# Algorithms for Pupil Detection

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

---

# Data Set

---

## 1.1 Labelled Pupils in the Wild (LPW)

### 1.1.1 Description

The data set "Labelled Pupils in the Wild" [1] or short LPW was created by the Max Plank Institution and and contains 66 high-quality, high-speed eye region videos for the development and evaluation of pupil detection algorithms. All videos are labeled with the center of the pupil. 22 participant's eye region with five different ethnicities, five different eye colors were recorded.

The goal of the data set was to record samples of participants under conditions that are present in the reality. By having strong reflections, wearing glasses, wearing make up and so on the data set becomes a difficult challenge for pupil detection algorithms and a good evaluation of the algorithms is possible.



**Figure 1.1:** three example frames from the LPW data set.

### 1.1.2 Procedure

The participants were asked to look at a moving red ball as it moved around. The recording location was randomly picked and was in and 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 was used that 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 1.1:** Location of recordings

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

**Table 1.2:** Light Source of recordings

### 1.1.3 Ground truth annotation

In many cases the pupil area has a clear boundary the center can be annotated easily. One or two points inside the pupil were manualy selected and used as seed points. From these point area with the same intensity value are extracted and used for annotation. But in some difficult scenarios this method was not possibl because of strong noise over the pupil. In this case the additional information gained from the participants following a red fall with their eyes was used to create the labels. The frames then were manually annotated and this data was then used as calibration data to cross reference the center of the pupil with the position of the red ball.

The complete data set has labels for the center of the pupil for every frame. Given the ground truth annotation, it is possible to evaluate algorithms and compare them to each other.
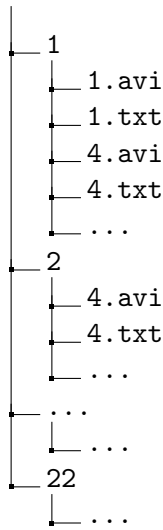
### 1.1.4 Folder structure

The folder stands for the part, file number stands the participants and the file extension for the file type. The text file contains the ground truth annotation of the pupil center in x and y coordinates for each frame.

The labels.ods file contains the 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.

The folder structure is shown in the following tree diagram.

```
LPW/
├── labels.ods
└── README.txt
```

```
├── 1
│   ├── 1.avi
│   ├── 1.txt
│   ├── 4.avi
│   ├── 4.txt
│   └── ...
├── 2
│   ├── 4.avi
│   ├── 4.txt
│   └── ...
├── ...
│   └── ...
└── 22
    └── ...
```

## 1.2 Eye characteristics

### 1.2.1 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. Then there are four different parts of the eye that matters for pupil detection. The white part of the eye is the **sclera** and is separated from the **iris** by the **limbus**.

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

**Table 1.3:** Oversight iris and pupil radius

The iris regulates the radius size of the **pupil** is and thereby regulates 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/hazle, brown, gray, amber and black. There are also other colors but they are in connection with health issues or genetic defects and play no role in this thesis.

The most important fact for pupil detection is that the iris, pupil and sclera have different brightness values. The pupil is the darkest area in the eye, the sclera is the brightest and the iris is 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
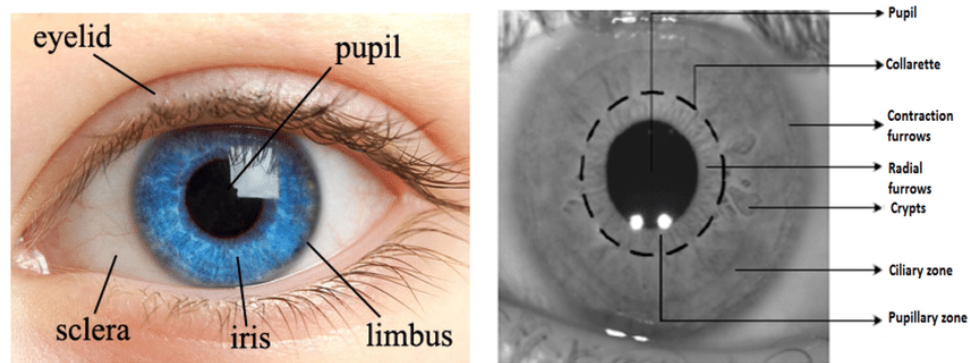
**Figure 1.2:** Overview of the different parts of the eye.

is called **purkinje reflection** and can temper with pupil detection algorithms accuracy and therefore is a challenge for pupil detection algorithms.

### 1.2.2 Image characteristics
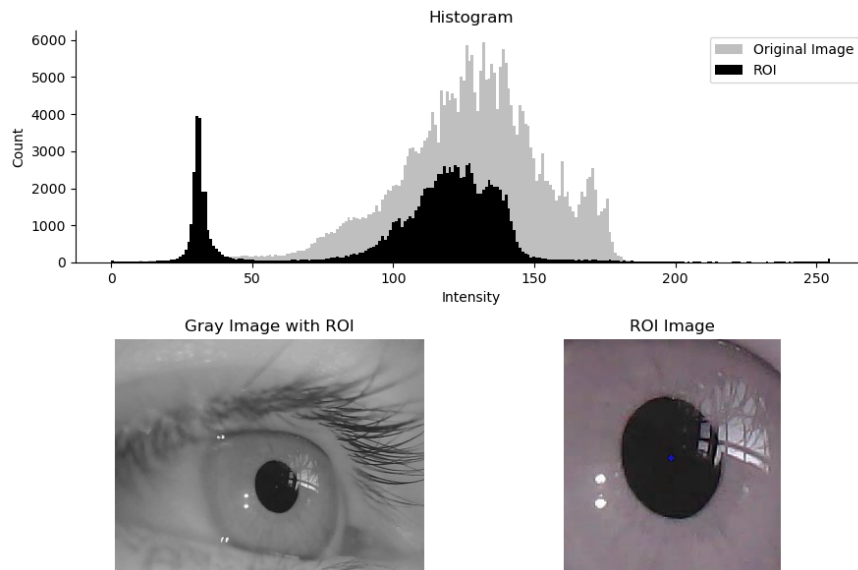
**Histogram**

When inspecting the histogram of a randomly chosen frame from the LPW data set, it can be seen that the histogram shows different peaks in different intensities. In this particular example it can be seen that the peak between 0 and 50 corresponds to the pupil it self. This can be be seen in figure 1.3 because when comparing both histograms the lowest peak stays unchanged. The intensity of the iris is therefore between 90 and 150.

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. Subtracting the ROI from the original picture will result in a histogram almost only containing the skin and sclera. The histogram is therefore a strong tool for eye segmentation.

If there is more reflection on the pupil, the histogram will also change in its 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. This ultimately makes it hard for using 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 one pixel row of the image with their intensity values. Here it can be observed that the pupil creates a valley in the intensity values. The reflection on the pupil generates a peak right after the valley. This can be seen in figure 1.4.

The x coordinate represents the pixel at this coordination and the y coordinate

**Figure 1.3:** Comparison histogram of the whole image and the region of interest.
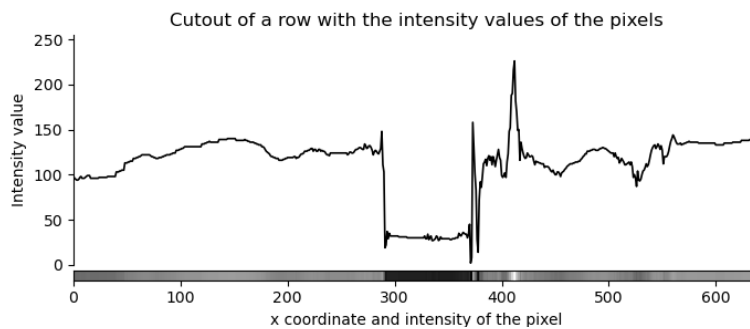


