lines.

Right image shows how the greedy search starts off at the end points of each bounded line to detect the most significant blade edgels in the edgel map to the left.

Step three results in a set of edgels along the four possible edges of the blade. However, some edgels may be wrongly detected as blade edgels, so simple filtration of the edgel points is commenced. The filtration simply removes edgels that are outside of the standard deviation of edgel points along the respective edge, and the result after filtration is seen in the left image in figure 4.4, which shows that the edgels are balanced around the highest concentration of edgel points.

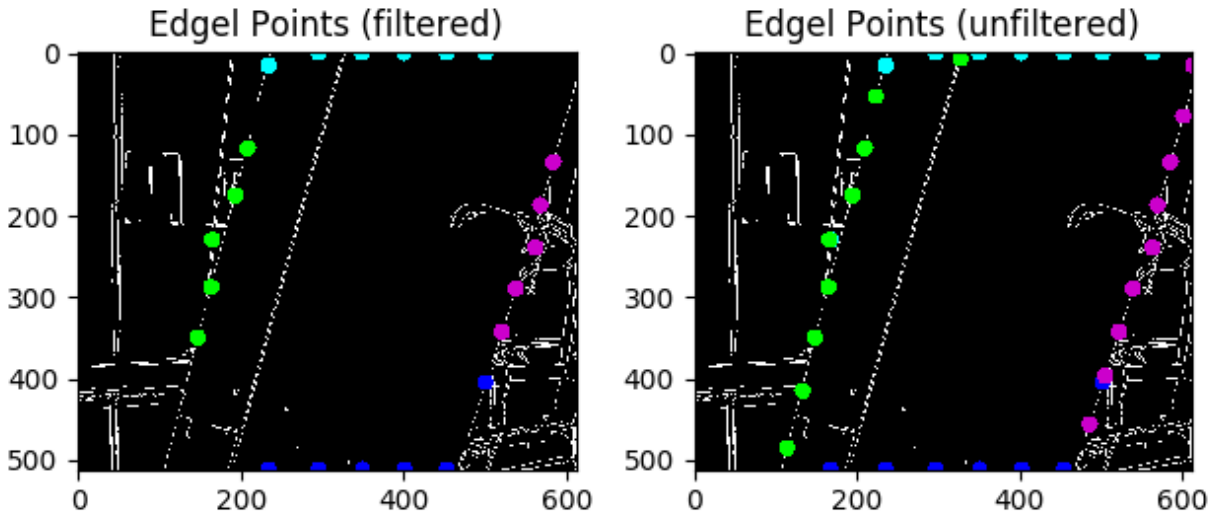


Figure 4.4: Edgel points

Left image shows the result of blade edge detection method with filtrated edgel points, while the right image shows the result with unfiltered edgel points.

The final step computes Hough line representations of the blade edges which is essential for finding a heading to follow, as discussed in chapter 5. Moreover, the Hough line transformation for each of the four edges is efficiently computed since only a few edgel points are parameterized during each transformation, and only horizontal or vertical degrees are considered for each horizontal or vertical blade edge, respectively. The horizontal degrees are limited to a range of radian degrees between $[0,..,\frac{\pi}{4}]$ and $[\frac{3\pi}{4},..,\pi]$, while the vertical degrees are limited to a range between $[\frac{\pi}{4},..,\frac{3\pi}{4}]$ radian degrees.

It was actually found that there were in many cases too few edgel points to give enough votes in the Hough parameter space, which could lead to strong areas of votes instead of a notable strong vote. This issue was solved by first dilating the acculumator of the Hough parameter space to expand strong votes, followed by smoothening the dilated accumulator using Gaussian

filtering to highlight the strongest vote in the strongest areas. Finally, both of the updated accumulators are summarized to compute a Gaussian strengthened accumulator.

Moreover, a given percent of the highest peaks in the Hough parameter space is chosen to represent the Hough lines along the respective blade edge, and the final Hough line is then computed by calculating the mean (ρ, θ) of those Hough lines. Step three and four of the method is implemented in the module $detectEdges$ as $DetectBoundaryEdges(..)$. Additionally, the result is illustrated in figure 4.5, with filtered edgel points in the left image and unfiltered edgel points in the right image.

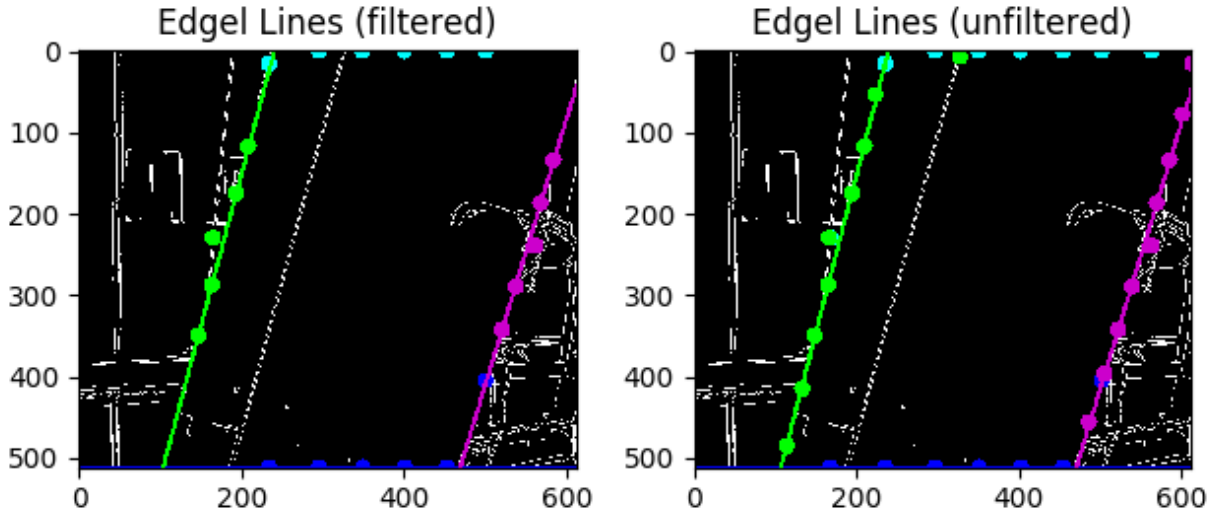


Figure 4.5: Edgel lines

Left image shows the result of blade edge detection method with filtrated edgel points, while the right image shows the result with unfiltered edgel points.

All of the steps are merged into a single function which is found in the $PointDetection$ module as $GetBoundaryHoughLines(..)$, and the computational delay for processing all of the respective steps was found to be approximately 0.10 seconds on a 512×612 sized frame. This is achieved by an extensive use of the numpy library and the OpenCV library to conduct efficient matrix operations.

4.3 Discussion & Comments

The blade detection method presented in this chapter efficiently segments the area of interest and detects the blade edges with good accuracy. Moreover, an important benefit of using this method is that the blade will be detected with acceptable accuracy in almost any circumstances, even if the Canny edge detector produces a highly disturbed edgel map, since the algorithm segments the area of interest using the feature points to minimize the edgel search along the respective blade edge. It is actually recommended to set the hysteresis thresholding low on the Canny edge detector, so that the blade edges are easily detected, although all else will become increasingly cluttered. An extreme example of this is illustrated in figure 4.6.

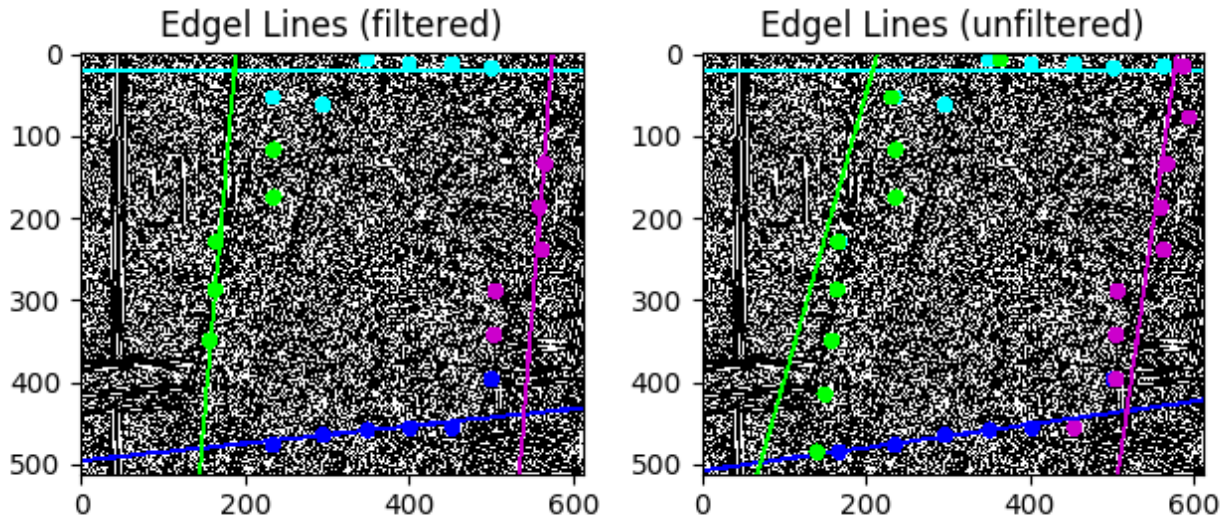


Figure 4.6: Edgel lines on a cluttered edgel map.

An extreme example of setting the hysteresis thresholding at minimum. The result from a badly tuned Canny edge detector is notably worse than when it is properly configured as seen in figure 4.5, although the result is acceptable.

As mentioned, the detected edgel points along the respective edge is filtrated by removing edgel points outside of the standard deviation of edgel points. This type of filtration is selected since it is easily implemented when considering filtration of two dimensional positions. The benefit of using this filtration method is that strong deviations are removed, since only the centered points remains. However, this also means that correct outliers also will be removed, which decreases accuracy of the final Hough line representation. A blade edge detection example is illustrated in figure 4.7, where it is seen that correct outliers are removed by the filtration, and that the centered edgel points remains.

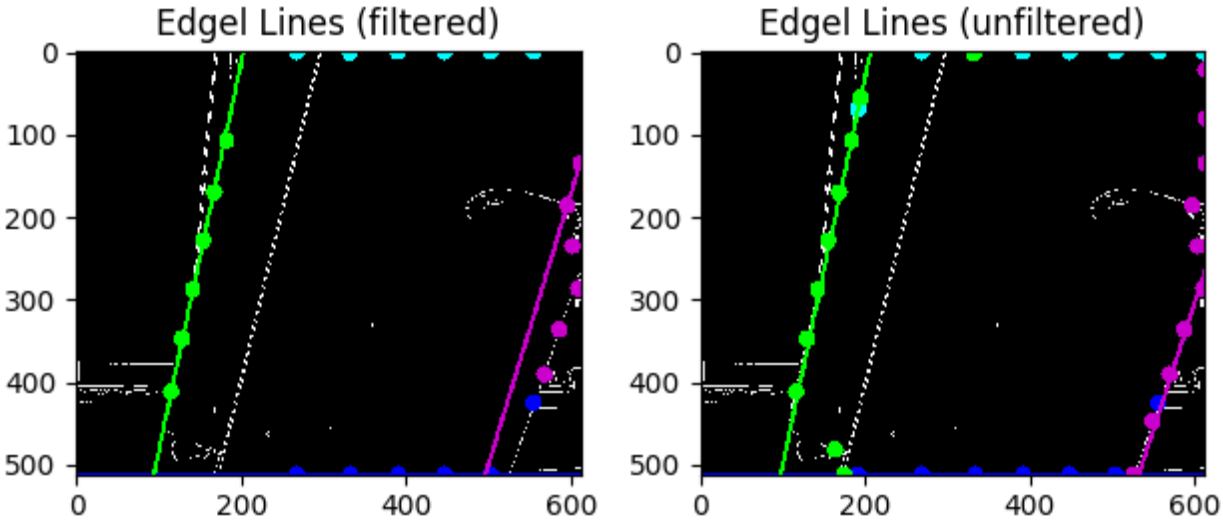


Figure 4.7: Unsuccessfull filtration of edgel points.

The unfiltered result is shown to give better result than the filtered result, due to that the filtration removes correct outliers.

However, it was considered that the benefits of using the filtration method outweighs the consequences, which was verified by testing the method in different circumstances. An example of this testing is presented in figure 4.8, where it is seen that some of the incorrect edgel points are removed, thus leading to a better result.

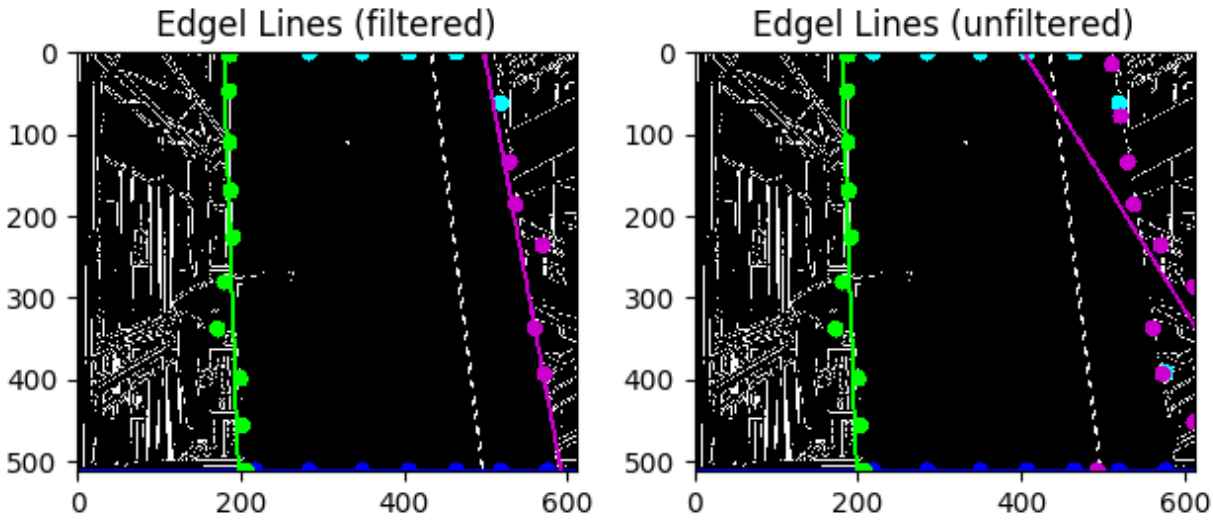


Figure 4.8: Successfull filtration of edgel points.

Some edgels along the blade edge, marked in purple, was not correctly detected. However, the filtration removed these edgel points since they are outside of the standard deviation of connected edgel points.

