Pupil Detection on Images

Semester Thesis

Anton Jakob Paris

23. February 2023

Dr.-Ing. Martin Weisenhorn

Image Processing and Computer Vision , OST Rapperswil

Abstract

In this thesis

Contents

Chapter 1

Introduction

Pupil detection on images is a fundamental task with diverse applications in various fields, for example computer vision, human-computer interaction and biometric systems. Accurate and efficient pupil detection is crucial in tasks such as gaze tracking, iris recognition, emotion recognition and medical diagnosis.

Over the years, numerous algorithms and techniques have been proposed for pupil detection, each with strengths and limitations. However, most of these algorithms cannot perform well under various conditions. Consequently, finding a robust and efficient combination of algorithms for pupil detection is still an intriguing research problem.

1.1 Problem Statement

The problem of pupil detection discussed in this thesis is the optimal combination of algorithms for pupil detection given the variability in imaging conditions, including variations in illumination, head poses, occlusions and quality of the images.

A single algorithm often fails to perform well under a wide span of conditions and lacks robustness and efficiency. Therefore this thesis aims to find a combination of algorithms that can perform well over a wide range of conditions without introducing machine learning, therefore, is independent of training data.

1.2 Motivation

The motivation behind this research is the lack of robustness and accuracy of current algorithms for pupil detection. Despite the extensive research on pupil detection, no universal solution exists that can consistently handle

the wide range of imaging conditions encountered in real-world scenarios. Therefore, exploring algorithmic combinations that can perform well under a wide range of conditions is a challenging and exciting research problem.

Improving the pupil detection algorithm's performance can increase the precision and reliability of medical diagnosis, human-computer interaction and biometric systems. The benefit of exploring different approaches for pupil detection is the possibility of combining the strengths of different algorithms to achieve a more robust and efficient solution.

This thesis presents a comprehensive analysis of various algorithms for pupil detection and proposes a combination of algorithms. Through extensive experimentation and evaluation, the thesis aims to provide valuable insights and guidelines for researchers and practitioners working on pupil detection. Overall, this research seeks to contribute to developing more effective pupil detection systems that can operate reliably under varying imaging conditions, paving the way for improved applications and advancing the understanding of pupil detection.

Chapter 2

Data Set

2.1 Labelled Pupils in the Wild (LPW)

2.1.1 Description

The data set "Labelled Pupils in the Wild" [1], or short LPW, was created by the Max Plank Institut and contains 66 high-quality, high-speed eye region videos for developing and evaluating pupil detection algorithms. All videos are labeled with the center of the pupil. Twenty-two participants with five different ethnicities and eye colors volunteered to record their eye movements.

The data set's goal was to record eye movement under natural conditions. With strong reflections, wearing glasses and wearing makeup, the data set becomes a difficult challenge for pupil detection algorithms and a good evaluation of algorithms is possible.

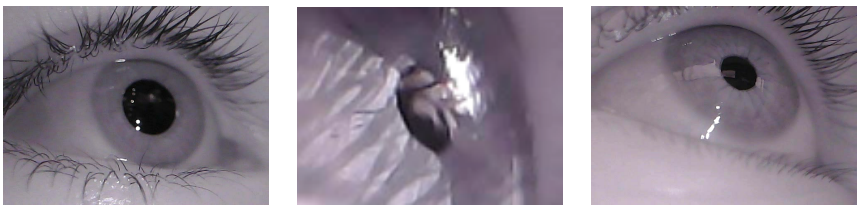


Figure 2.1: Three example frames from the LPW data set.

2.1.2 Procedure

The participants were asked to look at a moving red ball as it moved around. The recording location was randomly picked around several buildings. Each location was chosen once, creating a diverse range of real-life situations.

For the recording, a high-speed Pupil Pro head-mounted eye tracker took 95 frames per second with a resolution of 640x480 pixels. With this frame rate,

even fast eye movements last through several frames, making it more robust to detect the pupil.

Location	Number of Videos
Outside	34.3%
Inside	65.7%

Table 2.1: Location of recordings

Light source	Percentage of recordings
Natural light	84.7%
Artificial light	33.6%

Table 2.2: Light Source of recordings

2.1.3 Ground truth annotation

In many cases, the pupil area has a clear distinctive boundary and the center can be annotated easily. If the boundary was incomplete, one or two points inside the pupil were manually selected and used as seed points. From these points, the area with the same intensity value is extracted and used for annotation. However, this method was not possible in challenging scenarios because of the strong noise over the pupil. In this case, additional information was retrieved from the participants following a red ball with their eyes. This data was then used as calibration data to cross-reference the center of the pupil according to the position of the red ball and the eye movement.

The complete data set has ground truth annotations of the pupil's center for every frame. Given the ground truth annotation, it is possible to evaluate algorithms and compare them to each other.

2.1.4 Folder structure

The folder structure of the data set is straightforward. The data set is divided into 22 folders, one for each participant. Each participant folder contains three recordings in different locations with different light sources with their corresponding text file. The text file contains the ground truth annotation of the pupil center in x and y coordinates for each frame and more information about the participant.

The labels.ods file contains information about the location, light source, vision aid used, prescription, nationality, eye color and gender. The README.txt file contains the information about the data set and the folder structure.

2.2 Eye characteristics

2.2.1 Anatomy of the eye

The eye is surrounded by the **eye lid**. Its purpose is to shield the eye from debris and lubricate the eye by spreading tears over its surface with each blink. Four different parts of the eye do matter for pupil detection. As seen in figure 2.2.

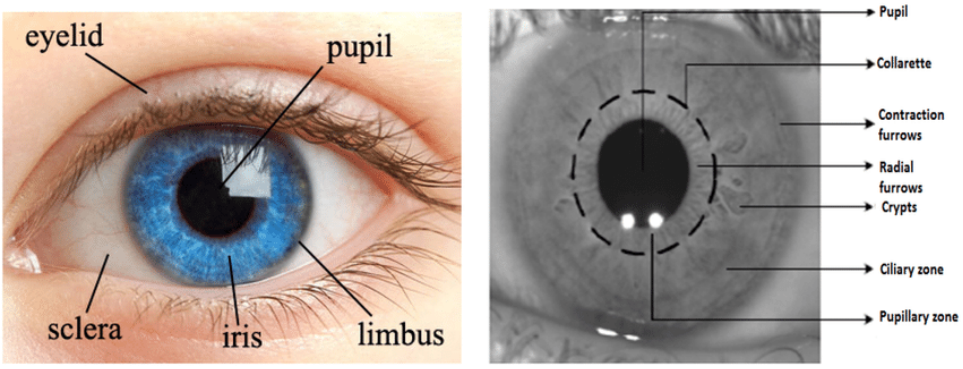


Figure 2.2: Overview of the different parts of the eye.

The white part of the eye is the **sclera** and is separated from the **iris** by the **limbus**. The table 2.3 shows the size difference between the iris and the pupil

Name	Radius	Additional Information
Iris	12 mm	in average
Pupil	2-9 mm	varies with light intensity

Table 2.3: Oversight iris and pupil radius

The iris regulates the radius size of the **pupil**, thereby regulating how much light comes through the pupil. The pupil is in the center of the iris and is the black part of the eye. The Darker the environment is, the larger the pupil radius. The iris is colored and can be blue, green/hazel, brown, gray, amber and black. There are also other colors that are connected with health issues or genetic defects and play no role in this thesis.

The critical fact for pupil detection is that the iris, pupil and sclera have different brightness intensity values. The pupil is the darkest area in the eye, the sclera is the brightest and the iris is something in between. Depending on the color of the iris, the brightness value can vary. Those characteristics are often used in pupil detection algorithms and will be discussed in the next

chapter.

There can also be reflections of the environment be seen in the eye. This reflection is called **corneal reflection** and can temper with pupil detection algorithms accuracy and therefore is a challenge for pupil detection algorithms.

2.2.2 Image characteristics

Histogram

When inspecting the histogram of a randomly chosen frame from the LPW data set, the histogram has peaks in different intensities. In figure 2.3, the peak between 0 and 50 corresponds to the pupil. This becomes clear, when comparing the histograms of the whole image with the region of interest (ROI), the lowest peak stays unchanged. The intensity of the iris is therefore between 90 and 150.

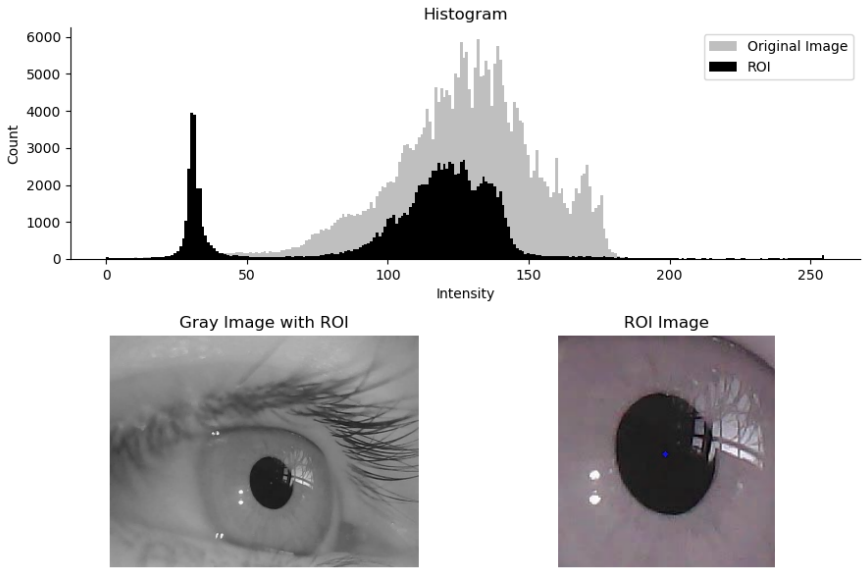


Figure 2.3: Comparing the histogram of the whole image and the region of interest.

The sclera is the brightest part of the eye, but in this frame, the skin intensity will be around the same magnitude as the sclera intensity.

If there is more reflection on the pupil, the histogram will also change shape. The peak created by the pupil will shrink and flatten out. The mean of the histogram will increase as dark points are substituted by brighter points. The presence of corneal reflection ultimately makes it hard to use a fixed threshold for segmentation. Using Histogram equalization increases the contrast of the image but stretches the peak into a wider intensity range.

Also interesting to inspect is a single pixel row of the image with their intensity values. Here it can be observed that the pupil creates a valley in the intensity values and the reflection on the pupil generates a peak right after or in the valley. The row intensity is shown in figure 2.4.

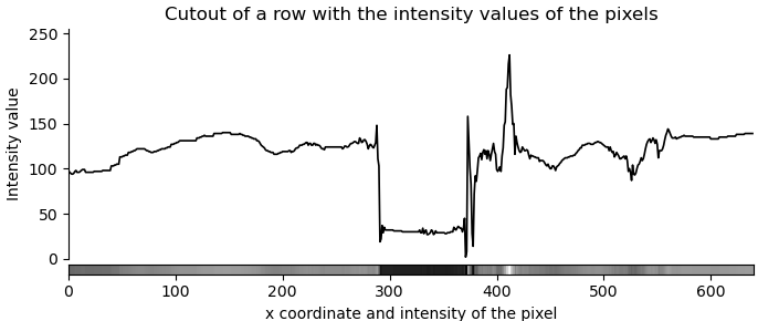


Figure 2.4: Plot of the intensity values of one pixel row.

The x coordinate of figure 2.4 represents the pixel at this coordination and the y coordinate represents the intensity value of the pixel.

The conclusion is, that the histogram is a solid tool to gain information about the pupil, but can vary a lot depending on the environment. Therefore adaptive algorithms have to be used.

Noise

In the data set are different kinds of noise. The most obvious one is the **corneal reflection**. The environment reflects on the eye and is seen as a bright spot on the pupil and destroys information of the pupil. Also **eyelashes** can be problematic as they are very dark and are often between the camera and the pupil.

Another Noise term can be that the eye is not constantly open. The **eyelid** can cover parts of the pupil or even the whole pupil. Glasses or lenses can add noise to the image as well as when the camera's focus is not on the pupil.

Chapter 3

Theory

3.1 General Definitions

This chapter discusses commonly used algorithms in image processing for edge detection, identifying areas of interest and applying them to pupil detection. At first, the algorithms will be discussed and analyzed on possible use cases, individual strengths and weaknesses. The algorithms use the same preprocessed images from the LPW data, so it is possible to showcase and compare the results. Even though different algorithms are use, the general approach for pupil detection can be summarized by finding the region of interest (ROI), then finding the pupil contour and finally approximating the pupil with an ellipse. Depending on the algorithm, the steps are sometimes extended or even combined. This chapter also discusses the possible combinations of algorithms.

3.1.1 Fundamental notation

Throughout this thesis, the following notation are used to describe the algorithms. The image with intensity level I is a function:

$f(x,y) : \mathbb{N}^2 \to \mathbb{N}$, where $f(x,y)$ is the intensity $I \in [0,255]$ at position (x,y)

In image processing, the coordinate system is defined differently than in mathematics. The origin is in the upper left corner and the x-axis points vertically down. The y-axis points horizontally to the right. this is shown in figure 3.1.