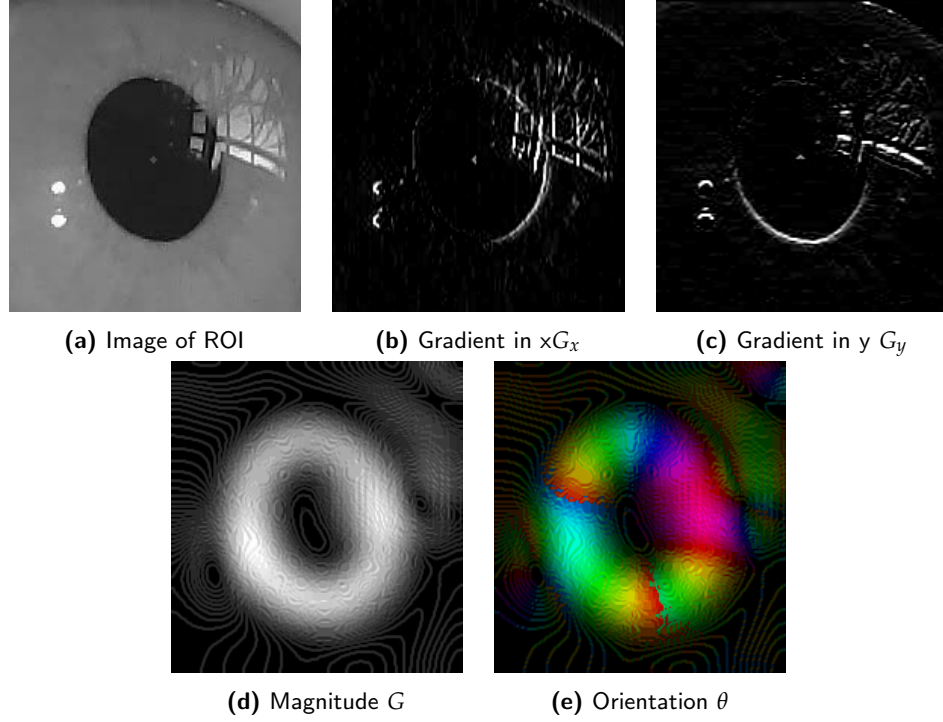**Figure 1.4:** Plot of the intensity values of one pixel row.

represents the intensity value of the pixel.

The conclusion is, that the histogram is a strong tool to gain insight information of the frame but can vary a lot depending on the environment, therefore adaptive algorithms have to be used.

**Gradients**

As already shown in the previous subsection, the pupil creates a valley in the intensity values. This is also reflected in the gradient of the image. The gradient responds strongly to the change of intensity in the transition from the pupil to the iris. The orientation of the gradient points from the pupil towards the iris and can be of great help for edge detection. The gradient is calculated by using the Sobel operator. The Sobel operator is a discrete

differentiation operator. It convolves the image with a differential kernel and therefore emphasizes regions with high spatial frequency. The Sobel operator will be discussed in a later chapter in more detail. But for know it's important to get an overview of the different characteristics of the eye.



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



**(d)** Magnitude $G$     **(e)** Orientation $\theta$

**Figure 1.5:** Plot of gradient characteristics of the ROI.

The gradient in x direction is shown in figure 1.5b and the gradient in y direction is shown in figure 1.5c. The magnitude of the gradient is shown in figure 1.5d and the orientation of the gradient is shown in figure 1.5e. Depending on the environment, the gradient itself could already sufficient for pupil detection. In image processing the gradient is often used for edge detection and all sorts of different filters. Used correctly it can be very powerful. The magnitude is calulated by the following formula:

$$G = \sqrt{G_x^2 + G_y^2} \tag{1.1}$$

Because we have the gradient in x and y direction, we can calculate the orientation of the magnitude by using the following formula:

$$\theta = \arctan \frac{G_y}{G_x} \tag{1.2}$$

The orientation is calculated in radians and can be converted to degrees by multiplying it with $\frac{180}{\pi}$. The orientation is perpenticular to the edge and points in the direction of the gradient. Because the pupil is darker than the iris, the gradient points from the pupil to the iris.

**Noise**

In the data set there are different kind of noise. The most common noise is the noise created by the camera. This noise is called **Gaussian noise**. Gaussian noise is a statistical noise having a probability density function equal to that of the normal distribution, which is also known as the Gaussian distribution. The noise is created by the camera and is therefore not dependent on the environment. The noise is therefore not a problem for the pupil detection algorithm.

Considering that the goal is to detect the pupil there can also be other noise that can be problematic. **Reflection** of the environment can have an impact on the accuracy of the algorithm and destroy information about the pupil. Also the **eyelashes** can be problematic. They are very dark and therefore temper with the edges of the pupil and iris. Especially when the eyelashes are in front of the pupil. Also if make up is used.

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. Then also glasses or lenses can add noise to the image.

Chapter 2

---

# Theory

---

## 2.1 Algorithms

This chapter will take a look at commonly used algorithms in image processing for edge detection, identifying areas of interest and applying them onto pupil detection. At first the algorithms will be discussed and analyzed on possible use cases, individual strengths and weaknesses. To show the nature of the algorithm, the same preprocessed images from the LPW data set are used and therefore it is possible to showcase the results and compare them. The general approach for pupil detection, even tough different algorithms are use, can be summarized by finding the region of interest (ROI), then find pupil edges and finally extract the pupil as ellipse. Depending on the algorithm, the steps can sometimes be extended or even combined. This chapter will therefore also discuss the possible combinations of algorithms.

### 2.1.1 Fundamental notation

Thru out this thesis the following notation is used to descripe 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 there the coordinate system is defined different than in mathematics. The origin is in the upper left corner and the x-axis is pointing to down vertically. The y axis is pointing to the right horizontally. this is shown in figure 2.1.

Also important to note it that the image is a discrete function, therefore each intensity value $I$ comes with an quantization error. This is also the case when using an algorithm on the intensity values of the image. So it is not possible to have an exact result, it is always an approximation of the real result.
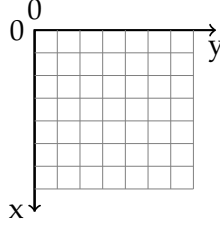
**Figure 2.1:** Coordinate system used in image processing.

**Relationship between pixels**

One also important theory in this thesis will be based on 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:
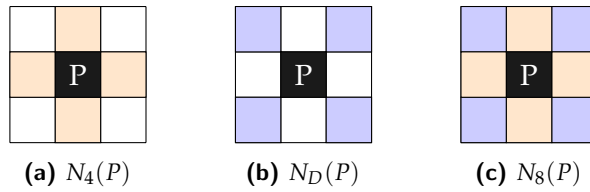
$$N_4(P) = \{(x, y+1), (x, y-1), (x+1, y), (x-1, y)\} \tag{2.1}$$

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:

$$N_D(P) = \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\} \tag{2.2}$$

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

$$N_8(P) = N_4(P) \cup N_D(P) \tag{2.3}$$



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

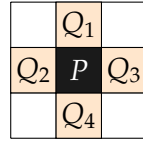**Figure 2.2:** 3 different neighborhoods of pixel $P$ at location $(x, y)$.

**Adjacency**    Mainly 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's possible to define different subsets

of $V$, containing the intensity values that are considered as adjacent 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. In this thesis the intensity range is $V = \{0, 1, 2, ..., 255\}$.

To keep it simple, the following explanations will be based on a binary image with $V = \{1\}$. Let 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 $Q \in N_4(P)$ and $f(Q) \in V$: The pixels that are directly above, below, left and right of the pixel.

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

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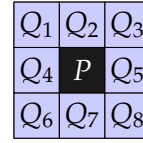


**(a)** 4-adjacency      **(b)** $8 - adjacency$

**Figure 2.3:** 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 exist a path (or curve) from $P$ to $Q$ that consists of a sequence of distinct pixels with coordinates

$$(x_0, y_0), (x_1, y_1), ..., (x_n, y_n), \text{ where } n \text{ is the lenght of the path.}$$

If the path is closed then $(x_0, y_0) = (x_n, y_n)$

**Region** Let's define $R$ as an 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 $R$ that are adjacent to pixels in $R$. In the definition of a boundary, the adjacency type is important. As a rule of thumb to define the boundary, the 8-adjacency is used. One important property of the outer boundary is, that it is a closed path. The inner boundary is the set of pixels that are in $R$ but are adjacent to at least one pixel that is not in $R$ and again the 8-adjacency is used. The inner boundary is not a closed path.
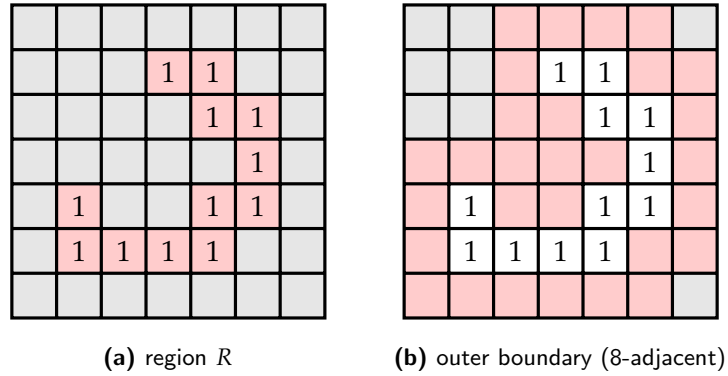
**(a)** region $R$        **(b)** outer boundary (8-adjacent)

**Figure 2.4:** Region and outer border.

## 2.1.2   Preprocessing

To be able to compare the different approaches, it is important to define the used algorithms that lead to the wanted result. An image taken has to be preprocessed first. The idea behind this step is to create a common ground for narrowing the deviation of the images down, so that the algorithms are able to recreate the same result over span of different images.