This method is proven to be an efficient blade edge detector, and it may also be used to detect a wide range of different objects such as road signs, building corners or even flag poles. The concept could also be adopted to detect other shapes, instead of a rectangular shape, as defined by the horizontal and vertical lines. This could be done by redefining the final step of the method to detect any shape that can be transformed to the Hough parameter space, such as circles or ellipses. As an example, the Hough curve transform, see section 2.3.2, will find circles using the detected edgel points, which will be more efficient than considering all possible edgels in the edgel map. Additionally, the detected object will be easy to work with in an arbitrary application since it is represented by the given shape. As in this case, the manouvering system will use the Hough line representations of the blade edges to estimate a heading along the respective blade. Finally, some additional samples of using the edge detection method on arbitrary objects are shown in figure 4.9.

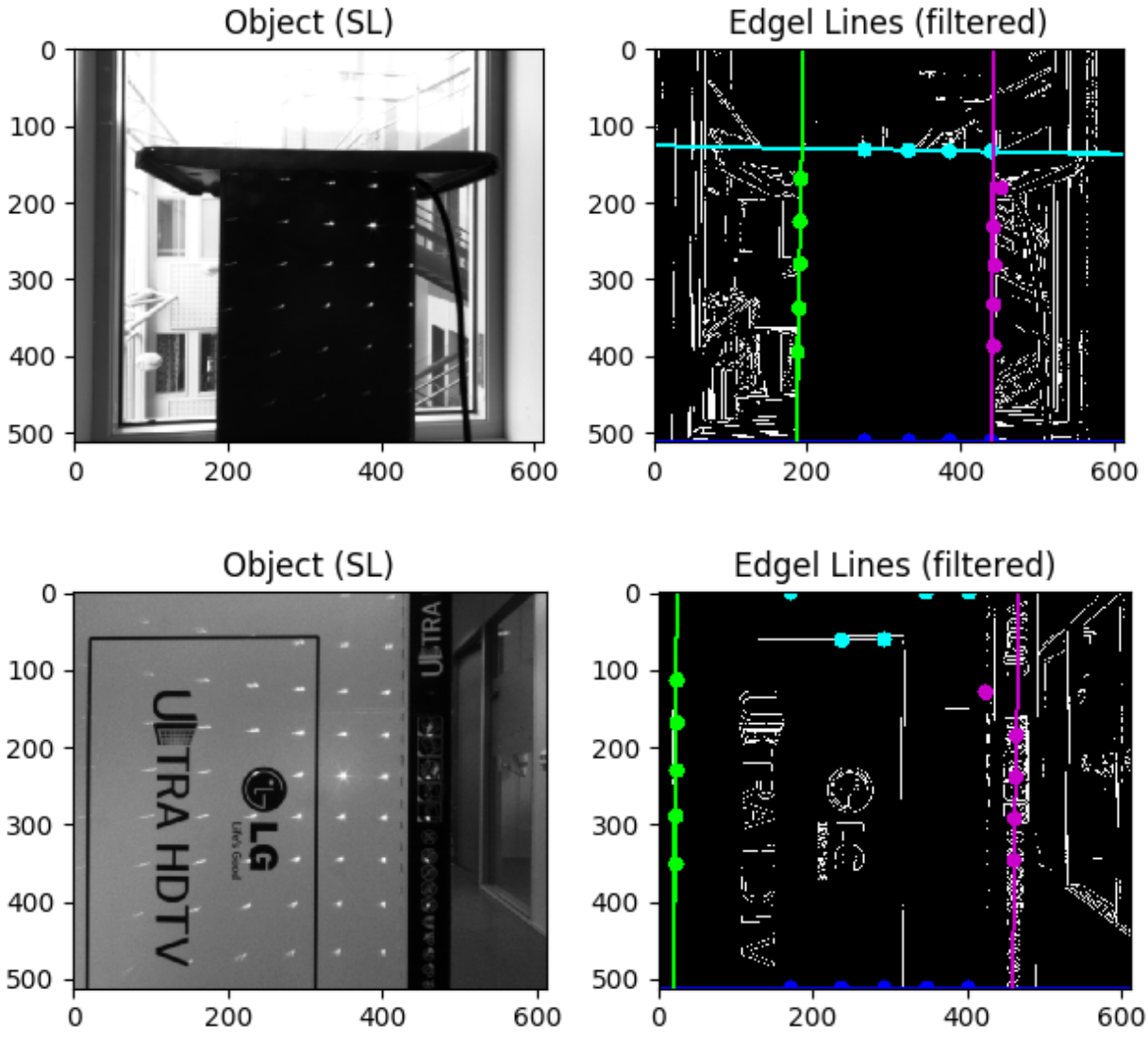


Figure 4.9: Edge detection on arbitrary objects.

Chapter 5

Manouvering Plan for Wind Blade Inspection

The previous chapters presented machine vision theories, which will be used to manouver the UAV along a wind blade. The edge processing methods discussed in chapter 2.3 estimates the edges of the blades as straight lines represented by angle θ and distance ρ according to a clockwise orientation from the image center. This chapter will address the issue of manouvering the UAV drone along the wind blades on the basis of the computer vision system, and the implemented version may be accessed through the $Heading$ module. Moreover, manouvering simulations of the essential manouvering propositions discussed in this is chapter will be presented in section 5.11. However, the implementation following this thesis does not include a Gimbal solution, although it is a part of the solution presented in this chapter.

5.1 Coordinate Frames

The coordinate frames according to the UAV body frame, the camera frame, the image frame and the windmill frame are referred to as b, c, i and W, respectively. The coordinate frame of the UAV drone is modelled by Kristian Klaussen, and the windmill is modelled by Torjus Sveier Ottemo [49], as illustrated in figure 5.1

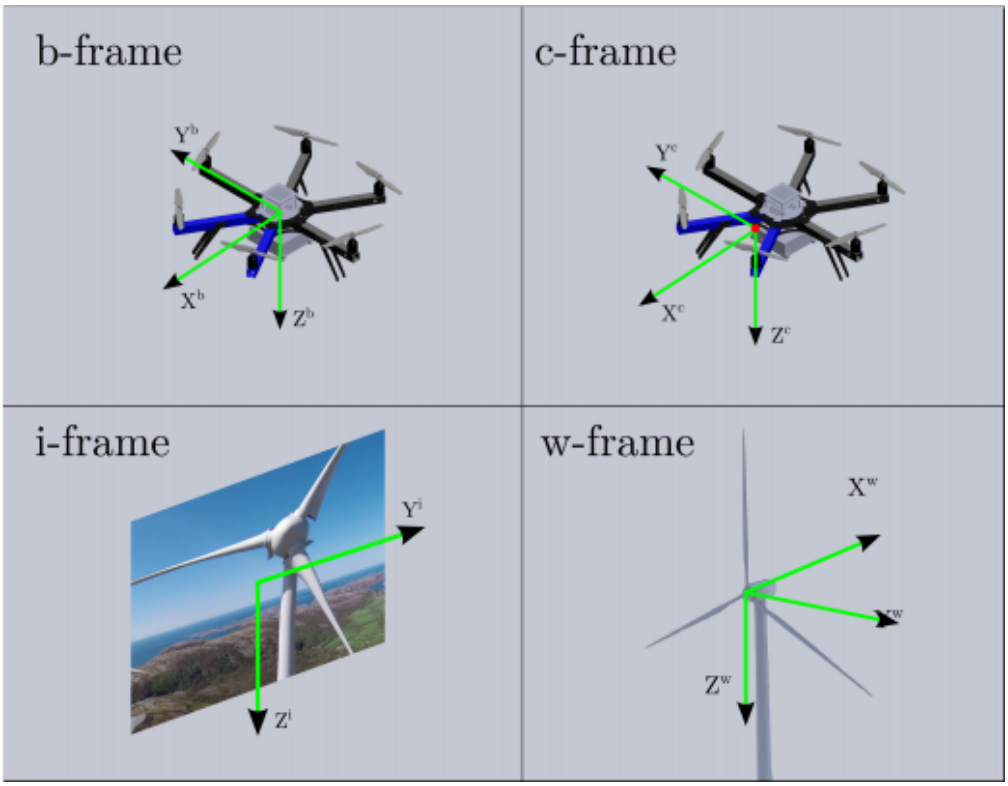


Figure 5.1: Coordinate frames

(Courtesy of Martin Stokkeland, Kristian Klaussen & Torjus Sveier Ottemo [49])

5.1.1 Body Frame

The body frame is centered at the centre of the geometrical centre of the UAV, and follows the standard 6 DOF, defined by the conventional surge, sway, heave, roll, pitch and yaw motion components as u, v, w, ϕ, θ and ψ, respectively. The surge, sway and heave corresponds to the translational motion along the longitudinal axis, X^b, the latitudinal axis, Y^b, and the altitudinal axis, Z^b.

5.1.2 Camera Frame

The parallel cameras will be mounted underneath the UAV on a gimbal which can move the pitch and yaw angle of the camera frames. The camera frames must therefore be rotated around the latitudinal and altitudinal axis according to the pitch and yaw angle of the camera frames, respectively, to relate it to the body frame. Furthermore, the left camera is chosen as the main camera frame related to the body frame and will be positioned close to the geometrical centre of

the UAV, so that the translational displacement between the body frame and the camera frame is negligible. Both cameras frames will therefore be referred to as a single camera frame, according to the left camera.

5.1.3 Image Frame

The image frame is the 2D projection of the corresponding camera, and will be centered in the centre of the image due to simplicity. The latitudinal and altitudinal axes of the image frame is parallel with the latitudinal and altitudinal axes of the corresponding camera frame. These statements contradicts the common image frame, as presented in section 2.4.2, where the origin is placed in the top left corner with axes defined according to the conventional right-hand rule. However, this image frame simplifies the transformation mapping from the camera frame by eliminating the difference in origin and direction of the axes. The transformation mapping from the image frame to the camera frame will then be defined by the effective pixel size only, defined by equation (5.1), where (S_z, S_y) are defined as the effective pixel sizes in the altitudinal and latitudinal axis respectively.

$$z = z_{im} S_z \qquad y = y_{im} S_y \tag{5.1}$$

Due to this simplification we neglect the effective pixel size throughout the rest of this chapter so that the perspective projection of 2D points to 3D points in the camera frame can be directly related to pixel coordinates. However, the implementation must consider that the origin of the image coordinates will initially be given in the top left corner and a transformation to the camera frame must include the effective pixel sizes.

5.1.4 Windmill Frame

The windmill frame is centered at the hub center of the wind turbine as seen from the direction of the windmill, with the longitudinal, latitudinal and altitudinal axes pointing from front to back, from right to left and from top to bottom, respectively. This is chosen so that it corresponds with the alignment of the body frame when the UAV is facing the windmill.

5.2 Detecting the Wind Blades

The initial position of the UAV will be directly in front of the windmill so that the body frame and windmill frame alignes. From that position, the UAV can detect the angle of each wind blade according to the hub center of the wind turbine, and sort out which blades have been inspected and which blade to do next. However, detecting the blades must be done from a distance where the turbine and blades are in view, and then proceed towards the chosen blade. This thesis will focus on the issue regarding manouvering along the chosen wind blade, thus there will only be a brief discussion regarding the issues with manouvering towards the selected wind blade.

The blades can be detected using the Hough transform, exploiting the straight lines of the blades when they are viewed from a distance. The angle of each blade can then be approximated according to the latitudinal axis of the windmill frame Y^W with a clockwise orientation as illustrated in figure 5.2. The clockwise orientation is chosen due to that the Hough transform measures the angles of the blade lines in the image frame according to a clockwise orientation, which will be usefull when manouvering along the wind blade. Approaching a selected wind blade can be managed by manouvering the UAV towards a fixed position at the beginning of the wind blade until the UAV is close enough for the stereo vision system to reconstruct valid 3D points from the structured light projected onto the blade, as well as using the structured light to detect the blade edges.

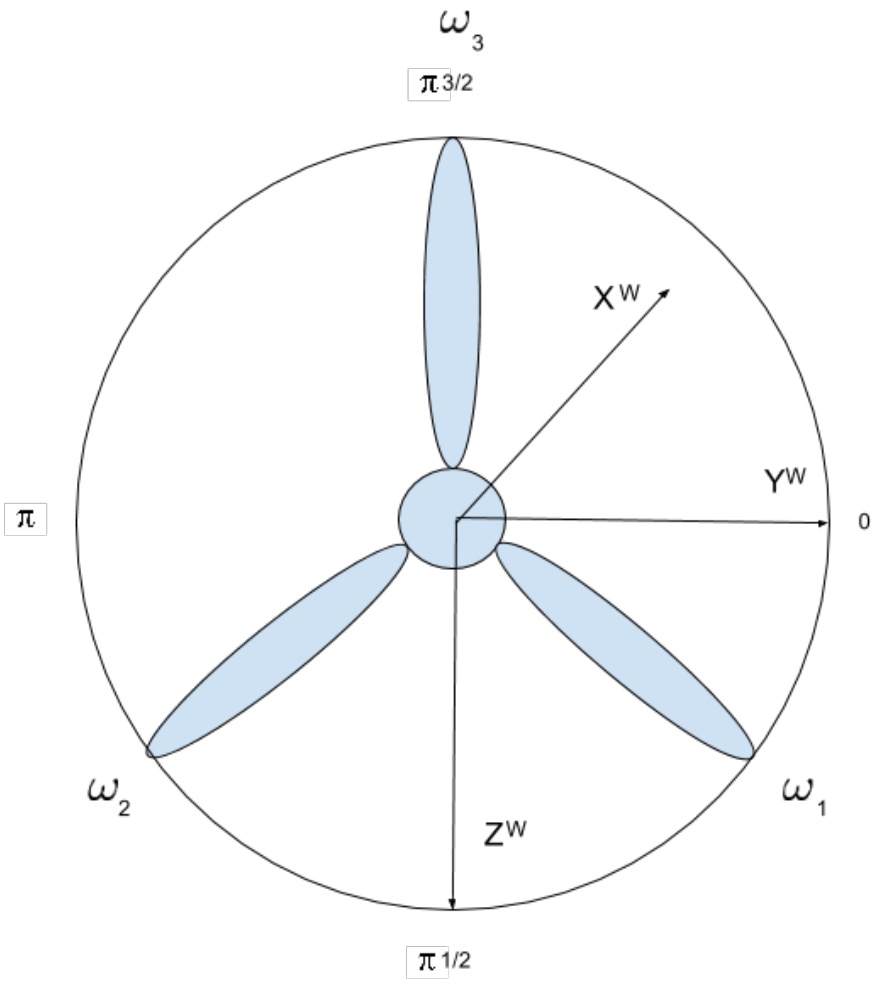


Figure 5.2: Angle, ω_i, of a wind blade according to the latitudinal axis Y^W.

The sway and heave of the body frame is found be decomposing the direction of heading V according to the blade angle ω_i as stated in equation (5.2), while the surge should be fixed so that the UAV drone slowly approaches the initial position of the wind blade. The initial position of the wind blade should be positioned at a fixed distance from the wind turbine, along the blade angle. Note that the latitudinal and altitudinal position according to a position along the wind blade follows the same decomposition as stated in equation (5.2).

$$\begin{aligned} v^b &= V\cos(\omega_i) \\ w^b &= V\sin(\omega_i) \end{aligned} \tag{5.2}$$

5.3 Initial Position

As discussed in chapter 3, a typical blade design indicates that the initial position of the UAV drone should be in the root region of the blade, as close as possible to the turbine to be able to inspect most of the blade. Furthermore, both sides of the blade can be inspected by the UAV when manouvering from the front of the windmill, due to the geometry of the blades. A blade inspection should start off by manouvering to the initial position on the top or bottom side of the blade, according to which side is chosen to inspect. The UAV will then proceed with manouvering along the side of the blade which is in view by the camera.

