Image-based extraction of eye features for robust eye tracking

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Wolfgang Fuhl

aus Freudenstadt

Tübingen
2019

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	29.03.2019
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Jun.-Prof. Dr. Enkelejda Kasneci
2. Berichterstatter:	Prof. Dr. Wolfgang Rosenstiel

Abstract

Eye-tracking technology is becoming increasingly important in current research and industrial applications as a means to study human behavior. Some examples include human-computer interaction, behavioral research, education and training, medicine, cognitive science, virtual/augmented reality and image processing. In all these areas, technological solutions are centered around the user. Currently available eye-tracking systems are built upon an image-based gaze estimation, which method consists of pupil detection, determination of the pupil center, and calibration for gaze estimation. Other important features are the eyelids and the degree of eye opening, which may provide additional information about a person's cognitive state.

In order for research to deliver reliable results and applications to be highly usable, the extraction of such features must be robust against a variety of challenges. These include the varying eye shapes, physiological differences of the pupil, make-up, reflections on spectacle lenses, changing light conditions, eyelashes that cover the eyes, birthmarks in the eye area, motion blur, and many others. Another very important challenge, especially in the field of applications, is the speed of eye movements. This understanding leads to the fact that high frame rates are crucial to capturing all types of eye movements for real-time estimation of the point of view, which only has a few milliseconds for calculation.

This thesis addresses the aforementioned challenges and introduces novel algorithms for real-time pupil detection as well as eyelid extraction both in a rule-based fashion and based on modern deep learning networks. Additionally, this thesis has focused on the automatic generation of new detectors and the fully automatic annotation of datasets based on machine learning approaches. Finally, new methods for data visualization, which are particularly helpful for eye-tracking data analysis, are considered and described. All algorithms developed during this thesis as well as large, accompanying annotated ground-truth data sets were made available to the research community in an open source fashion and have meanwhile proven their applicability in a variety of application areas.

Zusammenfassung

Die Eye-Tracking-Technologie gewinnt in der aktuellen Forschung und in industriellen Anwendungen zunehmend an Bedeutung als ein innovatives Instrument zur Erforschung des menschlichen Verhaltens. Einige prominente Anwendungsbeispiele liegen im Bereich der Human-Computer-Interaktion, in der Verhaltensforschung, in der Bildung, in der Medizin, in den Kognitionswissenschaften, in der Virtual/ Augmented Reality und in der Bildverarbeitung. In all diesen Bereichen wird Eye-Tracking zur Entwicklung nutzerzentrierter, technologischer Lösungen eingesetzt. Derzeit verfügbare Eye-Tracking-Systeme basieren auf einer bildbasierten Blickschätzung, die im Wesentlichen aus Pupillenerkennung, Bestimmung des Pupillenzentrums und Kalibrierung zur Blickschätzung besteht. Zusätzliche wichtige Merkmale sind dabei die Augenlider und der Grad der Augenöffnung, welche weiterführende Informationen über den kognitiven Zustand einer Person liefern können.
Damit Eye-Tracking basierte Forschung zuverlässige Ergebnisse liefert und Anwendungen hochgradig nutzbar sind, muss die Extraktion solcher Merkmale robust gegen eine Vielzahl von Herausforderungen sein. Dazu gehören beispielsweise die unterschiedlichen Augenformen, physiologische Unterschiede der Pupille, Make-up, Reflexionen an Brillengläsern, wechselnde Lichtverhältnisse, Verdeckungen durch Wimpern, Muttermale in der Augenpartie, Bewegungsunschärfe und vieles mehr. Eine weitere sehr wichtige Herausforderung, insbesondere im Anwendungsbereich, ist die Geschwindigkeit der Augenbewegungen, für deren Erfassung hohe Bildraten benötigt werden. Dies führt jedoch dazu, dass die bildbasierte Abschätzung der Blickinformation innerhalb von wenigen Millisekunden erfolgen muss.
Diese Dissertation widmet sich den oben genannten Herausforderungen und stellt neue, echtzeitfähige Algorithmen für die Pupillenerkennung sowie die Lidextraktion, sowohl regelbasiert als auch auf der Grundlage moderner Deep-Learning-Netzwerke, vor. Darüber hinaus beschäftigt sich diese Arbeit mit der automatischen Generierung neuer Detektoren und der vollautomatischen Annotation von Datensätzen auf Basis von maschinellen Lernansätzen. Schließlich werden neue Methoden der Datenvisualisierung betrachtet und beschrieben, die insbesondere für die Eye-Tracking-Datenanalyse hilfreich sein können. Alle in dieser Arbeit entwickelten Algorithmen, sowie große, begleitende, annotierte Ground-Truth-Datensätze, sind der Forschungsgemeinschaft auf Open-Source-Basis zur Verfügung gestellt worden und haben inzwischen ihre Anwendbarkeit in einer Vielzahl von Anwendungsbereichen bewiesen.

Contents

Table of Contents v

List of Figures ix

List of Tables xv

1 Introduction 1
- 1.1 Scope and contribution of this thesis . . . 3
- 1.2 Organisation of this thesis . . . 4

2 Fundamentals 7
- 2.1 Eye . . . 7
- 2.2 Eye movements . . . 8
  - 2.2.1 Fixations . . . 8
  - 2.2.2 Saccades . . . 9
  - 2.2.3 Smooth pursuits . . . 9
- 2.3 Eye-tracking technology and the measurement of eye movements . . . 9
  - 2.3.1 Head mounted eye-tracking technology . . . 9
  - 2.3.2 Remote eye-tracking technology . . . 11
- 2.4 Least squares fitting . . . 12
  - 2.4.1 Linear . . . 13
  - 2.4.2 Polynomial fit . . . 13
  - 2.4.3 Circle, ellipse fit . . . 14
- 2.5 Bezier splines . . . 15
- 2.6 Coordinate systems . . . 17
  - 2.6.1 Polar or angular coordinates . . . 17
  - 2.6.2 Barycentric coordinates . . . 18
- 2.7 Feature extraction . . . 19
  - 2.7.1 Convolution . . . 19
  - 2.7.2 Edge detection . . . 21
  - 2.7.3 Histogram of oriented gradients (HOG) . . . 22
- 2.8 Graph construction . . . 23
  - 2.8.1 Delaunay triangulation . . . 23
  - 2.8.2 Voronoi diagram . . . 23
- 2.9 Machine learning concepts . . . 24
  - 2.9.1 Support vector machine (SVM) . . . 26
  - 2.9.2 Artificial neural networks (ANN) . . . 29
  - 2.9.3 Convolutional neural network (CNN) . . . 32

3 Pupil detection **37**
3.1 State-of-the-art pupil detection for head mounted eye tracking 37
3.1.1 Starburst 37
3.1.2 Świrski 38
3.1.3 SET 39
3.2 State-of-the-art pupil detection for remote eye tracking 39
3.2.1 Droege and Paulus 39
3.2.2 Timm and Barth 39
3.2.3 George and Routray 40
3.3 Decision-based methods for pupil detection 40
3.3.1 ExCuSe 40
3.3.2 ElSe 46
3.3.3 Pupil detection on microscopy images 56
3.3.4 Pupil detection methodology 57
3.4 PupilNet: Pupil detection based on CNNs 64
3.4.1 Coarse positioning stage 65
3.4.2 Fine positioning stage 66
3.4.3 CNN training methodology 67
3.5 Evaluation of pupil detection algorithms on head mounted eye-tracking images 68
3.5.1 Data sets 68
3.5.2 Experimental results 69
3.6 Evaluation of pupil detection algorithms on remote eye-tracking images . 75
3.6.1 Data sets 75
3.6.2 Evaluation procedure 76
3.6.3 Results 76
3.7 Evaluation on dirt simulation images 78
3.7.1 Data sets 78
3.7.2 Results 80
3.8 Evaluation of pupil detection algorithms for microscopy images 81
3.8.1 Data sets 81
3.8.2 Results 82
3.9 Conclusion 84

4 Eyelid detection **85**
4.1 State-of-the-art algorithms for eye lid detection 85
4.2 Methods 86
4.2.1 Gradient map refinement method 86
4.2.2 Optimal oriented edge response method 91
4.3 Evaluation 96
4.3.1 Data sets 97
4.3.2 Results 97
4.4 Conclusion 98

5 Transferlearning for pupil and eyelid detection **99**
5.1 State-of-the-art algorithms for transfer learning . . . 99
5.1.1 Inductive learning . . . 99
5.1.2 Transductive learning . . . 100
5.1.3 Unsupervised learning . . . 101
5.2 General idea of MAM . . . 101
5.2.1 The MAM Algorithm . . . 103
5.3 Evaluation . . . 106
5.3.1 Remote driving datasets . . . 106
5.3.2 Evaluation without grid of detectors . . . 107
5.3.3 Evaluation with grid (3×3) of detectors . . . 108
5.3.4 Limitations . . . 111
5.4 EyeLad: Eye-tracking data annotation tool . . . 112
5.5 Conclusion . . . 114

6 Visualization of eye-tracking data **115**
6.1 State-of-the-art for automated AOI generation from eye-tracking data . . . 115
6.2 Data-driven methods for ROI generation . . . 117
6.2.1 Threshold-based ROI algorithm . . . 117
6.2.2 Gradient-based ROI algorithm . . . 119
6.2.3 Overlap clustering . . . 120
6.3 ROI experiments and method comparison . . . 124
6.3.1 Automatic vs. annotated statistics . . . 124
6.3.2 ROI generation in a classification task . . . 126
6.3.3 Discussion . . . 128
6.4 Saliency-based ROI generation . . . 131
6.4.1 Evaluation procedure . . . 133
6.4.2 Results of the saliency based ROI evaluation . . . 134
6.5 Conclusion . . . 136

7 All in practice: Gaze-based focus adaption **137**
7.1 Related work . . . 137
7.2 Hardware setup . . . 139
7.3 Method . . . 140
7.3.1 Focus measurement . . . 142
7.3.2 Graph representation . . . 142
7.3.3 Node correspondence collection . . . 144
7.3.4 Interpolation . . . 145
7.3.5 Rebuild graph . . . 146
7.3.6 Depth map creation . . . 146
7.4 Evaluation . . . 148
7.4.1 Data sets . . . 148
7.4.2 Results . . . 149
7.4.3 Algorithm evaluation . . . 150

7.4.4 Best index evaluation 151
7.5 Conclusion 153

8 Conclusion **155**

Bibliography **157**

List of Figures

1.1 Examples of commercial eye-tracker technology by Ergoneers (a) [52], Tobii (b,c) [237] and Smart Eye (d) [219]. . . . 1
1.2 Common challenges in video based eye tracking in real world environments. 2