**Converting to grayscale**

As already presented in the previous chapter, the only colorful part of the eye is the iris. But the color itself is of no interest for the detection. Therefore the frames are first converted to grayscale. This is done by converting the colors into a gray intensity value. During this chapter the grayscale images have certain properties:

It is important to note that these resolutions are resolutions are congruent with the resolutions used in the LPW paper [1]. This is important for the comparison of the results. When scaling an image there is always an image interpolation done to find the best approximation for the new intensity value $I$. The interpolation used in this thesis is the bilinear interpolation.[?] This is a linear interpolation in the x and y axis of the intensity values and solves equation 2.4. let $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$ and

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

**Table 2.1:** Scaling of the frames used in this thesis.

$Q_{22}(x_2, y_2)$ be the four surrounding points. The intensity value $I$ at $(x, y)$ is then calculated by equation 2.4. The point P at $(x, y)$ is the point of interest.
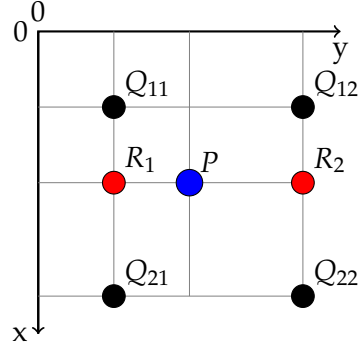


**Figure 2.5:** Bilinear interpolation.

$$v(x, y) = ax + by + cxy + d \tag{2.4}$$

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) = R_1(x, y_1) \tag{2.5}$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) = R_2(x, y_2) \tag{2.6}$$

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) = v(x, y) \tag{2.7}$$

**Histogram equalisation**

Another important aspect of preprocessing the frames is using Histogram equalization. This has the effect of increasing the contrast of the image. For this task Contras Limited Adaptive Histogram Equalization (CLAHE)[2] is used. CLAHE is an Histogram Equalization method that has the benefit that it is adaptive to the local contrast of the image. This is very useful if the contrast of the image is not uniform. By using Histogram Equalization on the frames, noise is added to the image. This can be dealt with by using a

low pass filter, like an Gaussian filter for example.

Using a normal Histogram Equalization would lead to a loss of information in the region around the pupil and the iris. This is due to the fact that the contrast in this region is already very high. The CLAHE method splits the image into smaller blocks called "tiles" and calculates the histogram on each block individualy and therefore does not lead to the same lose of important information at the pupil region. The CLAHE method is applied to the frames after they are converted to grayscale. The result of this step is shown in figure 2.6.
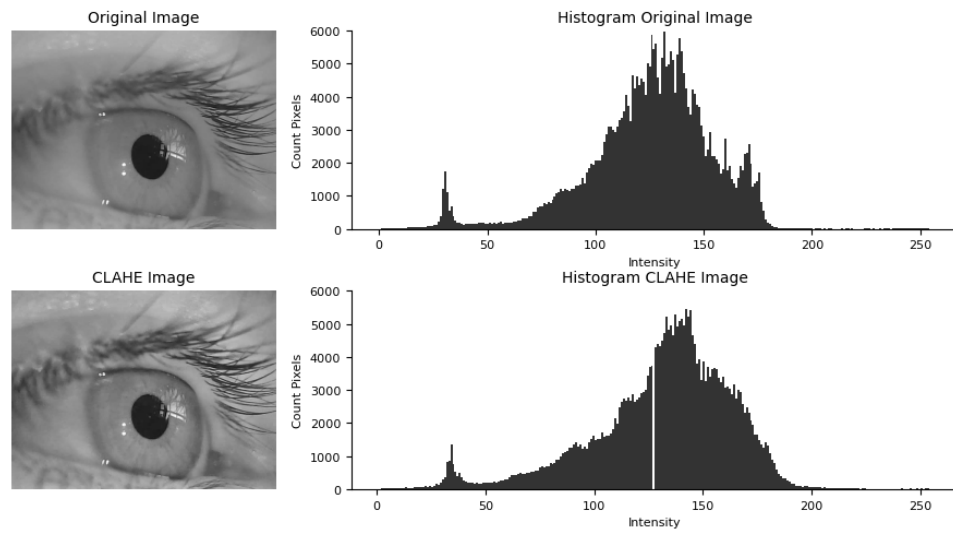


**Figure 2.6:** Example of CLAHE an its effect on the histogram.

These are the parameters used for the CLAHE plots

```
clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(11,11))
```

### 2.1.3 Edge Detection

The main goal behind edge detection is to find edges in the image. An edge is defined as a region that has a high contrast to its surrounding pixels, in other words a rapidly changing intensity in a small area. The edge detection is useful to filter the image for possible pupil contours. The edge detection analysis the image based on the change of intensity. Therefore a gradient calculation is used. There are different methods to calculate the gradient of an image. One of the most popular methods is the Sobel operator that makes use of the first differential of the image. The laplacian operator is another method that uses the second differential of 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 a hysteresis thresholding to filter the edges. The result of the edge detection is a on pixel thick binary edge map. All edges are now possible candidates for the pupil contour.

**Sobel Operators**

The Sobel Operators are used is a common algorithm for edge detection. It is a gradient calculation that uses a 3x3 differential kernel to calculate the gradient of the image. The Sobel gradient is calculated in x and y direction. The gradient in x and y direction are calculated by convolving the image with two different kernels. The image with gray values is defined as $f(x, y)$ and the kernels are defined as $k_x$ and $k_y$. Therefore the gradient is calculated as follows:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.8}$$

$$k_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{2.9} \qquad k_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{2.10}$$

$$G_x(x, y) = f(x, y) * k_x = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k_x(s, t) f(x + s, y + t) \tag{2.11}$$

$$G_y(x, y) = f(x, y) * k_y = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k_y(s, t) f(x + s, y + t) \tag{2.12}$$

$G_x(x, y)$ and $G_y(x, y)$ are the gradients in x and y direction. $f(x, y)$ is the image and $k_x$ and $k_y$ are the kernels. The kernels convolved with the image $f(x, y)$ and the result is the gradient magnitude in x and y direction. In other words the convolution of an image with $k_x$ or $k_y$ gives as result the change from pixel to pixel in x or y direction. In python the Sobel in x and y are calculated with the OpenCV Library for example:

```
G_x = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)
G_y = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)
```

The total gradient magnitude $G$ is calculated with this equation:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.13}$$

and the direction $\theta$ of the gradient is calculated with this equation:

$$\theta = \arctan \frac{G_y}{G_x} \tag{2.14}$$

The result of the convolution can be seen in figure 1.5 in chapter two.

**Canny Edge Detection**

The Canny Edge Detection is used to recieve single edge points form the gradient magnitude image. This algorithm 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 defined as:

$$f_{filter}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.15}$$

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

$$f_{smoothed}(x,y) = f(x,y) * f_{filter}(x,y) \tag{2.16}$$

**Step 2: Compute the gradient magnitude and direction**
The gradient magnitude is calculated with the equation 2.13 and the gradient direction is calculated with the equation 2.14.