It is essentially important for the UAV to keep the yaw angle of the body frame in the same yaw angle as of the windmill frame. This statement simplifies the manouvering of the UAV, since it will know which direction is heading towards the tip or root region of the blade. The yaw angle of the windmill is fixed, and can therefore be related to by a compass course registered when the UAV is directly facing the turbine hub. Yaw angle τ_ψ is found by measuring the present compass course of the UAV, representing the present yaw angle of the body, and proceed with a standard PID-regulator, as stated in equation (5.3).

$$\tau_\psi = K_p(\psi_W - \psi_b) + K_i \int_0^t (\psi_W - \psi_b)\,dt - K_d\dot{\psi}_b \tag{5.3}$$

5.4 Initialization of the UAV

Manouvering along the blade is not a straightforward approach due to the differing chord length and oval blade design, and the problem should therefore be kept as simple as possible. The first simplification will be to consider that the UAV will start at the initial position in the root region of the blade, with the longitudinal axis of the body frame parallel with the longitudinal axis of the windmill. The camera frame is initially focused at a fixation point on the blade by regulating the pitch and yaw of the gimbal, which will be described in section 5.5, so that the longitudinal axis of the camera frame is normal on the blade surface. Figure 5.3 illustrates a simplification of this approach, where the body frame starts at the initial position with the camera frame facing the blade. The blade will then be inspected in two sequences, first by inspecting the leading side of the blade when the UAV is moving towards the tip, and then inspecting the trailing side on the

way back to the root region of the blade. This approach is chosen since the oval design of the blade restricts the camera from inspecting the whole surface of the blade if it were to manouver straight above it.

The manouvering along the leading side will start by manouvering the UAV towards the leading side and then regulating the yaw and pitch angle of the camera frame so that the longitudinal axis of the camera are normal on the blade surface. A proposed fixation point on the leading side is chosen as the closest area of 3D points, while accounting for keeping the leading edge in view to the cameras. Furthermore, the yaw and pitch of the camera frame will then be fixed in respect to the windmill frame during the navigation towards the tip, and back to the root region. The camera frame needs to be regulated in response to the movements of the body frame during the navigation along the leading- and trailing side, to keep a steady angle according to the windmill frame. This solution will keep the longitudinal axis of the camera normal to the blade during the navigation along the edges, even though the body frame tilts when the UAV is moving or is tilted due to strong winds.

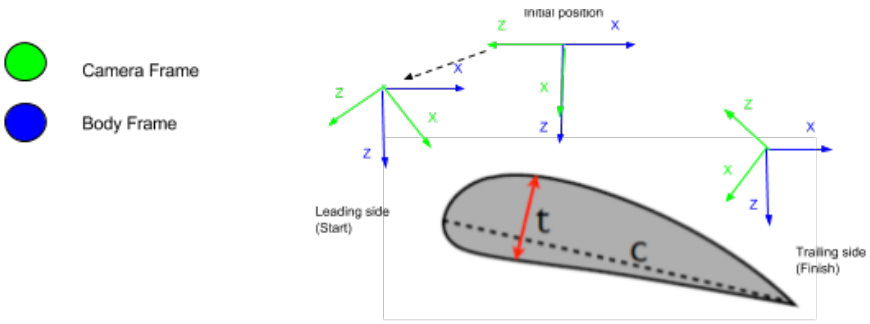


Figure 5.3: Relation between camera frame and body frame when positioning the UAV for blade inspection.

(This simplification neglects the translation along the Y-axis.)

Figure 5.4 illustrates a simplification on the issue of manouvering the UAV from the initial position to the leading side, or back from the trailing side to the root region. By considering that the camera frame is facing the blade at the initial position, the stereo vision system will be able to detect the edges following the leading and trailing side using the structured light, and estimate the distance to the blade by reconstructing corresponding 3D points. Note that the UAV might need to be positioned at an additional distance away from the blade at the initial position

to detect the leading and trailing edge in the same image. This additional minimum distance to the blade is decreased when the UAV is moving towards the leading edge. We assume an ideal situation, and the edge detection algorithm computes a straight line following the leading side and trailing side, respectively, using the method presented in chapter 4. The Hough transform of the blade edges provides the angle θ and distance from origin ρ in the image frame, which essentially is the heading direction towards the respective line. Thus, the Hough transform provides the heading direction towards the leading side according to angle θ_{ls} and distance ρ_{ls}. The heading direction is decomposed according to the latitudinal and altitudinal axis, as stated by equation (5.4), and provides the desired position $\mathbf{P}^i = [p_x^i, p_y^i, p_z^i]^\top$ in pixels according to the image frame.

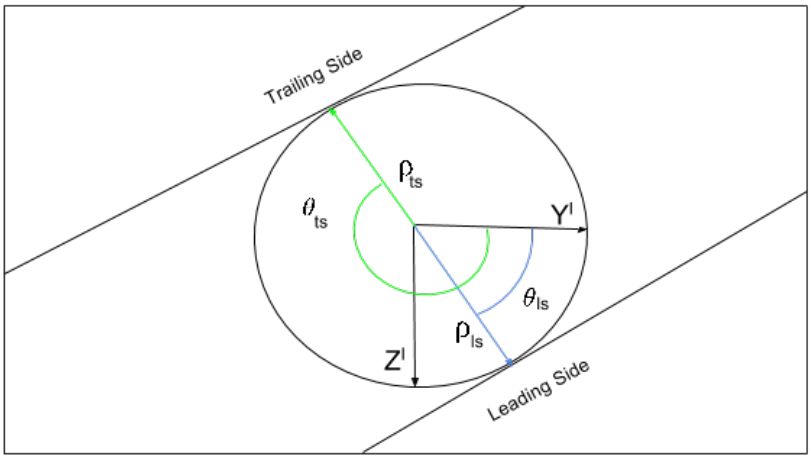


Figure 5.4: Heading direction to approach the leading side.

$$\begin{aligned} p_y^i &= \rho_{ls}\cos(\theta_{ls}) \\ p_z^i &= \rho_{ls}\sin(\theta_{ls}) \end{aligned} \tag{5.4}$$

The stereo vision system provides the longitudinal distance to the blade in real coordinates according to the camera frame, recall equation (2.37). Furthermore, the fundamental equations of perspective cameras, recall equation (2.39), relates the longitudinal position in the camera frame with camera coordinates in the same position, according to the focal length and pixel coordinates. The fundamental equations of perspective cameras, as stated by equation (5.5), is

rearranged with known longitudinal position, $X = f\frac{T}{d_x}$, which yields equation (5.6).

$$\begin{aligned} \frac{p_y^i}{p_y^c} &= \frac{f}{X} \\ \frac{p_z^i}{p_z^c} &= \frac{f}{X} \end{aligned} \tag{5.5}$$

$$\begin{aligned} p_y^c &= \frac{T}{d_x} p_y^i \\ p_z^c &= \frac{T}{d_x} p_z^i \end{aligned} \tag{5.6}$$

This relation is rearranged using equation (5.4), which yields equation (5.7). This equation relates the latitudinal and altitudinal position in the image frame with the corresponding position in the camera frame, according to the disparity d_x in that position.

$$\begin{aligned} p_y^c &= \frac{T}{d_x} \rho_{ls} \cos(\theta_{ls}) \\ p_z^c &= \frac{T}{d_x} \rho_{ls} \sin(\theta_{ls}) \end{aligned} \tag{5.7}$$

However, the longitudinal distance may also be estimated by considering that the blade is flat, when viewed by the camera, meaning the reconstructed 3D points of the respective feature points, as discussed in chapter 2.5, can be used to compute an estimate of the longitudinal distance close to the respective position. Desired position $\mathbf{P}^i$ may then be found by utilizing the fundamental equation of perspective cameras, as stated by equation (5.5).

Furthermore, the UAV should follow a fixed distance D_d to the blade according to the distance, $D_x = f\frac{T}{d}$, measured by the stereo vision system. By accounting for the relative error between the fixed distance and measured distancy, we derive the position along the longitudinal axis according to the camera frame to follow, which yields equation (5.8).

$$p_x^c = D_x - D_d = f\frac{T}{d} - D_d \tag{5.8}$$

By combining equation (5.7) and (5.8), we derive the position vector by equation (5.9) according to the camera frame.

$$\mathbf{P}^c = \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \end{bmatrix} = \begin{bmatrix} f\frac{T}{d} - D_d \\ \frac{T}{d_x}\rho_{ls}\cos(\theta_{ls}) \\ \frac{T}{d_x}\rho_{ls}\sin(\theta_{ls}) \end{bmatrix} \tag{5.9}$$

The positioning of the UAV according to a desired destination aquired from the camera frame is found by a rotation matrix around the latitudinal and altitudinal axis, as stated by equation (5.10). Recall that the translational displacement between the camera frame and body frame is negligible.

$$\mathbf{P}^b = R_c^b(\Theta_c)\mathbf{P}^c \tag{5.10}$$

$$R_c^b(\Theta_c) = R_{z,\psi_c}R_{y,\theta_c} = \begin{bmatrix} \cos\psi_c & -\sin\psi_c & 0 \\ \sin\psi_c & \cos\psi_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_c & 0 & \sin\theta_c \\ 0 & 1 & 0 \\ -\sin\theta_c & 0 & \cos\theta_c \end{bmatrix} \tag{5.11}$$

5.5 Rotating the Camera Frame

As mentioned in section 5.4, the camera frame needs to be tilted to focus the image projection to a fixation point on the wind blade. This is achieved by rotating the camera frame according to a yaw and pitch angle as ψ_{ref} and θ_{ref}, respectively, which are derived by focusing the longitudinal axis towards the fixation point. The fixation point is decomposed according to the latitudinal and altitudinal axis and derived using the fundamental equation of perspective cameras, recall equation (5.6). Furthermore, the longitudinal position of the fixation point is known by the stereo vision system as $X = f\frac{T}{d_x}$. By this, we can derive the yaw and pitch angle according to the pixel position in latitudinal axis and altitudinal axis from the image center as p_y^i and p_z^i, respectively, and the longitudinal position X according to the camera frame, which derives equation (5.12).

$$\begin{aligned}\tan(\theta_{ref}) &= \frac{p_z^c}{X} = \frac{p_z^i \frac{T}{d_x}}{f\frac{T}{d_x}} = \frac{p_z^i}{f} \\ \tan(\psi_{ref}) &= \frac{p_y^c}{X} = \frac{p_y^i \frac{T}{d_x}}{f\frac{T}{d_x}} = \frac{p_y^i}{f}\end{aligned} \tag{5.12}$$

The gimbal is then regulated using a PD-regulator according to the estimated yaw and pitch reference angle, as stated in equation (5.13). A PD regulator is chosen since the gimbal is assumed to be very responsive.

$$\begin{bmatrix}\tau_\theta \\ \tau_\psi\end{bmatrix} = K_p \int_0^t \begin{bmatrix}\theta_{ref} - \theta_c \\ \psi_{ref} - \psi_c\end{bmatrix} - K_d \begin{bmatrix}\dot{\theta}_c \\ \dot{\psi}_c\end{bmatrix} \tag{5.13}$$

5.6 Manouvering Along the Blade

The next step after approaching the start position on the blade is to follow the angle of the wind blade until the blade tip is detected, or the root region of the blade on the way back is detected. The basic approach of following the blade is to measure the angle ω of the blade to follow. By simplifying the issue, we consider that the edge processing algorithm has detected two straight lines, representing the edges of the blade as illustrated in figure 5.5.

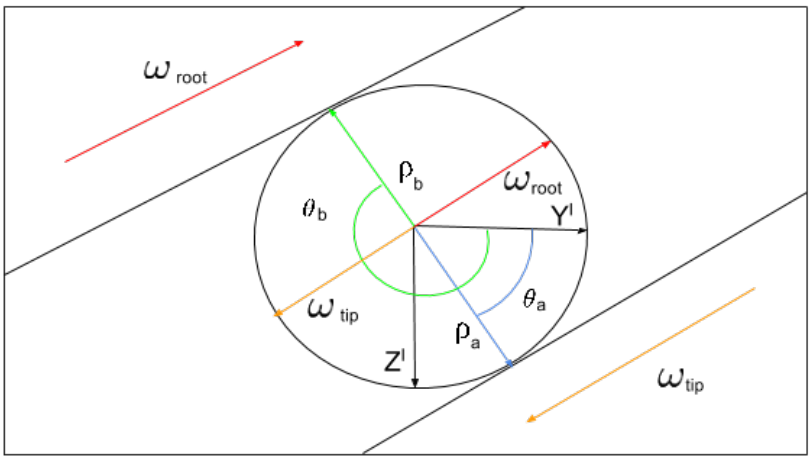


Figure 5.5: Heading direction ω according to the blade edges.

The direction of heading towards the tip is then measured by finding the angle of the blade ω_{tip} by simply deriving the average angle of the straight lines, as stated in equation (5.14). The angle back to the root region of the blade is found be rotating the heading angle towards the tip by π radians. However, it should be noted that this method is feasible only if the lines are on each side of the image center, thus in the opposite quadrant regions of the polar space.

$$\begin{aligned} \omega_{tip} &= \frac{\theta_a + \theta_b}{2} \\ \omega_{root} &= \omega_{tip} + \pi \end{aligned} \tag{5.14}$$

The UAV might occasionally drift away from the direction of heading, resulting in that the view of either one of the edges is lost, as illustrated in figure 5.6.

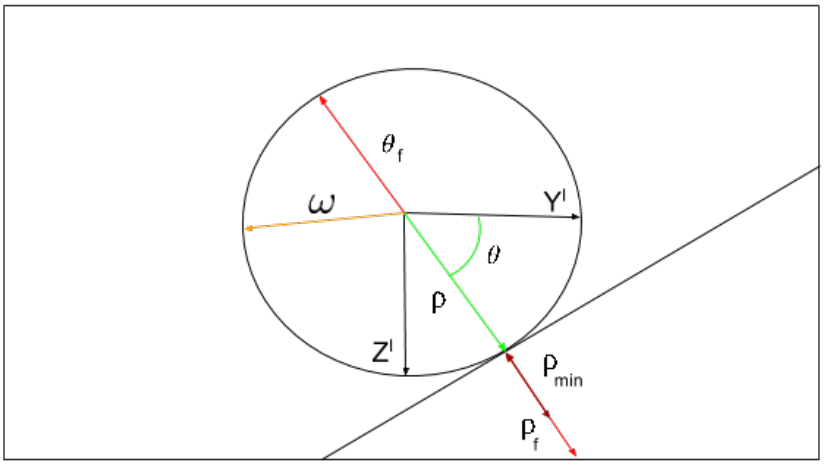


Figure 5.6: Heading direction ω when only the top edge is in view.