2.2 An overview of eye tracking technologies. . . . 9
2.3 The functionality of a modern head mounted eye tracker. . . . 11
2.4 Tasks and challenges for remote eye tracking. . . . 12
2.5 Visualization of different Bezier curves using one, two and four additional points to define the curve. . . . 16
2.6 Different Bezier splines for a given set of points (circles). . . . 17
2.7 Polar or angular coordinates. . . . 17
2.8 Three different polygons, where the color is interpolated between the edge points using Barycentric coordinates. . . . 18
2.9 The difference between convolution and correlation. . . . 20
2.10 The input image (a), the gradient in x (b) and y (c) direction, the angle (d) and the length (e). . . . 21
2.11 Canny edge detection. . . . 21
2.12 Shows the input image (a) with HOG features computed on different cell sizes (b) and (c). The difference between (c) and (d) is a smaller set of orientation bins for (d). . . . 22
2.13 Delaunay triangulation from a set of points. . . . 24
2.14 Delaunay triangulation (green), Voronoi diagram (blue), and data points(red cross). . . . 24
2.15 Shows red and green points that represent two classes in the feature space. The black line is a hyperplane separating both classes. . . . 26
2.16 Shows the margin (dashed lines) between samples from two classes (red and green dots) and the hyperplane (solid line). . . . 27
2.17 SVM feature dimension extension. . . . 28
2.18 An example of linearly separable data with two slack variables $e1$ and $e2$. 29
2.19 Explanation of artificial neural networks. . . . 30
2.20 (a) the influence of the weights and (n) the impact of the bias term. . . . 31
2.21 Illustrative example of gradient decent and algorithmic challenges. . . . 32
2.22 (a) a convolution example for one filter and (b) a pooling operation. . . . 33
2.23 Work flow of a CNN with a convolution layer consisting of four filters and a subsequent pooling operation. . . . 33
2.24 Convolution operation for consecutive convolution layers. . . . 34

3.1 The Starburst algorithm. . . . 38
3.2 The Świrski algorithm. . . . 38
3.3 The SET algorithm. . . . 39
3.4 The algorithmic work-flow in ExCuSe. . . . 40
3.5 Intensity histogram analysis. . . . 41
3.6 Edge filtering, thinning and breakup. . . . 41
3.7 Morphologic pixel manipulation patterns. . . . 42
3.8 Coarse positioning with the results of the AIPF calculated on the threshold image. . . . 44
3.9 Edge refinement using rays from the coarse position. . . . 46
3.10 Flowchart of the algorithm. Light gray boxes represent decisions, dark gray ellipses termination points, and white boxes represent processing steps. [79] 47
3.11 Morphologic patterns for edge manipulation. . . . 48
3.12 Algorithmic edge break up algorithm. . . . 49
3.13 Curve selection of ELSE. . . . 49
3.14 Calculation of the difference between the inner and outer area of an ellipse [79]. . . . 51
3.15 Downscaling operation by a factor of six using the mean between zero and the mean of the input image. . . . 52
3.16 Neighborhood regions of pixels close to each other in the downscaled image. 53
3.17 Mean and surface difference filter. . . . 54
3.18 Workflow of the coarse positioning. . . . 54
3.19 Position optimization using a threshold. . . . 55
3.20 Control inputs from the Opmi Pentero 900 and 800 operating microscopes (source Carl Zeiss AG https://www.zeiss.com) [85]. . . . 56
3.21 Images from remote, head-mounted, and microscope-integrated eye tracking. 57
3.22 Pupil monitoring and imaging system for a surgical microscope ocular [85]. 58
3.23 The algorithmic workflow for pupil center detection. Rounded boxes represent processing steps and squared boxes are the input and output. [85] . 58
3.24 (a) shows the input grayscale image (recorded from the imaging system in Figure 3.22). In (b) the image after preprocessing is shown [85]. . . . 59
3.25 Edge filtering in multiple resolutions. . . . 59
3.26 Workflow of the circle search. . . . 60
3.27 All remaining pupil center candidates (red dots) after outliers removal [85]. 60
3.28 Recalculation of the edge value outgoing from a possible pupil center. . . 61
3.29 Radial image extraction. . . . 62
3.30 Edge detection in the radial image. . . . 63
3.31 Radius selection from a histogram. . . . 63
3.32 Workflow of PupilNet. . . . 64
3.33 The downscaled image is first divided in subregions (24×24) with a stride of one pixel (a), which are then rated by the first stage CNN (b) [83]. . . . 65
3.34 The coarse position stage CNN. . . . 66
3.35 The workflow of the accuracy improvement. . . . 68
3.36 Example images of the data sets provided by [231],[66], [79], [83] and, [240]. 72

3.37 Detection rates of the algorithms ElSe, ExCuSe, PupilNet, SET, Starburst, and Świrski for each of the data sets described in Table 3.1. 74
3.38 Example images from the BioID data set [73]. 75
3.39 Example images from the GI4E data set [73]. 75
3.40 Example images from the data set [73]. The left two images are grayscale converted RGB images. The remaining ones are infrared recorded [73]. . 76
3.41 Challenges posed in the data set [73] [73]. 76
3.42 Proportion of correctly detected eye regions by means of the Haar Cascade classifier togther with the KLT tracking [73]. 77
3.43 Euclidean distance in pixels and normed with the eye box diagonal. . . . 78
3.44 Example images selected from the respective data sets published by [66], [74]. 79
3.45 Simulation results for different focal lengths on one image. 200 particle of size group 2 were inserted [74]. 79
3.46 Simulation results for different particle size groups on one image. The particle amount is set to 200 and the focal length is 5.6 [74]. 79
3.47 Simulation results for different amounts of particles on one image. The particle size group is set to 2 and the focal length is 5.6 [74]. 80
3.48 Results on all data sets without dust simulation. 80
3.49 Results of all algorithms for different focal length (2.8, 4.0 and 5.6), changing amount of dirt particles (50 to 500) and altering size groups (1 to 6). . 81
3.50 Challenges for pupil center detection arise in the data sets. The green line in the images below show the pupil border [85]. 82
3.51 Results for all evaluated algorithms on the microscope data sets. 83

4.1 Some of the challenges caused by the eyelids, such as occlusion and motion blur. 85
4.2 Downscaling window size (on the top) and stride (on the bottom) – not in scale in relation to each other [81]. 87
4.3 Smoothed mean horizontal intensity values distribution (b), from which local maxima and minima (c) are identified. 88
4.4 Function performed by each stage in the eyelid detection algorithm – normalized per image [81]. 89
4.5 Graphical representation of edge selection metrics for an edge pair. 90
4.6 Edge (E), upper and lower eyelid Bézier curves, and the resulting ellipsis with the aperture estimation (minor axis, in cyan) [81]. 90
4.7 General overview of the algorithm work flow [67]. 91
4.8 The histogram of horizontally oriented edge values. 92
4.9 Overlay showing the first (and wrong) selected position, which violates the method's assumptions. 93
4.10 The lower eyelid approximation procedure. 94
4.11 Second outliers removal step illustration. 95
4.12 Upper eyelid approximation procedure. 95
4.13 Labels for eye corners (red) and eyelid points (blue) labels. 97

4.14 Examples from the data sets [67]. 97
4.15 Overall results in terms of outline similarity, eyelid aperture estimation, and cumulative detection rate [67]. 98

5.1 MAM tries to extend its knowledge of the object. 102
5.2 Workflow of the MAM algorithm. 104
5.3 Subset of challenges which arise in pupil center detection. Deformations, reflections, motion blur, nearly closed eyes and contact lenses are shown. 105
5.4 Exemplary images of the dataset from [70], where two consecutive pictures represent the same subject. 106
5.5 Exemplarily eyelid and pupil annotations. 107
5.6 Exemplary eye detections that are valid but not annotated in the data set. . 107
5.7 Points used for eyelid evaluation. One marks the left eye corner, two the upper eyelid center, three the right eye corner and four the lower eyelid center. 110
5.8 The graphical user interface of our labeling tool. 112
5.9 Each row shows the original input image (first column) and normalized images (second and third columns) [80]. 113
5.10 The three kinds of features used for tracking. 114

6.1 Workflow of the threshold based ROI algorithm. 118
6.2 Calculation of the cutoff threshold for one local maximum. 118
6.3 Workflow of the gradient based ROI algorithm. 119
6.4 Heatmap density in (a) and its first derivative, the gradient, in (b). 120
6.5 Overlap clustering procedure. 121
6.6 Fixations are the outlines of ellipses, whereas the dots represent the gaze points. Dots and ellipses of the same color belong together. 122
6.7 Overlap clustering example. 123
6.8 Visual comparison of ROI algorithms. 124
6.9 Comparison of ROI algorithms for fixations and gaze points. 124
6.10 Shows the used clusters for experiments in 6.3.2. 127
6.11 Exemplary weak and strong point visualizations for the different methods. 128
6.12 Jackson Pollock's "Convergence" with cumulative clusters (red ellipses). . 129
6.13 Jacopo Tintoretto's "The Last Supper" with cumulative clusters calculated using the overlap clustering. 130
6.14 Shows the clear difference between experts and novices for images (e) and (f) from Figure 6.13. R1 and R2 are the region identifiers [77]. 130
6.15 ROIs computed on different saliency maps. 132
6.16 ROIs computed on saliency maps in comparison to ROI results extracted from heatmaps. 133
6.17 Whisker plot over all feature combinations S_{1-16}. 136

7.1 The system for image recording consisting of a digital camera (XIMEA mq013mge2) and an optotune lens (el1030). 139

7.2 The GUI of the system. 140
7.3 The algorithmic workflow. 140
7.4 Canny edge based in focus estimation for one input image 7.4a. 141
7.5 Maximum responses in the set of images. 142
7.6 Maximum magnitude responses (7.6a) and the assigned depth index (7.6b) in the set of images. 143
7.7 In 7.7a the nodes which have an equal or larger magnitude value compared to their connected neighbors in G_{all} are shown. 144
7.8 The graph build on G_{max} using Delaunay triangulation [82]. 146
7.9 Interpolation using barycentric coordinates. 147
7.10 Computed depth map and 3D model. 148
7.11 Shows all objects used to generate the data sets. Below each object image stands the title which will be used further in this document [82]. 148
7.12 The results on all data sets from [82] are shown. 149
7.13 The results on the data sets alley [158], balcony [158], shelf [158] and zeromotion [228] are shown. 150
7.14 The results for the data set tin. 151
7.15 The results for the data set lego steps. 152
7.16 The results for the data set tape steps. 152
7.17 The results for the data set plastic tower. 152

List of Tables

3.1 Five publicly available data sets containing 266,781 ground-truth eye images were employed for the evaluation of pupil detection algorithms. . . . 71
3.2 Performance comparison of SET, Starburst, Świrski, ExCuSe, ElSe and PupilNet. . . . 73
3.3 The eye region resolutions in pixel for all data sets including the detected eye boxes [73]. . . . 76
3.4 Detection rate of all algorithms with a relative error of 20% [73]. . . . 77
3.5 The 5 pixel error detection rate on the original, non-modified data sets for all algorithms. . . . 80

5.1 Overview of the eye detection results for each data set with preliminary annotated images or a used detector in advance. . . . 108
5.2 Overview of the head mounted pupil center detection results for each individual data set. . . . 109
5.3 Overview of the remote pupil center detection results for each individual data set. . . . 110
5.4 Overview of the remote eyelid point detection results for each individual subject. . . . 111
5.5 Overview of the runtime of the algorithms for each experiment. For the method, the resulting detector as grid of five detectors were evaluated [70]. 111

6.1 The parameters used to calculate the images are shown in Figure 6.9. Letters behind the method name refer to Figure 6.9 [77]. . . . 125
6.2 Averaged statistic results over all nine subjects (mean gaze position of both eyes) for generated ROIs and annotated ones. . . . 125
6.3 Area similarity of the generated ROI and the manually annotated one using the formula $\frac{A \cap B}{A \cup B}$. . . . 126
6.4 The parameters used to calculate the images shown in Figure 6.10. Letters behind the method name refer to Figure 6.10 [77]. . . . 128
6.5 The classification results for all ROI generation algorithms with different kernels [77]. . . . 128
6.6 Advantages and disadvantages for each implemented method [77]. . . . 129
6.7 Maximal classification result per k for each ROI generation algorithm (using all ROIs) [78]. . . . 134
6.8 The calculated score (mean) for each method using Equation 6.3 (using all ROIs) [78]. . . . 135