**Step 3: Non-maximum suppression**
The non-maximum suppression is used to thin the edges out, so that the edges are only one pixel wide. This can be achieved with an loop that goes over all edges and checks if the current pixel, belonging to the edge, is the local maximum in the direction of the $\pm$gradient vector. If that is the case the pixel is kept, otherwise it is set to zero. Because as already described in 2.1.1 the coordinate system is defined different. The gradient direction is in reference to the $x$ axis.

Because an image is quantized, this also means that $\theta$ needs to be quantized to four directions to evaluate their neighbors.

This leads following quantizations:

$$\theta_q = \begin{cases} 90, & \text{if } 67.5° < \theta \le 112.5° \vee -112.5° < \theta \le -67.5° \\ -45°, & \text{if } 22.5° < \theta \le 67.5° \vee -157.5° < \theta \le -112.5° \\ +45°, & \text{if } 112.5° < \theta \le 157.5° \vee -67.5° < \theta \le -22.5° \\ 0°, & \text{if } -22.5° < \theta \le 22.5° \vee -157.5° < \theta \le 157.5° \end{cases} \tag{2.17}$$
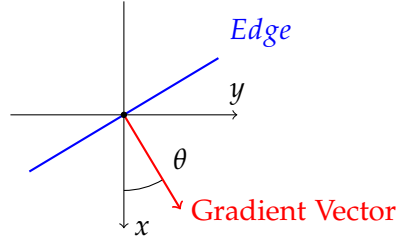
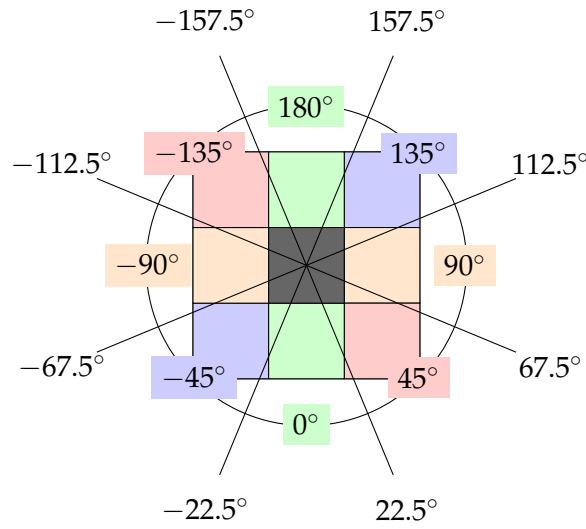**Figure 2.7:** Definition of the gradient direction



**Figure 2.8:** Quantization of the gradient direction

It is important to note that when following the gradient direction, the two neighboring pixels are used to evaulate the gradient magnitute maximum. This is shown in figure 2.8 and 2.9. If the gradient is maximal at the current pixel at $(x, y)$, meaning it is a local maximum in the previous defined neighborhood in respect to the gradient direction, the value of the pixel is written into $g_n(x, y)$, otherwise it is set to zero $g_n(x, y) = 0$. This is called non-maximum suppression. Therefore $g_n(x, y)$ contaians only the thinned edges.

**Step 4: Double thresholding and connectivity analysis**
After Step 3, $g_n(x, y)$ still shows edges that can be thicker than one pixel. $g_n(x, y)$ is then thresholded with a high an low threshold (hysteresis thresh-

**(a)** Neighbors: 90°   **(b)** Neighbors: 45°   **(c)** Neighbors: -45°   **(d)** Neighbors: 0°
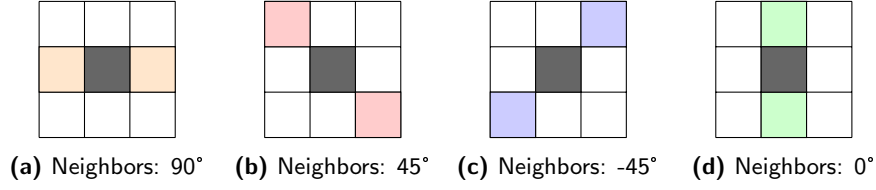
**Figure 2.9:** The gradient magnitude is evaluated in the direction of the gradient.

olding) creating two images:

$$g_{low}(x,y) = \begin{cases} g_n(x,y), & \text{if } g_n(x,y) \geq T_{low} \\ 0, & \text{otherwise} \end{cases} \tag{2.18}$$

$$g_{high}(x,y) = \begin{cases} g_n(x,y), & \text{if } g_n(x,y) \geq T_{high} \\ 0, & \text{otherwise} \end{cases} \tag{2.19}$$

Because two different thresholds were used, there is still overlap between $g_{low}$ and $g_{high}$. All non zero pixels in $g_{high}$ are considered strong edge pixels. To recieve all weak edge pixels, the strong edge pixels are substracted from $g_{low}$. The remaining pixels are considered weak edge pixels.

$$g_{weak}(x,y) = g_{low}(x,y) - g_{high}(x,y) \tag{2.20}$$

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 is done until no more weak edge pixels are found. The result is a binary image $g_{final}(x,y)$ containing all edges. But this still does not return a one pixel thick edge.

To solve this the edges are passed on an edge-thinning algorithmus. Let's define the edges as set A and B as a structuring element. The equation for thining then becomes:

$$A \otimes B = A - (A \circledast B) \tag{2.21}$$

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

**Results**   In a frame where the pupil region clearly can be distinceted from the rest, the Canny edge detector can be used to find the pupil region. 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, eye brows 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 varriety of environments.

**(a)** Original image, scaling 0.5          **(b)** Clear pupil region, Canny edge detection
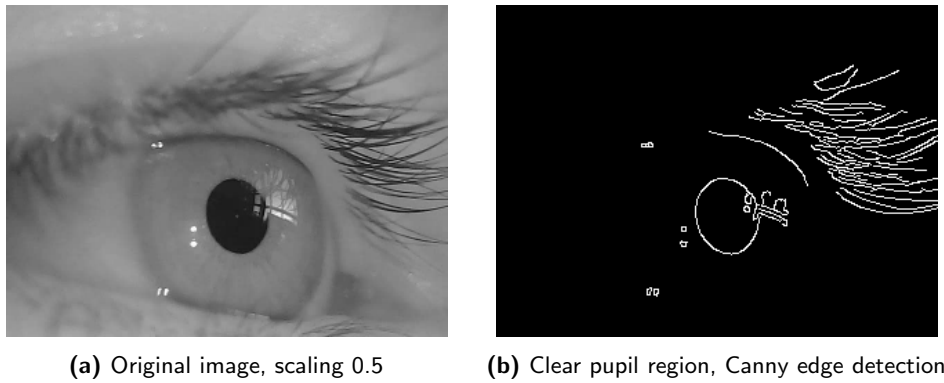
**Figure 2.10:** Canny edge detection on a clear pupil region



**(a)** Original image, scaling 0.5          **(b)** Weak pupil region, Canny edge detection
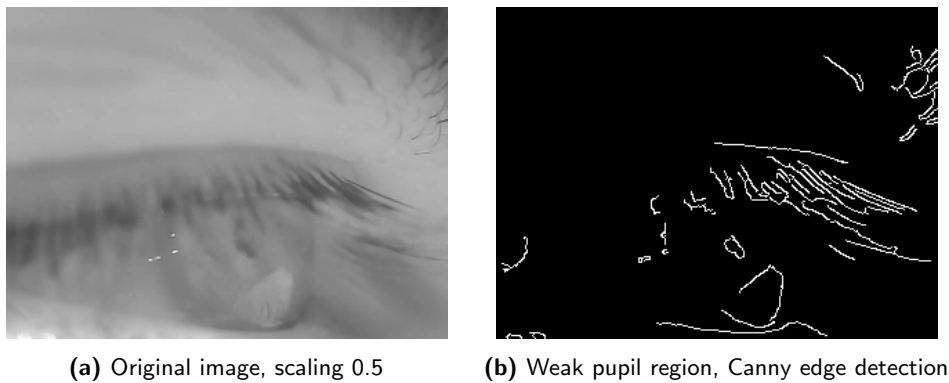
**Figure 2.11:** Canny Edge detection on a frame with weak pupil region
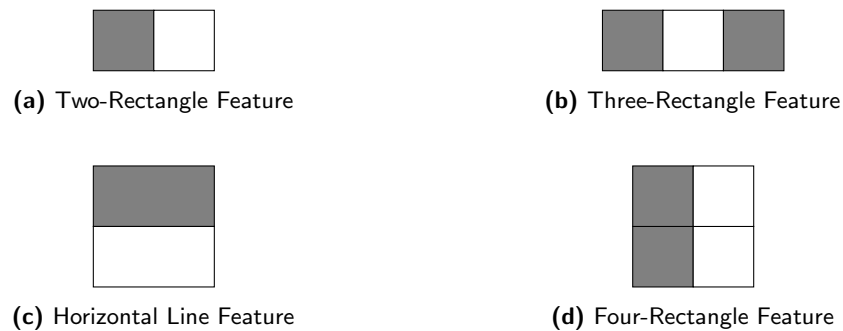
### 2.1.4   Haar like feature detection