In such a case, the UAV will consider the angle θ and distance ρ_f, according to the edge in view, to estimate the heading angle ω. The distance ρ_f is the measured distance from the line to the image edge, and is found by considering the distance ρ to the line according to the length and width of the image as I_l and I_w, respectively. Distance ρ_f from the line to the image edge is found by subtracting ρ with the distance to the image edge from origin, as stated by equation (5.15). It should be noted that the image edge is referred to as an elliptical boundary of the image from the center, which is easier to consider and implement.

$$\rho_f = \frac{1}{2}\sqrt{I_w^2\cos^2(\theta) + I_l^2\sin^2(\theta)} - \rho \tag{5.15}$$

The heading angle is then estimated by minimizing the distance ρ_f, while manouvering the UAV towards the tip or root region of the blade. Moreover, it should be noted that it is essentially important to restrict the UAV from loosing the edge which is still in view, so the UAV will be restricted from searching for the lost edge according to the minimum distance ρ_{min}, meaning $\rho_{min} \leq \rho_f$. Furthermore, ρ_{min} may be defined as a fixed percent of the diagonal distance from center to the image edge, as defined by equation (5.16).

$$\rho_{min} = k\frac{1}{2}\sqrt{I_w^2\cos^2(\theta) + I_l^2\sin^2(\theta)}, \quad 0 \leq k \leq 1 \tag{5.16}$$

Furthermore, the heading angle towards the respective tip or root region of the blade is estimated by rotating the line angle θ by $\pm\frac{\pi}{2}$ radians, according to which direction the UAV is heading. Additionally, the rotation of the line angle will be a weigthed rotation to counteract the lost edge while not loosing the edge in view. The weighted rotation parameter W_f is an estimate of the proportional relation between the distance ρ and minimum distance ρ_{min}. By conclusion, the heading angle is stated by equation (5.17) while the weighted rotation parameter is stated by equation (5.18). As an example, the weighted rotation will move the heading angle closer to θ_f if $\rho_{min} \leq \rho_f$, thus the drone will be navigated to get more of the blade in view. On the other hand, it will move the heading angle closer to the line angle θ if $\rho_{min} > \rho_f$, which navigates the drone to get less view of the blade.

$$\omega = \theta \pm \frac{\pi}{2}W_f \tag{5.17}$$

$$W_f = \begin{cases} 2 - \frac{\rho_{min}}{\rho_f}, & \rho_{min} \leq \rho_f \\ \frac{\rho_f}{\rho_{min}}, & \rho_{min} > \rho_f \end{cases} \tag{5.18}$$

It should be noted that this manouvering solution considers that the image center is focused on the blade, which means that the blade edge is in the correct quadrants of the polar space. The respective edge will be located in the wrong quadrant if the blade is not localized at the

image center. However, a simple solution to this issue is to redefine equation (5.15) and (5.16) to estimate the distance ρ_f and ρ_{min} in the opposite direction, as follows in equation (5.19) and (5.20), to estimate a heading that relocates the blade at the image center.

$$\rho_f = \frac{1}{2}\sqrt{I_w^2\cos^2(\theta+\pi) + I_l^2\sin^2(\theta+\pi)} + \rho \tag{5.19}$$

$$\rho_{min} = k\frac{1}{2}\sqrt{I_w^2\cos^2(\theta+\pi) + I_l^2\sin^2(\theta+\pi)}, \quad 0 \leq k \leq 1 \tag{5.20}$$

Moreover, this issue will also result in redefiniton of equation (5.17) and (5.18), whereas the problem is mirrored as defined by the following equations of (5.21) and (5.22). The equation for estimating the weighted rotation parameter W_f is simplified since this issue will cause ρ_{min} to always be shorter than ρ_f.

$$\omega = \theta \mp \frac{\pi}{2} W_f \tag{5.21}$$

$$W_f = \frac{\rho_{min}}{\rho_f} \tag{5.22}$$

Moreover, the UAV will counteract the issue of having lost both edges by manouvering towards the leading or trailing side according to either that the UAV is heading towards the tip or root region of the blade, respectively. The response of locating the lost edge along the leading or trailing side is found by registering the last measured angle θ of the line following the leading or trailing side, and counteracting the manouvering accordingly. An alternative approach, is to manouver the UAV back to the last known position of the blade edge, however, a solution may be a combination of both approaches.

The proposed manouvering plan requires a fixed velocity for the UAV to follow in the angle of direction, and it should be noted that a slow UAV is safer to manouver along the blade due to response delay from the guidance system. Moreover, the camera frame will be tilted according to the pitch and roll of the body frame which is proportional to the velocity along the latitudinal and longitudinal axes. It is therefore essential that the velocity of the UAV is kept at a minimum, in order to minimize the tilt of the camera frame.

Finally, the direction of heading towards the tip or root region of the blade is computed by the same principle as stated in equation (5.9). Accordingly, the heading angle ω, respectively towards the tip or root region, and a fixed step distance D_m in meters yields equation (5.23). As noted, the fixed step distance D_m should be kept at a minimum to minimize the velocity of the UAV. The guidance of the UAV according to the body frame is derived from equation (5.10).

$$\mathbf{P}^c = \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \end{bmatrix} = \begin{bmatrix} f\frac{T}{d} - D_d \\ D_m \cos(\omega) \\ D_m \sin(\omega) \end{bmatrix} \tag{5.23}$$

5.7 Detecting the Tip

The UAV must continue to manouver towards the tip region of the blade until the tip is detected. From the design properties of the wind blade, as presented in chapter 3, we can distinguish a small curved edge at each side of the tip which might resemble a corner following the leading and trailing side of the blade. This assumption means that the tip can be detected by searching for a corner along the leading edge and trailing edge using the Harris corner detector. A simplified illustration of the corners C_a and C_b at the blade tip is shown in figure 5.7. Additionally, the machine vision system will also match the corners with an infinite background distance in the region following the blade edges and next to the corners to verify that the tip is detected. As mentioned, the structured light will provide information on where the blade is located in the image, meaning the detected feature points of the structured light may be used as a basis for detecting the blade corners, as well as the edges. As mentioned in section 2.3.3, the Harris corner detector is non-invariant to image scale meaning the distinct corners along the tip might be left undetected if the kernel size of the Harris corner detector is too small to detect the corners. This issue is resolved by considering the fixed scale distance between the structured light points, so that the kernel size can be adapted to the estimated corner scale according to the feature points.

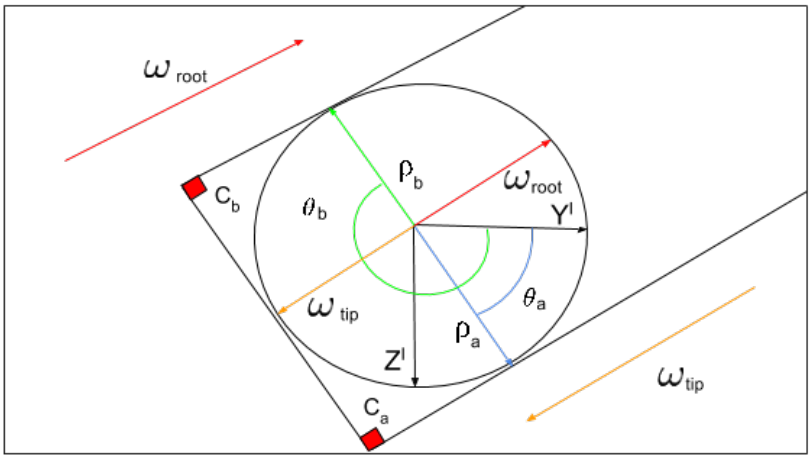


Figure 5.7: Corner detection at the blade tip.

Another approach is to consider that only a maximum of two edges should be detected when manouvering along the blade, while a third edge, approximately vertical to the leading and trailing edges, will be detected when the drone is at the tip of the blade. Both approaches was tested to verify if the respective method is feasible, and the result is presented and discussed in section 5.11.2.

5.8 Manouvering Back To the Root

The next movement of the UAV will be to inspect the trailing side of the blade on the way back to the root region, as mentioned in section 5.4. This movement involves positioning the UAV and camera frame according to the trailing side of the blade, so that the UAV will inspect the trailing side on the way back to the root. First of all, the UAV knows which side of the blade relates the leading edge and trailing edge, which means that the UAV can manouver the body frame towards the trailing edge. A proposed manouvering of the body frame to position the UAV on the trailing side of the blade is managed by two steps. The first step involves manouvering the body frame towards the detected edge on the trailing side, as illustrated in figure 5.8. Moreover, the second step involves repositioning the camera frame towards the leading side so that the camera projection is focused towards the blade.

This movement is managed by utilizing equation (5.15), where distance ρ_f, according to

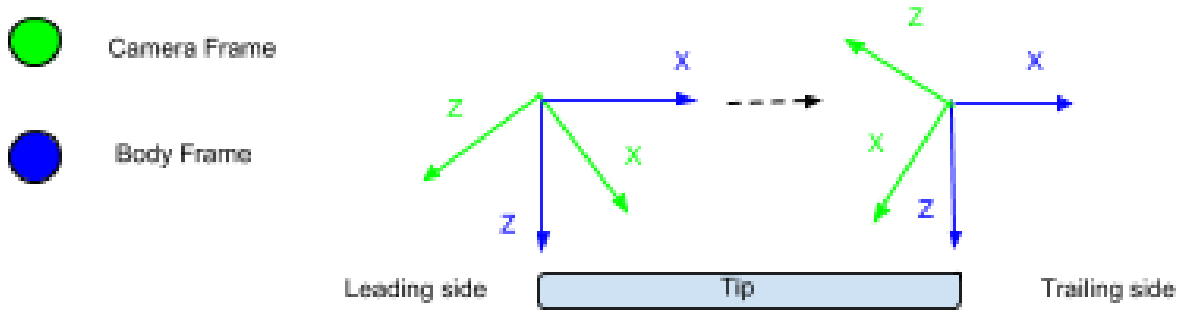


Figure 5.8: Repositioning of the UAV according to the trailing side.

the leading side, will be minimized by the minimal distance ρ_{min} to compute the measured difference ρ_d given in equation (5.24) and illustrated in figure 5.9.

$$\rho_d = \begin{cases} \rho_f - \rho_{min}, & \rho_{min} \leq \rho_f \\ 0, & \rho_{min} > \rho_f \end{cases} \tag{5.24}$$

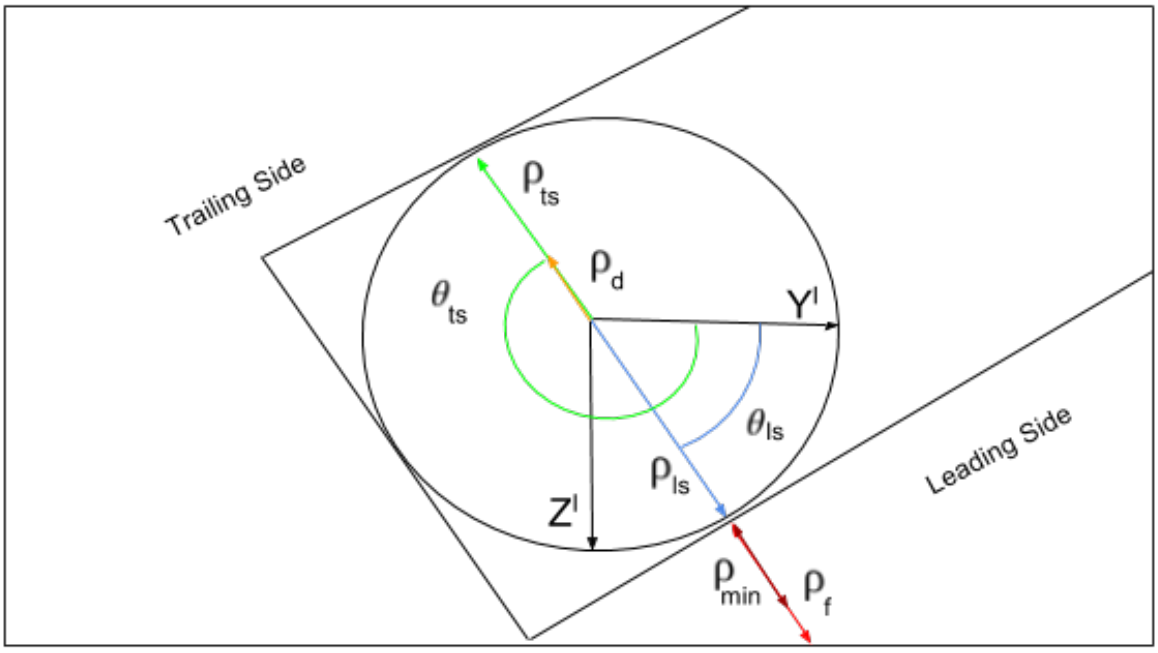


Figure 5.9: Repositiong of the UAV according to distance ρ_d.

The measured distance ρ_d concludes the distance towards the trailing side to move the UAV,

without loosing the view of the leading edge. Furthermore, the manouvering is given by the position in the camera frame, yielded by equation (5.25), and rotated according to equation (5.10) to relate the manouvering to the body frame.

$$\mathbf{P}^c = \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \end{bmatrix} = \begin{bmatrix} f\frac{T}{d} - D_d \\ \frac{T}{d_x}\rho_d \cos(\theta_{ts}) \\ \frac{T}{d_x}\rho_d \sin(\theta_{ts}) \end{bmatrix} \tag{5.25}$$

As mentioned, the second step involves repositioning the camera frame such that the image projection is focused towards the blade. This rotation of the camera frame is managed by the same principle as manouvering the body frame from the leading side to the trailing side. However, the camera frame will be rotated towards the leading side, minimizing the view of the trailing edge. Therefore, equation (5.24) is utilized by measuring the distance ρ_f according to the trailing edge, to find a fixation point towards the leading side as illustrated in figure 5.10.