Also important to note is that the image is a discrete function: Therefore, each intensity value I comes with a quantization error. The quantization error is also present when using an algorithm on the image's intensity values or position (x,y). So it is not possible to have an exact result, it is always an approximation of the real result.

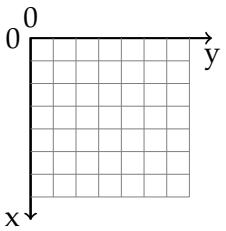


Figure 3.1: Coordinate system used in image processing.

3.1.2 Relationship between pixels

Another important theory in this thesis will be based on the relationship between pixels. In this subsection, the terms **neighborhood**, **adjacency**, **connectivity**, **region** and **boundaries** will be introduced and visualized so that they can be used in the following chapters.

Neighborhood

A pixel P at location (x, y) has two vertical neighbor pixels and two horizontal neighbor pixels in a 2D image. These neighbors are defined as $N_4(P)$ with coordinates:

$$N_4(P) = \{(x, y+1), (x, y-1), (x+1, y), (x-1, y)\} \tag{3.1}$$

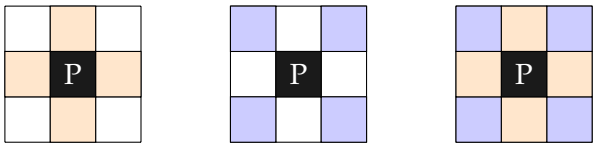
A pixel P at location (x, y) has four diagonal neighbor pixels in a 2D image. These neighbors are defined as $N_D(P)$ with coordinates:

$$N_D(P) = \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\} \tag{3.2}$$

Combining the neighbors from $N_4(P)$ and $N_D(P)$ results in the 8-neighborhood $N_8(P)$ of pixel P:

$$N_8(P) = N_4(P) \cup N_D(P) \tag{3.3}$$

Those three different types of neighbors are visualized in figure 3.2.



(a) $N_4(P)$ **(b)** $N_D(P)$ **(c)** $N_8(P)$

Figure 3.2: 3 different neighborhoods of pixel P at location (x, y).

Adjacency

There are three types of adjacent pixels in a 2D image. Let V be the set of intensity values used to define adjacency. Depending on the intensity range of the image, it is possible to define different subsets of V, containing the intensity values considered adjacent if they are present in the neighborhood. In a binary image, V is often defined as $V = \{1\}$, where 0 stands for background and 1 for foreground (This can also be considered a binary mask). In a grayscale image, V can be defined as any subset of the intensity range.

To keep it simple, the following explanations uses a binary image with $V = \{1\}$. Let's define P as a pixel at location (x, y) and Q as a pixel at location (x', y'). P and Q are considered adjacent if Q is in the neighborhood of P and $f(Q)$ is $\in V$. These are the three adjacency types:

- 4-adjacency, if $Q_i \in N_4(P) and f(Q_i) \in V$: The pixels directly above, below, left and right of the pixel.
- 8-adjacency, if $Q_i \in N_8(P) and f(Q_i) \in V$: The pixels directly above, below, left, right and the pixels diagonally adjacent to the pixel.

These two adjacency types are visualized in figure 3.3.

- m-adjacency $\begin{cases} \text{if } Q \in N_4(P) \text{ and } f(Q) \in V \\ Q \in N_D(P) \text{ and } N_D(P) \cap N_4(Q) \text{ has no intensities} \in V \end{cases}$

(a) 4-adjacency **(b)** $8 - adjacency$

Figure 3.3: 3 different neighborhoods of pixel P at location (x, y).

Connectivity

The connectivity of point P is a set of points that can be reached in n steps with a given adjacency type and intensity set V. If point P is connected with Q, then there exists a path (or curve) from P to Q, that consists of a sequence of distinct pixels with coordinates:

$$(x_0, y_0), (x_1, y_1), ..., (x_n, y_n),$$

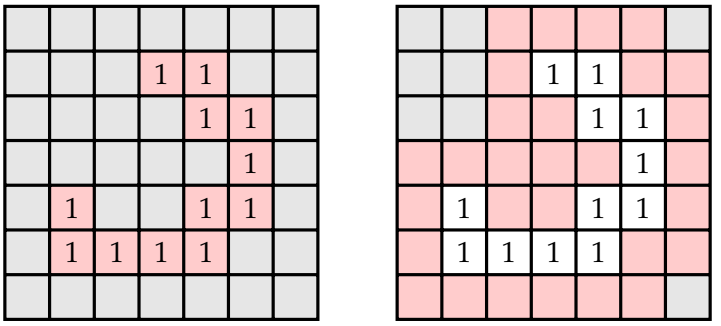
Where n is the length of the path. If the path is closed then $(x_0, y_0) = (x_n, y_n)$

Region

Let's define R as a subset of pixels in an image. R is a region if all points $\in R$ are a connected set, meaning that all points in R are connected with each other and therefore form a region. This does not mean that the path connecting all points is closed. Two regions can be adjacent to each other, if their union again forms a connected set.

Boundary

The outer boundary of a region R is the set of pixels not in R adjacent to pixels in R. In the definition of a boundary, the adjacency type is of importance. As a rule of thumb the 8-adjacency is used to define the boundary. One critical property of the outer boundary is, that it is a closed path. The inner boundary is the set of pixels in R but are 8-adjacent to at least one pixel $\notin R$. In figure 3.4 it can be seen that the outer boundary is a closed path whereas the inner boundary, in this example identical to the region, is not a closed path.



(a) region R **(b)** outer boundary (8-adjacent)

Figure 3.4: Region and outer border.

3.1.3 Preprocessing

The preprocessing must be defined first to compare the different approaches for pupil detection. The idea behind this step is to create a common ground for narrowing the deviation of the images down so that the algorithms can recreate the same result for different images.

Converting to grayscale

As presented in the previous chapter, the iris is the only colorful part of the eye. However, the color itself is of no interest for the detection. Therefore the frames are first converted to grayscale and this is done by converting the

images into grayscale. In this thesis the grayscale images have the intensity values $I \in [0, 255]$ and the image size is depending on the scaling as shown in table 3.1. The scaling is done to reduce the computational cost of the algorithm.

Scaling	Shape	numpy array type
100%	640x480	unit8
50%	320x240	unit8
25%	160x120	unit8
12.5%	80x60	unit8
6.25%	40x30	unit8

Table 3.1: Scaling of the frames used in this thesis.

It is important to note that these resolutions are congruent with the LPW paper [1]. This is important for the comparison of the results. When scaling an image, an interpolation is alway done to find the best approximation for the new intensity value I at location (x, y).

The interpolation used in this thesis is the bilinear interpolation[2]. This is a linear interpolation in the x and y-axis of the intensity values and solves equation 3.4.

$$v(x, y) = ax + by + cxy + d \tag{3.4}$$

Let $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$ and $Q_{22} = (x_2, y_2)$ be the four surrounding points. The intensity value v at (x, y) is calculated by equation 3.4.The point P at (x, y) is the point of interest. The visualization of the bilinear interpolation is shown in figure 3.5.

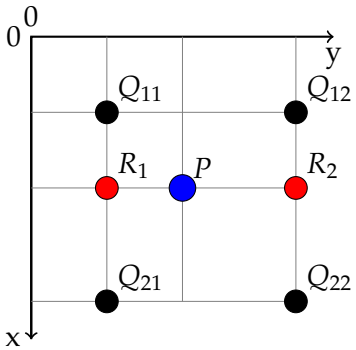


Figure 3.5: Bilinear interpolation.

First the intensity values $I_1 = f(R_1)$ and $I_2 = f(R_2)$ are calculated. $f(R_1)$ is

the linear interpolation between Q_{11} and Q_{21}

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) = R_1(x, y_1) \tag{3.5}$$

$f(R_2)$ is the linear interpolation between Q_{12} and Q_{22}.

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) = R_2(x, y_2) \tag{3.6}$$

Then the intensity value $I = f(P)$ at P is calculated by the linear interpolation between R_1 and R_2.

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) = v(x, y) \tag{3.7}$$

Therefore $f(P) = v(x, y)$ is the intensity value at P calculated with the bilinear interpolation of Q_{11}, Q_{12}, Q_{21} and Q_{22} at P.

Histogram equalization

Another vital aspect of preprocessing the frames is using Histogram equalization [3]. Histogram equalization has the effect of increasing the contrast of the image. This thesis uses Contras Limited Adaptive Histogram Equalization (CLAHE). CLAHE is a histogram equalization method that has the benefit that it is adaptive to the local contrast of the image. Local contrast equalization is very useful if the contrast of the image is not uniform in all regions of the image. Using Histogram Equalization adds additional noise to the image. The additional noise can be dealt with using a low pass filter, like a Gaussian filter for example, to minimize the noise.

Using a standard Histogram Equalization approach would lead to a loss of information in the region around the pupil and the iris because this is the region with the highest contrast already. Therefore, more noise would be added to the pupil and iris region. The CLAHE method splits the image into smaller blocks called ”tiles”. CLAHE then calculates the histogram equalization on each block individually without losing the same amount of information in the pupil region compared to the standard histogram equalization. The CLAHE method is applied to the frames after they are converted to grayscale. The result of this step is shown in figure 3.6.

3.2 Edge Detection

The main goal behind edge detection is to find edges in the image. An edge is defined as a region with high contrast to its surrounding pixels, in other words, a rapid change of intensity in a small area. Edge detection is helpful to filter the image for possible pupil contours. The edge detection

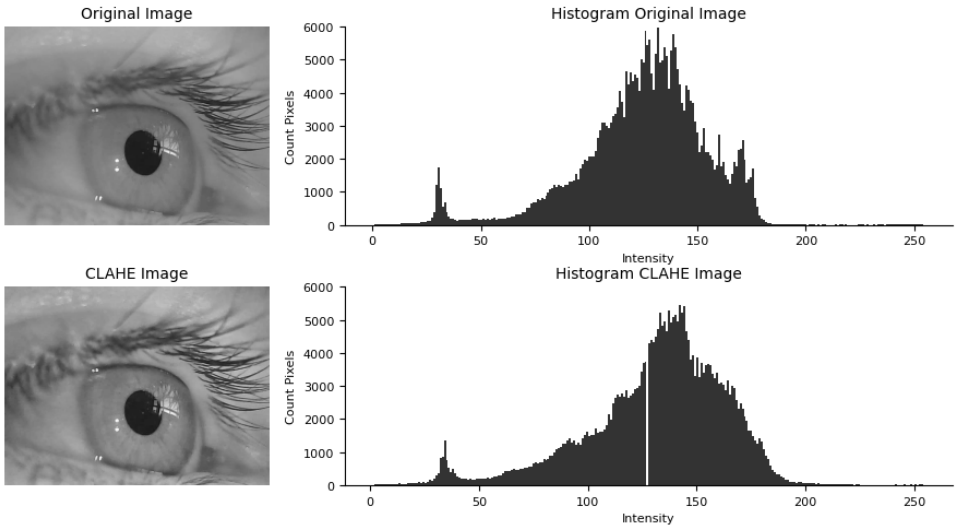


Figure 3.6: Example of CLAHE an its effect on the histogram.

analysis is based on a gradient calculation. There are different methods to calculate the gradient of an image. One of the most popular methods is the Sobel operator, which uses the first differential of the intensity change in the image. The Laplacian operator is another method that uses the second differential of the intensity change in the image. In this thesis the Sobel operator is used to calculate the gradient of the image. After calculating the gradient, Canny edge detection is used to refine the edges. Canny edge detection is a multi-step algorithm that uses hysteresis thresholding to filter the edges. The edge detection result is a one pixel thick binary edge map. These edges are now possible candidates for the pupil contour and need additional processing steps to find the pupil contour.