6.9 The calculated standard deviation for each method using Equation 6.4 (using all ROIs) [78]. 135

List of Abbreviations

3D EMoSIFT	3D enhanced motion scale-invariant feature transform
AIPF	Angular integral projection function
ANN	Artificial neural network
AOI	Area of interest
CNN	Convolutional neural network
CRF	Conditional random fields
dur	Duration
ElSe	Ellipse selector
EOG	lectro-OculoGraphy
ExCuSe	Exclusive curve selector
Feat	Feature
Fix	Fixation
GLVM	Modified gray level variance
GP	Gaze point
HOG	Histogram of oriented gradients
IPF	Integral projection function
KLT tracker	Kanade–Lucas–Tomasi feature tracker
LAP3D	Laplacian in 3D window
LSDA	Large scale detection through adaptation
MAM	Multiple annotation maturation
MicPup	Microscope pupil detection
MLAP	Modified Laplacian
MoSIFT	Motion scale-invariant feature transform
MSER	Maximally stable extremal regions
NN	Neural network
PupilNet	Pupil center detection convolutional neural network
RANSAC	Random sample consensus
ROI	Region of interest
Sac	Saccade
SDP	Semidefinite programming
SIFT	Scale-invariant feature transform
SURF	Speeded up robust features
SV	Support vector
SVM	Support vector machine
VarDepth	Variational depth
WAVR	Ratio of wavelet coefficients
WAVV	Variance of wavelets coefficients

1 Introduction

In many research and application areas, eye movements are considered a rich source of information about the user. Therefore, the eye movement signal is used to create a deeper understanding of human cognition; answering questions from psychology, medicine, marketing research, advertisement, application control, only to mention a few. In these fields, different types of technologies have been developed to capture the gaze of users. These, eye-tracking technologies are separated into two main categories, head-mounted and the remote eye tracking.



(a) (b) (c) (d)

Figure 1.1: Examples of commercial eye-tracker technology by Ergoneers (a) [52], Tobii (b,c) [237] and Smart Eye (d) [219].

Figure 1.1 shows examples of two head-mounted eye-trackers from Ergoneers [52] (a) and Tobii [237] (b). Head-mounted eye trackers consist of two or more cameras recording the subject's eyes and the field of view. One of the main advantages of such eye trackers is that they can be attatched to the head, and incorporate the user's head movements. In contrast to head-mounted eye tracking, remote eye-tracker consists of at least one stationary camera (e.g. Figure 1.1 [219], [237](c,d)). Remote eye-tracking technology, which consists of one or more cameras recording the entire environment, is usually employed in settings that require an unrestricted field of view of the subject's, such as driving. Here additional challenges arise due to the detection of the subject's head and eyes. Several companies such as SMI, Tobii, Ergoneers, Pupil Labs and others build specialized hardware and develop algorithms for both types of technologies continuously. However, towards a broadly available and affordable eye-tracking technology, fundamental challenges still need to be resolved. Video-based gaze estimation consists essentially of feature extraction (face, eye, and pupil detection), the determination of the pupil center, and a calibration step. The calibration procedure determines a mapping between the position of the pupil center and the position in the subject's field of view.
In order for the technology to be deployable, the gaze signal must be robust against a variety of challenges. These include robustness against different eye shapes, physiological differences of the pupil, make-up, reflections on spectacle lenses, changing light conditions, eyelashes that cover the eyes, birthmarks in the eye area, motion blur and many others

(Figure 1.2). Another very important challenge, especially in the field of application, is the speed of eye movements. Therefore, high frame rates are necessary to capture all types of eye movements, and the point of view must be calculated within very few milliseconds.

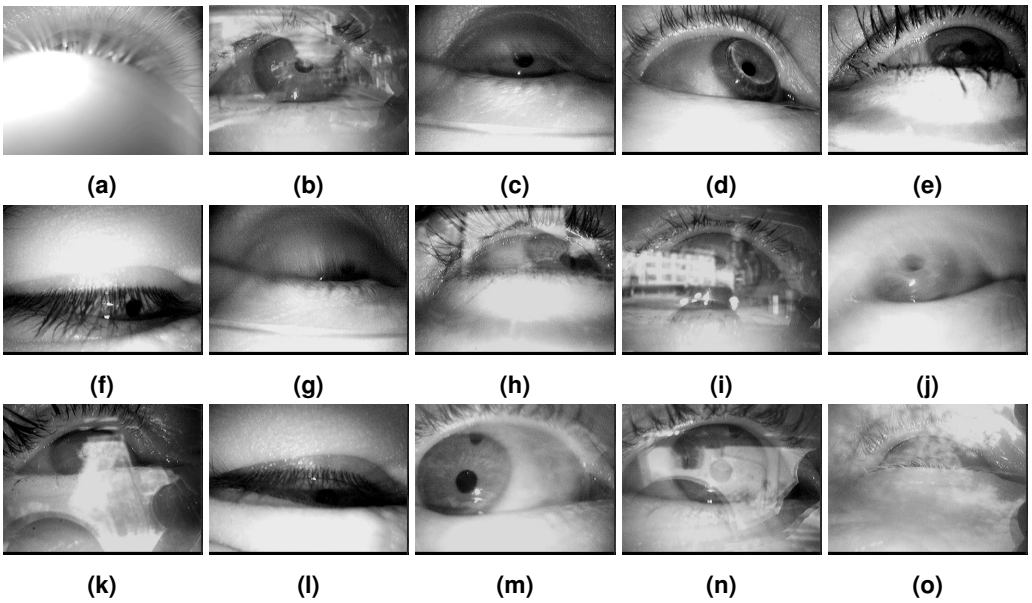


Figure 1.2: Common challenges in video based eye tracking in real world environments.

Schnipke and Todd [209] reported several difficulties arising in eye-tracking applications, e.g., changing illumination conditions, the intersection of eyelashes with the image of the pupil, glasses, etc. (Figure 1.2). Frequent illumination changes are often caused by ego-motion, which is especially the case while driving or moving fast. Another scenario for rapid illumination changes are night rides in cities: such as flickering environment lighting or moving light sources like car headlights. Reflection in the near infrared spectrum, which is commonly employed, is caused by ambient light sources and can be difficult to extract. These reflections are on either the cornea of the eye or the glasses which are worn (e.g. Figure 1.2(i,n)). Additionally, the placement of the camera also influences the quality of feature extraction. It should be placed highly off-axial to not limit the user's field of view. While several of the aforementioned problems have been solved under laboratory conditions [95], [122], [136], [138], [143], [180], [231], [243], [269], studies employing eye tracking in real-world scenarios regularly report low pupil detection rates [118], [119], [140], [217], [224], [241]. Therefore, the data has to be processed post-experimentally which is a laborious task and impossible in the application scenario. Additionally, in order to evaluate the cognitive state of a person, higher information extraction steps are required, such as eye movement detection [117], [204] and human visual exploration behavior evaluation [118], [126]. These steps reveal the information necessary to estimate a person's mental state but rely on high quality and robust feature extraction. They can be seen as a chain through which the error propagates and from which critical misjudgments are made. Besides the gaze signal, other eye-related features can be beneficial to infer the user's cognitive state.

Eyelids, for example, serve as protection and maintenance system, hindering particles from reaching the eye and limiting the amount of light entering the pupil [61]. Furthermore, blink frequency, eye opening and, eye closure velocity hold information about the vigilance, fatigue, and, drowsiness [154], [229], [263] of a person.

1.1 Scope and contribution of this thesis

First, three methods for pupil detection and pupil center determination are presented, all of which meet the real-time condition on a conventional CPU. Two of these methods are rule-based, while the third is based on convolutional neural networks (CNN) [66], [79], [86]. Each of these procedures was published together with a data set to further advance research in the field of image-based pupil recognition. Two of these algorithms can be used for head-mounted and remote eye-tracking [73]. In a further publication, these methods were evaluated under very challenging conditions with simulated dirt on lenses or eye glasses [74]. This challenge is especially important for individuals who wear glasses as well as for permanently installed cameras that are not cleaned regularly. A rule-based algorithm is also presented for pupil detection in a microscopy application [85]. Here, the eye was captured through the microscope's lens structure, which greatly enlarges the pupil. This expands the challenges because the pupil no longer has a smooth border, but jagged edges that overflow into the iris. Furthermore, it is not always possible to capture the entire pupil, meaning the center of the pupil is out of the image area. The presented algorithm was also published together with a data set in [85].
In addition, two methods were developed to determine the degree of eye opening and extract the eyelids in an automated fashion [67], [81]. These methods can be used for both remote and head-mounted eye tracking. The first algorithm is rule-based to meet the real-time conditions [81]. The second approach is based on an optimization procedure, which also fulfills the real-time conditions together with a coarse positioning [67]. Both algorithms were published along with manually annotated data sets.
The last detection approach presented in this thesis uses a very small set of annotated data (≈ 10 images) to automatically annotate entire videos [70]. The result of the algorithm is an annotated data set and real-time detectors for it. It uses an aging function to generate new training sets and a grid of detectors to compensate for displacements and deformations. The results of this algorithm exceed those from the state-of-the-art by a large margin. Again, a manually annotated data set was published along with the method. The motivation for this algorithm comes from the application multiplicity of the image based detection. Therefore, a method is needed that allows generating good detectors with only a small amount of annotated data. In addition, the shapes and characteristics of humans are different and the challenges in new application areas vary. For the labeling process, a special annotation software was developed which uses the previous methods to support the user [80].
Further presented methods from the field of applications include an algorithm for real-time autofocus determination for surgical microscopes and algorithms to assist in the analysis of eye-tracking data. Our Auto Focus algorithm uses a fluid lens to take images for different focal lengths within a few milliseconds and then creates a depth map [82]. The algorithm

also marks areas where the depth cannot be determined. An additional contribution of this thesis was in the area of eye-tracking data analysis and visualization, where several algorithms for automatic calculation of important areas are presented [76]–[78]. These areas are used for extracting statistics for classification and for visual analysis of eye-tracking data. An extension of this procedure with saliency maps is also presented.

As mentioned above, the research presented in this thesis has been published in renowned journals and conferences. In the area of eye movement analysis research, works were published in *Acta Neurochirurgica* [49], *International Conference on Intelligent User Interfaces* [48], [50], *Vision Sciences Society Annual Meeting* [11], *Internationales Stuttgarter Symposium Automobil- und Motorentechnik* [25], *Communications in Computer and Information Science* [128]. An eye velocity simulator together with automatic detector creation was published in *Modeling Cognitive Processes from Multimodal Data* [68], [69], [84] and *Egocentric Perception, Interaction and Computing* [75]. Methods supporting eye movement analysis were published in the *European Conference on Eye Movements* [76] and the *European Conference on Computer Vision Workshop VISion for ART Analysis* [131]. Additional methods were published in *Eye Tracking and Visualization* [77] and in the *Symposium on Vision, Modeling and Visualization* [78]. Methods for pupil center detection were published in *Computers in Biology and Medicine* [85], in the *Conference on Pervasive and Ubiquitous Computing* [73], in the *ACM Symposium on Eye Tracking Research and Applications* [79], in the *Conference on Computer Analysis of Images and Patterns* [66] and a summerized evaluation in the journal of *Machine Vision and Applications* [86] and the *Journal of Eye Movement Research* [74]. Additional work was published in the *ACM Symposium on Eye Tracking Research and Applications* [71], [72]. Algorithms for eyelid detection were published in the *Conference on Pervasive and Ubiquitous Computing* [81] and the *Winter Conference on Applications of Computer Vision* [67]. A self adapting algorithm was submitted to *Egocentric Perception, Interaction and Computing (EPIC)* [70]. Methods for calibration of an eye tracker and eye movement extraction were published in the *Conference on Human Factors in Computing Systems (CHI)* [207] and the *ACM Symposium on Eye Tracking Research and Applications* [204]. Additional tools for research were published in *Imaging and Computer Graphics Theory and Applications* [80], [88], [206] and the *International Conference on Health Informatics* [218].