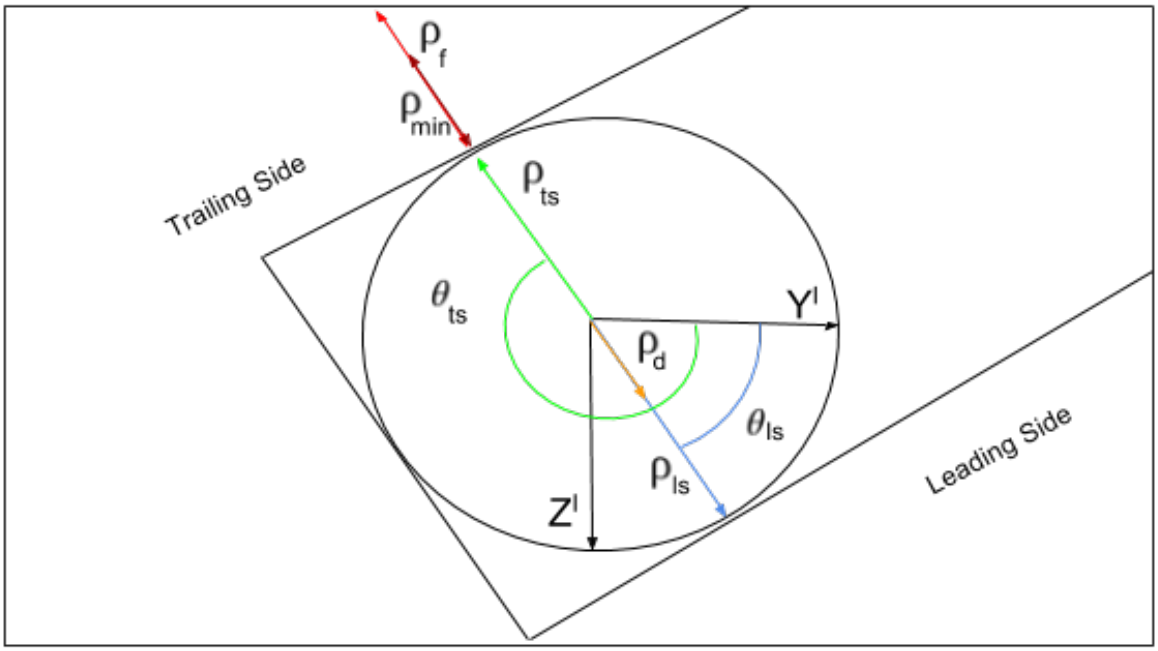


Figure 5.10: Rotation of the camera frame according to distance ρ_d.

The fixation point is given in pixel coordinates as stated by equation (5.26), while the camera frame is rotated according to section 5.5. The UAV will then be ready to continue the inspection along the trailing side.

$$\begin{bmatrix} p_y^i \\ p_z^i \end{bmatrix} = \rho_d \begin{bmatrix} \cos(\theta_{ls}) \\ \sin(\theta_{ls}) \end{bmatrix} \tag{5.26}$$

5.9 Detecting the Root

The final step of inspecting the wind blade is to detect the root region of the blade, where the UAV will consider the inspection of the blade as finalized. It is essentially important for the UAV to stop the manouvering towards the root region before it is too close to the wind turbine.

By studying the HAWT design of a wind blade, recall figure 3.2, we recognize an increasing chord length as seen from the tip to the root region. However, the chord length drops in the root region, which can be utilized by the machine vision system to detect the root region when the UAV is manouvering along the trailing side. Therefore, a simple approach to this issue is to consider that the chord length will increase when the UAV is moving towards the root region, and a sudden drop of the chord length will signal that the root region is reached. This is, however, a simplification of the issue due to that the measured chord length cannot be correctly estimated when the UAV is manouvering along the trailing side since the camera does not project the whole surface of the blade. Moreover, the camera projection of the trailing side will change when the UAV is moving, meaning that the measured chord length can only be a filtrated estimate. So, a solution will be to measure a continuous estimate of the chord length, as seen from the trailing side by the camera projection, during the manouvering towards the root region. Finally, the root region is detected when there is a continuous decrease of the chord length, which means that the UAV has finished inspecting the wind blade.

Additionally, this solution can be improved by introducing a GPS on the UAV, which keeps track of how far away the UAV is from the wind turbine hub. The GPS is considered to be too inaccurate for the UAV to locate the root region solely on a GPS location, however, it may assist the machine vision system on wether the UAV is close to the root region.

Another approach may also be considered, which utilizes a sonar to detect the distance to

neighboring close objects, such as the root region of the windmill. This solution may be more reliable, and should be easily implemented, however, it was not possible to install and test such a device during the time this thesis was conducted.

5.10 Collision Avoidance

The distance from the UAV to the blade is measured by the stereo vision system, and the main motivation is to restrict the UAV from colliding with the blade by measuring the closest and most critical distance to it. Secondly, the UAV is also suppossed to keep a fixed distance to the blade according to a given reference distance D_d, recall section 5.4. The relative position of the blade to the camera is found by 3D reconstruction of feature points on the blade, as discussed in section 2.5, so an average distance D_x to the blade may be computed using these 3D points. The drone will then be set to follow a steady blade distance given by the referance distance D_d, which may be regulated using a common PID regulater. Moreover, the reference distance D_d should be set according to the maximal chord length of the blade, while accounting for the distance error δZ which is proportional to an increasing distance to the blade, recall section 2.4.4.

Moreover, a solution for avoiding a collision with the turbine hub is to utilize the proposed solution with detecting the root, recall section 5.9. The UAV may register its current GPS position or detect the turbine hub using a sonar to keep a minimum distance to the turbine hub.

5.11 Simulations of the Manouvering Plan

The essential manouvering methods presented in this chapter is implemented in the $Heading$ module which mainly considers simple manouvering along an arbitrary blade and how to detect the blade tip, as discussed in section 5.6 and 5.7, respectively. Unfortunately, it was decided that the other manouvering solutions was too complex to implement during the limited time this thesis was conducted, and it would have required acquiring and implementing a Gimbal for rotating the camera frame.

5.11.1 Simulations of Blade Manouvering

Section 5.6 presents two methods for manouvering along a wind blade. The first method is defined by equation (5.14) and presents a simple solution for following a wind blade when the trailing edge and leading edge of the wind blade is in view by the camera.

First of all, the implementation considers that the virtual drone follows a path along the horizontal or vertical edges of the blade, relative to the image frame, thus simplifiying the problem of selecting which of the detected edges are the leading and trailing blade edges.

The simulation was conducted on a broken wind blade of a drone and some simulation samples are illustrated in figure 5.11. The estimated heading is represented in radian degrees according to the image frame and is drawn as an orange arrow, which points in the estimated heading direction. Moreover, each of the vertical and horizontal boundary lines are colored with a different color, and marked with an open circle to mark the perpendicular point from the image center. Furthermore, only the lines along the trailing and leading side of the blade is detected as possible edges, thus marked by the yellow arrows, and used to estimate the respective heading according to equation (5.14). The simulation samples shows that the method estimates accurate heading directions, and the benefit is that it is very simple to conduct when considering that the blade edges are detected as Hough lines according to the image center.

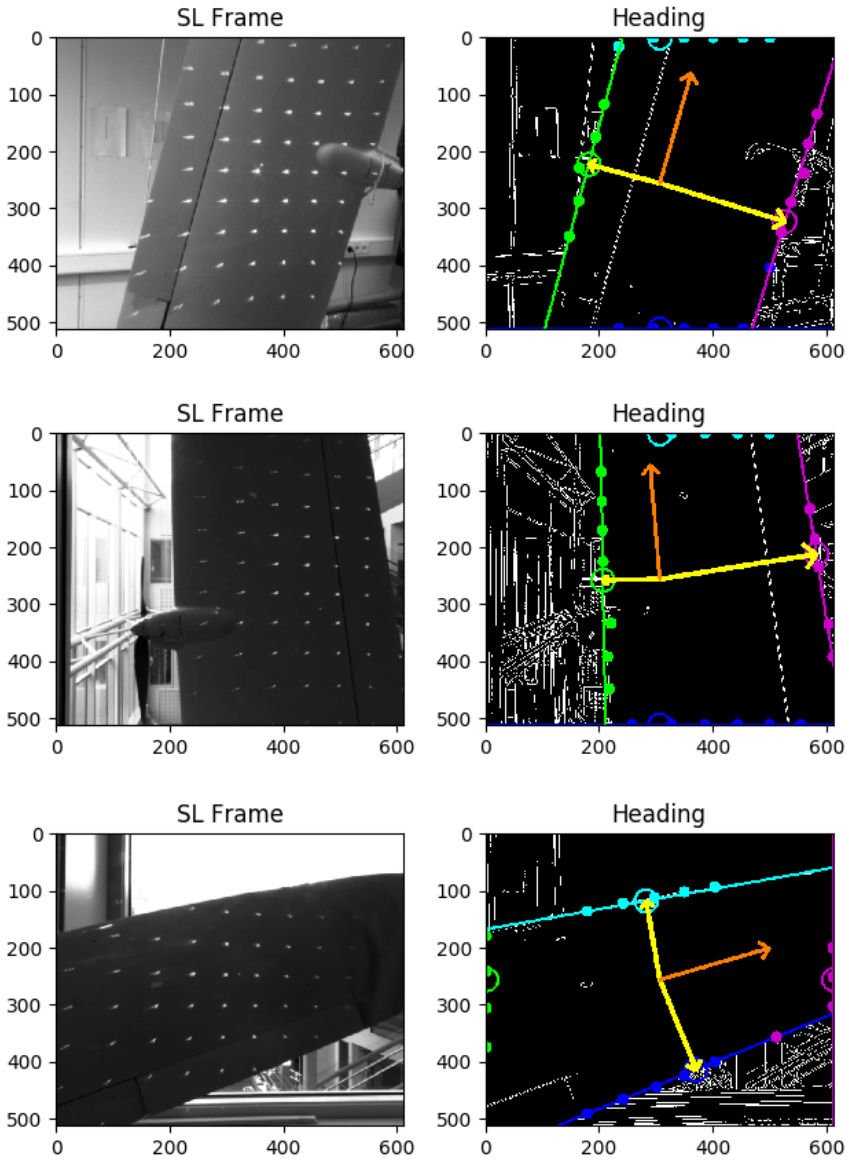
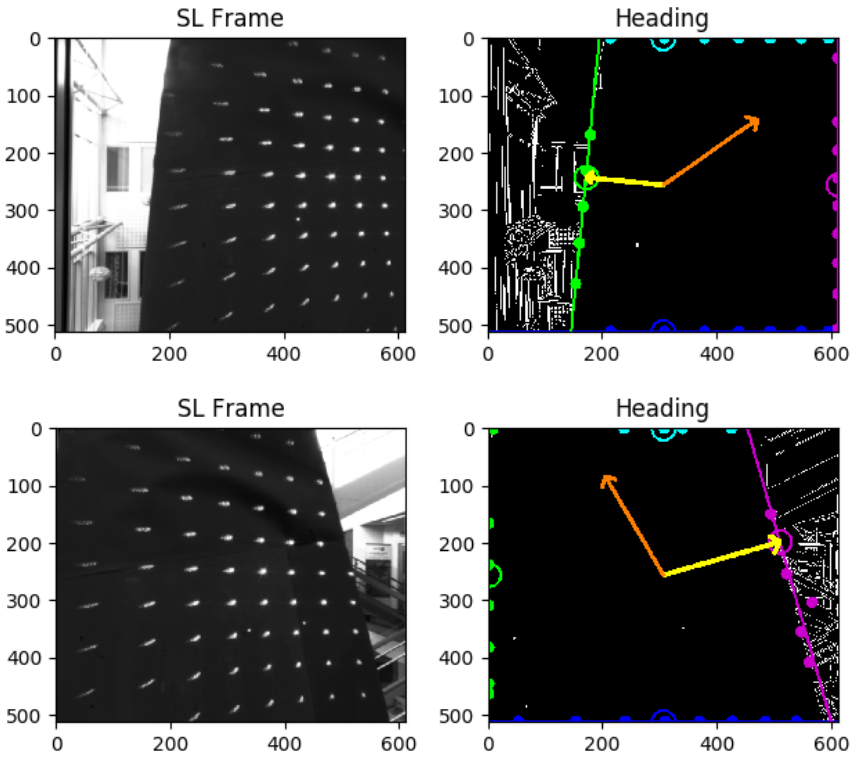


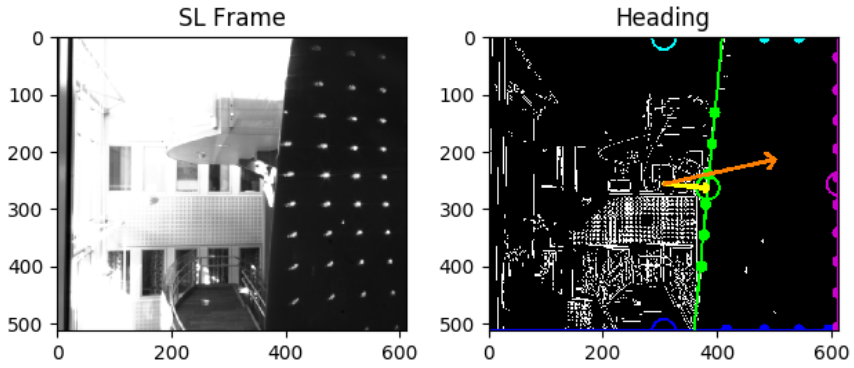
Figure 5.11: Simulation samples of manouvering along a wind blade when the leading and trailing edge is in view.

Boundary lines are colored with the respective color, and marked with an open circle to mark the perpendicular point from the image center. The heading direction is drawn as a orange arrow, and estimated according to the Hough line coordinates of the blade edges, which is marked by the yellow arrows.

Another consideration is that equation (5.14) is feasible only if the trailing and leading edge are on each side of the image center, thus in the opposite quadrant regions of the polar space. However, this problem was solved by selecting the closest edge of the trailing or leading edge and using the method for following a single edge to estimate a heading. The solution of following a single edge is defined by equation (5.15) to (5.18), which defines a simple regulation scheme designed to follow the respective edge while maximizing the view of the blade, whereas three simulations of the respective manouvering schemes are illustrated in figure 5.12. The simulations illustrated in figure 5.12a shows how the manouvering scheme of following a single edge estimates a heading which follows the blade edge while bringing more of the blade in view to the camera. Moreover, the simulation illustrated in 5.12b, shows how the heading is estimated when the respective blade edge is localized in the wrong quadrant area of the polar space, thus utilizing equation (5.19) to (5.22) to estimate a heading which also aims at bringing more of the blade in view.



(a) Simulations of manouvering along a blade edge which is too close to the image center, thus the heading is estimated to get more of the blade in view to the camera.



(b) Simulation of manouvering along a blade edge which is localized in the wrong quadrant of the polar space, thus informing that the blade is drifting out of view.

Figure 5.12: Simulations of manouvering along a wind blade when only a single edge is in view.

Boundary lines are colored with the respective color, and marked with an open circle to mark the perpendicular point from the image center. The heading direction is drawn as a orange arrow, and estimated according to the Hough line coordinate of the blade edge, which is marked by the yellow arrow.

Moreover, following simulations, illustrated in figure 5.13, shows examples of heading estimates which aims at following the respective edge while reserving that the edge will still be in view to the camera.