3.2.1 Sobel Operators

The Sobel Operators [4] is commonly used for edge detection. It is a gradient calculation that uses two 3x3 differential kernels to calculate the gradient of the image. The Sobel gradient is calculated in the x and y direction with their corresponding kernels k_x and k_y, and the kernels are defined as:

$$k_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{3.8}$$

$$k_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3.9}$$

The gradient vector is $\nabla f(x,y) = [G_x, G_y]^T$ and is calculated by convolving the image $f(x,y)$ with the Sobel kernels k_x and k_y.

$$G_x(x,y) = f(x,y) * k_x = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k_x(s,t) f(x+s, y+t) \tag{3.10}$$

$$G_y(x,y) = f(x,y) * k_y = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k_y(s,t) f(x+s, y+t) \tag{3.11}$$

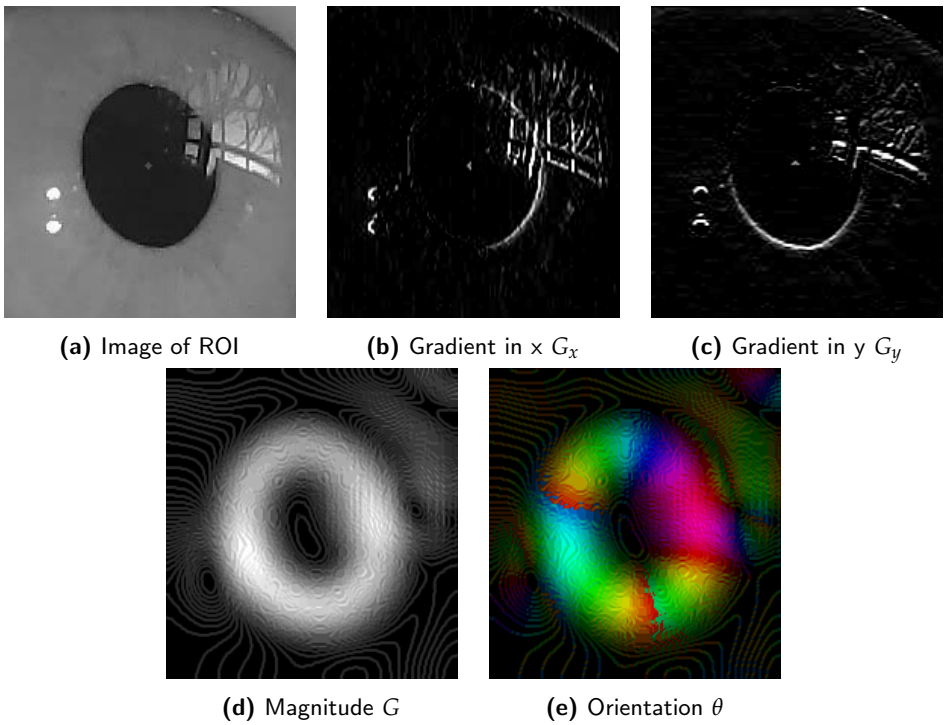
$G_x(x,y) = \partial f / \partial x$ and $G_y(x,y) = \partial f / \partial y$ are the gradients in x and y direction and the total gradient magnitude G of the gradient vector is calculated with the euclidean norm:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.12}$$

the orientation θ of the gradient is :

$$\theta = \arctan 2 \frac{G_y}{G_x} \tag{3.13}$$

The result of the gradient calculation can be seen here: Important to note is



(a) Image of ROI **(b)** Gradient in x G_x **(c)** Gradient in y G_y

(d) Magnitude G **(e)** Orientation θ

Figure 3.7: Plot of gradient characteristics of the ROI.

that figure 3.7d and figure 3.7e are the results of a strong gaussian blurred initial image.

3.2.2 Canny Edge Detection

The Canny Edge Detection [5] calculates a single edge points from the gradient vector image. Canny edge detection can summarized in four steps:

1. Noise reduction, smoothing the image with a Gaussian filter
2. Compute the gradient magnitude and direction
3. Non-maximum suppression to the gradient magnitude image
4. Use double thresholding and connectivity analysis to detect and link edges

Step 1: Noise reduction
The first step is to reduce noise from the input image. This is done by convolving the image with a low pass filter. For this task a Gaussian filter is used. The Gaussian filter is:

$$f_{filter}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.14}$$

The Gaussian filter is then convolved with the image to smooth the image.

$$f_{smoothed}(x,y) = f(x,y) * f_{filter}(x,y) \tag{3.15}$$

Step 2: Compute the gradient magnitude and direction
The gradient magnitude G is calculated with equation 3.12 and the gradient direction θ is calculated with equation 3.13.

Step 3: Non-maximum suppression
The non-maximum suppression is used to thin the edges out so the edges are only one pixel wide. This can be achieved with a loop iterating over all edge pixels and individually checking if the current pixel is the local maximum in the gradient vector's direction $\pm\theta$. If the pixel indeed is the local maximum, the pixel is kept, otherwise it is set to zero. Because as already described in 3.1.1 the coordinate system is defined different. The gradient direction θ is in reference to the x axis

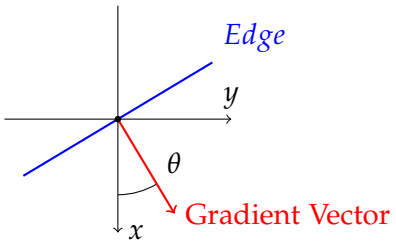


Figure 3.8: Definition of the gradient direction

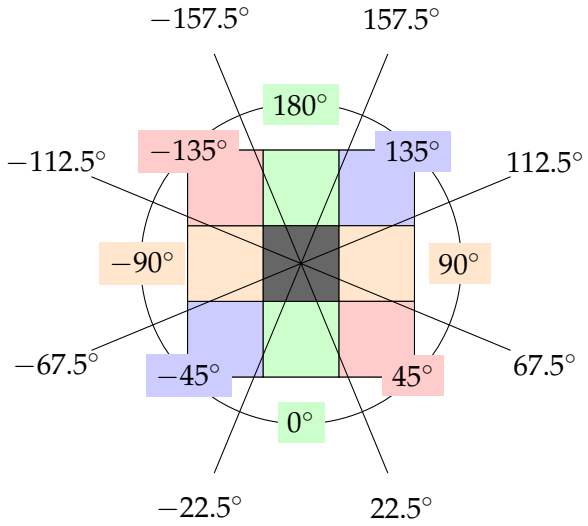


Figure 3.9: Quantization of the gradient direction

Because an image is quantized, this also means that θ needs to be quantized in four directions to evaluate their neighbors.

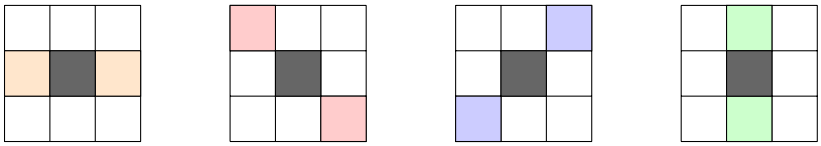
This leads to following quantizations:

$$\theta_q = \begin{cases} 90, & \text{if } 67.5° < \theta \leq 112.5° \vee -112.5° < \theta \leq -67.5° \\ -45°, & \text{if } 22.5° < \theta \leq 67.5° \vee -157.5° < \theta \leq -112.5° \\ +45°, & \text{if } 112.5° < \theta \leq 157.5° \vee -67.5° < \theta \leq -22.5° \\ 0°, & \text{if } -22.5° < \theta \leq 22.5° \vee -157.5° < \theta \leq 157.5° \end{cases} \tag{3.16}$$

It is important to note that two neighboring pixels are used to evaluate the gradient magnitude maximum. This is shown in figure 3.9 and 3.10. Suppose the maximum gradient magnitude is at the current pixel at (x, y), meaning it is a local maximum in the previously defined neighborhood with respect to the gradient direction, the value of the pixel is written into $g_n(x, y)$. Otherwise, it is set to zero $g_n(x, y) = 0$. $g_n(x, y)$ is the non-maximum suppressed edge image. Therefore $g_n(x, y)$ contains only the thinned edges.

Step 4: Double thresholding and connectivity analysis
After Step 3, $g_n(x, y)$ still contains edges that can be thicker than one pixel. $g_n(x, y)$ is then thresholded with a high a low threshold value (hysteresis thresholding), creating two images:

$$g_{low}(x, y) = \begin{cases} g_n(x, y), & \text{if } g_n(x, y) \geq T_{low} \\ 0, & \text{otherwise} \end{cases} \tag{3.17}$$



(a) Neighbors: 90° **(b)** Neighbors: 45° **(c)** Neighbors: -45° **(d)** Neighbors: 0°

Figure 3.10: The gradient magnitude is evaluated in the direction of the gradient.

$$g_{high}(x,y) = \begin{cases} g_n(x,y), & \text{if } g_n(x,y) \geq T_{high} \\ 0, & \text{otherwise} \end{cases} \tag{3.18}$$

Because two different thresholds are used, there is still an overlap between g_{low} and g_{high} edges. All non-zero pixels in g_{high} are considered strong edge pixels. To find all weak edge pixels, the strong edge pixels are substracted from g_{low}.

$$g_{weak}(x,y) = g_{low}(x,y) - g_{high}(x,y) \tag{3.19}$$

The remaining pixels are considered weak edge pixels.

The next step is to connect the weak edge pixels to the strong edge pixels. This is done by checking the 8-neighborhood of each strong edge pixel. If there is a weak edge pixel in the neighborhood, it is considered a strong edge pixel. This continues until no more weak edge pixels are found. The result is a binary image $g_{final}(x,y)$ containing connected edges. $g_{final}(x,y)$ still doesn't consists of one-pixel thick edges.

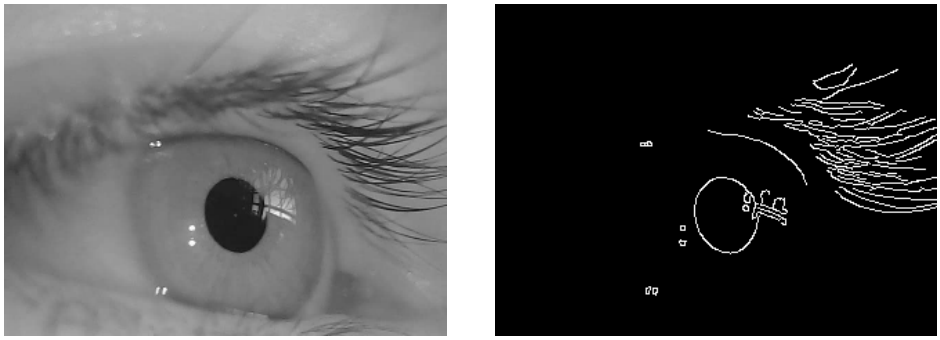
An edge-thinning algorithm solves this problem and returns the wanted image with only one-pixel thick edges. Let's define the edges as set A and a structuring element as B. The equation for thinning is:

$$A \otimes B = A - (A \circledast B) \tag{3.20}$$

Where $\otimes$ is the thinning operator, $\circledast$ is the dilation operator.

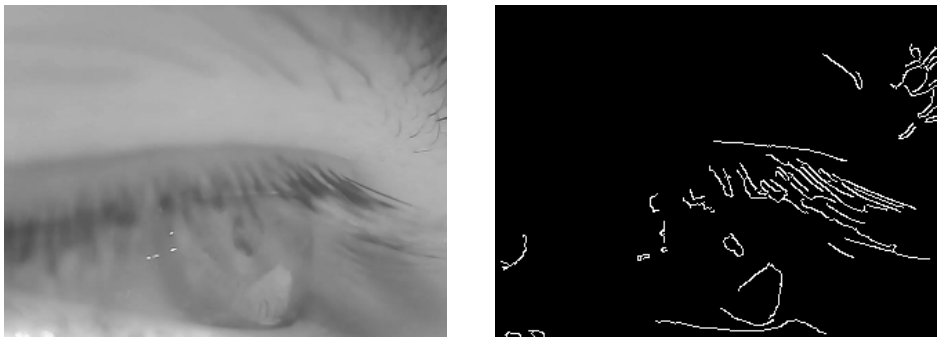
Results

In a frame where the pupil region is clearly distinguishable from the rest, the Canny edge detector can be used to find the pupil boundary. But as soon as more noise to the pupil is added, the hysteresis thresholding becomes more tricky and the detection accuracy decreases immensely. Also is it not possible to differentiate between the eye leashes, eyebrows and the pupil. Therefore by using only the Canny edge detector. The pupil edges can not be found reliable and the algorithm itself is not adaptable to a great variety of environments. The problem with Canny edge detection is visualized in figure 3.11 and 3.12.



(a) Original image, scaling 0.5 **(b)** Clear pupil region, Canny edge detection

Figure 3.11: Canny edge detection on a clear pupil region



(a) Original image, scaling 0.5 **(b)** Weak pupil region, Canny edge detection

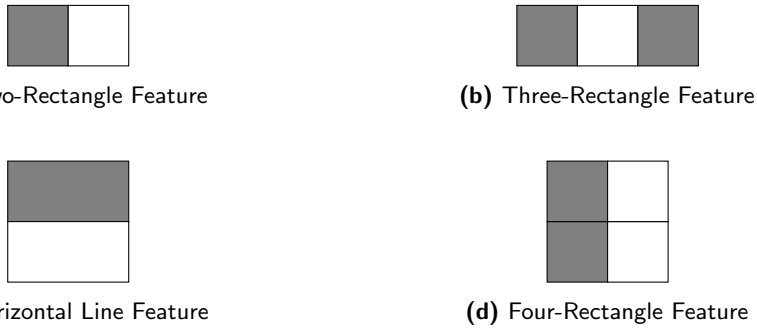
Figure 3.12: Canny Edge detection on a frame with weak pupil region

3.3 Haar like feature detection

3.3.1 Concept

Haar-like features [6] are based on the observation that objects have a particular local intensity variation in an image which can be useful for identifying objects. Haar-like features use the intensity variation and divide the image into rectangular regions and then calculate the difference between the sum of the pixel's intensity in the white and gray regions. An overview of the structure of features is given in 3.13.

The calculation is done for all possible rectangular positions in the image. The result is a response matrix that classifies each pixel in the image and gives insight into the object's position. In theory, the feature is convolved with the image to calculate the response matrix, but the integral image is used to make the detection faster. The integral image is calculated once and used to calculate the response matrix at each possible position in the image. The result is an easy and fast calculation of the response matrix. The structure of the feature bring different abilities to detect specific local intensity patterns



(a) Two-Rectangle Feature **(b)** Three-Rectangle Feature

(c) Horizontal Line Feature **(d)** Four-Rectangle Feature

Figure 3.13: Haar-like Features

that are an indicator for a particular object or image region property.