1.2 Organisation of this thesis

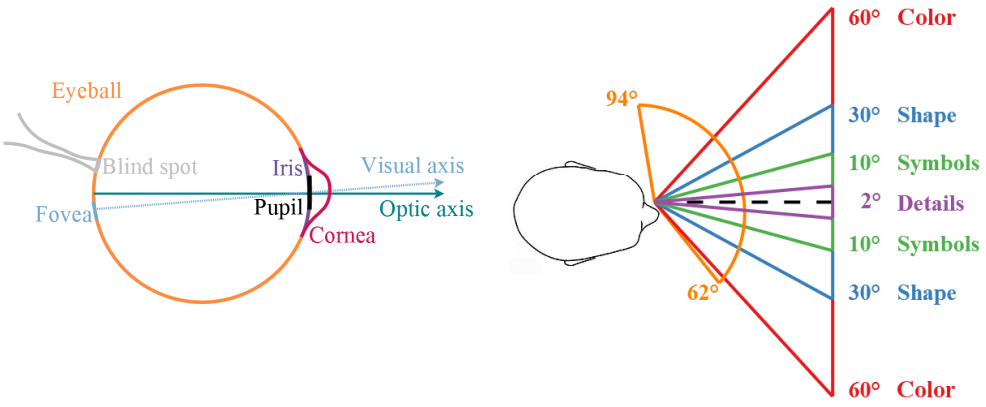
The remainder of this thesis is structured as follows. The second chapter explains fundamental concepts related to the physiology of the eye and to eye tracking. Afterward, fundamental concepts that are used in the algorithms in this thesis are explained. The remaining of the document is separated into four main chapters. Chapter 3 introduces our methods for pupil detection for both head-mounted and remote eye-tracking images. Additionally, an algorithm designed for a surgical microscope is presented. In the focus of Chapter 4 is a robust eyelid extraction for head-mounted eye trackers. Here a rule-based approach and an optimization formulation are presented. Chapter 5 presents an algorithm for detector creation and automatic data annotation. This approach is capable of perform-

ing pupil center detection as well as eyelid area extraction. Additionally, it is capable of labeling large amounts of data without human intervention and creates specialized detectors automatically. Chapter 6 presents algorithms that use the gaze signal for visualization and classification applications. Chapter 7 describes a real-time setup together with an algorithm for gaze-based focus control, while Chapter 8 concludes this thesis.

2 Fundamentals

This chapter describes the basic concepts related to the eye physiology and eye tracking as well as the methodological concepts necessary for the understanding of this work. Section 2.1 describes the structure of the human eye and basic concepts of vision. Based on this information, Section 2.2 deals with human eye movements, while Section 2.3 describes modern eye tracking technologies. Section 2.4 introduces the least squares fitting, which is used for eye tracker calibration (polynomials) as well as pupil outline determination (ellipse fitting). For the robust calculation of more complex shapes, Bezier splines are described in Section 2.5, which are used in this work for eyelid extraction. Section 2.6 describes different coordinate systems for calculating features and interpolating surfaces. Section 2.7 covers feature extraction methods, which are used in this thesis in combination with machine learning methods (Section 2.9) for detection and classification. In Section 2.8, procedures that enables graph creation from sets of points are presented; they help to reduce the computational effort.

2.1 Eye



(a) Visualization of the basic structure of the human eye adapted from [18].

(b) The visual field of a human for the left eye [18].

The eye is the sensory organ that converts light stimuli into electrical signals that enable us to perceive our environment. These stimuli are recorded with the help of photoreceptors [249]. These light-sensitive nerve cells change their state of excitation based on the different wavelengths of the light. A small wavelength range corresponds to a color. The area perceptible to humans is in the range of 400 to 700 nanometers (nm) [17], [249]. The range below 400nm is the ultraviolet light and above 700nm comes the infrared range [194].

Color perception is made possible by three types of nerve cells, which react to different wavelengths of light (short, medium and long). These three types correspond to the colors red, green and blue with the nerve cells for red light having their absorption maximum in the yellow-green range. Each of these neurons generates an electrical signal as soon as light from their wavelength range falls on them [249]. This signal is transmitted through the optic nerve (blind spot Figure 2.1a) to the visual centre of the brain. These electrical signals are processed into optical perception.
This optical perception consists not only of color, but also of sharpness and adaptation to different light conditions. The area of sharpest vision is in the fovea (Figure 2.1a). Here is the mostdense accumulation of photoreceptors which results in a high resolution of the visual perception [18]. Outgoing from the fovea, the number of photoreceptors and color perception is reduced towards the periphery of the visual field (Figure 2.1b).
Light enters the eyeball through the pupil (Figure 2.1a). The pupil is dilated or reduced in size by a surrounding ring muscle, which is covered by the iris. This muscle enables us to control the incident light and thus to adapt to different lighting conditions. Behind the pupil lies the lens, which projects the incident light on the retina. The center of the pupil together with the center of the eyeball forms the optical axis. Since the center of the sharpest vision is located in the fovea, which deviates about 5 degrees from this axis [17], [44], the real visual axis (fovea center to pupil center) is individually different. Another important component is the cornea over the pupil (Figure 2.1a), which not only bundles the light but also protects the pupil from particles to be removed by the eyelids from the cornea.

2.2 Eye movements

Due to the physiological properties of the eye, only a small area can be perceived sharply (Figure 2.1b). This limitation means the eye must move to capture whole scenes. The muscles controlling its movement follow a complex structure; they can rotate the eyeball around all three axes. The x and y axis represent a shift of the pupil vertically and horizontally. Rotating the eye around the z axis represents a rotation to compensate for small oblique positions of the head. The general categorization of eye movements are three groups, which are described in the following subsections according to [18]. In the following, we will only consider three eye movement types that are measurable by most eye-tracking devices.

2.2.1 Fixations

Time ranges where the eye does not move or moves minimally are called fixations. This enables the visual stimuli to be perceived. The duration of a fixation ranges from 100 to 600 miliseconds (ms) and depends on the dynamics of the scene, the light conditions, as well as the underlying interest of the person [17], [44]. An example of an underlying interest would be looking at a painting. Long fixations would fall on areas that appeal to the person, whereas areas with no salient objects for the person would only be briefly catch the eye. Another example are objects that are hard to recognize or partially occluded; here a person needs a long time to recognize the object. During a fixation, minimal eye movements also

occur, which are called microsaccades, drifts or microtremors [148]. Those movements are difficult to measure and go beyond the scope of this work.

2.2.2 Saccades

A movement between two fixations is called a saccade [17], [44]. Thus, a saccade is a fast eye movement, which locates the focus of the eye to a new position. The duration of a saccade is between 10ms and 100ms depending on the length [17], [44]. During a saccade, no visual stimuli is perceived [17], [44]. As a result, saccades have to be planned in advance and cannot be interrupted [44], [65], [213].

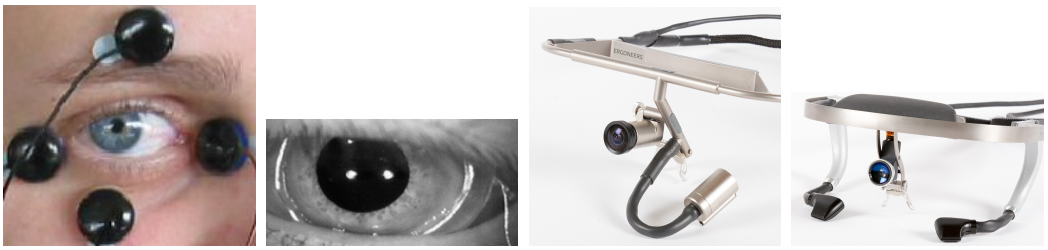
2.2.3 Smooth pursuits

When looking at a moving target, the respective eye movement id known as a smooth pursuit. This type of eye movement corresponds to pursuing the target visually [197]. The eye movement corresponds to the speed of the object, which suggests that our eye has a closed-loop feedback system [197]. This system corrects itself in case of speed changes. These smooth pursuits correspond to a combination of saccades and fixations.

2.3 Eye-tracking technology and the measurement of eye movements

Eye tracking refers to the determination of the viewing direction or viewing position. Since the discovery of the first usable eye tracker [266] many different approaches have been proposed. In general, a distinction is made between head-mounted and remote eye trackers. In this section, both types of eye-tracking technologies and challenges associated with gaze estimation are introduced.

2.3.1 Head mounted eye-tracking technology



(a) (b) (c) (d)

Figure 2.2: An overview of eye tracking technologies. In (a) Electro-OculoGraphy [156] is shown, where the black dots are the measurement units. (b) shows the scleral coil [109] which is placed below the eyelids surrounding as much as possible of the eyeball. (c) and (d) show video based eye trackers, where (c) only records one eye and (d) both eyes [51].

The main advantages of head-mounted eye trackers is that the head orientation does not need to be determined. In addition, all measurements can be done with less influence of the environment. Furthermore, supplementary steps such as detection of the person are not necessary. An overview of different techniques is shown in Figure 2.2. This work focuses only on video-based eye tracking but, for completeness, two further procedures are described below.

Electro-OculoGraphy

The Electro-OculoGraphy (EOG) is based on the measurement of changing electrical potentials on the skin. Electrical potential is measured with electrodes (Figure 2.2(a)) and is generated by muscle movement. Therefore, this method is only capable of measuring the torsion change to the eye. To estimate the gaze location of a person, an initial location has to be set, where changes are applied. This approach has two additional disadvantages. First, the head position has to be measured separately or a scene camera has to be added to the subject's head on which the gaze location is projected. The second disadvantage is that the error integrates over time: Wrong or inaccurate movement detections influence all subsequent estimations.

Scleral contact lens

Figure 2.2(b) shows the scleral coil or contact lens [196]. As can be assumed, it is very uncomfortable for the subject and also limits the time for eye tracking experiments. However, it is the most precise way of eye tracking [195]. The most used techniques are reflecting phosphors, line diagrams and wire coils. For the first two techniques, either an additional person is needed or an image processing based approach has to be applied, to measure the pupil center location. In comparison, The wire coil measures the position in a magnetic field, which is highly accurate and can be evaluated automatically. Therefore, an additional apparatus is necessary to generate the magnetic field surrounding the subject. This method is independent of head orientation as long as the area of the magnetic field is large enough.

Video-based eye-tracking

Most modern approaches use video cameras to record the subject's eyes and the scene location, since this method is non intrusive if the head frame is neglected. Two examples can be seen in Figure 2.2(c,d). In Figure 2.3, the functionality of a video based eye tracker is shown. The pupil center location of a person is projected into the image of the scene camera, which represents the gaze position. This mapping is possible due to the static locations of the camera recording the eye and scene. For the estimation task, an additional step is required to map a location from one image into the other. This mapping is done by computing a function which is usually a second order polynomial. Other approaches use machine learning techniques, such as neural networks or support vector machines, to learn this function [13], [149]. These methods are also capable of learning more complex functions; meaning they include the pupil center detection or, in this case, feature extraction.