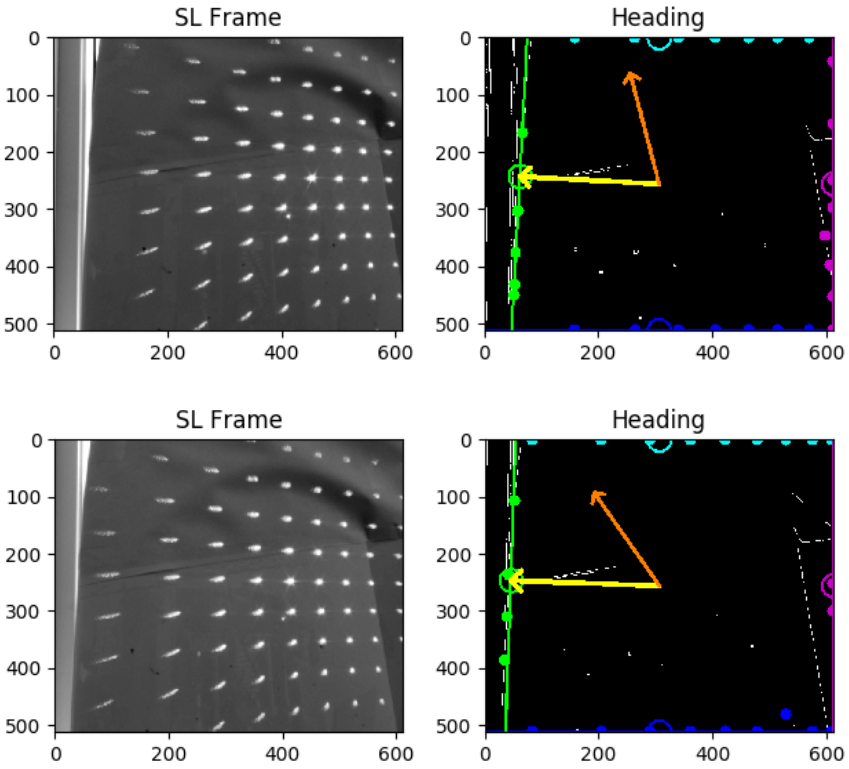


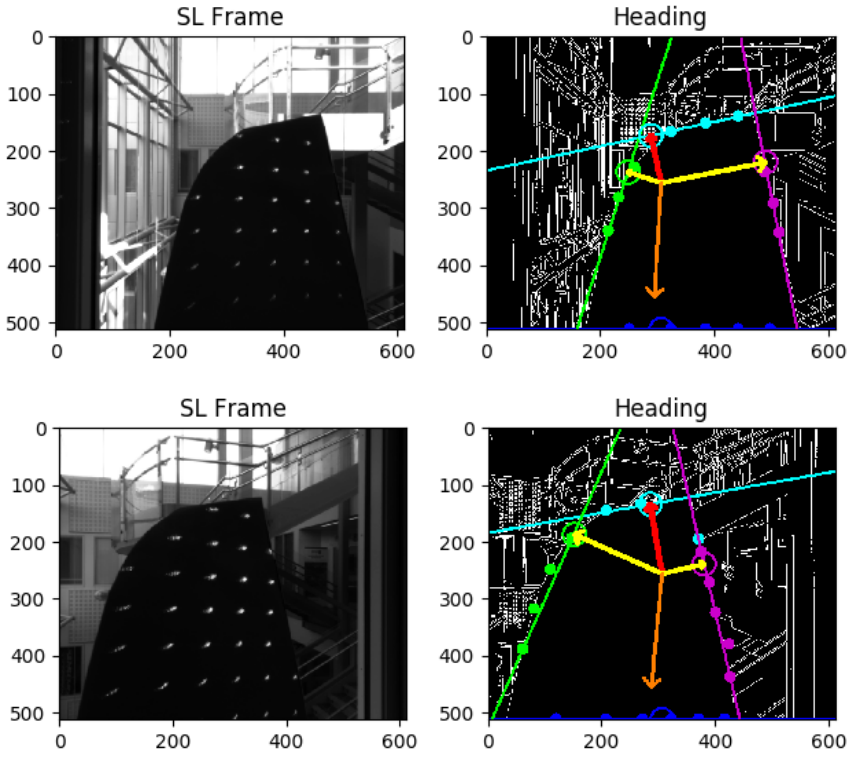
Figure 5.13: Simulations of manouvering along a blade edge which is too far away to the image center, thus the heading is estimated to get less of the blade in view to the camera.

Boundary lines are colored with the respective color, and marked with an open circle to mark the perpendicular point from the image center. The heading direction is drawn as a orange arrow, and estimated according to the Hough line coordinate of the blade edge, which is marked by the yellow arrow.

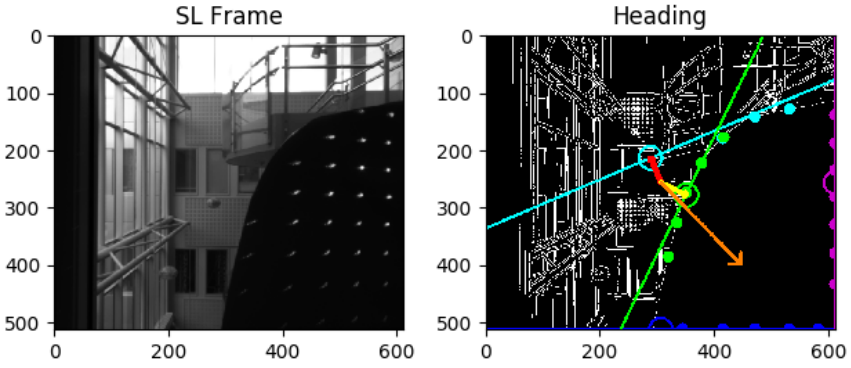
5.11.2 Simulations of Blade Tip Detection

Detecting the tip was found to more difficult then anticipated, and it was concluded that the corner detection method was not a feasible method for detecting the blade tip. It was found that the feature points could not provide the corner structure necessary in the area of an aribtrary corner tip, and the distance between the feature points was too big to create a feasible Harris corner detector for the respective scale.

The second approach was found to be a feasible and simple solution, whereas a possible perpendicular edge to the trailing and leading edges of the blade would be considered as the tip. The consequences of using this method is that the the scale of the tip must be big enough to be detected by multiple boundary points for beeing accurately detected as a line in the Hough parameter space, but it does not change the outcome of a detected tip since at least one boundary point will be detected at the tip, regardless of scale. As mentioned, the implementation considers that the virtual drone follows a path along the horizontal or vertical edges of the blade, which then makes it simple to decide that the third edge in the perpendicular direction of the chosen direction is the edge tip, as shown by the simulations in figure 5.14. Notice that the scale of the tip is big enough for multiple boundary points to be detected, thus the tip is accurately detected as a line in the Hough parameter space.



(a) Simulations of reversing the heading when the tip is detected.



(b) Simulation of detecting the tip while following a single edge.

Figure 5.14: Simulations of reversing the heading when the tip is detected.

Boundary lines are colored with the respective color, and marked with an open circle to mark the perpendicular point from the image center. The heading direction is drawn as a orange arrow, and estimated according to the Hough line coordinate of the blade edge, which is marked by the yellow arrow. The tip edge is marked by a red arrow, which flags that the blade tip is reached, thus the heading is flipped to manouver the virtual drone back to the root.

5.11.3 Discussion & Comments

The simulations conducted and presented intends to show the feasibility of using the proposed manouvering solution, and it is shown that the heading is accurately estimated to follow the direction of the blade, regardless of orientation, scale or wether one or more edges of the blade is detected. Moreover, the proposed regulation scheme of following a single edge solves the issue of keeping a maximized portion of the blade in view, which is beneficial for both inspection purposes and the manouvering scheme. Finally, simulations of blade tip detection were conducted and presented to show how the tip is easily detected by considering that the tip edge is detected in the perpendicular direction of heading. The simulations shows that the tip may be accurately detected when it is big enough to be detected by multiple boundary points. Additionally, it was mentioned that the consequences of detecting a spiked tip is that only one boundary point will be detected at the tip, thus the Hough line representation of the tip will be invalid. However, such an issue does not change the outcome of signalling that the tip is detected, so the drone will be manouvered back to the root of the blade, regardless of the tip scale. It should also be noted that the HAWT blade design, as discussed in chapter 3, illustrates a round tip which is notably similar to the blade tip used in the simulations.

Chapter 6

Kalman Filter

The Kalman filter is a recursive filtration algorithm which aims to remove white noise and colored noise of a linear or nonlinear system [11, p. 296], while estimating a least-square optimal estimate of the present state vector [6, p. 141]. The computer vision system may fail occasionally on estimating a navigation path due to unexpected loss of features which can lead to inaccurate detection of blade edges. The Kalman filter is therefore a suitable tool for fault recovery by predicting a navigation path based on prior path estimations. Moreover, the Kalman filter solves the issue of inconsistent path estimation, which may be a result of disturbances to the movements of the drone. The Kalman filter is therefore briefly reviewed by considering a linear discrete-time Kalman filter [6, p. 143], although it is not implemented in the program following this thesis.

6.1 Literature Review of the Kalman Filter

First of all, the Kalman filter considers a linear system as follows in equation (6.1). $\mathbf{x}_k$ is the system state vector with corresponding state transition matrix Φ_k, $\mathbf{u}_k$ is the control input vector with corresponding control input model $\mathbf{G}_k$ and $\mathbf{w}_k$ is the process noise.

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \mathbf{G}_k \mathbf{u}_k + \mathbf{w}_k \tag{6.1}$$

Moreover, the observation of the system is modelled by the linear model in equation (6.2), where $\mathbf{z}_k$ is the measurement vector. $\mathbf{H}_k$ is the matrix connecting the state vector to the mea-

surement vector, while $\mathbf{v}_k$ is the measurement noise.

$$\mathbf{z}_k = \mathbf{H}_k + \mathbf{v}_k \tag{6.2}$$

The measurement noise and process noise are considered as independent white noise, and modelled by the process covariance matrix $\mathbf{Q}_k$ and measurement covariance matrix $\mathbf{R}_k$, respectively.

The Kalman filter computes the following steps in sequence, as depicted by equation (6.3) to (6.7) [6, p. 147], during each iteration. An estimate is depicted with a hat, and with a possible "super minus" noting that this is the best estimate so far [6, p. 144]. Moreover, the first iteration starts with initial prior state estimate $\hat{\mathbf{x}}_0^-$ with corresponding error covariance $\hat{\mathbf{P}}_0^-$.

1. Compute Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{R}_k)^{-1} \tag{6.3}$$

2. Update estimate with measurement $\mathbf{z}_k$:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \tag{6.4}$$

3. Compute error covariance for updated estimate:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^- \tag{6.5}$$

4. Compute predictions of states and error covariance:

$$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k \tag{6.6}$$

$$\mathbf{P}_{k+1}^- = \Phi_k \mathbf{P}_k \Phi_k^\top + \mathbf{Q}_k \tag{6.7}$$

It should be noted that the Kalman filter estimate may have to be ignored when the UAV must follow a path which deviates strongly from prior estimates, such as following a path to locate a lost blade edge.

Chapter 7

Program Design & Hardware

This chapter reviews program design and selected hardware for implementing the methods discussed in the previous chapters.

7.1 Program Design

The program is designed as a master/slave system, whereas the master is connected to the left camera and the slave is connected to the right camera. In general, the master controls all aspects of the program, such as triggering the cameras to capture new frames and commanding slave to retrieve a captured frame from its respective camera. Moreover, master and slave will be given static IP addresses so that they may be connected using TCP on a local ethernet network connected through a switch. The Transmission Control Protocol (TCP) was chosen due to its reliability, which is critical for sustaining a reliable master/slave communication. Figure 7.1 illustrates necessary hardware and how they are connected, whereas the trigger wires for the laser and cameras are connected to the master device so that the cameras are triggered simultaneously, and with or without the laser turned on. The laser contains a diffractive lens that splits the laser beam into laser spots, creating a dot matrix designed structured light.

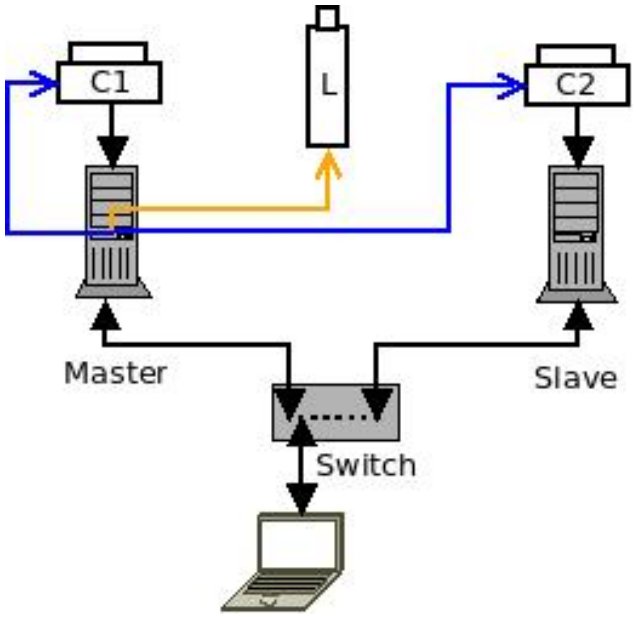


Figure 7.1: Hardware connections. Trigger wires to the cameras are blue colored, while the trigger wire to the laser is yellow.

The main objective of the program is to handle the master/slave system and compute an estimated distance to the object in view, as well as detecting the edges of the object to estimate a heading to follow. Figure 7.2 illustrates the program flow between the master and slave, whereas the master follows a sequential flow while the slave interactively acts on requests from master. As an example, the master starts the main program flow by requesting slave to capture two new frames, with and without structured light patterns. The slave acts on the request and initiates a separate flow to capture frames from its respective camera. However, the slave will be waiting for the camera to capture a frame, which is triggered by the master, forcing the cameras to capture the frames simultaneously. The master continues to trigger the cameras, and captures a frame from its respective camera, until it continues to turn on the laser to capture the next frame with structured light patterns. Next step involves processing the frames to compute keypoints, as mentioned in section 2.5, which is done simultaneously by both the slave and master. The slave saves the keypoints locally so that the master may request the keypoints when it is ready. However, the master will receive a wait response if the slave is not finished computing the keypoints, or even a failure response if it failed detecting any keypoints. The benefit of this program flow is that the master and slave shares some of the work load by computing keypoints simultaneously. Additionally, it is drastically more efficient to send a payload of keypoints instead of a

frame.