3.3.2 Features characteristics

To better understand the different features and their characteristics of the resulting response matrix four features are explained a little more in detail.

The key concept to remember is that the features generate a response based on the difference of the sum of the pixels in the white and gray regions and therefore are able to detect local intensity patterns. The response of each feature also depends on the size of the feature and stands in relation to the size of the object to be detected. The result of the Haar-like features provides information about the presence or absence of a certain pattern reflected in the structure of the feature used. Because an object can have more than one fitting feature, the features are often used in a cascade of classifiers. The first classifier is a very simple classifier that can detect general patterns. The following classifiers are more complex and are only used in a region where the previous classifier could detect a certain pattern. Using a cascade approach leads to speeding up the detection of the object, which saves computational resources and time. Often Machine learning is involved in classifying whether an object is present based on the response matrix.

3.3.3 Haar-Like Feature for pupil detection

One handy feature structure [7] for finding a point in the pupil is given by a feature constructed differently but is used the same way as the other features to calculate a response matrix. The feature is constructed as follows: The total size of the feature is $6rx6r$ whereas the center is of the size $2rx2r$. The feature mimics the shape of a pupil with a larger boundary around it that mimics a lighter iris. The variable radius makes detecting the strongest response of different pupil sizes possible. The feature is then used to calculate the response matrix repeatedly, using different radii and each radius generates

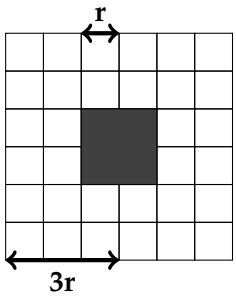


Figure 3.14: Haar-like feature for Pupil Detection

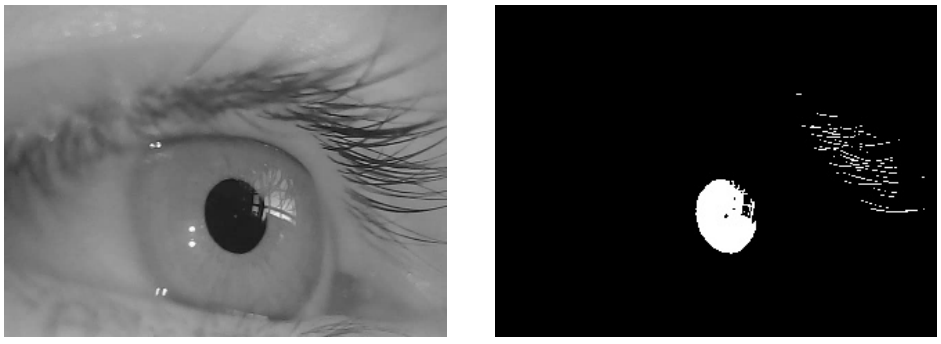
a different response matrix. All response matrixes are compared and the position (x, y) with the strongest response can be considered inside the pupil.

3.4 Thresholding and Ellipse Fitting

3.4.1 Thresholding

In thresholding pupil's characteristic is used. The pupil is the darkest part of the eye, as seen in 2.3. It is essential to find a specific thresholding value to only extract the pupil. Considering the best scenario, the pupil is unaffected by too much noise: no eyelashes or reflections. This method can be reliable in finding the pupil region, is less computation expensive and finds the pupil fast.

The most challenging part is to find a fitting threshold value. The threshold value can be approximated by inspecting the histogram of the frame. The lowest peak in the histogram is the given center value for thresholding. There is also the possibility for adaptive thresholding, for example, otsu thresholding. However, throughout the work with otsu. However, otsu was found less reliable than the histogram approach.

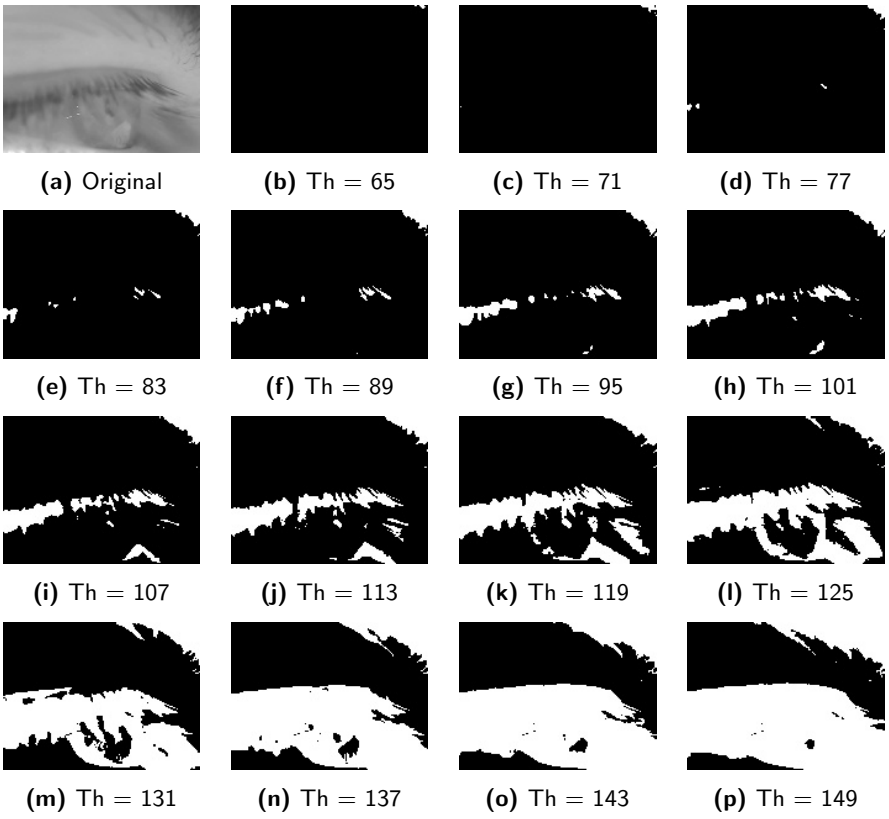


(a) Original frame **(b)** thresholded frame

Figure 3.15: Original and thresholded frame

In figure 3.15, it can be seen that the thresholding is extremely volatile to reflections, eyelashes, and other noise, concluding that thresholding only has a good performance in a strictly defined environment with almost no noise.

To showcase the volatility and dependency on the threshold value, figure 3.16 shows a series of the same frame with different threshold values (Th). It can be seen that the threshold value has to be chosen very carefully. If the value is chosen too low, the pupil region is not found. If the value is chosen too high, too much information is extracted and the pupil region can not be differentiated from the rest.



(a) Original **(b)** Th = 65 **(c)** Th = 71 **(d)** Th = 77

(e) Th = 83 **(f)** Th = 89 **(g)** Th = 95 **(h)** Th = 101

(i) Th = 107 **(j)** Th = 113 **(k)** Th = 119 **(l)** Th = 125

(m) Th = 131 **(n)** Th = 137 **(o)** Th = 143 **(p)** Th = 149

Figure 3.16: Thresholded images

3.4.2 Ellipse fitting

Ellipse fitting [8] based on minimizing the algebraic distance using least squares fitting is a common method to fit an ellipse to a set of points. Given a set of n 2D points $p_i = (x_i, y_i)$ where $i = 1, 2, ..., n$ and the equation of an ellipse in standard form is:

$$F(x, y) = Ax2 + Bxy + Cy2 + Dx + Ey + F = 0 \tag{3.21}$$

Where $a = [A, B, C, D, E, F]^T$ are the parameters to be determined. To exclude the trivial solution $a = 0$, the constraint $||a||^2 = 1$ is added. Another important constraint is that the ellipse is not a hyperbola or a parabola. Therefore adding the constraint $B^2 - 4AC < 0$ ensures ellipses as a result. Let's define the design Matrix D as:

$$D = \begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & x_ny_n & y_n^2 & x_n & y_n & 1 \end{pmatrix} \tag{3.22}$$

And let's define C as the constraint matrix so that the constraint can be expressed as $a^TCa = 1$

$$C = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.23}$$

Now the method of Lagrange multipliers [9] gives the conditions for solving the least square fit with the scatter matrix being $S = D^TD$:

$$Sa = \lambda Ca \tag{3.24}$$

$$a^TCa = 1 \tag{3.25}$$

This return at most six solutions for a_j and λ_j. The solution with the smallest λ_k and the corresponding eigenvector a_k best fits the ellipse based on the least square fit.

3.4.3 Contours

When working with thresholds, the contour is of interest. The contour is defined as the outer boundary of the threshold binary matrix and is the foundation for the ellipse fit method. Here it is not given to extract only one contour from the binary mask; therefore, the contours need to be filtered based on which contour represents the pupil the best. Mainly three criteria are used.

Circularity / Compactness

Circularity is described as the contour area A multiplied with 4π and divided by the integral over the outer boundary of the contour, also known as the

permitter of the contour.

$$\text{circularity} = \frac{4\pi A}{(\oint_{\partial S} dS)^2} \tag{3.26}$$

Similarity

For calculating similarity, the OpenCV library is used. This method compares a given shape with another. As the base for the comparison, a simple ellipse is taken and then compared with all possible contours.

Area

The Area of the contour should be maximal to find the largest pupil area and filter out smaller area contours. Each algorithm that returns a list of contours needs to be checked with these three conditions, and in the best-case scenario, only the pupil's contour survives. The result is then the contour on which an ellipse can be fitted.

3.5 Random Sample Consensus (RANSAC)

The Random Sample Consensus (RANSAC) [10] algorithm is an iterative method used to estimate the parameters for a known problem. For ellipse fitting, a contour or a set of possible points inside or on the contour C is the input. The algorithm then randomly selects a subset S_c of 5 randomly selected points in C. These 5 points are then used to fit an ellipse as described in the section Ellipse Fitting.

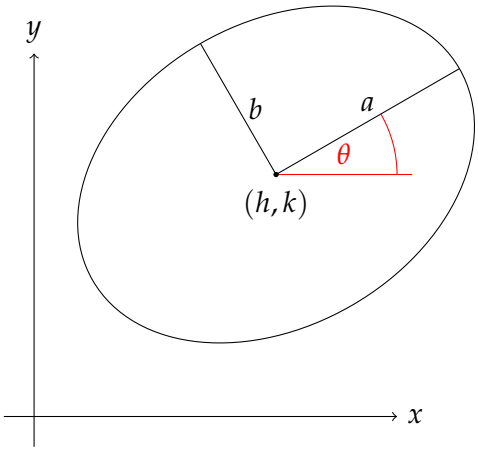


Figure 3.17: Ellipse with center (h,k), semi-axes a and b, and rotation θ

The equation 3.27 represents an ellipse with major axis a, minor axis b, and the ellipse's center at (h,k). The angle θ is the ellipse rotation in the coordinate system.

$$\frac{((x-h)\cos(\theta)+(y-k)\sin(\theta))^2}{a^2}+\frac{((x-h)\sin(\theta)-(y-k)\cos(\theta))^2}{b^2}=1 \tag{3.27}$$

When comparing the ellipse equation 3.27 with the general covariance ellipse equation 3.28

$$\mathbf{A}x^2+\mathbf{B}xy+\mathbf{C}y^2=1 \tag{3.28}$$

and compare coefficients, a relation between the ellipse parameters and the covariance matrix can be found

$$\mathbf{A}=a^2*\sin(\theta)^2+b^2*\cos(\theta)^2 \tag{3.29}$$

$$\mathbf{B}=2(a^2-b^2)*\sin(\theta)*\cos(\theta) \tag{3.30}$$

$$\mathbf{C}=a^2*\cos(\theta)^2+b^2*\sin(\theta)^2 \tag{3.31}$$

The covariance matrix is :

$$\mathbf{V}=\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{B} & \mathbf{C}\end{bmatrix} \tag{3.32}$$

The benefit of using the covariance matrix is that the eigenvalues and eigenvectors directly relate to the major and minor axis and the rotation of the ellipse. The eigenvalues represent the axis's major and minor, and the eigenvectors represent the rotation matrix R that rotates the ellipse in the coordinate system.

Important to note is that the eigenvectors need to be ordered to the size of the eigenvalues, and the eigenvector calculated with the largest eigenvalue comes first. This has the benefit that all parameters are known to transform all points into a new coordinate system where the calculated ellipse equation is represented in a unit circle.

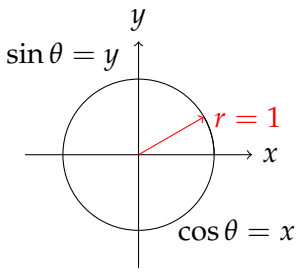


Figure 3.18: Unit circle with trigonometric identities

Using the unit circle allows for a simple distance calculation. Because $\lambda_{1,2}$ are known, then all points in $\in C$ of the old coordination system can be transformed onto the unit circle by subtracting $x - h$ and $y - k$, are then multiplied by the transposed eigenvectors to rotate the points into the new system and then divide by the eigenvalues. Let p be a point in C and p' the transformed point in the new coordinate system.

$$p' = R^T \begin{bmatrix} p_x - h \\ p_y - k \end{bmatrix} \odot \begin{bmatrix} \frac{1}{\lambda_1} \\ \frac{1}{\lambda_2} \end{bmatrix} \tag{3.33}$$

$$r = \sqrt{p_x'^2 + p_y'^2} \tag{3.34}$$

So the distance calculation of a point p to the ellipse curve simplifies to the distance of the transformed point p' to the unit circle.

$$d = \sqrt{p_x'^2 + p_y'^2} - 1 \tag{3.35}$$

The RANSAC algorithm iterates n times to find the best ellipse with the most inliers. The points are classified into three groups:

$$p = \begin{cases} \text{inlier} = \{p \in C | d < 0\} \\ \text{border} = \{p \in C | d = 0\} \\ \text{outlier} = \{p \in C | d > 0\} \end{cases} \tag{3.36}$$

The RANSAC algorithm calculates the ellipse parameters with five random points S_c for n iterations and evaluates the number of inliers for each iteration. The ellipse with the most inliers is considered the best fit.