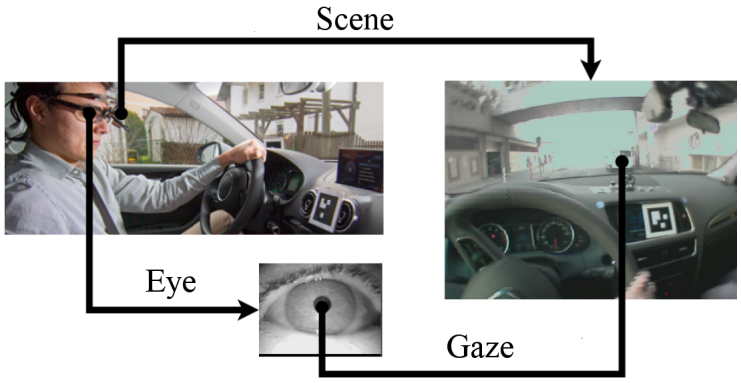


Figure 2.3: The functionality of a modern head mounted eye tracker.

This approach is called appearance-based gaze estimation [256]. In [232], a model based approach was proposed. For model based approaches, the eyeball is estimated based on the deformation of the pupil in relation to its position in the eye image. Afterwards, this model can be used to compute the gaze location. Newer approaches use multiple low resolution cameras to record the eye from different angles [239]. Based on those images, the gaze location is estimated using a preliminary calibration.

Correction for the fovea displacement

As can be seen from the illustration in Figure 2.1a, no approach can measure the gaze position accurately for all subjects [44], due to the visual and the optical axis displacement based on the fovea location. Therefore, all approaches have to perform a personal calibration or adaption to the subject. During calibration, images with the corresponding viewing position of the subject are collected and a function that describes this transition is calculated. In practice, the polynomial approach is used in most cases together with a personal calibration [44]. Therefore, the learned function includes the displacement correction together with the mapping. This function is learned by multiple relations of pupil center position to gaze location, which have to be collected before an experiment. Other approaches like the model-based or appearance-based approach only need one calibration point, where the subject has to attend to one spot moving his head slightly [170]. This procedure has to be done only once per subject and can be reloaded if the subject is using the eye tracker again. The one point calibration approach is also applied to the polynomial fitting with the difference that the subject has to move his or her head in a larger area and the fovea correction cannot be stored separately [170].

2.3.2 Remote eye-tracking technology

In remote eye tracking, the subject is recorded by one or more video cameras, which are usually placed at some distance to the subject, as shown in Figure 2.4. This approach includes additional challenges like the detection of the subject, the estimation of the head orientation and the computation of the gaze vector in relation to the camera's global coordinate system.

Additionally, the resolution of the pupil, eyes, faces, and other landmarks are much smaller, therefore the expected inaccuracy is higher.

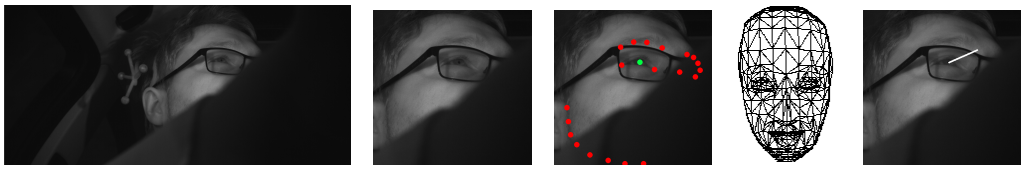


Figure 2.4: Tasks and challenges for remote eye tracking. In (a), a common scene for remote eye tracking is shown. (b) and (c) show the steps necessary to compute the gaze of a person (d) in relation to one camera.

Figure 2.4(a) shows a common scene for a remote eye tracker. It can be seen that occlusions and background distractors represent enormous challenges. The camera hardware itself can produce out of focus images, thus making it difficult to handle illumination distributions. In Figure 2.4(b), the first step of remote gaze estimation is shown, i.e. face detection. Afterwards, the landmarks (eye corners, pupil, mouth shape and face shape) are detected (Figure 2.4(c)). On those locations, a face model (Figure 2.4(d)) has to be mapped. This model in relation to the general frontal model yields a transformation that represents the head orientation. Based on this head orientation, the gaze is estimated using the pupil centers (Figure 2.4(e)). All of these steps result in a gaze vector in relation to a single camera. If multiple cameras are used, the gaze vector is transformed into the global coordinate system. In addition, multiple estimations can be used to correct each other and to compute a 3D model. Although this approach lacks accuracy, it is non intrusive to the subject and, together with infrared illumination, applicable at night. Gaze estimation can be done in real time using weak detectors for landmarks in combination with a global model of the face of a person [34], [121], [268]. The detection under real world scenarios is however still a challenging task, but still applicable in real time due to the combination of modern tracking methods and hardware. Other more simple approaches for desktop systems also exist. Here, the subject is recorded and glints (first Purkinje image) are projected into the eye [38]. Glints are infrared reflections on the cornea that are produced using directed infrared illumination. On start, the subject has to perform a calibration where glint center, pupil center, and gaze location relations are collected. Afterwards, a polynomial fitting method is applied.

2.4 Least squares fitting

Least Squares Fitting refers to the problem of estimating the parameters of a model (function $f(x)$) to a given set of data points [87]. The best fit has to minimize the sum of the squared residuals. For each data point, the residual is the difference between the estimated value of the model and the expected result which is given by the set of data the model is fitted to. These residuals form an error function, which is the sum of the residuals. To

form a continuous differential quantity, this sum must be squared. This squaring has the disadvantage that outliers in the collected data set can have a high impact on the result.

$$R^2 = \sum i = 1n(y_i - f(x_i))^2 \tag{2.1}$$

Equation 2.1 [87] shows the error function for least squares. x_i and y_i are the collected data samples and $f(x_i)$ is the estimation of the model. The parameters of this model can be computed by setting the gradient (first derivation) to zero. This differentiation leads to two types of least squares fitting approaches. The first is the linear least squares where the parameters of the model are linear dependent, which is the case for polynomials. In the second type, these parameters are non-linear dependent, which means they cannot be computed directly unless the equation is transformed to a linear problem, which has other disadvantages. Least squares fitting will be applied in this thesis for ellipse approximation and gaze estimation.

2.4.1 Linear

In the linear case, our model is $f(x_i) = u_1 + u_2 * x_i$, where u_i are the unknowns of the model that we want to compute. Inserted in Equation 2.1 yielding $R^2 = \sum i = 1n(y_i - (u_1 + u_2 * x_i))^2$,the error function for the line equation. The solution for u_1 and u_2 is where the gradient of this function is minimal.

$$\begin{array}{l} \frac{\partial R^2}{\partial u_1} = -2 * \sum_{i=1}^{n}(y_i - (u_1 + u_2 * x_i)) \\ \frac{\partial R^2}{\partial u_2} = -2 * \sum_{i=1}^{n}(y_i - (u_1 + u_2 * x_i)) * x_i \end{array} \tag{2.2}$$

The partial derivatives are shown in Equation 2.2 [87]. Multiplying out and setting this equation to zero results in Equation 2.3 [87].

$$\begin{array}{l} \sum_{i=1}^{n} y_i = n * u_1 + u_2 * \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} y_i * x_i = u_1 * \sum_{i=1}^{n} x_i + u_2 * \sum_{i=1}^{n} x_i^2 \end{array} \tag{2.3}$$

Equation 2.3 is the final linear equation system that has to be solved for u_1 and u_2 and can be directly computed out of the collected data samples. This system can be also written in matrix notation (Equation 2.4 [87]).

$$\underbrace{\begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} y_i * x_i \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} n + \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} x_i^2 \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}}_{U} \tag{2.4}$$

Now, the solution of $A = BU$ can be computed by the inverse of B multiplied with A ($U = B^{-1}A$) [87].

2.4.2 Polynomial fit

In comparison, the more general case to form the line is by polynomials. Here the model or function has the form $f(x) = \sum_{j=1}^{k+1} u_j * x^{j-1}$ [87]. The residual function is therefore given

by $R^2 = \sum i = 1n(y_i - (\sum_{j=1}^{k+1} u_j * x_i^{j-1}))^2$ [87]. The gradients of this function can again be written in matrix form (Equation 2.5 [87]).

$$\underbrace{\begin{pmatrix} \sum_{i=1}^{n} y_i \\ \vdots \\ \sum_{i=1}^{n} y_i * x_i^k \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} n & \sum_{i=1}^{n} x_i & \dots & \sum_{i=1}^{n} x_i^{k-1} & \sum_{i=1}^{n} x_i^k \\ & & \vdots & & \\ \sum_{i=1}^{n} x_i^k & \sum_{i=1}^{n} x_i^{k+1} & \dots & \sum_{i=1}^{n} x_i^{2k-1} & \sum_{i=1}^{n} x_i^{2k} \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} u_1 \\ \vdots \\ u_{k+1} \end{pmatrix}}_{U} \tag{2.5}$$

Therefore, the solution of Equation 2.5 [87] is given by $U = B^{-1}A$. A more elegant representation of this linear equation system is obtained by using the Vandermonde matrix of B. This representation is shown in Equation 2.6 [87] and represents the linear equation system which we obtain for the residual without squaring.

$$\underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}}_{Y} = \underbrace{\begin{pmatrix} 1 & x_1 & \dots & x_1^{k-1} & x_1^k \\ 1 & x_2 & \dots & x_2^{k-1} & x_2^k \\ & & \vdots & & \\ 1 & x_{n-1} & \dots & x_{n-1}^{k-1} & x_{n-1}^k \\ 1 & x_n & \dots & x_n^{k-1} & x_n^k \end{pmatrix}}_{V} \underbrace{\begin{pmatrix} u_1 \\ \vdots \\ u_{k+1} \end{pmatrix}}_{U} \tag{2.6}$$

Here the Vandermonde matrix is V. If we multiply this equation $Y = VU$ [87] with the transposed V^T we obtain $V^TY = V^TVU$ which is the same as Equation 2.5 [87]. The solution for this equation system is then written as $U = (V^TV)^{-1}V^TY$ [87].