**Concept**

Haar like features is based on the observation that object have a certain local intensity variation in an image and this fact can be useful to identify objects. Haar like features makes use of this and divides the image into rectangular regions and then calculates the difference between the sum of the pixels in the white and gray regions, show in figure 2.12. This is done for all possible rectangular positions in the image. The result is a feature vector that is used to classify the image and give insight about the position of the object. In theory the feature is convolved with the image to calculate the feature vector but to make the detection faster, the integral image is calculated once and then used to calculate the feature vector at each possible position in the image. This leads to an easy calculation of the feature vector than now only needs four values to be calculated. The properties of the different features bring different abilities to detect specific local intensity patterns that are an indicator for a certain object or image region property.

19

**Features characteristics**

To get a better understanding of the different features and their characteristics of the resulting feature vector the four features are explained a little more in detail.



**(a)** Two-Rectangle Feature

**(b)** Three-Rectangle Feature

**(c)** Horizontal Line Feature

**(d)** Four-Rectangle Feature

**Figure 2.12:** Haar-like Features

The key concept to keep in mind 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 that is to be detected. The result of the Haar-like features provides information about the presence or absence of a certain pattern that is 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 is able to detect a lot of features. The following classifiers are more complex and are only used if the previous classifier was able to detect a certain feature. This leads to a very fast detection of the object. This safes computational resources and time. Often Machine learning is involved to classify if an object is present or not, based on the feature vectors.

**Haar-Like Feature for pupil detection**

One very useful feature for finding a point in the pupil is given by a feature described in the paper [?]. This feature is constructed differently but used the same way as the other features to calculate an feature vector or response map.
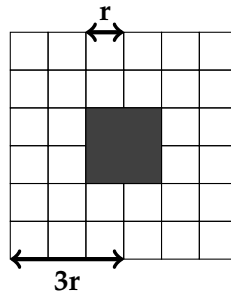
## 2.1.5  Thresholding and Ellipse fitting

**Thresholding**

In Thresholding the characteristic of the pupil is used, that it is the darkest part of the eye as seen in 1.3 . Here it is important to find a certain thresholding value to only extract the pupil. Considering the best scenario that the pupil is not affected by too much noise: no eyelash, no reflection. This method can be reliable to find the pupil region. This process shows to be less computation expensive and finds the pupil fast.

The most difficult part is to find a fitting threshold value. This could be done by using the histogram of the frame. When inspecting the histogram, one could consider the lowest peak in the histogram as the given center value for thresholding. But this is not always the case. There is also the possibility for adaptive thresholding for example otsu thresholding. but throughout the work with thresholding otsu was also tested but was found to be less reliable than the histogram approach itself.
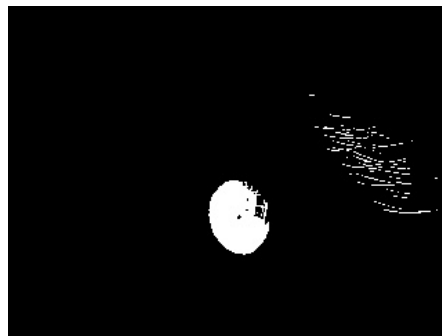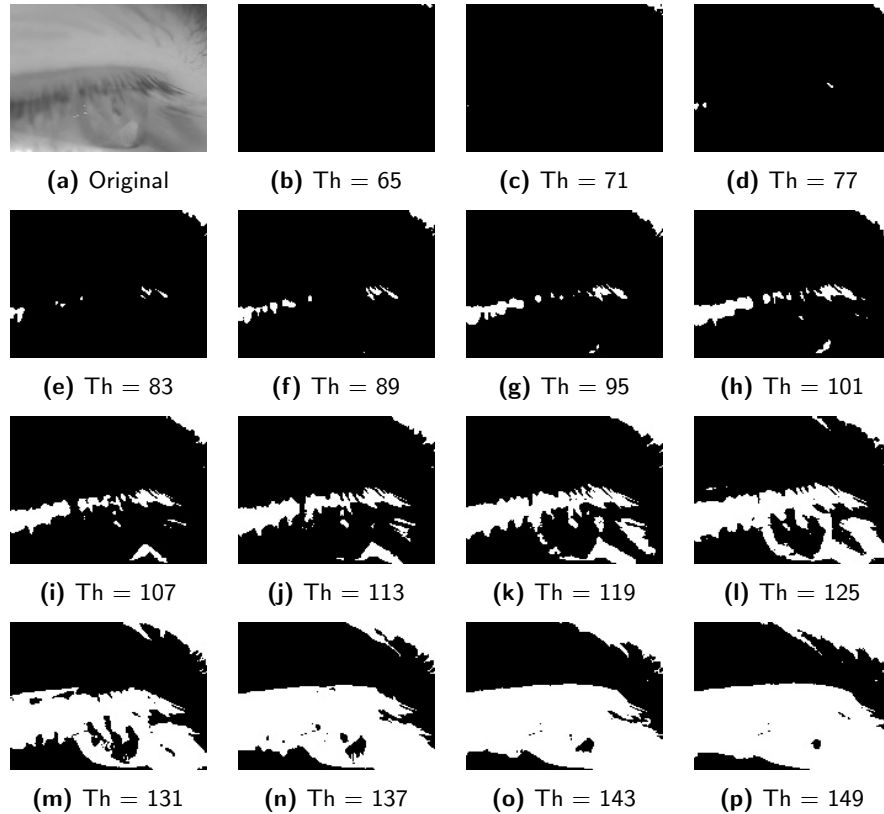


**(a)** Original frame          **(b)** thresholded frame

**Figure 2.13:** Original and thresholded frame

in 2.13 it can be seen that the thresholding is extremely volatile to reflections, eyelashes and other noise. This is why thresholding only has a good performance in a strictly defined environment with almost no noise.

To showcase the volatility and the dependency on the threshold value, in 2.14 a series of the same frame is shown with different threshold values (Th). It can be seen that the threshold value has to be chosen very carefully. If the value is chosen to low, the pupil region is not found. If the value is chosen to high, the 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 2.14:** Thresholded images

**Ellipse fitting**

Ellipse fitting is based on finding a contour and fit an ellipse to it. The accuracy of the ellipse fit is strongly dependent on the quality of the contour extracted. The goal is to find the pupil as a connected region and than use ellipse fit to reconstruct the pupil as an ellipse. This involves solving equations with an approximation algorithm. For example least squares fitting. The least squares fitting is a method to find the best possible fit based on a set of data points. In this case points on the contour. The least squares fitting then calculated the distance from the possible ellipse curve to all points and minimizes the distance to all points. This is done by solving the following

equations:

The general equation for an ellipse with center $(x_c, y_c)$, major axis $a$ and minor axis $b$ is:

$$\frac{(x - x_c)^2}{a^2} + \frac{(y - y_c)^2}{b^2} = 1 \tag{2.22}$$

To find all the values there is a need for at least 5 points on the contour of an possible ellipse.

$$J(a, b, x_c, y_c) = \sum_{i=1}^{n} \left( \frac{(x_i - x_c)^2}{a^2} + \frac{(y_i - y_c)^2}{b^2} - 1 \right)^2 \tag{2.23}$$

where $n$ is the number of data points. The goal is to find the values of $a$, $b$, $x_c$, and $y_c$ that minimize this cost function.