The differentiation between on-border and inlier is essential to weigh the focus on enclosing ellipses. Because the distance calculation does not return an integer, it is possible that the distance is not exactly zero, but the point p is on the border. So it is crucial when comparing floats with each other to always use a small epsilon value that d can differ from 0. This epsilon is also defined as threshold. In Python, functions like *isclose* can be used to compare floats with each other. The number of iterations to get a 99% chance of finding the best ellipse with the most inliers can be calculated with the following formula:

$$n = \frac{\log(1 - P)}{\log(1 - w^s)} \tag{3.37}$$

Where P = is the probability of finding the best ellipse and w is the ratio of inliers to outliers (Probability that a point is an inlier). This formula estimates the number of iterations n for the RANSAC algorithm to be 99% sure to find the best ellipse with s sample taken each iteration.

In ellipse fitting $s = 5$, $p = 0.99$, and for w, an approximation must be made. Because the number of inliers is unknown, the ratio of inliers to outliers can not be calculated and needs to be estimated. In this case, we assume that the ratio of inliers to outliers is $w = 0.90$ which can be adapted later on.

$$n = \frac{\log(1 - 0.99)}{\log(1 - 0.90^5)} = 5.158 \tag{3.38}$$

So the number of iterations to find the best ellipse with the most inliers is 6. But it is important to keep in mind that this is only true if 90% of the points are located on the ellipse contour. Otherwise, the number of iterations needs to increase because the ratio of inliers to outliers changes drastically. Also an important fact is that the number of iterations needs to be chosen high enough because throughout the video, the number of border points changes drastically, and n can not be estimated perfectly. Therefore n is set to 500 but can still be optimized.

If only the border points are considered inliers and only $\frac{1}{5}$ of the border is visible, the number of iterations would jump to 4′713. The calculation of the inliers is discussed in detail in section: 4.2.4.

Visualizing RANSAC looks like this:

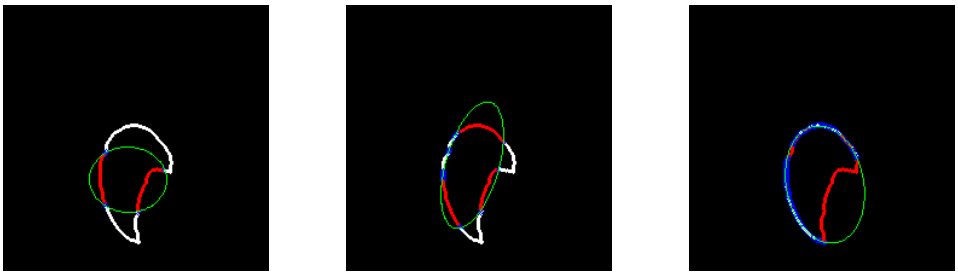


Figure 3.19: Example of RANSAC ellipse fit iterations

In figure 3.19, the green ellipse is the ellipse fitted to the five randomly chosen points. The red points are the inliers of the ellipse, and the blue points are the points that lay on the boundary within a given threshold. The white points are the outliers of the ellipse. The number of inliers is the sum of the red points and the number of boundary points is the sum of blue points. The goal is to find the ellipse with the most inliers, boundary points, and smallest area. These conditions lead to an enclosing ellipse with minimal area. All points together are the contour of the binary mask received from the ACWE algorithm explained in section 3.6.1.

3.6 Active Contouring

Active contour segmentation [11] is a method that is used to find the boundary curve of an object given that the object exists. There are many different approaches in active contouring, but three variants will be discussed in this thesis. The first one is the classic snake's approach (Kass et al., 1988) [12] with a simple energy function, and the second variant is the active contouring without edges (ACWE) based on level sets. The third variant is active contouring with ellipse parameters. In all cases, an initial contour is set and, through iterative changes, tries to find the boundary curve of an object. Each iteration the contour is changed by a small amount and then evaluated with an energy function. The energy function is a measurement of how well the contour fits the contour of the object, and in both cases, the energy function is minimized.

Let $C(q) : [0,1] \rightarrow \mathbb{R}^2$ be a parametrized planar curve and let $I : [0,a] \times [0,b] \rightarrow \mathbb{R}^+$ be a given image used to detect the object boundaries with active contouring. The classical snake's approach (Kass et al., 1988) [12] associates the curve C with an energy given in 3.40 But the most basic description for the energy minimization can be summarized in E_{int} and E_{ext}.

$$E = E_{int} + E_{ext} \tag{3.39}$$

$$E(C) = \underbrace{\alpha \int_0^1 |C'(q)|^2 \, dq + \beta \int_0^1 |C''(q)|^2 \, dq}_{\mathrm{E}_{int}} \underbrace{- \lambda \int_0^1 |\nabla I(C(q))| \, dq}_{\mathrm{E}_{\mathrm{ext}}} \tag{3.40}$$

The goal is to minimize the energy function, and by doing so, the contour C will find local minima and depending on the sign of λ be attracted to light or dark edges. The first two terms of the energy describe the intern energy of the contour and the last term represents the external energy. In other words, the intern energy describes the curve's smoothness and the external energy describes the relation of the curve to the edge at the curve's boundary. In the following, the these terms will be discussed in more detail.

To use these Equations, they have to be applied on a discrete grid. The curve C is discretized into N points $c_i(x_i, y_i)$ with $i \in [0, N-1]$. The energy function is then calculated for each point c_i and summed up to get the total energy of the curve. The energy function is minimized by using the gradient descent method. The gradient of the energy function is calculated and the points $c_i(x_i, y_i)$ are changed by a small amount in the direction of the gradient. The gradient descent method is repeated until the energy function converges to a local minima or the optimal solution.

The formula can be discretized by sampling the initial curve position evenly, then the formula can be expressed as:

$$E = \sum_{i=0}^{N-1} \alpha(c_{i+1} - c_i)^2 + \beta(c_{i+2} - 2c_{i+1} + c_i)^2 - \lambda|\nabla I(c_i)| \tag{3.41}$$

Here it is important to note that the points are circular, when i equals $N-1$, the next point is c_0. Therefore the indices can also be interpreted as $c_{(i+1 \bmod N)}$ and $c_{(i+2 \bmod N)}$

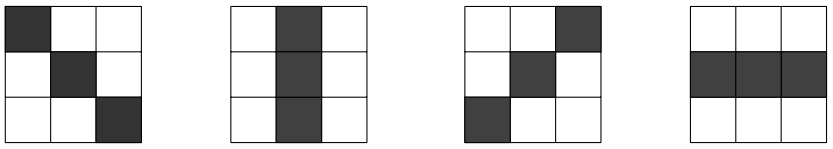
$$\text{substitute: } \begin{cases} \sum_{i=0}^{N-1}(c_{i+1} - c_i) = C'(q) \\ \sum_{i=0}^{N-1}(c_{i+2} - 2c_{i+1} + c_i) = C''(q) \end{cases}$$

Where $C'(q)$ calculates the length of the curve and $C''(q)$ calculates the curve's curvature.

Because the classic snake's implementation searches for the best boundary curve that locally minimizes the energy function, the contour must be initialized close to the best solution. Otherwise, the algorithm will converge to a local minima and not the global minima. The classic snakes approach searches for the minima with PDEs that are solved using the steepest descent method, which brings performance issues and is prone to terminate early if a local minimum is found.

3.6.1 Active Contouring without Edges (ACWE)

In ACWE [11], the same principle is used to find the best object boundary. But in this case, the energy function is minimized using level sets and morphological operations to solve the PDEs. **??** The morphological operations use four different structuring elements.



(a) Element 1 **(b)** Element 2 **(c)** Element 3 **(d)** Element 4

Figure 3.20: The four morphological structuring elements used in ACWE.

ACWE is based on Geodesic Active Contours (GAC) []. GAC can already solve critical problems from classical active contouring. Based on a threshold, the contour is iteratively evaluated and then compared to the threshold to decide if it flows and expands the contour to a new pixel. GAC is based on classic active contours and geodesic curves in a Riemannian space. The level sets have the benefit that they can easily be implemented in a discrete grid and open the door to solve the PDEs using morphological operations. The level set approach brings more stability and is less sensitive to the

initial contour than the classical active contouring method. It is also possible to detect the interior and exterior boundary of an object. In the classical approach, the internal force is based on the first and second derivatives of the curve and results in stability, computational complexity, and performance issues. The internal energy is solved in the GAC by using the evolution of an implicitly defined curve. The GAC algorithm is based on a level set function u and is defined as:

$$\frac{\partial u}{\partial t} = \underbrace{g(I) \cdot |\nabla u| \cdot \operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right)}_{\text{Smoothing force}} + \underbrace{g(I) \cdot \nu \cdot |\nabla u|}_{\text{Balloon force}} + \underbrace{\nabla g(I) \cdot \nabla u}_{\text{External force}}. \tag{3.42}$$

Where u is the level set function described as a binary mask. u is 1 inside the mask and 0 outside. I is the image and $g(I)$ is the edge indicator function. ν is a variable that decides in which direction and how strong the curve should move. For choosing $g(I)$, there are different variants but two of the most common are the following:

$$g(I) = \begin{cases} 1/(1 + |\nabla I| * \alpha) & (1) \\ -|\nabla I| & (2) \end{cases} \tag{3.43}$$

In (1) $g(I)$ is low when the gradient is high and vice versa. Meaning that $g(I)$ regulates the influence of the forces depending on the gradient. It is around 0 when the gradient is high. In other words, the curve stops moving when the pixel has reached an intensity transition, such as an edge. α is a variable that regulates how strong the influence of the external energy is. In (2) $g(I)$ is negative when the gradient is high. So both definitions of $g(I)$ will stop the curve from moving when it reaches an edge.

Smoothing Force

The smoothing force is the approximation of the curvature term of the classical snake's method. The difference is that in GAC, the smoothing force is calculated using the gradient of the level set function u instead of the curve.

$$g(I) \cdot |\nabla u| \cdot \operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right) \tag{3.44}$$

The smoothing force is calculated using a recursive call of SI and IS. Together they build operator $F(u)$, which approximates the curvature flow based on PDEs.

$$\mathrm{F}(u) = SI(u) \circ IS(u) \tag{3.45}$$

Where SI means supremum of infimum and IS means infimum of supremum. In other words, SI stands for eroding the current level set u four times

separately with each of the structuring elements once and taking for each pixel its highest result, and IS stands for dilating the current level set u four times separately with each of the structuring elements once and taking for each pixel its lowest result. One step of smoothing looks like the following:

$$1)u_{SI\circ IS} = \text{erode}(u) \circ \text{dilate}(u)$$
$$2)u_{IS\circ SI} = \text{dilate}(u) \circ \text{erode}(u)$$

by repeating step 1 and step 2 the smoothing force can be stronger or weaker. But it is crucial to always use step 1 and step 2 in the same order.

Balloon Force

The balloon force is essential in regions where the gradient is low and hinders the curve flow from getting stuck at a local minimum. Especially when looking at the external force, where the velocity of the curve is determined by the gradient of $g(I)$ and the gradient of u. If $g(I)$ goes to zero, the curve will have no external velocity. Therefore the balloon force is vital to make the algorithm more robust.

$$g(I) \cdot \nu \cdot |\nabla u| \tag{3.46}$$

In GAC, the ballon is also calculated differently than in the classical snake's method. The Balloon force $g(I) * |\nabla u| * v$ with parameter $v \in -1, 1$ can be substituted with dilation using the four different neighborhood structuring elements and then taking for every pixel the minimum value received from the four dilations. Dilation is a morphological operation and is computed very efficiently and also gets rid of the PDEs that are solved in the classical snake's method. By using dilate, the balloon force leads to the desired result of making the curve grow.

$$\text{dialate}(u) = u \bigoplus Element_i \quad \text{for} \quad i \in \{1, 2, 3, 4\} \tag{3.47}$$

External force

The external force is the same as in the classical snake's method, but in GAC, it is weighted by $g(I)$ to stop the curve from moving when it reaches an edge.

$$\nabla g(I) \cdot \nabla u \tag{3.48}$$

The change in the level set function u is only applied to the points where $g(I) > T$, where T is a defined threshold to control the velocity and leads to the characteristic that the curve moves faster in areas where the gradient is low. In other words, it leads to an algorithm that moves faster in a region with the same intensity and slower in regions with a high gradient.