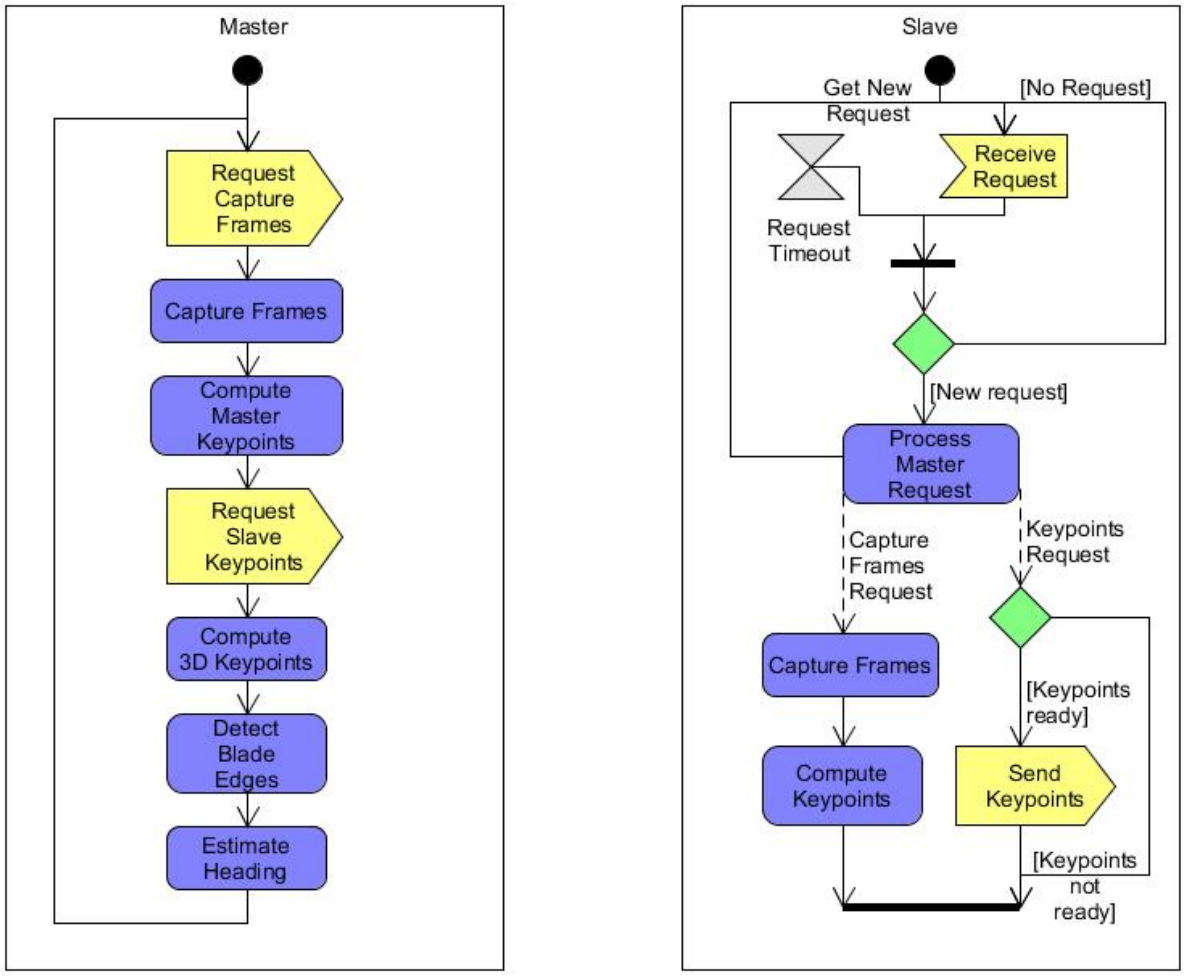


Figure 7.2: Activity diagram of the program flow.

Table 7.1: Approximate delay for the respective processing steps.

Process	≈Delay (seconds)
Convert to grayscale	0.05
Downscale	0.05
Undistort	0.05
Compute feature points	0.15
Detect blade edges	0.10
Reconstruct 3D points	0.03
Estimate heading	0.001
Estimated processing delay	0.431

7.2 Computational Delay

It is important to address the computational delay of conducting the main image processing steps necessary to detect the blade edges, and finally to estimate a heading. Only the image processing delays will be considered, thus the time it takes for the camera to capture a frame will not be considered. Table 7.1 lists the approximate delay for each of the respective processes necessary to estimate a heading according to the blade in view, and the processes were conducted on an Odroid-XU4 microcontroller, as specified in chapter 7.3.1. It should be noted that the raw frames are given as colored images with an original shape of 2048×2448, so it is therefore necessary to first convert the frames to grayscale before downscaling the frames to the standard shape of 512×612.

7.3 Hardware Overview

This section introduces selected hardware, whereas figure 7.3 shows the resulting implementation. Moreover, the implementation was built on a simple rig and powered by a 14.8V battery to make it portable, and the cameras and laser were fixed on a sliding ruler for easily beeing able to adjust the baseline between the cameras. The laser was positioned next to the cameras due to lack of space between the cameras, however, it does not change the outcome since the position of the laser is fixed relative to the cameras.

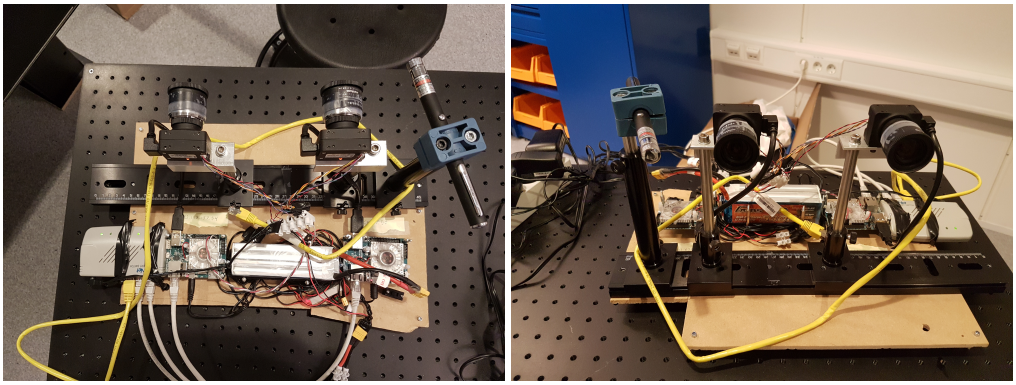


Figure 7.3: Connected hardware, including a battery and a sliding ruler for testing the stereo vision system on different baselines.

7.3.1 Microcontroller

The Odroid-XU4 was selected as a suitable microcontroller due to its USB 3.0 and GPU (Graphics Processing Unit) capabilites, which is essential since the cameras are connected through a USB 3.0 interface and since the GPU drastically accelerates image processing. Moreover, multiple GPIO pin options are accessible for triggering the laser and cameras, and the hardkernel was installed with the latest Ubuntu 16.04 arm version for Odroid-XU4. More technical details are summarized in figure 7.4.

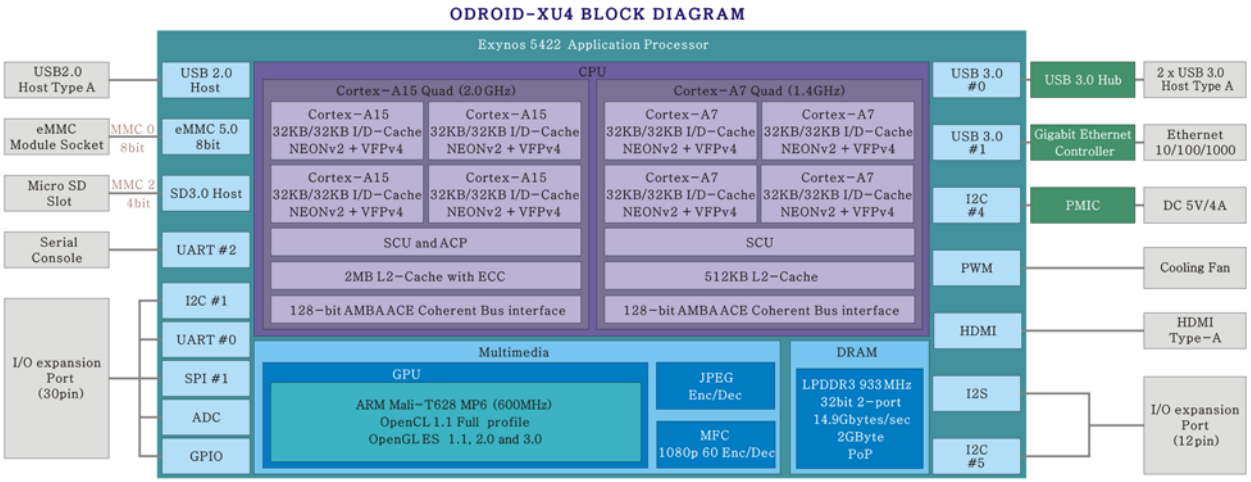


Figure 7.4: Technical details about Odroid-XU4.

7.3.2 Camera

It is especially important that the selected camera has triggering capabilities since frame capturing will be manually controlled. Moreover, it is preferred that the camera provides colored frames with high resolution so that small cracks may be detected and identified during inspection. The Chameleon3 camera, supplied by PtGrey, was the appropriate choice since it is designed for computer vision systems and fulfills all the mentioned criterias. Technical details are summarized in table 7.2.

Table 7.2: Technical details about the Chameleon3 camera.

	Chameleon3
Resolution	2448 × 2048
Pixel size	3.45μm
Optical format	2/3
Sensor size	8.8 × 6.6mm
Sensor type	Color, global shutter, CMOS
Lens mount	CS-mount
FPS	35
Interface	USB 3.0
Power	5-24V via GPIO or USB 3.0
Dimensions	44 × 35 × 19.5mm (case enclosed)
Weight	54.9g

7.3.3 Lens

A suitable lens must provide a field of view that covers most of a standard HAWT wind blade, specified by table 3.1, at an appropriate distance. A distance of 3 meters between the blade and the drone was concluded to be a suitable standard distance. Furthermore, it is important that the optical format of the lens is bigger, or preferrably equal, to the optical format of the camera, or else the frames will be cropped accordingly. From the theories presented in section 2.1 and a maximal blade chord length of 3.5 meters, it was concluded that a 8.5mm fixed focal length lens is appropriate. The field of view is estimated to be 3105 × 2329mm at a focus depth of 3 meters, with the given lens, which should cover most of the respective wind blade. It was also decided to use a manual iris since it is easy to configure when calibrating the camera. Technical specifications for the selected lens are summarized in table 7.3. Note that the lens is a C-mount so it is necessary to use a CS to C mount 5mm spacer adapter to fit it on the Chameleon3 camera.

Table 7.3: Technical details about the 8.5mm Fixed Focal Length Lens.

	8.5mm Fixed Focal Length Lens
Focal Length (f)	8.5mm
Aperture	f/1.3 - f/16
Min working distance	186mm
Working distance	200mm - inf
Optical format	2/3
Lens mount	C-mount
Iris	Manual

7.3.4 Laser

The main objective of the laser is to project structured light as a dot matrix of feature points. Moreover, it is important that the fan angle of the laser is big enough for the dot matrix to cover the field of view of the cameras. Given the specifications, it was decided to use the 3D PRO Laser Mini Green supplied by prophotonix. A fan angle of 60° was found to be optimal, however, it was not possible to combine this fan angle with a dot matrix diffraction option, so a laser with an 11×11 dot matrix diffraction option and a 28.2° fan angle had to be selected. Additionally, the laser is supplied with TTL modulation which means it is controlled by a digital high/low GPIO pin.

7.4 Software Overview

The program is written in python, which is a widely used high-level programming language. It was chosen due to its simplicity and its many open-source package extensions, such as the numpy package for scientific computation in which the program heavily relies on. Another essential tool is the open-source computer vision library OpenCV. The OpenCV library is written in optimized c/c++ with a strong focus on real-time applications, and has capabilites of enabling OpenCL for enabling the GPU to accelerate computation. Additionally, it comes with python bindings and may be installed on Linux, Mac OS, iOS, Windows and Android.

Chapter 8

Summary & Recommendations for Further Work

8.1 Summary and Conclusions

This report reviews computer vision techniques used to develop a method for segmenting and detecting wind blade edges, as well as conducting 3D reconstruction of the segmented area. Additionally, the method is proven to be accurate and computationally efficient for use on real-time applications. The method represents the blade edges as Hough lines which is utilized to propose a manouvering solution for following the blade edges from root to tip, and the solution is proven to be invariant to both rotation and scale while it follows a manouvering scheme to maximize the blade area in view. Moreover, a simple block matching method is developed to accurately match corresponding feature points for 3D reconstruction, and all of the methods are developed using structured light which is of a fixed scale and orientation due to the fixed relative position of the camera and laser.

The proposed manouvering solution includes solutions for collision avoidance, manouvering of the UAV back to the root region of the blade, and proposes to use the blade design combined with a GPS or sonar to detect the turbine hub. However, these solutions remains to be implemented and simulated since it was found that they were too complex to implement during the limited time this thesis was conducted. Moreover, the solution also reviews how to include a Gimbal to inspect a wind turbine blade from all angles, including how to transform coordinates

in the image frame to the body frame.

Finally, the algorithms for the respective methods discussed and developed in this report are included in the program mentioned in chapter 7, whereas review of relevant hardware is included.

8.2 Discussion

8.2.1 Structured Light & Accuracy

As mentioned, the methods are developed using structured light designed as a dot matrix, which enables accurate object segmentation and feature matching. Therefore, it is important to address that the structure of the dot matrix determines the accuracy of the respective methods developed in this report. Given that the laser projects a dot matrix with a fixed fan angle, then an increasing number of dots in the dot matrix will increase accuracy since neighboring dots will be closer, thus the accuracy of the segmentation method will improve. However, this comes at a cost of 3D reconstruction accuracy, as mentioned in the concluding section 2.5.6 about 3D reconstruction, where the baseline is constrained according to the fixed scaling between the dots in the dot matrix. In short, matching dots in the dot matrix will cross if the object is too close relative to the baseline length, which will make it nearly impossible to accurately match correct dots from the left and right dot matrix. It is therefore necessary to consider a tradeoff between accuracy of object segmentation and 3D reconstruction. Moreover, the optimal tradeoff may be found by considering a minimum baseline for an acceptable 3D reconstruction error estimate at a given depth range, and then maximizing the number of dots in the dot matrix relative to the fan angle of the laser projection and the estimated baseline. Additionally, the minimum distance that can be estimated is found by considering that the maximum disparity will be the distance between neighboring dots.

8.2.2 Guidance System

The proposed manouvering solution does not fully implement a guidance system, however, it works as a basis for manouvering along a wind blade. Further development towards a guidance

system may be to use the manouvering solution as a model combined with a Lyapunov designed controller and integrater backstepping, see Fossen [11, p. 457], or a PID controller to stabilize the navigation. Additionally, the guidance system will need to address real-time decision making on several issues that will occur during the navigation along the respective wind blade. First of all, the manouvering system offers a solution for accurately estimating a heading towards either the tip or root of the blade, but the guidance system needs know in which direction the tip is relative to the turbine hub, thus deciding which way to rotate the heading.

8.2.3 Wind Blade Inspection

The purpose of this report is to address the issue of autonomous inspection of wind turbines, with a focus on inspecting the wind blades. It is therefore essential that the camera captures images of highest possible resolution and quality, to ensure that all details of the blade are captured. It should also be noted that the manouvering solution maximizes the view of the blade if only a single edge of the blade is in view, which is crucial for beeing able to record the whole area of the blade. However, it will be impossible to get recordings of the whole blade if the drone is too close, thus the guidance system must be set to keep a distance to the blade so that at least half the blade will be in view during the whole inspection. This distance may be found by considering the maximum chord length of the respective blade.