We can use calculus to find the values of $a$, $b$, $x_c$, and $y_c$ that minimize $J$. The partial derivatives of $J$ with respect to $a$, $b$, $x_c$, and $y_c$ are:

$$\frac{\partial J}{\partial a} = 4 \sum_{i=1}^{n} \frac{(x_i - x_c)^2}{a^3} \left( \frac{(x_i - x_c)^2}{a^2} + \frac{(y_i - y_c)^2}{b^2} - 1 \right) \tag{2.24}$$

$$\frac{\partial J}{\partial b} = 4 \sum_{i=1}^{n} \frac{(y_i - y_c)^2}{b^3} \left( \frac{(x_i - x_c)^2}{a^2} + \frac{(y_i - y_c)^2}{b^2} - 1 \right) \tag{2.25}$$

$$\frac{\partial J}{\partial x_c} = -8 \sum_{i=1}^{n} \frac{x_i - x_c}{a^2} \left( \frac{(x_i - x_c)^2}{a^2} + \frac{(y_i - y_c)^2}{b^2} - 1 \right) \tag{2.26}$$

$$\frac{\partial J}{\partial y_c} = -8 \sum_{i=1}^{n} \frac{y_i - y_c}{b^2} \left( \frac{(x_i - x_c)^2}{a^2} + \frac{(y_i - y_c)^2}{b^2} - 1 \right) \tag{2.27}$$

We can set these partial derivatives to zero and solve for $a$, $b$, $x_c$, and $y_c$ to find the values that minimize $J$. This can be done using numerical methods such as Newton's method or gradient descent.

Once we have the values of $a$, $b$, $x_c$, and $y_c$, we can use the equation for an ellipse to draw the fitted ellipse on the data points.

Least square ellipse fitting is commonly used in image processing, computer vision, and pattern recognition to extract features from objects that can be approximated by ellipses, such as eyes, faces, or particles. When solving for the five different parameters, using the conic way described as:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{2.28}$$

defining an ellipse if this condition is satisfied:

$$B^2 - 4AC < 0 \tag{2.29}$$

or if using the parametric was described as:

$$(x - h)^2/a^2 + (y - k)^2/b^2 = 1 \tag{2.30}$$

the solution the the least square problem becomes a non linear problem and needs some sort of approximation. Depending on the kind of approximation different ellipses will be calculated. Also important to keep in mind, the ellipses are multiplied with an rotation matrix $R$ to fit the ellipse also to rotated ellipses.

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.31}$$

**Contours**

When working with thresholds the contour is of interests. 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 one single contour and therefore the contours need to be filtered based on which contour represents the pupil the best. Mainly three criteria are used.

**Circularity / Compactness**

$$\frac{4\pi A}{\oint_{\partial S} dS} \tag{2.32}$$

Where A is the Area of the contour, and it is devided by the integral over the outer boundary of the contour also known as the permitter of the contour.

**Similarity**   The for the similarity the OpenCV library is used. This methods compares a given shape with another. As base for the comparison a simple ellipse is taken and then compared.

**Area**   The Area of the contour should be maximal to find the greatest pupil area and filter out smaller area contours.

Each algorithm that returns a list of contours needs them to be checked with these four conditions and in the best case scenario only the contour of the pupil survives. This is then the contour on which a ellipse will be fitted.

### 2.1.6   Random Sample Consensus (RANSAC)

The Random Sample Consensus (RANSAC) algorithm is an iterative method that is 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 random selected points in $C$. These 5 points are then used to fit an ellipse as described in the section Ellipse Fitting.
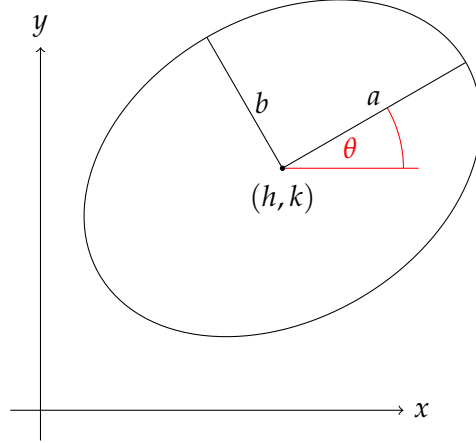


**Figure 2.15:** Ellipse with center $(h,k)$, semi-axes $a$ and $b$, and rotation $\theta$

$$\frac{((x-h)\cos(\theta)+(y-k)\sin(\theta))^2}{a^2} + \frac{((x-h)\sin(\theta)-(y-k)\cos(\theta))^2}{b^2} = 1$$
(2.33)

The equation 2.33 represents an ellipse with major axis $a$, minor axis $b$ and the center of the ellipse at $(h,k)$. The angle $\theta$ is the rotation of the ellipse in the coordinate system. When comparing it with the general ellipse equation and compare coefficients the covariance matrix can be derived:

$$\mathbf{A}x^2 + \mathbf{B}xy + \mathbf{C}y^2 = 1$$
(2.34)

$$\mathbf{A} = a^2 * \sin(\theta)^2 + b^2 * \cos(\theta)^2$$
(2.35)

$$\mathbf{B} = 2(b^2 - a^2) * \sin(\theta) * \cos(\theta)$$
(2.36)

$$\mathbf{C} = a^2 * \cos(\theta)^2 + b^2 * \sin(\theta)^2$$
(2.37)

The covariance matrix is then defined as:

$$\mathbf{V} = \begin{bmatrix} \mathbf{A} & \mathbf{B}/2 \\ \mathbf{B}/2 & \mathbf{C} \end{bmatrix}$$
(2.38)

The benefit of using the covariance matrix is, that the eigenvalues and eigenvectors have a direct relation to the major and minor axis and the

rotation of the ellipse. The eigenvalues represent the major and minor of the axis and the eigenvectors represent the rotation matrix $R$ to rotate the ellipse in the coordinate system. 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 an unit circle.
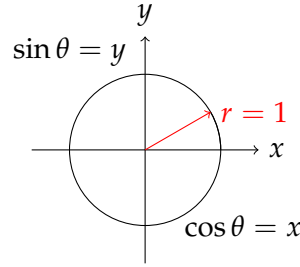


**Figure 2.16:** Unit circle with trigonometric identities

This allows for a simple distance calculation. Because $\lambda_{1,2}$ are known, the new coordsystem can be transformed to the unit circle by dividing the x and y values by the eigenvalues. The points form $C$ 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.

$$p' = R^T \begin{bmatrix} p_x - h \\ p_y - k \end{bmatrix} \odot \begin{bmatrix} \frac{1}{\lambda_1} \\ \frac{1}{\lambda_2} \end{bmatrix} \tag{2.39}$$

$$r = \sqrt{p_x'^2 + p_y'^2} \tag{2.40}$$

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.

$$d = \sqrt{p_x'^2 + p_y'^2} - 1 \tag{2.41}$$

By using this knowledge, the RANSAC algorithm iterates $n$ times to find the best ellipse with the most inliers. The inliers are calculated like following:

$$p = \begin{cases} \text{inlier} = \{p \in C | d < 0\} \\ \text{border} = \{p \in C | d = 0\} \\ \text{outlier} = \{p \in C | d > 0\} \end{cases} \tag{2.42}$$

The RANSAC algorithm calculates the ellipse parameters with five random points $S_c$ for $n$ iterations and evaluates each iteration the number of inliers. The ellipse with the most inliers is then used to calculate the final ellipse

parameters with the best fit. The differentiation between on border an inlier is important to weight the focus on enclosing ellipses or on the border of the ellipse. Because the calculation of the distance $d$ is not an integer, it is possible that the distance is not exactly zero but still on the border. So it is important to keep in mind that comparing floats with each other, to alway use a small epsilon value that d can differ from 0 to set the threshold. In python functions like *isclose* from the math module can be used to compare floats with each other. The number of iterations to find a 99% chance of finding the best ellipse with the most inliers can be calculated with the following formula:

$$n = \frac{\log(1-p)}{\log(1-w^s)} \tag{2.43}$$

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 can then be used to estimate the number of iterations $n$ for the RANSAC algorithm to be 99% sure to find the best ellipse with size $s$ as sample. In our case we use $s = 5$, $p = 0.99$ and for $w$ an approximation needs to 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.

$$n = \frac{\log(1-0.99)}{\log(1-0.90^5)} = 5.158 \tag{2.44}$$

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 inliers are the sum of border and inliers. Otherwise the number of iterations needs to increase because the ratio inliers to outliers changes drastically.