The external force or the attracting force is calculated for each point p in the binary mask from the before modified level set function u and evaluated with the following equation:

$$p = \begin{cases} 1 & iff \nabla u * \nabla g(I) > 0 \\ 0 & iff \nabla u * \nabla g(I) < 0 \end{cases} \tag{3.49}$$

Summary

To conclude, the equation 3.49 describes the deformation of u, the binary level set function. and is the main equation for Geodesic Active Contours (GAC).

3.6.2 Convert GAC to ACWE

The GAC only iteratively inspects places where the curve could move, and there lies the main difference to the ACWE. ACWE, also known as the Chan-Vese algorithm, considers the complete image to evaluate the boundary or curvature flow of the level set. Let c_i be the mean intensity of a subset Ω of the image $I(x, y)$. Let c_1 be the mean intensity inside the curve or level set u defined as Ω_1 and c_2 the mean intensity outside the level set u defined as Ω_2.

$$c_i = \frac{\int_{\Omega_i} I(x)\, dx}{\int_{\Omega_i} dx} \tag{3.50}$$

Then the external force function $F(c_1, c_2, C)$ is defined as

$$\begin{aligned} F(c_1, c_2, C) =& \mu \cdot \text{length}(u) + \nu \cdot |\text{inside}(u)| \\ &+ \lambda_1 \cdot \int_{\Omega_1} (I(x) - c_1)^2\, dx \\ &+ \lambda_2 \cdot \int_{\Omega_2} (I(x) - c_2)^2\, dx \end{aligned} \tag{3.51}$$

where the first two terms $\mu \cdot \text{length}(u) + \nu \cdot |\text{inside}(u)|$ are the same as in the GAC. The third and fourth terms calculate the unscaled intensity variance in the different subsets Ω_1 and Ω_2.

The only difference between GAC and ACWE is that the external force is calculated differently. In GAC, the external force is evaluated with $\nabla g(I) \cdot \nabla u$ and then, based on a threshold, decides if a point is inside or outside the curve, as seen in equation 3.49.

In ACWE, the external force is calculated with the mean intensity of the subsets Ω_1 and Ω_2 and the intensity of the image $I(x)$ as seen in equation 3.51. Converting the equation onto a discrete grid it can be written as:

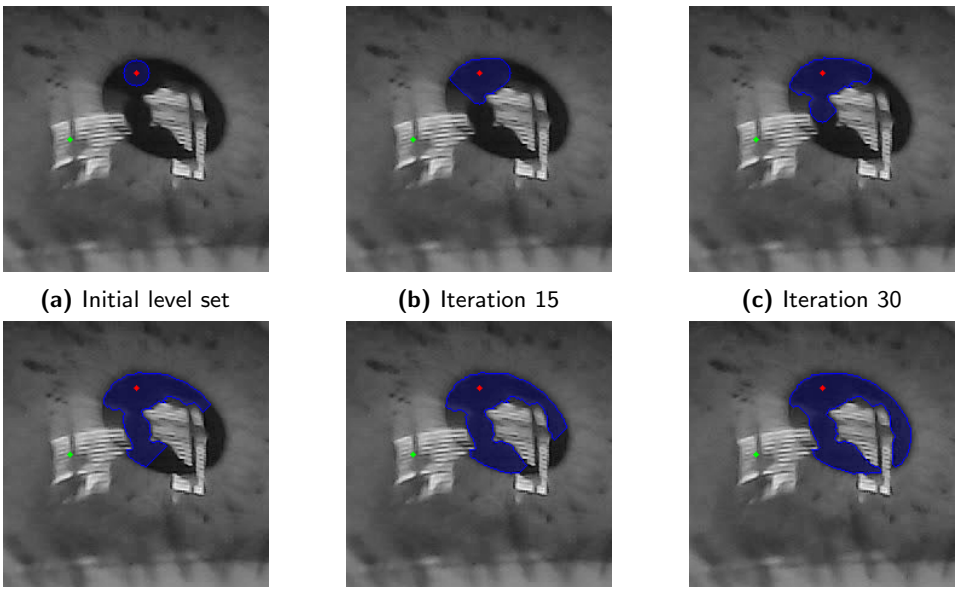
$$\xi = \underbrace{\lambda_1 \cdot |\nabla u_{mask}| \cdot (I(x) - c_1)^2}_{Inside} - \underbrace{\lambda_2 \cdot |\nabla u_{mask}| \cdot (I(x) - c_2)^2}_{outside} \tag{3.52}$$

ξ is then used to evaluate the new points in the level set function u with the following condiction:

$$u(x,y) = \begin{cases} 1 \text{ if } \xi < 0 \\ 0 \text{ if } \xi > 0 \\ \text{do nothing otherwise} \end{cases} \tag{3.53}$$

The gradient of the mask will only be nonzero at the contour of the mask. Now the mean is subtracted from $I(x)$, and each point on the contour will have two values, their inside and outside weighted intensities. By subtracting the mean intensity from the pixel intensities on the boundary of u, multiplying it with λ_i and subtracting the outside term from the inside term, a relationship between the inside and outside intensities is created and can be tuned with the sensitivity parameters λ_i .

This relationship is then used to calculate the new points in the level set function u and the process is repeated until the level set function converges. When the level set reaches the object boundary, the inside and outside intensities will be the same, the curve will not move anymore, and the equation will go to zero. Here is an example of the evolution the level set u when the ACWE algorithm is used with a given starting point inside the pupil.



(a) Initial level set **(b)** Iteration 15 **(c)** Iteration 30

(d) Iteration 60 **(e)** Iteration 90 **(f)** Iteration 115

Figure 3.21: ACWE iterations

3.6.3 Active contouring with ellipse parameters

A different approach is to let an ellipse grow and modify its parameters based on the gradient information. By sampling an amount of equally spaced points on the ellipse and evaluating an energy function based on the scalar product of the image intensity gradient vector and the normal vector of the ellipse at those points.

$$E = -\sum_{i=0}^{N-1} \nabla I(p_i) \cdot n(\vec{p_i}) \tag{3.54}$$

Where p_i is a point on the ellipse and $n(\vec{p_i})$ is the normal vector of the ellipse at that point. $\nabla I(p_i)$ is the gradient of the image at point p_i. The energy function is iteratively minimized by changing the ellipse parameters. Finding

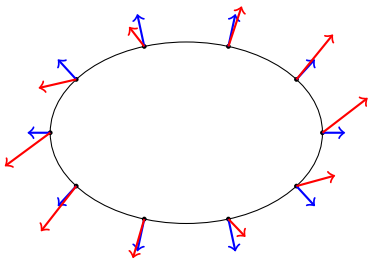


Figure 3.22: Ellipse with normal vector (blue) and gradient vectors (red)

the best fit turns into a minimization problem. The energy function still needs a term to ensure the curve grows and does not get stuck at a local minimum. Here it is not possible to use the morphological operators because the shape of the ellipse is not given as a level set. Instead the ellipse is based on only the five ellipse parameters: center(x, y), axis: (a, b), angle θ.

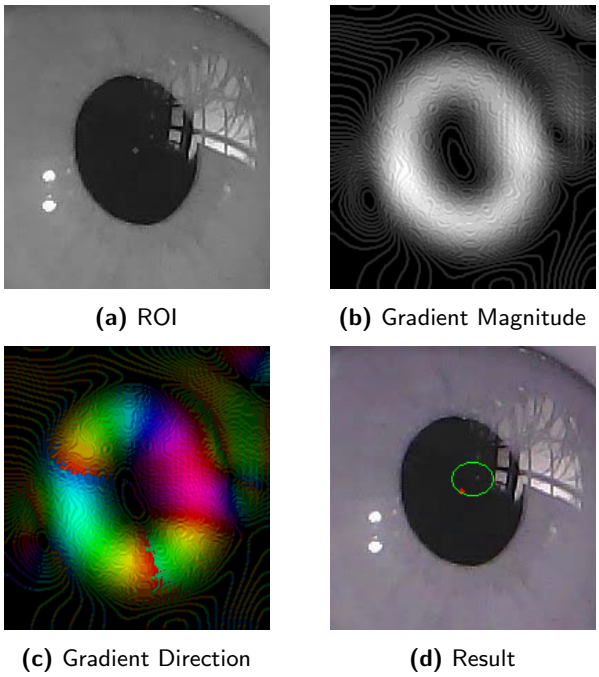
The energy function is the lowest when at each point, the gradient vector and normal vector have the same direction, and the magnitude of the gradient is maximal. The gradient vector consists of the magnitude of the gradient and the orientation. Whereas the normal vector is normalized and points outwards in the direction of the normal of the ellipse curve at that point.

The problem with this approach is overlapping with classical active contouring. The initial contour needs to be close to the optimal solution; otherwise, the algorithm stops at a local minimum, and convergence to the optimal solution is not guaranteed. Because it is a minimization with five parameters, it is a very complex problem, and the algorithm needs multiple iterations to find the best local solution.

For this approach, the image needs preprocessing and the algorithm is very sensitive to noise. Therefore, first applying a low pass filter onto the image is

a must. But because of blurring, the image gradients and directions are not as accurate as they could be and the scalar product struggles with reflections and other noise.

Here is an example with the convergence to a local minimum. The initial ellipse was initialized far from the optimal solution and the algorithm converges to a local minimum.



(a) ROI **(b)** Gradient Magnitude

(c) Gradient Direction **(d)** Result

Figure 3.23: Convergence to local minimum

The problem with expanding the energy function with more terms is that the algorithm gets more complex. For example, when the size of the ellipse is used as a reward for the energy function (the bigger, the better), it is not given anymore that the minimization stops at the object's boundary. The parameters must be chosen carefully to ensure the algorithm converges to the optimal solution.

Chapter 4

Algorithm Implementation

In this chapter, some of the algorithms of the theory chapter are combined, tested and evaluated. The goal is to find the most robust combination of algorithms to detect the pupil, even in a very demanding data set like the LPW. In general, the evaluation can be split up into two parts.

The first part is the localization of the pupil and the second part is finding the pupil contour estimation. The importance of the localization is that it is possible to create a region of interest (ROI) with the information on the pupil's location. Extracting the ROI has the benefit that the image size is already decreased in the second part, and the computational effort is reduced.

Important to note is that all algorithms make the basic assumption that the pupil is always visible. Blinking is not counted as a failure and is excluded from the evaluation if possible. For detecting blinking, other algorithms need to be used to preprocess every frame and detect if the eye is closed or not.

4.1 Localization

For the localization, mainly thresholding, edge detection and haar-like features were implemented and can now be evaluated. The task of a localization algorithm is to extract a ROI that contains the complete pupil.

4.1.1 Thresholding

An adaptive algorithm was created to make thresholding more flexible when choosing the best-fitting threshold value t. The histogram is used to determine the highest peak in the low-intensity range

$$t = \text{argmax}\{h(i)|i \in [0, 255]\} \tag{4.1}$$

With $h(i)$ being the histogram of the image. The threshold value t is then used to calculate a range for using double thresholding to extract the pupil.

The image is modified by setting all values below and above the threshold value to 0.

$$f(x,y) = \begin{cases} 0 & iff \quad I(x,y) < t - 35 \quad \text{or} \quad I(x,y) > t + 25 \\ I(x,y) & otherwise \end{cases} \tag{4.2}$$

Important to highlight is that the peak intensity is not centered in the thresholding range. This assumption derives from testing and can be justified with the probability that a pixel with a lower intensity value as t belongs to the pupil is higher than that of a pixel with higher intensity values as t. This approach works well in an environment with almost no noise and no reflections. However, as already mentioned in the theory section, reflections lead to a less high peak in the histogram, and the possibility exists that the threshold value has no peak in the lower value range, leading to a faulty result that can not be used to create an ROI.

The mask created has intensity values between $[t - 35, t + 25]$. The possibility of creating a binary mask exists and helps create a ROI. To create a ROI all the contours of the binary mask are evaluated on their circularity and the location of the contour with the best circularity and the greatest area will be used as a ROI. [RESULTS] This method works in an environment with almost no noise flawless and almost in real-time. Thresholding is keen to struggle with noise and is therefore not useful for the LPW data set.

4.1.2 Edge detection

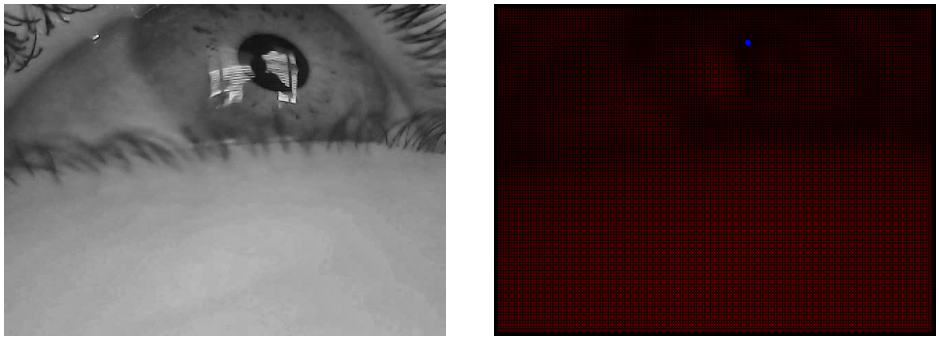
Edge detection is strongly affected by noise, and therefore preprocessing is vital for valuable results. Every frame undergoes the same preprocessing as in the other algorithms, but a Gaussian blur is used additionally to weaken small fast-changing intensity regions that can be considered noise.