Depth of Field

As described in section 2.1.1, the depth of field depends on the aperture and focal length, which means the lens must be adjusted to the respective distance to the blade to avoid blurred images. However, the distortion coefficients changes relative to the intrinsic values of the camera, thus it also depends on the aperture and focal length. This report uses a fixed focal length lens and the aperture was fixed close to the pin hole model to avoid blurring regardless of depth of field, so the calibration had to be commenced ones for the fixed intrinsic parameters. Implementing a lens with auto-iris, which enables automatic control of the aperture, requires a set of calibration samples in response to the aperture size to conduct undistortion according to the changing intrinsic parameters. However, it is recommended to avoid this issue by using a manual iris, as this report does, and to adjust the aperture close to the pin hole model. Additionally, it is

considered that it will be quite bright when inspecting a windmill during the day, thus a small aperture should not pose a problem since required exposure time will be low.

Localizing Cracks & Defects

Cracks and defects on the blade should be located according to where they are detected on the blade, meaning it would be preferable to map current position of the UAV with each image frame. Unfortunately, the approach proposed in this report infer that the system will navigate in real-time by estimating next manouver from each image frame until next goal is reached, which is either the tip or root region of the blade, without considering the current position of the UAV on the blade. In practice, this means that it will be difficult to map the location on the blade to each image, since the system only considers the current image frame. An alternative approach is to consider a model of the blade, which the UAV follows, hence knowing its current location on the blade at all times. However, this requires an exact model of the blade so that the guidance system is able to compute the direction to follow along the blade. Moreover, the blade does not have distinct feature points, except from a decreasing chord length proportionally closer to the tip region, which is necessary if the guidance system follows a model of the blade. Therefore, it is considerably easier to follow the blade edges until a simple and fixed goal is reached, which in this case is the tip- and root region of the blade. Still, localizing an image frame to an estimated position on the blade can be achieved by estimating the travelled distance using a GPS or simply deriving the distance based on known elapsed time and travel speed. The latter approach is feasible since the blade is considerably straight from root to tip.

Moreover, automatic identification of cracks and defects may drastically decrease inspection time, instead of manually going through the recordings. This is also beneficial for beeing able to automatically detect wether a crack has changed during a given period of time. Cracks and defects may be detected using rotational- and scale invariant feature detection methods, such as SIFT, or by conducting line detection with an edge linking method. Moreover, machine learning is essential for identifying different types of cracks and defects, whereas a typical solution would be the 'bag-of-words' model [27] which quantizises a set of feature points by clustering to match the features with words or keys. A machine learning approach was proposed by Zhang [53] to recognize and classify cracks and defects on wind blades. The report concludes on using a line

detection method and edge linking techniques to recognize cracks, while using a self-learning mechanism to identify and classify the cracks automatically.

8.2.4 Segmentation & Detection of Arbitrary Objects

As mentioned in section 4.3, the blade detection method developed in this report may also be used to detect a wide range of different objects, such as the windmill tower or the turbine hub. Moreover, the turbine hub will be detected by the segmentation method as presented in this report, although it will be represented as a rectangle of Hough transformed lines. As mentioned in section 4.3, the segmentation method may be adjusted to detect any type of shape which can be represented in the Hough parameter space, whereas the turbine hub can be detected as an ellipse using an elliptical Hough transformation, see Ballard [4, p. 113]. Moreover, the benefits of using this segmentation method is that the object is efficiently segmented using a small set of boundary points which minimizes the computational delay of the Hough transformation, while improving accuracy since the voting stage is limited to the peak points of the respective boundary points. Additionally, this method can be used to detect road signs, flag poles or building corners, whereas the manouvering solution also can be adopted to follow the respective structures.

8.3 Recommendations for Further Work

Short-term

This report concludes on a blade detecting and manouvering solution, without implementing a Gimbal, GPS or possibly a sonar. These devices are necessary to implement to continue simulations for rotating the camera frame according to the manouvering, and to test wether it is possible to detect the root region of the blade by recognizing a rapidly decreasing chord length or by using a GPS or sonar to detect the distance to the turbine hub, as discussed in section 5.9. Moreover, it is important to address the response of the proposed collision avoidance system, as discussed in section 5.10, which depends on accurate 3D reconstruction of feature points.

Medium-term

Recommendations for future work in the medium-term is to develop a guidance system that utilizes the manouvering solution developed in this report combined with a suitable controller to stabilize the navigation. Moreover, each recorded frame must be mapped according to the time and position of the respective recording to address the issue of localizing cracks and defects along the blade.

Long-term

Future work in the long term should address a solution of inspecting the whole windmill. This includes combining the work of Stokkeland et al. [50] with this solution, and developing a solution for inspecting the turbine hub and navigating towards the initial position in the root region of the wind blade. Additionally, more research and development on how the UAV will manouver towards the next blade needs to be addressed.

Appendix A

Acronyms

UAV Unmanned Aerial Vehicle

HT Hough Transform

PPHT Progessive Probabilistic Hough Transform

SSD Sum Of Squares Difference

HAWT Horizontal Axis Wind Turbines

FLANN Fast Library for Approximate Nearest Neighbors

SIFT Scale Invariant Feature Transform

DoG Difference-of-Gaussian

Appendix B

Glossaries

Occlusion Occlusion occurs when some elements cannot be shown due to some error. In the terms of stereopsis, occlusion occurs when an element is present in the first image, but not in the second.

Image features Image features are local, meaningful and detectable parts of the image [51, p. 68].

Focal length Focal length of a lens is the distance between the lens and image sensor.

Disparity Difference in retinal position between corresponding elements in two images.

Baseline distance Displacement between the center of projection of two cameras.

Stereopsis Depth perception, computed by a stereo vision system.

Stereo vision *The ability to infer information on the 3-D structure and distance of a scene from two or more images taken from different viewpoints* [51, p. 140]

Parallax Change in relative angular displacement of correlated image points across different camera frames.

Epipolar Geometry Geometry of stereo.

Extrinsic parameters The parameters that describe the relative position and orientation of two cameras.

Intrinsic parameters The transformation mapping from an image point to pixel coordinates.

Coplanar vectors Vectors parallel to the same plane, or lie on the same plane.

Lambertian reflectance Surface points appear equally bright from any view point.

Bibliography

[1] (2011). Automated Turbine Inspection. http://www.windsystemsmag.com/media/pdfs/Articles/2011_Oct/1011_AutoCopter.pdf. Last accessed: 2016-12-07.

[2] (2016). Operational and Maintenance Costs for Wind Turbines. http://www.windmeasurementinternational.com/wind-turbines/om-turbines.php. Last accessed: 2016-12-07.

[3] (2016). Wind Turbine Inspection (Aerialtronics). http://www.aerialtronics.com/applications/inspection/wind-turbine-inspection/. Last accessed: 2016-12-07.

[4] Ballard, D. H. (1981). Generalizing The Hough Transform To detect Arbitrary Shapes. *Pattern Recognition*, 13:111–122. http://comp-eng.binus.ac.id/files/2012/04/D.H.-Ballard-Generalizing-the-Hough-Transform-to-Detect-Arbitrary-Shapes1.pdf. Last accessed: 2016-10-11.

[5] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded Up Robust Features. *Proc. 9th European Conference on Computer Vision (ECCV'06) Springer Lecture Notes in Computer Science 3951*, pages 404–417.

[6] Brown, R. G. and Hwang, P. Y. (2012). *Introduction to random signals and applied kalman filtering: with matlab exercises.* John Wiley & Sons Ltd.

[7] Danish Wind Industry Association. Rotor aerodynamics. http://xn--drmstrre-64ad.dk/wp-content/wind/miller/windpower%20web/en/tour/wtrb/rotor.htm. Last accessed: 2017-01-18.

[8] Derpanis, K. G. and Hart, P. E. (2004). The Harris Corner Detector. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.482.1724&rep=rep1&type=pdf. Last accessed: 2016-10-11.

[9] Duda, R. O. and Hart, P. E. (1971). Use Of The Hough Transformation To Detect Lines And Curves In Pictures. *Comm. ACM.*, 15:11–15. http://www.dtic.mil/dtic/tr/fulltext/u2/a457992.pdf. Last accessed: 2016-10-11.

[10] Fisher, R., Perkins, S., Walker, A., and Wolfart, E. (2003). Gaussian Smoothing. http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth_htm. Last accessed: 2016-10-11.

[11] Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons Ltd.

[12] Frank (2006). Probabilistic Hough Transform. http://phdfb1.free.fr/robot/mscthesis/node14.html. Last accessed: 2016-10-11.

[13] Geng, J. (2011). Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*, 3:128–160.)doi:10.1364/AOP.3.0001281943-8206/11/020128-33. Last accessed: 2017-03-28.

[14] Harris, C. and Stephens, M. (1988). A Combined Corner And Edge Detector. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.434.4816&rep=rep1&type=pdf. Last accessed: 2016-10-11.

[15] Hartley, R. and Zisserman, A. (2004). Multiple View Geometry.

[16] Heggem, H. E. (2016). Autonomous Wind Blade Inspection - Project Thesis.

[17] Høglund, S. (2014). Autonomous Inspection of Wind Turbines and Buildings using an UAV. https://brage.bibsys.no/xmlui/handle/11250/261286. Last accessed: 2016-10-21.

[18] Kaspers, A. Blob Detection. *Biomedical Image Sciences, Image Sciences Institute, UMC Utrecht*.

[19] Kemao, Q. (2016a). Image Enhancement In The Frequency Domain, Course slides - computer vision CZ4003, NTU.

[20] Kemao, Q. (2016b). Image Enhancement In The Spatial Domain, Course slides - computer vision CZ4003, NTU.

[21] Kemao, Q. (2016c). Imaging and Camera Systems, Course slides - computer vision CZ4003, NTU.

[22] Kemao, Q. (2016d). Imaging Geometry, Course slides - computer vision CZ4003, NTU.

[23] Kiryati, N., Eldar, Y., and Bruckstein, A. M. (1991). A Probabilistic Hough Transform. *Pattern Recognition*, 24:303–316. http://www.cs.technion.ac.il/FREDDY/papers/39.pdf. Last accessed: 2016-10-11.

[24] Kong, W.-K. A. (2016a). 3D Stereo Vision, Course slides - computer vision CZ4003, NTU.

[25] Kong, W.-K. A. (2016b). Image Edge Processing, Course slides - computer vision CZ4003, NTU.

[26] Kong, W.-K. A. (2016c). Image Region Processing, Course slides - computer vision CZ4003, NTU.

[27] Kong, W.-K. A. (2016d). Object Recognition - Part II, Course slides - computer vision CZ4003, NTU.

[28] Lindeberg, T. (1993). Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention.

[29] Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales.

[30] Lindeberg, T. (2012). Scale Invariant Feature Transform. 7(5):10491. revision #153939.

[31] Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proc. of the International Conference on Computer Vision, Corfu (Sept. 1999)*, pages 1150–1157.

[32] Macdonald, I. (2016). Probabilistic Hough Transform. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/macdonald.pdf. Last accessed: 2016-10-11.

[33] Marghany, M., Tahar, M. R. B. M., and Hashim, M. (2011). 3D Stereo Reconstruction Using Sum Square Of Difference Matching Algorithm. *Academic Journals*, 6:6404–6423. http://www.academicjournals.org/article/article1380810673_Marghany%20et%20al.pdf. Last accessed: 2016-10-12.

[34] Matas, J., Galambos, C., and Kittler, J. (1998). Progressive Probabilistic Hough Transform. ftp://147.32.84.2/pub/cvl/articles/matas/matas-bmvc98.pdf. Last accessed: 2016-10-11.

[35] Muja, M. and Lowe, D. G. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. pages 331–340.

[36] OpenCV, D. T. (2015). Introduction to SIFT (Scale-Invariant Feature Transform). http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html. Last accessed: 2017-03-23.

[37] OpenCV, D. T. (2016a). Depth Map From Stereo Images. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html#py-depthmap. Last accessed: 2016-10-12.

[38] OpenCV, D. T. (2016b). Epipolar Geometry. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html. Last accessed: 2016-10-12.

[39] OpenCV, D. T. (2016c). Harris Corner Detector. http://docs.opencv.org/2.4/doc/tutorials/features2d/trackingmotion/harris_detector/harris_detector.html. Last accessed: 2016-10-11.

[40] OpenCV, D. T. (2016d). Image Pyramids. http://docs.opencv.org/2.4/doc/tutorials/imgproc/pyramids/pyramids.html. Last accessed: 2016-10-11.

[41] OpenCV, D. T. (2016e). openCV. http://opencv.org/. Last accessed: 2016-12-14.

[42] OpenCV, D. T. (2017). Simple Blob Detector. http://docs.opencv.org/trunk/d0/d7a/classcv_1_1SimpleBlobDetector.html#details. Last accessed: 2017-03-29.

[43] Project, S. E. (2006). Gaussian Pyramid Generation. http://sepwww.stanford.edu/data/media/public/docs/sep124/ssen1/paper_html/node3.html. Last accessed: 2016-10-11.

[44] Reinforced Plastics (2012). Wind turbine blade production – new products keep pace as scale increases. *Elsevier Ltd*, 12:22–29. http://www.materialstoday.com/download/79552/. Last accessed: 2017-05-30.

[45] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: an efficient alternative to SIFT or SURF . *Computer Vision (ICCV), 2011 IEEE International Conference on*.

[46] Schäfer, B. E., Picchi, D., Engelhardt, T., and Abel, D. (2016). Multicopter unmanned aerial vehicle for automated inspection of wind turbines. *Pattern Recognition*. http://ieeexplore.ieee.org/abstract/document/7536055/. Last accessed: 2016-12-07.

[47] Schubel, P. J. and Crossley, R. J. (2012). Wind Turbine Blade Design. *Energies*, 5:3425–3449. http://www.mdpi.com/1996-1073/5/9/3425/htm. Last accessed: 2017-01-19.

[48] Silpa-Anan, C. and Hartley, R. (2008). Optimised KD-trees for fast image descriptor matching.

[49] Stokkeland, M. (2014). A Computer Vision Approach for Autonomous Wind Turbine Inspection using a Multicopter. http://www.diva-portal.org/smash/get/diva2:744160/FULLTEXT01.pdf. Last accessed: 2016-10-21.

[50] Stokkeland, M., Klausen, K., and Johansen, T. A. (2014). Autonomous visual navigation of unmanned aerial vehicle for wind turbine inspection. http://folk.ntnu.no/torarnj/inspec.pdf. Last accessed: 2016-10-21.

[51] Trucco, E. and Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Pearson.

[52] Vestas. Vestas V90 - 2MW. https://www.ledsjovind.se/tolvmanstegen/Vestas%20V90-2MW.pdf. Last accessed: 2017-01-19.

[53] Zhang, H. (2016). Reducing Uncertainty in Wind Turbine Blade Health Inspection with Image Processing Techniques. http://www.imse.iastate.edu/files/2014/03/

Zhang-Huiyi-Reducing-Uncertainty-in-Wind-Turbine-Blade-Health-Inspection-with-Image-P
pdf. Last accessed: 2016-12-15.