### 2.1.7 Active Contouring

Active contour segmentation is a method that is used to find the boundary curve of an object given that the object exists. There a many different approaches in active contouring but in this thesis two variants will be discussed. The first one being the classic classical snakes approach (Kass et al., 1988) with a simple energy function and the other variant being the active contouring without edges (ACWE) based on level sets. In both cases a 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 is fitting to the edge 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 snakes approach (Kass et al., 1988) **??** associates the curve $C$ with an energy given in 2.46 But the most basic description for the energy minimization can be summarized in $E_{int}$ and $E_{ext}$

$$E = E_{int} + E_{ext} \tag{2.45}$$

$$E(C) = \underbrace{\alpha \int_0^1 |C'(q)|^2 \, dq + \beta \int_0^1 |C''(q)|^2 \, dq}_{E_{int}} \underbrace{- \lambda \int_0^1 |\nabla I(C(q))| \, dq}_{E_{ext}} \tag{2.46}$$

The goal is to minimize the energy function and by doing so the contour $C$ will find local minimals and depending on the sign of *lambda* 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 describes the external energy. In other words, the intern energy describes the curve smoothness and the external energy describes the relation of the curve to the edge at the curves boundary. In the following the three 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 then minimized by changing the position of the points $c_i(x_i, y_i)$ and the process is repeated until the energy function converges. 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 converge to a local minimal.

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

$$E = \sum_{i=0}^{N-1} \alpha (c_{i+1} - c_i)^2 + \beta (c_{i+2} - 2c_{i+1} + c_i)^2 - \lambda |\nabla I(c_i)| \tag{2.47}$$

Here it is important to note, that the points are circular, when $i$ is equal to $N-1$ 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 $C'(q)$ calculate the length of the curve and $C''(q)$ calculates the curvature of the curve.

Because the classic snakes implementation searches for the best boundary curve that localy minimizes the energy function, the initial contour need to be

initialized close to the best solution. Otherwise the algorithm will converge to a local minimum and not the global minimum. The classic snakes approach searches for the minimal with PDEs that is solved using the steepest descent method which brings performance issues and is prone to terminate early if a local minimum is found.

## Active Contouring without Edges (ACWE)

In ACWE 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 operation use four different structuring elements.

**(a)** Element 1    **(b)** Element 2    **(c)** Element 3    **(d)** Element 4
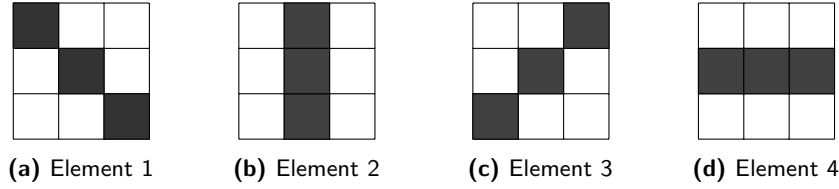
**Figure 2.17:** The four morphological structuring elements used in ACWE.

ACWE is based on Geodesic Active Contours (GAC). GAC has the benefit that it already solves key problems from the classical active contouring. Based on thresholds the contour is evaluated and decided if it flows expands the contour to a new pixel or not. 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 compared to the classical active contouring method. It is also possible to detect the interior and exterior boundary. In the classical approach the interior force is based on first and second derivatives of the curve. This leads to stability, computational complexity and performance issues. This is solved in the GAC by using the evolution of an implicitly defined curve. The GAC algorithm based on a level set function $u$ is defined as following:

$$\frac{\partial u}{\partial t} = \underbrace{g(I) \cdot |\nabla u| \cdot \mathrm{div}\left(\frac{\nabla u}{|\nabla u|}\right)}_{\text{Smoothing force}} + \underbrace{g(I) \cdot v \cdot |\nabla u|}_{\text{Balloon force}} + \underbrace{\nabla g(I) \cdot \nabla u}_{\text{External force}}. \qquad (2.48)$$

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

$$g(I) = \begin{cases} \frac{1}{1+|\nabla I|*\alpha} & (1) \\[2ex] -|\nabla I| & (2) \end{cases} \tag{2.49}$$

In (1) $g(1)$ 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 it stops the curve from moving when the the pixel has reached an edge. $\alpha$ 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 definition of $g(I)$ will stop the curve from moving when it reaches an edge.

**Smoothing force**   The smoothing force is the same as the curvature term in the classical snakes 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.

$$g(I) \cdot |\nabla u| \cdot \text{div}\left(\frac{\nabla u}{|\nabla u|}\right) \tag{2.50}$$

Now the smoothing force is calculated with using an recursive call of SI and IS of operator F that approximates the curvature flow that is based on PDE.

$$F(u) = SI(u) \circ IS(u) \tag{2.51}$$

Where SI means supremum of infimum and IS means infimum of supremum. In other words, SI stands for erode and IS stands for dilate. One step of smoothing looks like 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 to 2 the smoothing force can be stronger or weaker. But it is important so always use step 1 and step 2 in the same order.

**Balloon force**   The balloon force is important in regions where the gradient is low and hinders the curve flow to get stuck at a local minimum. Expecially when looking at the external force, where the velocity of the curve is determined by the gradient of $g(I)$ and the gradient of $u$. Ifh $g(I)$ goes to zero the curve will have no external velocity, therefore the balloon force is important to make the algorithm more robust.

$$g(I) \cdot v \cdot |\nabla u| \tag{2.52}$$

To make GAC more computational efficient the ballon force and also the smoothing force are computed differently. The Balloon force is $g(I) * |\nabla u| * v$

with parameter $v \in -1, 1$ can be substituted with dilation over the four different neighborhood structuring elements. Because dilation is a morphological operation it is computed very efficient and also gets rid of the PDEs that need to be solved in the classical snakes method. By using dialate the balloon force leads to the wanted result by making the curve grow.

$$\text{dialate}(u) = u \bigoplus Element_i \quad \text{for} \quad i \in \{1, 2, 3, 4\} \tag{2.53}$$

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

$$\nabla g(I) \cdot \nabla u \tag{2.54}$$

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 the characteristic that the curve moves faster in areas where the gradient is low. This 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:

$$p = \begin{cases} 1 & iff \nabla u * \nabla g(I) > 0 \\ 0 & iff \nabla u * \nabla g(I) < 0 \end{cases} \tag{2.55}$$

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

### Convert GAC to ACWE

The GAC only inspects places where the curve could move iteratively and this is the main difference to ACWE. ACWE also known as Chan-Vese algorithm takes the complete image into consideration to evaluate the boundary or the curvature flow of the level set. Let $c_i$ the the mean intensity of an subset *Oomega* of the image $I$ and let $c_1$ be the mean intensity inside the curve or level set $u$ defined as $\Omega_1$ and $c_2$ the mean intensity outside level set $u$ defined as $\Omega_2$.

$$c_i = \frac{\int_{\Omega_i} I(x) \, dx}{\int_{\Omega_i} dx} \tag{2.56}$$

Then the external force function $F(c_1, c_2, C)$ can be defined as equation **??** where the first two terms $\mu \cdot \text{length}(u) + v \cdot |\text{inside}(u)|$ are the same as in the GAC The third and fourth terms sort of calculate the unscaled intensity

variance in the different subsets $\Omega_1$ and $\Omega_2$.

$$
\begin{aligned}
F(c_1, c_2, C) = & \mu \cdot \text{length}(u) + v \cdot |\text{inside}(u)| \\
& + \lambda_1 \cdot \int_{\Omega_1} (I(x) - c_1)^2 \, dx \\
& + \lambda_2 \cdot \int_{\Omega_2} (I(x) - c_2)^2 \, dx
\end{aligned}
\tag{2.57}
$$

Only difference now between the GAC and the ACWE is the external force that is calculated different. In GAC the external force is calculated with $\nabla g(I) \cdot \nabla u$ and considering if a point is inside or outside the curve based on a threshold as seen in equation 2.55. In ACWE the external force is calculated with the mean intensity of the subsets $\Omega_1$ and $\Omega_2$ and the intensity of the image $I(x)$ as seen in equation **??**. Translating onto an discrete grid the equation can be written as:

$$
\zeta = \underbrace{\lambda_1 \cdot |\nabla u_{mask}| \cdot (I(x) - c_1)^2}_{Inside} - \underbrace{\lambda_2 \cdot |\nabla u_{mask}| \cdot (I(x) - c_2)^2}_{outside}
\tag{2.58}
$$

$\zeta$ is then used to calculate the new level set function $u$ with the following equation:

$$
u(x, y) = \begin{cases} 1 \text{ if } \zeta < 0 \\ 0 \text{ if } \zeta > 0 \\ \text{do nothing otherwise} \end{cases}
\tag{2.59}
$$

The Gradient of the mask will only be nonzero at the contour of the mask. Now $I(x)$ is subtracted by the mean and each point on the contour will have two values, their inside and outside Intensities. By subtracting the mean intensity from the pixel intensities on the boundary of u, multiplying it with $\lambda_i$ and subtracting the outside term from the inside term an relationship between the inside and outside intensities is created and can be tuned with the sensitivity parameters $\lambda_i$. This relationship is then used to calculate the new level set function $u$ and the process is repeated until the level set function converges. In a case where the function reaches the object boundary the inside and outside intensities will be the same and the curve will not move anymore. Here is an example of the evolution the the level set u when the ACWE algorithm is used with a given starting point inside the pupil.