Even though Gaussian blur is used, pupil contour information is still missing because the contour is covered by noise and can not be recovered. The edges are detected, but the same problem as with thresholding arises here. By using Sobel and then the Canny edge detection, the results are useful in an environment with almost no noise. Canny edge detection needs two threshold parameters to work correctly and also here arises the problem that these parameters need to be adaptive to the environment, which can be tricky. [RESULTS]

4.1.3 Haar-like features

The Haar-like feature is the approach that is proposed by this thesis to find the region of interest. It has the best detection rate of all approaches tested on the LPW data set and is therefore the best approach to finding the ROI.

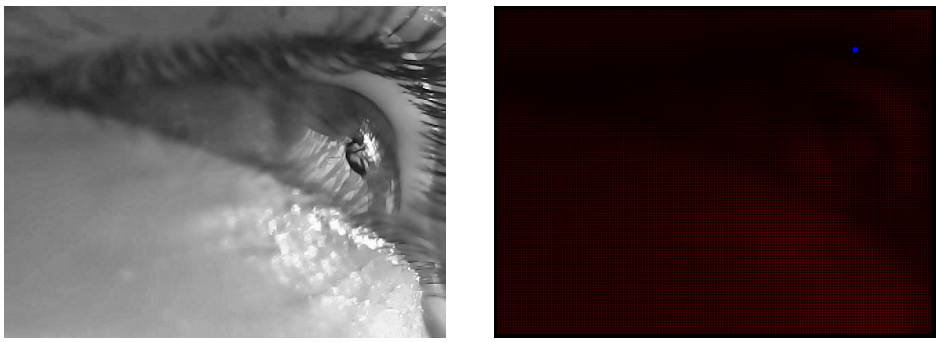
The Haar-like feature is constructed as described in the theory part and is shown in **??**. The calculation of the response matrix is done by using the integral image and is calculated multiple times with variating radius r. All response matrixes are compared, and the location of the highest response is returned as a point that can be considered to be inside the pupil.



(a) Original Frame **(b)** feature vector, blue is the best response

Figure 4.1: Feature vector for pupil detection

The strongest response lies within the pupil, and the location is used to create a ROI. In this case, a ROI of $(220x220)$ was the norm size, but depending on the rescaling of the video, this needs to be adapted. Important to note is that the returned point in the pupil is not given to be in the center, and this in an important fact when choosing the size of the ROI. Nevertheless, Haar-like feature detection still needs improvement and even though it can handle noise, there are limits to the amount of noise until the algorithm fails. Also, the algorithm is not able to detect the pupil when the eye is closed.



(a) Extreme noise example **(b)** feature vector result with best response

Figure 4.2: Limits to the Haar-like feature approach

The problem with the LPW data set is that it is so versatile and the conditions change during the recording. Therefore, finding an approach that can adapt

to all the conditions and perform well in a given time interval is hard. Of all the localization algorithms Haar-like feature outperformed every single one and is the most adaptable to changing environmental conditions and different eye positions. The implementation can still be improved and sped up, in the Haar-like feature detection the following libraries were used to speed up the algorithm:

```
from concurrent.futures import ThreadPoolExecutor
from numba import njit
```

ThreadPool Executor makes it possible to use multithreading and njit compiles the sliding window over the integral image and the calculation of the response matrix for the Haar-like feature to machine code. The use of those two libraries makes the algorithm run about ten times faster.

4.2 Ellipse parameter estimation

In the second part of pupil detection, it is necessary to make use of the location information from the first part and build on this foundation to find the five ellipse parameters: center: (x, y), axis: (major, minor) and angle. The LPW has labels for the center only of the pupil, and therefore only the center can be used to evaluate the performance of the algorithms. A second evaluation has to be done manually by inspecting the fit of the ellipse to the pupil contour. This is hard to evaluate with numerical methods and the results must therefore be taken with a grain of salt. In this section four different algorithms will be discussed and evaluated: The OpenCV ellipse fit based on contours (binary thresholding), Canny edge detection with OpenCV Ellipse fit, ACWE with OpenCV Ellipse fit, and ACWE combined with RANSAC.

4.2.1 Thresholding and OpenCV ellipse fit

The benefit of this method is that it can run in real-time and still perform well in specific frames when the noise amount is low, and the pupil is completely visible. However, as mentioned in the localization part, this combination is not robust enough to handle the LPW data set. The thresholding is done with the same approach as in the localization part and the binary mask is then used to find the contours. The contours are then evaluated by their circularity and the similarity to an ellipse. The contour with the biggest area is further used with the OpenCV ellipse fit function.

The OpenCV ellipse fit function is based on the least square method and is robust and fast. The problem with the least square method for ellipse fitting is that the method tries to find an ellipse that minimizes the distance from every point on the boundary of the binary mask to the ellipse curve. When

the pupil is partially covered by noise, the contour can not match the original shape of the pupil and is not the total pupil boundary. Using the least square method leads to a faulty ellipse fit where the result is not usable. The amount of outliers is too high, and the ellipse fit is not accurate enough because there is information missing. The problem can be seen in the visualization of the OpenCV ellipse fit:

(a) ROI **(b)** Binary thresholding **(c)** OpenCV Ellipse fit

Figure 4.3: Binary thresholding with OpenCV ellipse fit

4.2.2 Canny edge detection with OpenCV ellipse fit or RANSAC

For this combination, the first step is to find the ROI, this is done with the Haar-like feature. This is from great importance because otherwise, there is no chance of finding the pupil boundary because there will be many edges that are possible pupil edges. The image is blured with a Gaussian blur. Afterward, the Canny edge detection is used on the ROI (Using the gradient, by calculating the sobel gradient).

The edges are evaluated by their circularity. The best matching contour is further processed with the OpenCV ellipse fit or the RANSAC alrogithm. This results in the pupil ellipse approximation. The problem with this approach is that the boundary must be clearly visible and the threshold values must be chosen carefully. Here the already known intensity value from the given point inside the pupil is used to make a decision for the thresholding values needed by the Canny edge detection. The benefit of this algorithm is the speed at which the frames can be processed. However, this combination is not considered to be robust enough to handle the LPW data set and is therefore not further evaluated.

4.2.3 ACWE with OpenCV ellipse fit

This approach uses the OpenCV ellipse fit to interpret the results from the ACWE, the benefit is that it runs faster than the RANSAC but it is strongly dependent on the quality of the binary mask returned by the ACWE. If the

mask is not completely on the boundary of the pupil, the ellipse fit will also not fit the pupil because of the least square fit used in OpenCV ellipse fit.

ACWE and OpenCV ellipse fit also only work in low noise environments, just as the thresholding approach and the Canny edge detection. When comparing the threshold and contour segmentation with the ACWE with OpenCV ellipse fit, it can be seen that the quality of the approximation is almost the same. Nevertheless, in terms of speed, ACWE with OpenCV ellipse fit is decently slower than the thresholding approach. Especially when the pupil is large, the ACWE takes a long time to converge to the boundary of the pupil because the level set u of the ACWE can only grow for a certain amount each iteration. In contrast, thresholding finds the contour of the pupil in a few iterations and in a fraction of the time compared to ACWE.

4.2.4 ACWE combined with RANSAC

The benefit of using ACWE discussed in section 3.6.1and the RANSAC method already discussed in section 3.5 is that the ellipse fit is not anymore based on the least square method but instead uses a different approach as already introduced in the RANSAC section. The RANSAC algorithm leads to a better fit but with the tradeoff of computational efficiency. This tradeoff is worth it when the images contain much noise and the pupil boundary is not completely visible. If only a part of the boundary is visible, the accuracy of the ACWE combined with RANSAC is more significant than when just using the OpenCV Ellipse fit.

Haar-like feature detection returns a point inside the pupil. This location is the center position for the initial contour creating the level set u. When ACWE converges, the binary mask is dilated and eroded to remove small noise pixels and to make the mask smother. Then the mask is used as input for the RANSAC algorithm to fit an ellipse to the contour of the mask. The parameter of the approximated pupil contour can be retrieved from the ellipse fit.

As already foreshadowed, the process contains many steps and ACWE and RANSAC are iterative algorithms, making this approach the slowest but most accurate of all discussed algorithms and their combination. The RANSAC algorithm uses a subset of five points and fits an ellipse to these points. To evaluate the ellipse fit, the distance from all points of the mask's contour to the ellipse curve is calculated and categorized. The fit is then calculated with

these equations:

$$\text{InlierRatio} = \frac{n_{\text{inliers}}}{n_{\text{total\_points}}} \tag{4.3}$$

$$\text{BorderRatio} = \frac{n_{\text{border}}}{n_{\text{total\_points}}} \tag{4.4}$$

$$\text{Area} = \pi \cdot a \cdot b \tag{4.5}$$

$$\text{Fit} = \alpha \cdot \text{InlierRatio} + \beta \cdot \text{BorderRatio} - \frac{\text{Area}}{n_{\text{total\_points}}} \cdot \pi \tag{4.6}$$

with $\alpha = 150$, $\beta = 200$, a and b are the major and minor aixs of the ellipse. The *InlierRatio* is the ratio of inliers to the total number of points, the *BorderRatio* is the ratio of points on the border of the ellipse to the total number of points and the area is the area of the ellipse. The Fit is then calculated by a weighted sum of the *InlierRatio*, *BorderRatio* and the *Area*. The weights are chosen by testing and are not optimal but work well enough and can be further adapted.

In each iteration of the RANSAC the fit is calculated, and if the current ellipse parameters have a greater fit score than all the previous ellipse parameters, the current ellipse parameters are saved as the best fit. After N iterations, the best fit is returned as the approximation of the pupil ellipse.

The reason for using a fit equation is that the individual conditions can exclude each other. When using conditional logic, the evaluation of a fit becomes tricky and strongly dependent on the order of the ellipses generated. For example, suppose the ellipse has a small area, which is considered a good fit, but only 20% of inliers. In that case, the conditional logic rejects all following ellipses based on the smaller area. Therefore conditional logic is not considered for the evaluation of the ellipses. Here is an example fit in a very noisy environment to show the robustness of the algorithm:

Chapter 5

Proposal

5.1 Proposed Algorithm

After intensive research and analysis, the algorithm proposed for pupil detection consist of the following steps:

1. **Preprocessing:** The image is converted to grayscale, and then the histogram equalization method CLAHE is used to improve the image's contrast.
2. **Haar-like features:** From the image the response matrix is calculated using the Haar-like feature for pupil detection proposed by 3.3. The response matrix is then used to find the strongest response in the image, and this point is considered to be inside the pupil area.
3. **ACWE** The active contour without edges algorithm is applied to the image with the point returned by the Haar-like feature detection as center of the initial contour. ACWE then returns the contour of the pupil.
4. **RANSAC** The RANSAC algorithm is applied to the mask returned by the ACWE algorithm. Iterates over the mask contour and fits a circle to a random subset of the contour points. RANSAC returns the circle with the best fit.

Chapter 6

Results

In this chapter, the proposed algorithm is evaluated on the LPW data set, and numerical results are presented, followed by a discussion of the results and possible improvements.

6.1 Evaluation

The evaluation of the proposed algorithm is done by calculating the error of the pupil center. The data set labels the center of the ellipse, not the actual ellipse parameter. Therefore, the evaluation of the actual fit of the ellipse was conducted manually, and no numerical results for the fit are presented. For the evaluation of the pupil center, the ground truth notation explained in subsection 2.1.3 is used. The error is calculated in the x and y direction separately, and a KDE Plot is used to display the error density of the estimation. The error is calculated for each frame, totaling 2000 frames per video. The standard deviation is calculated and drawn into the KDE plot as an ellipse, giving insight into the algorithm's accuracy. The noise changes throughout the video because of eye movement and blinking. As mentioned in the introduction, the algorithm cannot detect if the eye is open or closed. Therefore, a significant error deviation is possible and filtered out if it is more significant than three times the standard deviation of the complete video. To filter the error, the z score is introduced. The z score is a statistical measure that tells how far a point is from the mean of a data set in terms of standard deviations. All errors with a z score greater than three are declared outliers for the evaluation.

6.2 Discussion

The algorithm's performance can be split up into Accuracy, Speed, Robustness and Noise. The error of the pupil center measures the accuracy. The

algorithm's speed is given in frames per second and is calculated with the mean of the total time taken for processing 2000 frames. The robustness is measured by the number of outliers the algorithm produces. The noise is hard to evaluate because the data set is very diverse and eye movement during the video changes the amount of noise present.

6.2.1 Accuracy

The algorithm's accuracy can be seen in the KDE plot of an example video. The KDE plot is shown in figure 6.1 and shows the error density of the pupil center estimation.