2.4.3 Circle, ellipse fit

For circles and ellipses, there exist two general ways of applying the least squares fitting [87]. The direct approach can be applied to the algebraic representation [185]. For a circle, the general equation is $(x-x_c)^2 + (y-y_c)^2 = r^2$, where x_c, y_c is the center and r its radius. It can be rewritten as $-(x^2+y^2) + 2x * u_1 + 2y * u_2 + u_3 = 0$ where $u_1 = x_c$, $u_2 = y_c$ and $u_3 = r^2 - x_c^2 - y_c^2$ [185]. When writing this approach in the matrix form as done for the polynomials, we get Equation 2.7 [185].

$$\underbrace{\begin{pmatrix} x_1^2+y_1^2 \\ \vdots \\ x_n^2+y_n^2 \end{pmatrix}}_{Y} = \underbrace{\begin{pmatrix} 2x_1 & 2y_1 & 1 \\ & \vdots & \\ 2x_n & 2y_n & 1 \end{pmatrix}}_{V} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}}_{U} \tag{2.7}$$

As can be seen it is the least squares formulation, which can be solved by $U = (V^TV)^{-1}V^TY$ [185]. This equation has to be zero, but is not related to the geometric distance of each point. Therefore, we can not expect to obtain a good result for an overdetermined system, where not all points are on the outline of this circle. In practice, usually the mean of the data points is extracted first, which improves the result. There are other

more robust approaches such as [59], [185], which are not discussed here in detail. This example should only show the general idea behind the transformation of the circle equation. Although the result of Equation 2.7 is not robust, it delivers a good initial position for an iterative approach. In general, the residual function is $R^2 = \sum i = 1n(\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - r)^2$ [87], which is the geometric distance of the points. It leads to a non-linear least squares problem and can be solved by computing a correction vector in each iteration (Gauss-Newton method). Given an initial parameter vector u with x_c, y_c, r, we are able to compute a correction vector h using the Jacobian matrix (partial derivatives). Therefore, solve the linear equation system $f(u) + J(u)h \approx 0$ or $J(u)h \approx= -f(u)$ [87]. The update of u is then computed by $u_{i,t+1} = u_{i,t} + h_i$.

$$\underbrace{\begin{pmatrix} (\sqrt{(x_1 - u_{1,t})^2 + (y_1 - u_{2,t})^2} - u_{3,t})^2 \\ \vdots \\ (\sqrt{(x_n - u_{1,t})^2 + (y_n - u_{2,t})^2} - u_{3,t})^2 \end{pmatrix}}_{U} = \underbrace{\begin{pmatrix} \frac{u_{1,t} - x_1}{\sqrt{(x_1 - u_{1,t})^2 + (y_1 - u_{2,t})^2}} & \frac{u_{2,t} - y_1}{\sqrt{(x_1 - u_{1,t})^2 + (y_1 - u_{2,t})^2}} & -1 \\ \vdots & & \\ \frac{u_{1,t} - x_n}{\sqrt{(x_n - u_{1,t})^2 + (y_n - u_{2,t})^2}} & \frac{u_{2,t} - y_n}{\sqrt{(x_n - u_{1,t})^2 + (y_n - u_{2,t})^2}} & -1 \end{pmatrix}}_{J} \underbrace{\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}}_{H} \tag{2.8}$$

Equation 2.8 [87] is the linear equation system to solve, where J is the Jacobian matrix. Applying the transposed ($H = (J^T J)^{-1} J^T U$) assigns an equation to each h_i as before. This geometric distance fit can also be computed directly for circles and ellipses [184]. While in this section we only described the fitting for circles (since it is more descriptive), the same approaches apply for ellipses. In the case of ellipses, where the direct fitting equation is usually the conic section, it has to be mentioned that, in comparison to the geometric fit, it is also a capable of detecting non ellipses, e.g. when the result forms a hyperbola instead of an ellipsis.

2.5 Bezier splines

Splines are usually applied if it is required that the model or function to be computed contains the given a set of points P_j. Therefore, the values between those points have to be interpolated. The most popular function to compute an arbitrary curve between two points is the Bezier curve. The shape of this curve can be modified by changing the location of the support points B_i for $i \in [1, n-1]$ [21].

$$BC(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} * B_i \tag{2.9}$$

These support points define control polygons and the curve is computed using Equation 2.9 [21]. This function is defined for $t \in [0,1]$, and the support points B_0 and B_n are P_j and P_{j+1} respectively. The disadvantage of the Bezier curve is that the support points have to be set manually. Figure 2.5 shows examples of Bezier curves. It can be seen that these functions are capable of defining complex shapes using a different amounts of support points. The black dashed lines are the control polygons defined by the support points and show how the placement affects the course of the curve.
The Bezier spline consists of $m-1$ such Bezier curves and is computed based on the

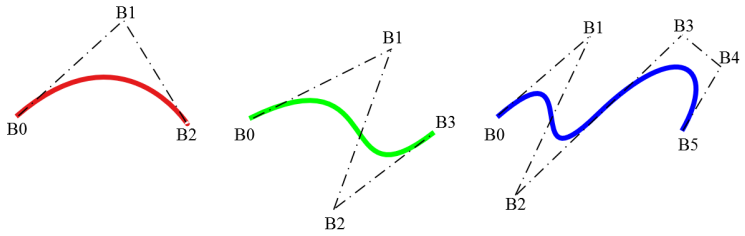


Figure 2.5: Visualization of different Bezier curves using one, two and four additional points to define the curve.

placement of the given set of points P_j. For each of these segments, the support points have to be computed based on the gradient and gradient direction of the adjacent curves. Meaning, the first and second derivative have to be equal [21].

$$BC(t) = (1-t)^3 B_0 + 3(t - 2t^2 + t^3)B_1 + 3(t^2 - t^3)B_2 + t^3 B_3 \tag{2.10}$$

The cubic Bezier curve ($n = 3$) is shown in Equation 2.10 [21]. It is used as base function for computing the spline.

$$\begin{aligned} BC'(t) &= -3(1-t)^2 B_0 + 3(1 - 4t + 3t^2)B_1 + 3(2t - 3t^2)B_2 + 3t^2 B_3 \\ BC''(t) &= -6(1-t)B_0 + 3(-4 + 6t)B_1 + 3(2 - 6t)B_2 + 6tB_3 \end{aligned} \tag{2.11}$$

Equating the first derivative ($BC'(t)$ from Equation 2.11) at positions $t = 0$ and $t = 1$ results in $-3B_{0,j+1} + 3B_{1,j+1} = -3B_{2,j} + 3B_{3,j}$. Given $B_{0,j+1} = P_j$ and $B_{3,j} = P_j$, it is simplified to $2P_j = B_{1,j+1} + B_{2,j}$. For the second derivative, the same is done resulting in $-2B_{1,j+1} + B_{2,j+1} = B_{1,j} - 2B_{2,j}$ [21]. The last two equations are defined for each segment connection. Meaning, the linear equation system is under determined. Therefore, the outer segments (P_1, P_2 and P_{m-1}, P_m) are enforced to be linear by setting the second derivative to zero. It leads two the linear equation system shown in Equation 2.12 [21].

$$\begin{aligned} 2P_j &= B_{1,j+1} + B_{2,j} \\ -2B_{1,j+1} + B_{2,j+1} &= B_{1,j} - 2B_{2,j} \\ P_1 - 2B_{1,1} + B_{2,1} &= 0 \\ -P_m - B_{1,m-1} + 2B_{2,m-1} &= 0 \end{aligned} \tag{2.12}$$

This system can be solved using Gaussian elimination and results in a defined curve given a set of points with the property that each point is on the curve. Figure 2.6 shows computed Bezier splines for a set of four points. The circles represent the given points and the black curve is the Bezier spline. This visualization also shows the effect of the placement and order of those points. Bezier splines are used in this thesis to estimate the eyelid outline.

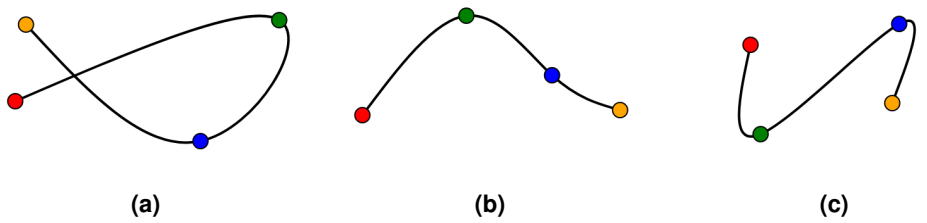


Figure 2.6: Different Bezier splines for a given set of points (circles).

2.6 Coordinate systems

This section introduces some coordinate systems that are used in this thesis. In general, an image is represented using Cartesian coordinates with the center at the top left corner. The positive y axis points down and represents the row index. Left to right is the direction of the x axis, which are used to select the column index. While there exist numerous other transformations, we limit ourselves here to the coordinate systems used in this thesis.

2.6.1 Polar or angular coordinates

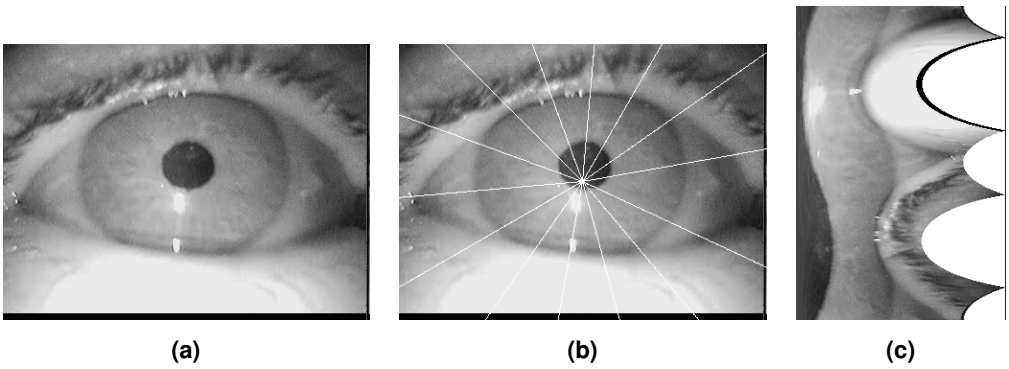


Figure 2.7: (a) Input image. (b) Center of the angular coordinate system (intersection of the white lines) and the extraction location with an intermediate spacing of 25°. (c) is the angular representation of (a). The white region on the right side of (c) represent areas where no image content was available due to the non circularity of (a).

The angular or polar transformation is shown in Figure 2.7. The image is transformed into a new representation consisting of an angle and a distance to the coordinate center. In Figure 2.7(b), the new center is shown by the intersection of all white lines. These white lines represent vectors along which the image intesities are extracted. Each of these vectors has an angle, which represents the new y axis. The x axis is now the length of this vector to the extraction location.

$$\begin{pmatrix} \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \\ arctan(\frac{y_i - y_c}{x_i - x_c}) \end{pmatrix} \tag{2.13}$$

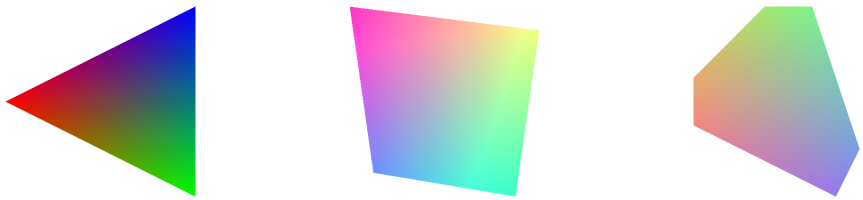
Equation 2.13 [163] defines this transformation, where x_c, y_c is the center of the angular transformation and x_i, y_i are the positions in Cartesian image coordinates. The transformation of Figure 2.7(a) can be seen in (c). Due to the square shape of (a), not all distances have an intesity value in (c): Shown as white areas on the right side.

2.6.2 Barycentric coordinates

Barycentric coordinates describe the position of a point in a polygon. Therefore, the coordinate system is defined by the outer points of this polygon and the amount of points also defines the dimension of the Barycentric coordinate system.

$$a_i(P) = \frac{\Delta P \cup A \setminus A_i}{\Delta A} \tag{2.14}$$

Equation 2.14 [163] computes the Barycentric coordinates a_i for a point P based on the polygon defined by the set of points A. Δ represents the computation of the area of a set of points and can be computed using the Gaussian trapezoidal formula. Therefore, the coordinate for a point P in a triangle with edge points X, Y, Z is $(\frac{\Delta PYZ}{\Delta XYZ}, \frac{\Delta XPZ}{\Delta XYZ}, \frac{\Delta XYP}{\Delta XYZ})$ [163]. It can be seen that the coordinates are the area similarities between the polygon, and the areas that can be constructed by replacing a point with P. The formulation in Equation 2.14 [163] has a disadvantage, which is that these coordinates can be assigned to points outside of the polygon. Therefore, it has to be checked if the point is within the polygon or not. Figure 2.8 shows polygons where the coloring is based on the Barycentric coordinates. In



(a) **(b)** **(c)**

Figure 2.8: Three different polygons, where the color is interpolated between the edge points using Barycentric coordinates.