**Active contouring with ellipse parameters**

A different approach is to use the let an ellipse grow and modify it's parameters based on the gradient information. By sampling an amount of equally spaced points on the ellipse and evaluate an energy function based on the image intensity gradient and the normal vector of the ellipse at those points.
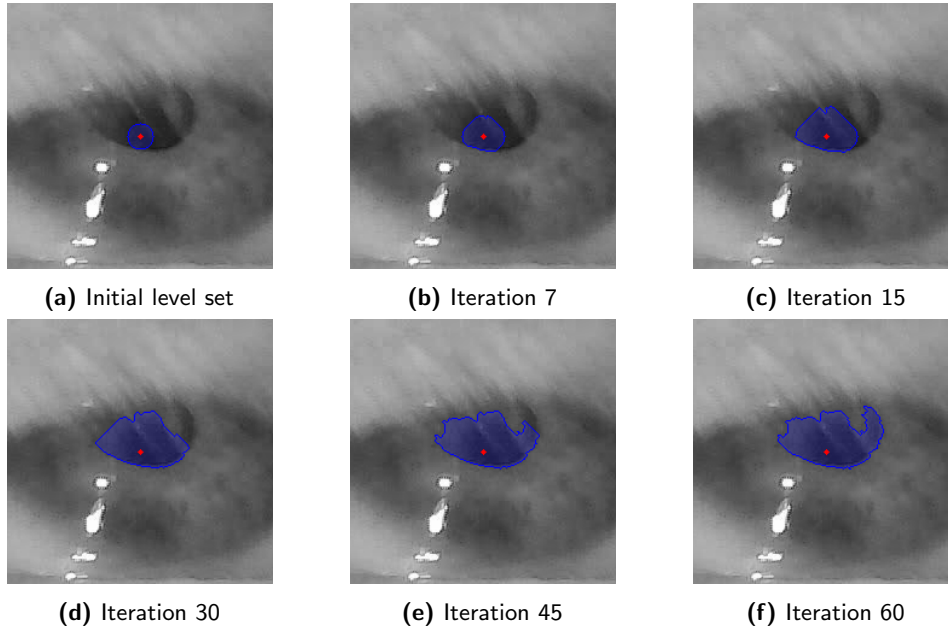
**(a)** Initial level set      **(b)** Iteration 7      **(c)** Iteration 15

**(d)** Iteration 30      **(e)** Iteration 45      **(f)** Iteration 60

**Figure 2.18:** ACWE iterations

The energy function can be described as the scalar product of the gradient and the normal vector of the ellipse curve.

$$E = - \sum_{i=0}^{N-1} \nabla I \cdot \vec{n} \tag{2.60}$$

this converts the problem into a minimization problem. The energy function still needs a term to make sure that 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 is based on only the five ellipse parameter: center, axis, angle.
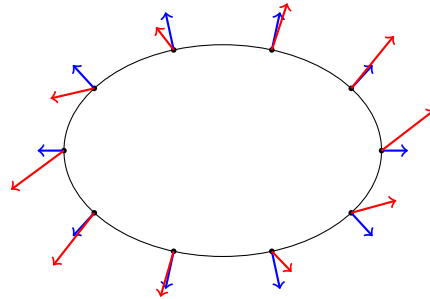


**Figure 2.19:** Ellipse with normal vector (blue) and gradient vectors (red)

The energy function is the lowest when at each point the gradient vector and

normal vector point in the same direction. The gradient vector consists of the magnitude of the gradient and the orientation. Whereas the normal vector is normalized and points in the direction of the normal of the ellipse at that point.

The problem with this approach is overlapping with the classical active contouring. The initial contour needs to be close to the optimal solution. Otherwise the algorithm stops at an local minimum and convergence to the optimal solution is not guaranteed. Because it is an minimization with five parameters it is a very complex problem and the algorithm needs to be run multiple times to find the best local solution. Also the image needs strong preprocessing and the algorithm is very sensitive to noise. Therefore it is a must to first apply a low pass filter onto the image. But because of blurring the image gradients and directions aren't as accurate as the could be and struggles with reflections and other noise. Here is an example with the convergence to an local minimum. The initial ellipse is nowhere near to the optimal solution and the algorithm converges to an local minimum.



**(a)** ROI



**(b)** Gradient Magnitude



**(c)** Gradient Direction



**(d)** Result
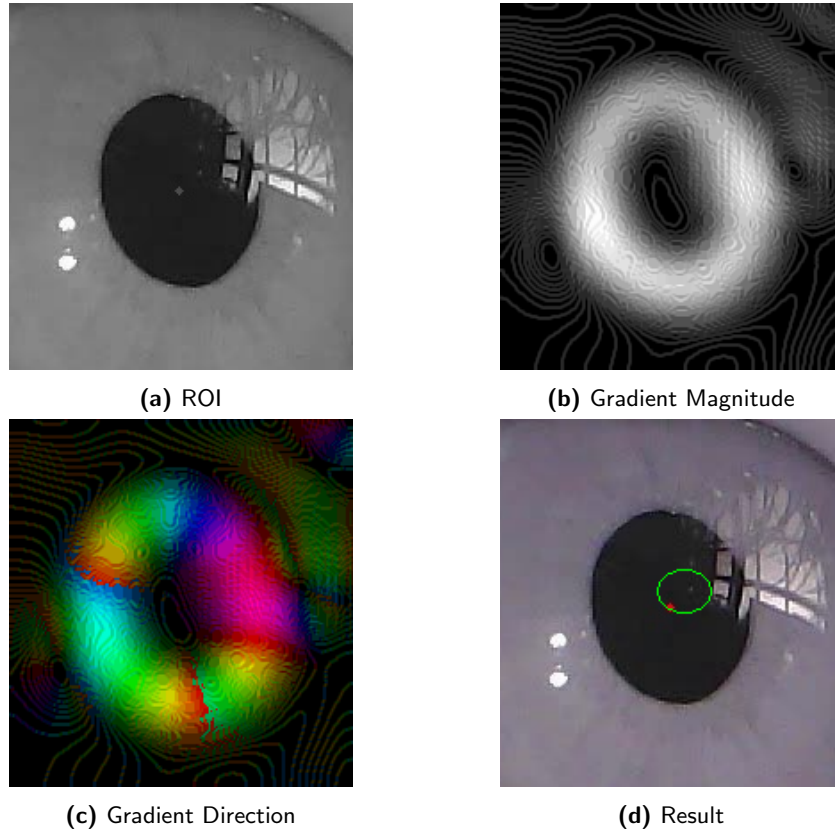
**Figure 2.20:** Convergence to local minimum

The problem with expanding the energy function with more terms is, that the algorithm gets more complex and for example the size is used as a reward for the energy function (the bigger the better), it is not given anymore to stop at the boundary of the object. The parameters have to be chosen carefully.

Appendix A

# Dummy Appendix

# Bibliography

[1] Tonsen Marc, Zhang Xucong, Sugano Yusuke, and Bulling Andreas. Labelled pupils in the wild (lpw): Pupil detection in unconstrained environments. `https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/labelled-pupils-in-the-wild-lpw`. Accessed: 28.04.2023.

[2] OpenCV. Contrast limited adaptive histogram equalization. `https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html`. Accessed: 28.04.2023.

# 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.*