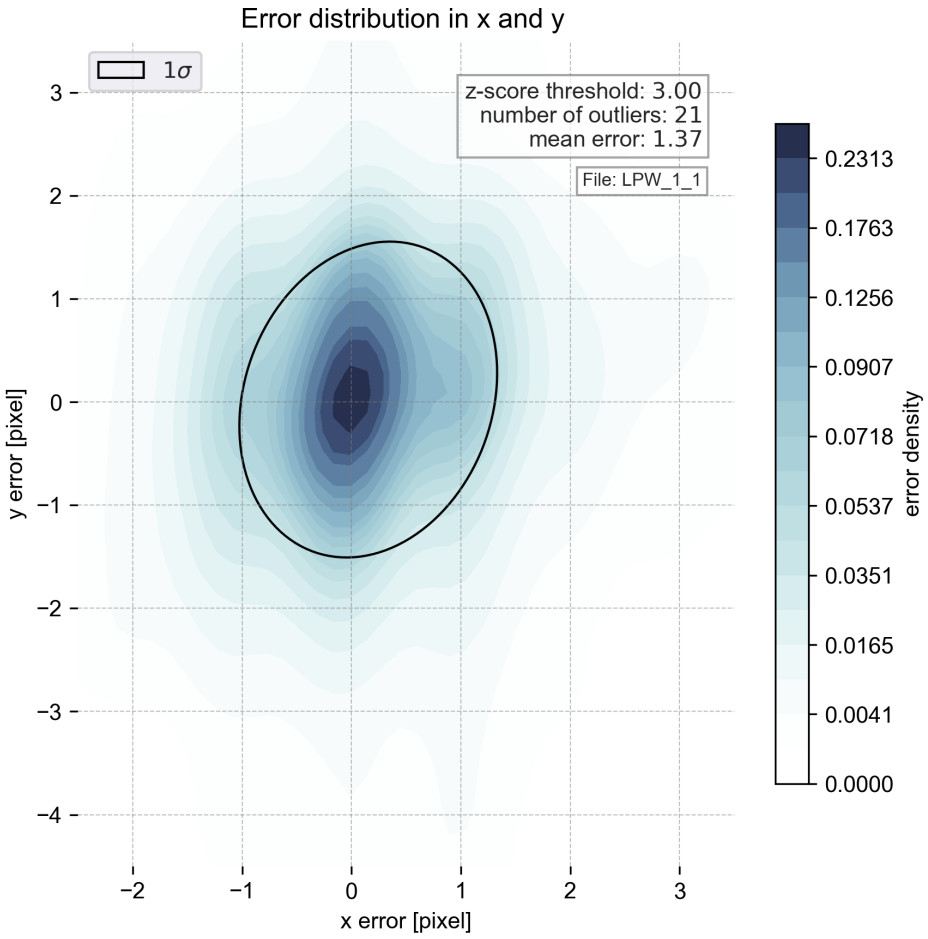
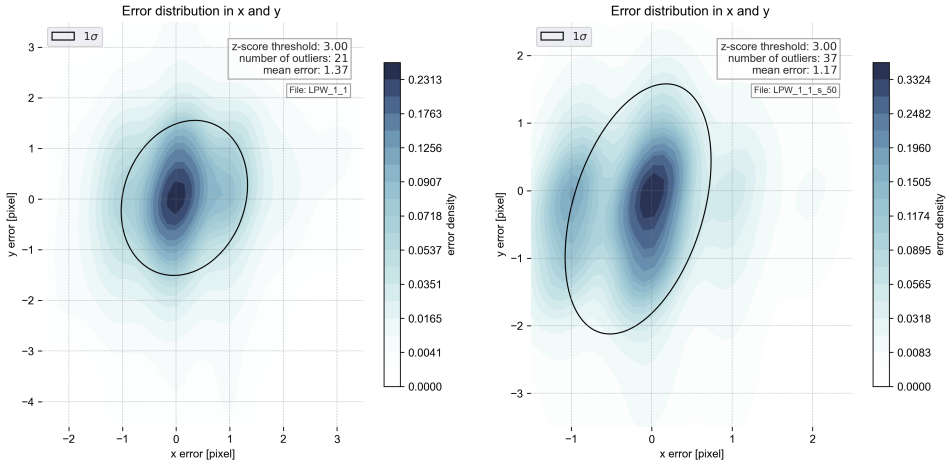


Figure 6.1: Error KDE plot of the pupil center estimation

Here, the standard deviation in the y direction is bigger than in the x direction. The reason for the difference in the axes has to do with the experimental

setup of the LPW data set. Because in most of the videos, the light source is reflected in the pupil when the participant is looking to the left, leaving only a small portion of pupil contour on the right side of the pupil. Because the contour information is not complete and more information on the y-axis is missing. Therefore the y-axis probability error fluctuates more than the x-axis. The same observation persists and increases when using different scaling of the video. Here is a comparison between the full-size video (640x480) with the scaling of 0.5 (320x240) Even though the mean error in figure 6.2b



(a) Full image size (640x480) **(b)** Image size scaled by 0.5 (320x240)

Figure 6.2: Comparing Error KDE plots of different image sizes

seems to be lower, it needs to be considered that the number of outliers almost doubled when scaling the frames with a factor of 0.5 and the standard deviation also increases in the y direction but the mean error decreased. Nevertheless, notable is that the Haar-like feature detected the pupil in every frame, no matter the scaling. This statement is not valid for all the LPW data set videos. By scaling the image, information is lost, including information about the pupil boundary. Important to note is that the parameters for the Haar-like feature and ACWE must be adapted to the scaling of the images. It is possible to get more accurate results by increasing the number of iterations of the RANSAC algorithm as well as λ_1 and λ_2 of the ACWE algorithm. This comes with the cost of speed, discussed in the next section. Another aspect of the accuracy is the fit of the ellipse itself. Even though the center has a considerably low error, the fit of the ellipse contour also tends to jump in a range of one or two pixels. This effect can also be seen when using scaled frames but does not necessarily increase with the scaling. The reason for this is the RANSAC algorithm that is used to fit the ellipse to the binary mask. This can be solved by using a higher number for iterations. The ACWE

algorithm can also be a limiting factor because it stops at the pupil's outer boundary, which is not necessarily the same boundary as visually perceived.

6.2.2 Speed

The algorithm's speed is measured by the time it takes to process 2000 frames. The mean of the total time per frame is considered the speed of the algorithm. Currently, the algorithm runs at 1.2 fps, which equals 0.83 seconds per frame. This time was measured on full-scale images of the size 640x480. Decreasing the size significantly improves the speed and increases the ACWE algorithm's error, which is reflected in the ellipse fit. The parameters for the Haar-like feature and ACWE must be recalibrated for the new scaling. There is no general rule of thumb to scale the parameters. Also, the algorithm's speed does not scale linearly with the size of the image.

6.2.3 Robustness

The algorithm's robustness is measured by the amount of outliers the algorithm produces. Calculating the z-score and setting the threshold to three times the standard deviation of the complete video makes it possible to filter outliers out and get a numerical value for robustness. In a full-scale video with reasonable noise, the haar-like feature can find a point in the pupil in every frame. The smaller the image gets, the less reliable the haar-like feature method becomes. However, even at a scaling of 0.5, the Haar-like feature detection performs at a success rate of 100 percent. The ACWE algorithm is the limiting factor for robustness. Because the RANSAC algorithm expects a certain amount of boundary points to fit the ellipse, the error will also be visible in the boundary approximation if the ACWE algorithm cannot extract the pupil boundary. The fewer border points are detected, the more iteration N RANSAC needs to find the best fit.

6.2.4 Noise

6.3 Possible Improvements

The algorithm has many parameters that can be tuned to improve the performance. The parameters that can greatly impact the performance are the parameters of ACWE introduced in section 4.2.4 and RANSAC parameters introduced in section3.5. The parameters are:

$$\text{ACWE: } \lambda_1, \lambda_2, \text{smoothing iteratios}, \text{Iterations}$$
$$\text{RANSAC: } \alpha, \beta, \text{Iterations}, \text{Threshold}$$

Because the binary mask of the ACWE is used for the RANSAC, λ_1 and λ_2 are of great importance for the model's accuracy. If those parameters are

chosen incorrectly, ACWE cannot extract the pupil area correctly. Therefore, the RANSAC will not be able to fit an accurate ellipse to the boundary of the pupil. λ_1 and λ_2 are sensible to the iris intensity. Darker colors need especially careful tuning.

The number of Iterations is less critical for the ACWE because it has a stopping criteria when the contour converges, but it has to be chosen high enough so that the ACWE is not abruptly stopped before the contour has converged. If more smoothing iterations are chosen, the contour will grow and converge faster but will also be more inaccurate. So there is a tradeoff between speed and accuracy.

The RANSAC algorithm is also sensitive to the choice of parameters. α and β do not influence the speed but the accuracy. They are the weights for the importance of inliers or boundary points. The number of iterations affects the speed and accuracy. The more iterations are chosen, the more different ellipses are fitted, and the chance for a better fit is higher with the cost of speed.

The *threshold* used to decide if a point is on the boundary or not also affects the accuracy. The threshold can impact the accuracy because the binary mask will not have a perfect elliptic boundary. Therefore, the RANSAC algorithm needs to have a certain threshold to decide if a point is on the boundary.

More testing should be done to find the best values for the parameters. The proposed algorithm is currently unable to run in real-time and still needs some improvements and bug fixing to maximize the performance. The proposed algorithms should be implemented in C++ with more multithreading and parallelization to improve the performance further.

Chapter 7

Conclusion

7.1 Summary

In this thesis, a diverse set of existing algorithms for pupil detection, each demonstrating its unique strengths and weaknesses, have been examined. These algorithms were analyzed under various imaging conditions, giving insight into their robustness and efficiency.

The Results show that finding a robust and efficient combination of algorithms is challenging and that the proposed algorithm cannot solve this task in real-time. The proposed algorithm presents a valid combination but still has weaknesses and needs adaptation to the specific use cases. The parameters must be repeatedly tuned on each data set and can not be seen as just working. The thesis aimed to evaluate and compare different algorithms and show their limitations, and this goal was achieved, and an overview of different approaches is given. The proposed algorithm can only be regarded as a solution that outperforms every method in some use cases.

Although the combination of Haar-like features, ACWE and RANSAC can perform well in even tricky conditions, there is still room for improvement and further research. Especially the Haar-like feature detection stood out with exceptional reliability for finding points inside the pupil and is considered the best method to gain information about the pupil's location.

The estimation of the pupil boundary with the proposed algorithm is valid but not perfect. The ellipse fit can still be improved by further tuning the parameters of the individual algorithms. The performance can be increased by implementing the code into C++ and using more multithreading and parallelization. It is unknown if the proposed algorithm can run in real time, but there is a high chance it is possible with further improvements. The estimation quality is sufficient to be used as a starting point for training a highly reliable AI-based eye Tracker or Iris recognition system and gaze

tracing.

The code published on GitHub can be used as a starting point to build more complex tasks on.

Appendix A

Appendix

A.1 Choosing the right model

Depending on the level of noise, different algorithms can be considered. Here only algorithms that are implemented in the thesis are considered. Even more algorithms could be used for object sedimentation, for example, MSER or Machine Learning.

Low noise If there is almost no noise in the image, meaning the pupil is visible all the time and there are no reflections in the pupil area, it is possible to use the Haar-like feature to localize the pupil. Create a ROI and then use the pixel's intensity value with the best response from the Haar-Like Feature to get a threshold value. Then create a binary mask, inspect all contours in the ROI, and choose the contour with the best circularity and similarity to an ellipse. Use OpenCV ellipse fit to retrieve the ellipse parameters. This approach is fast and accurate in low-noise conditions and can run almost in real-time.

Medium noise / High noise Here it becomes more tricky to choose the right mode. In general, the more noise is introduced, the more robust the algorithm has to be. This comes with the cost of speed. Here the proposed algorithm should be used to extract the pupil parameters.

A.2 Code

All the code can be found on GitHub under: https://github.com/parisj/Algorithms_Eye_Detection. The code is implemented in Python and for the packages used, take a look a the requirement.txt file.

Bibliography

[1] X. Zhang, Y. Sugano, and A. Bulling. Max-planck-institut für informatik: Labelled pupils in the wild (LPW). [Online]. Available: https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/labelled-pupils-in-the-wild-lpw

[2] R. C. Gonzalez and R. E. Woods, "Bilinear interpolation," in *Digital Image Processing*, 4th ed., p. 77.

[3] OpenCV: Histograms, histogram equalization. [Online]. Available: https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html

[4] R. C. Gonzalez and R. E. Woods, "Sharpening (highpass) spatial filters," in *Digital Image Processing*, 4th ed., p. 184.

[5] ——, "Canny edge detection," in *Digital Image Processing*, 4th ed., p. 732.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features." [Online]. Available: https://www.merl.com/publications/docs/TR2004-043.pdf

[7] L. Swirski, A. Bulling, and N. Dodgson, "Robust real-time pupil tracking in highly off-axis images," pp. 173–176.

[8] A. Fitzgibbon, M. Pilu, and R. Fisher, "Direct least square fitting of ellipses," vol. 21.

[9] W. Gander, "Least squares with a quadratic constraint," vol. 36, no. 3, pp. 291–307. [Online]. Available: https://doi.org/10.1007/BF01396656

[10] K. G. Derpanis, "Overview of the RANSAC algorithm."

[11] A. Vondráček, "Image segmentation using a morphological operator for curvature-driven motion."

[12] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," vol. 1, no. 4, pp. 321–331. [Online]. Available: http://link.springer.com/10.1007/BF00133570

OST
Ostschweizer
Fachhochschule

Declaration of originality

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

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

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

Title of work (in block letters):

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

Name(s):	**First name(s):**

With my signature I confirm that

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

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

Place, date	**Signature(s)**

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