(a) the corners of the triangle correspond to the color red, green and blue respectively. The Barycentric coordinates of the points inside the triangle are multiplied with those colors, which is a color interpolation. In Figure 2.8(b) and (c), two additional shapes are shown with the same interpolation method.

2.7 Feature extraction

Feature extraction is the first process of data reduction. In the case of images where two or three dimensional arrays are processed, feature extraction is a fundamental operation. An image consists mainly of redundant and unimportant information, e.g. the background surrounding an object. Intuitive features are the outline, shape and color. These properties of important regions or pixels have to be extracted and can afterwards be used for classification tasks or to build graphs as described in the previous section. In comparison to such primitive features (outline, shape etc), more complex and robust features have been developed. The first are Maximally Stable Extremal Regions (MSER) [151], where regions of an image are extracted that are stable under different intensity thresholds. Another technique is the Scale-Invariant Feature Transform (SIFT) [146]. For those features, first possible extrema points in an image are computed using a Laplacian of Gaussian with different standard deviations. Therefore, an extrema is selected over its neighbors and the scale space. Since edges and positions with low contrast also represent such extrema, only points with high contrast and no unilateral orientation are selected. The rotation invariance is then achieved by calculating the gradient over a larger area surrounding this extrema. This gradient direction is used to align key points in the matching process, which is used to connect positions in two different images which represent the same. The final descriptor of such a key point is created by calculating the orientation of subregions surrounding it. The faster version of those features is called Speeded Up Robust Features (SURF) [15]. The main difference to the SURF features is the approximation of the Laplacian of Gaussian by Haar features operating on an integral image. The same technique is used to calculate the orientations, which is done by Haar approximations of wavelet filters. While there are numerous further feature calculation algorithms, here we focus on methodology that was applied for pupil and eyelid detection.

2.7.1 Convolution

Convolution is the product of two functions (e.g. $f(x)$ and $g(x)$) which form a third ($(f * g)(x)$) function [60]. This new function is the weighted average of f, where g is the weight function.

$$(f * g)(x) = \int f(p)g(x-p)dp \tag{2.15}$$

Equation 2.15 [60] is the mathematical definition of a convolution. This means that $(f * g)(x)$ is the overlapping area between both functions, where the weighting function g is inverted $g(-p)$ and shifted by x. As an example, this convolution can be used for averaging or to calculate gradients of a function. These operations are usually applied to signals, which are approximated using time-related measurements. Those measured values form a discrete function, which is shown in Equation 2.16 [60].

$$(f * g)(x) = \sum_{-\infty}^{\infty} f(p)g(x-p) \tag{2.16}$$

In the case of an image, a two dimensional function (the image I(x,y)) is used. Therefore, the discrete convolution can be computed by shifting a kernel function $K(x,y)$ over an image.

$$(I*K)(x,y) = \sum_{i=-\frac{n}{2}}^{i=\frac{n}{2}} \sum_{j=-\frac{n}{2}}^{i=\frac{n}{2}} I(x+i,y+j)K(i,j) \tag{2.17}$$

In Equation 2.17 [60], I represents the image and K is the kernel with a size $n+1, n+1$. It computes the element-wise product at position x, y. As can be seen, it is not the convolution as defined before $(-p)$. Therefore, it is called correlation and is identical to the convolution for symmetric kernels. The correlation function has its application in template matching, where the kernel represents a shape that is searched. For the correlation, Equation 2.17 is a simple change to Equation 2.18 [60] by inverting the indices.

$$(I*K)(x,y) = \sum_{i=-\frac{n}{2}}^{i=\frac{n}{2}} \sum_{j=-\frac{n}{2}}^{i=\frac{n}{2}} I(x-i,y-j)K(i,j) \tag{2.18}$$

This inversion is a rotation of the kernel function by 180°; which is important to make the computation associative. Meaning, $f*(g*h)$ is equal to $(f*g)*h$. An example herefore

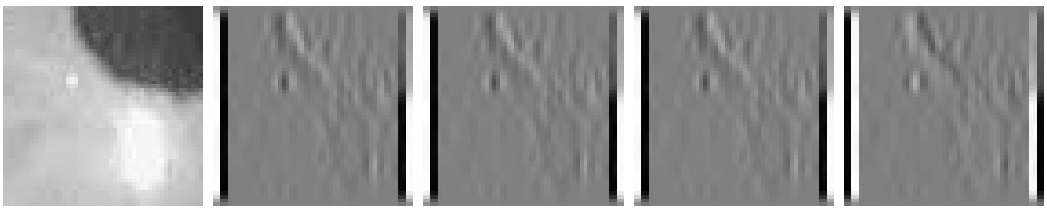


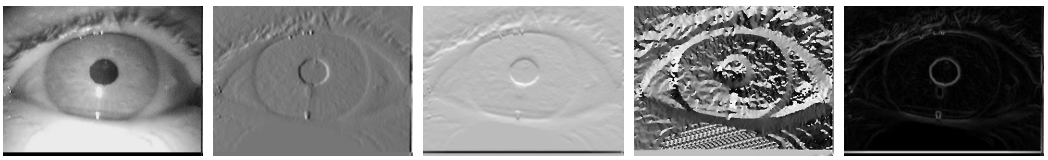
Figure 2.9: Shows the difference between convolution and correlation. (a) is the input image. (b,c) are the convolution and (d,e) the correlation with two identical non symmetric filters.

is shown in Figure 2.9. The used kernel is $\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$ with which the operation was performed twice. Figure 2.9(b,c) and (d,e) represent the operations $(I*K)*K$ and $I*(K*K)$ respectively. It is apparent, there is a difference between (d) and (e), since the correlation is not associative.

The convolution in image processing can be used to compute smoothing, sharpening, gradients, reliefs, and much more. It makes it possible to treat an image like a mathematical function and is therefore unalterable. The disadvantage of convolution are the high computational costs. Equation 2.18 [60] for example, has to be computed for each image position and under the assumption that the kernel has the same size as the image this operation would have the complexity of $O(n^4)$. The costs can be reduced to $n*log(n)$ by using the Fourier transform and the powerful convolution theorem [60]. The Fourier transform splits the image into individual frequencies, which are sin and cosine waves of different wavelength [60]. For each of these frequencies the amplitude are computed. These amplitudes

describe the portion of the associated frequency in the image. Meaning, frequencies that are not present in the image have an amplitude of zero. Therefore, the first obvious usage of the Fourier transform is image compression, which can be easily done by only storing the amount of frequencies that describe the largest portion of the image. The usefulness in case for convolutions is described by the convolution theorem. These concepts are used in this thesis to extract features like edges out of an image. Convolutions are also a central part of CNNs which enable the machine learning approach to learn stationary invariant features.

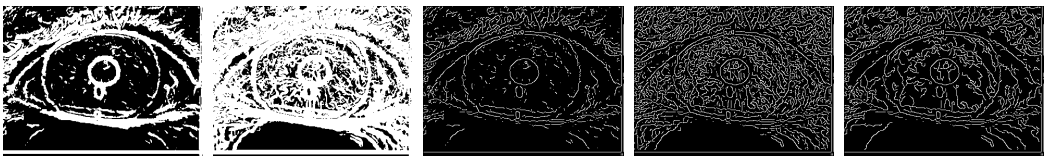
2.7.2 Edge detection



(a) (b) (c) (d) (e)

Figure 2.10: The input image (a), the gradient in x (b) and y (c) direction, the angle (d) and the length (e).

The Canny edge detection [32] by John Francis Canny is probably the most famous algorithm for edge computation. In the first step, the gradients or the first derivate of an image are computed. An example is shown in Figure 2.10, (b) and (c) are the derivatives in the x and y direction of image (a). In practice, a preliminary step is always the smoothing of the image due to noise. Based on those partial derivatives in x and y direction, the angle of the gradient is computed (Figure 2.10(d)). This angle is the orthogonal along the edge that the algorithm aims to extract. The length of a gradient or the magnitude is the Euclidean distance and shown in Figure 2.10(e). For the computation of the edges, the algorithm starts by defining a threshold (T_1) for the magnitude image. All edges that are above this threshold are possible edge candidates (Figure 2.11(a)). These edge candidates form surfaces



(a) (b) (c) (d) (e)

Figure 2.11: Shows the thresholded gradient magnitude with T_1 (a) and T_2 (b). In (c) and (d)non maximum supression is applied to the thresholded images. (e) is the final result of the canny edge detector.

surrounding the real edge, since the gradient grows towards an edge. Therefore, non maximum suppression is applied. This technique selects only the maximum along the gradient direction. An example result of this can be seen in Figure 2.11(c). This result is already acceptable but one threshold is not particularly robust and it could easily happen that edges

are extracted incompletely. For another lower threshold T_2, (b) represents the candidates and (d) the selected edges after non maximum suppression. As can be seen, there are too many edge pixels selected. Therefore, Canny proposed to use a method called hysteresis, which combines two thresholds. It uses all candidates selected using the high threshold T_1 with non maximum suppression (c) and analyses the selected candidates from the lower threshold T_2 if they are connected to a high threshold pixel. This process is done not only by a direct neighborhood but also through several candidates. The final result can be seen in Figure 2.11(e).

2.7.3 Histogram of oriented gradients (HOG)

While edges represent a feature that we understand, other features that represent local structures hold more information if processed by a computer. The Histogram of Oriented Gradients (HOG) [39], [153] feature is also computed on the gradients of an image. As the name suggests, those gradients are binned in a histogram, where each bin represents an orientation. These orientation histograms are represented as small stars in Figure 2.12(b,c,d).

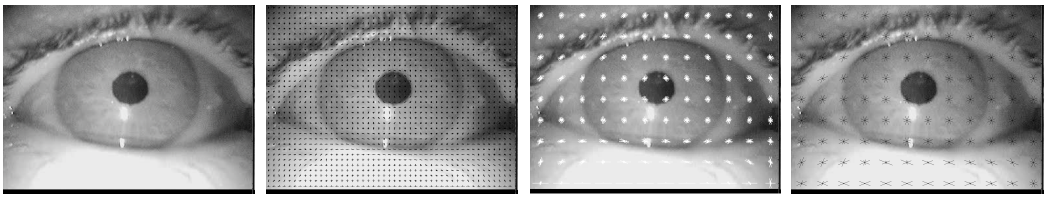


Figure 2.12: Shows the input image (a) with HOG features computed on different cell sizes (b) and (c). The difference between (c) and (d) is a smaller set of orientation bins for (d).

Each histogram is computed in a cell, which has the shape of either a square or circle [60]. These areas are set by defining a grid over an image. For each cell, such a histogram is computed, where the amount added per gradient is usually a value computed on the magnitude of this gradient. Based on the amount of bins defined for each histogram, the amount of rays in the illustration from Figure 2.12(c,d) changes. The amount of orientation bins influences the quality of subsequent processing steps like classification, therefore it has to be mentioned that too many bins lead to ungeneralized features [60]. In contrast, too few bins lead to a feature that contains low information. In [39] for example, nine bins were used.

These features computed on the magnitude leads to the problem that, in comparison to the global magnitude, they can be feeble [60]. In addition, changes in illumination would change the feature too. Therefore, those features have to be normalized. It is done over locally connected cells; such a set of cells is called block. The normalization is calculated by dividing each orientation bin in each histogram by the Euclidean distance of all orientation bins in the block. The result represents one feature vector. For the entire image, such a block is shifted over the grid of cells; therefore each cell contributes to different features [60].

2.8 Graph construction

The graph construction of point clouds is useful to represent a scene only by a subset of locations. It improves the runtime of algorithms by reducing the data that has to be processed. The point cloud for images consists out of feature points like edges or computed by other methods like SIFT [146], SURF [15], or MSER [151]. In such a graph, the connections and the area can be used for interpolation. It allows for example to distribute a 3d reconstruction on the whole image. In the following, we describe two general methods to compute such graphs but it should also be pointed out that the way in which a graph is to be constructed depends on its intended use.

2.8.1 Delaunay triangulation

The Delaunay triangulation forms a graph of triangles under the usage of one fundamental property, namely that the circumcircle associated to each triangle is not allowed to contain another point q_i from the given point cloud Q in two dimensions. For the third dimension, the circumcircle criterium changes to a circumsphere and the triangle to a tetrahedra. Due to the fact that these shapes can be extended in every dimension, the Delaunay triangulation is applicable for all dimensions. However, it has to be said that the topology of the triangulation is not unique [134].
For the construction of a triangulation in such cases there exist numerous algorithms, e.g. the flip algorithm [46] operating at a complexity of $O(n^2)$. The complexity can however be reduced to $O(nlog(n))$, if the triangles are stored in a tree [124]. An example of a Delaunay triangulation for a given point cloud (Figure 2.13(a)) can be seen in Figure 2.13(c). The proximity of this graph to a Voronoi diagram (see below) becomes obvious when image (d) is viewed.

2.8.2 Voronoi diagram

In Figure 2.13(b,) a Voronoi diagram is shown. The definition of such an enclosed area in such a diagram is given by the set of points $p_i \in \mathbb{R}^2$, which are closest to $q_i \in Q$ [124]. Where Q is the set of points given (Figure 2.13(a)) and $\mathbb{R}^2$ is the two dimensional plane. The borders of such a Voronoi diagram are the points that have the same distance to both neighbors. The proximity to the Delaunay triangulation is that these areas can be computed directly out of the triangulation [124]: It also applies vice versa. In Figure 2.14, it is shown more intuitively. It can be seen that the connections of the Delaunay triangulation are orthogonal to the connections of the Voronoi diagram [124]. In addition, the intersection of both lines is always at the center of the delauny triangulation. It can be used by selecting one triangle corner and calculating the intersections of the orthogonal lines. It is done for all triangles, in which a point is part of. After computing all points, the regions can be connected along those orthogonal lines. For the inverse (Voronoi diagram to Delaunay triangulation), only the data points with shared borders have to be connected.

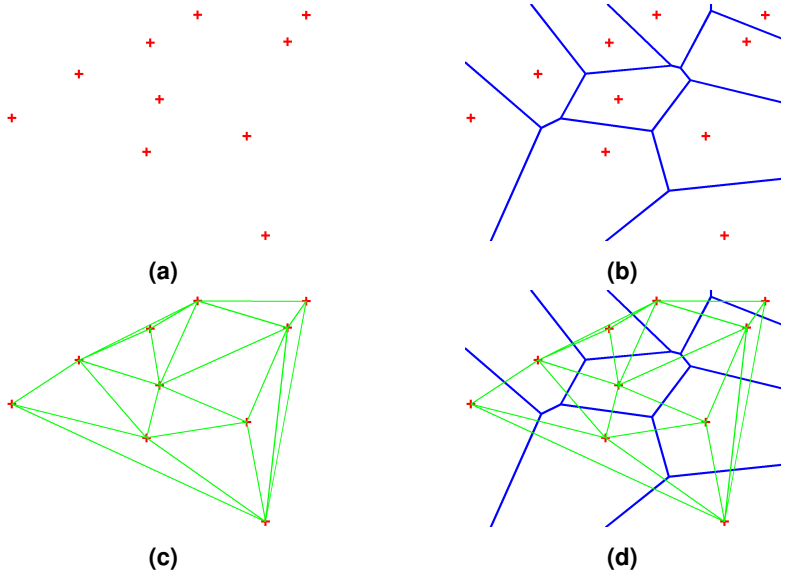


Figure 2.13: Delaunay triangulation (c) from a set of points (a). In (b) the Voronoi diagram is shown and in (d) both the Delaunay triangulation (green) and the Voronoi diagram (blue) are plotted.

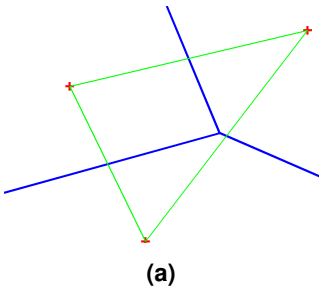


Figure 2.14: Delaunay triangulation (green), Voronoi diagram (blue), and data points(red cross).

2.9 Machine learning concepts

Machine learning is used as an umbrella term for methods that learn generalized solutions by means of examples [166]. There are different methods in machine learning such as supervised learning, semi-supervised learning, reinforcement learning, unsupervised learning, and active learning [36], [63], [106], [171], [255]. Supervised and semi-supervised learning are the most commonly used methods. Here the algorithm is given a set of data with ground truth annotations. Based on the ground truth information, the algorithm searches for a function, which results in the desired annotation when entering the corresponding data [166]. The difference between supervised and semi-supervised learning is that for the semi-supervised case, only a part of the given data is annotated [166]. In reinforcement learning, the algorithm is trained by reward and punishment [166]. Meaning in case of failure, the algorithm is forced to drop a part of his previous approach or a part of his memory is removed. In the opposite case, where the algorithm succeeds, parts of the learned approach

are hardened, and therefore, less probable to forget. Unsupervised learning, in comparison is the approach above mentioned, is where the machine learning method is given a set of data for which a better representation or clustering has to be found [166]. Some well-known representatives are the expectation maximization [41], Principal Component Analysis [178], [211], and Autoencoder [102] technique. The core idea of the expectation maximization is to start with a randomly selected initialization of the model. Afterwards the data is assigned to classes based on the model parameters. It is the expectation step on which the maximization step follows. Here, the model parameters are adapted based on the data assignment until convergence. The Principal Component Analysis is used to simplify extensive data sets by approximating a large number of statistical variables with a smaller number of meaningful variables. Therefore, the correlation of the multidimensional data is minimized by a transformation into a new vector space with a new base. The axis of this new base are the principal components. The arrangement of these components is ordered according to the proportion of the total variance of the data along this axis. Therefore, the first component has the largest share in the variance of the data. For the reduction of the data only the first n components are selected for further processing. Autoencoder [102] are an unsupervised example for Artificial Neural Networks [104], [199]. Here, the given data is the input and output of the network, where the network structure consists of significantly fewer neurons than the dimension of the data. After training the weights of each neuron, they can be used to reduce the dimension or feature extractors [102]. With active learning, the algorithm is allowed to request the annotation of the data, but it has to learn to formulate this request. Therefore, the algorithm learns to formulate a request of annotations, which promise a high information gain with as few requests as possible [259].
The generality of a trained classifier represents the performance on unseen data [166]. In case of supervised or semi-supervised learning, it is apparent. For the unsupervised case, it means that, for example, the auto encoder is capable to successfully reducing the data for unseen inputs. Successful in this context means that the steps following the autoencoder reduction are also successful. In reinforcement learning, it concerns the developed method of rewarding and punishment to be applicable to new problems and also that the learned classifier is able to handle new scenarios. For example, if an algorithm learns how to play a computer game with some scenarios or levels, it should be capable of playing other unseen scenarios as well. The generality in active learning can be understood similarly to that in reinforcement learning. In contrast to the generality of a classifier, is the over fitting. Meaning, if a classifier, performs only well on seen or a specific subsets of data, it is fitted to this problem and therefore not applicable to data representing other challenges. Well-known representatives of machine learning methods are the Naive Bayes classifier [201], Decision trees [62], [157], Random Forrest [26], Random Ferns [173], Hidden Markov Models [27], Support Vector Machines [58], [210], Artificial Neural Networks [104], [199], Convolutional Neural Networks [133], and others. In the following, the last three techniques will be explained in more detail since they are used through out this work.

2.9.1 Support vector machine (SVM)

The starting point for building a Support Vector Machine is a set of training objects $(\vec{x}_i, y_i)$ where $\vec{x}_i$ is the feature vector of an object and y_i the corresponding class label (e.g. A, B in case of a two-class SVM) [29], [99], [100], [221]. The support vector machine now tries to find a hyperplane in this feature vector space that separates the data. This hyper plane is also called model [29], [99], [100], [221]. In order for a class assignment to be described mathematically, each class has to be represented by a number. For this class assignment, the position to the hyperplane is used. Each point can lay either above or below it. This also leads to the model function of the SVM that is the scalar product between the normal vector of the hyperplane $\vec{n}$ and the feature vector $\vec{x}$ together with a shift b.

$$y_i = sgn(\vec{n} \circ \vec{x}_i + b) \tag{2.19}$$

Where $\circ$ is the scalar product and $sgn()$ the sign function (Equation 2.19 [100]). Meaning, every point on this hyperplane using Equation 2.19 [100] is zero and for all the others, the result is either negative or positive. The choice of our classes $1, -1$ is therefore the geometrical separation of the feature space, where it has to be noted that points on the line are assigned to the negative class. It leads to the equation system in Equation 2.20 [100],

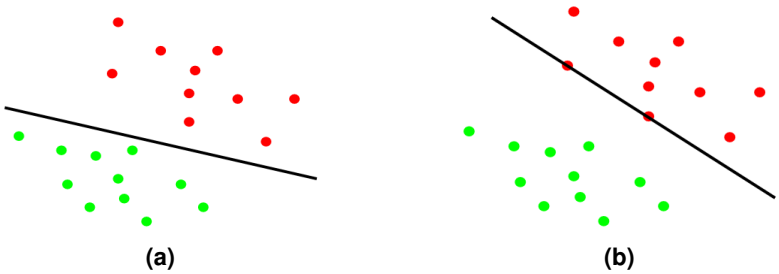


Figure 2.15: Shows red and green points that represent two classes in the feature space. The black line is a hyperplane separating both classes.

where $\vec{x}_i$ and b has to be estimated. The problem with this formulation can be seen in Figure 2.15, where the black line represents the hyperplane.

$$\begin{aligned} \vec{n} \circ \vec{x}_i + b > 0, \forall y_i = 1 \\ \vec{n} \circ \vec{x}_i + b \leq 0, \forall y_i = -1 \end{aligned} \tag{2.20}$$

This hyperplane is not optimal and there are various solutions for it. In addition, there is no formulation of a margin, which can be seen in Figure 2.15(b). Therefore, to ensure that the margin on both sides is equal, we define a margin in Equation 2.20 [100]. Leading to Equation 2.21 [100], where the closest class example is only allowed to be on the margin border (Figure 2.16).

$$\begin{aligned} \vec{n} \circ \vec{x}_i + b \geq 1, \forall y_i = 1 \\ \vec{n} \circ \vec{x}_i + b \leq -1, \forall y_i = -1 \end{aligned} \tag{2.21}$$

It can be combined in one equation $y_i(\vec{n} \circ \vec{x}_i + b) \geq 1$ [100] due to the fact that correct classified examples multiplied by their label are always positive. The width of this margin is only dependent on $\vec{n}$, since b only represents the displacement of the hyperplane (see point normal equation $\vec{n} \circ (\vec{x}_i - \vec{x_0}) = 0$). Therefore, the width of the margin can be controlled by the Euclidean leangth $||\vec{n}||$ as normalization. It leads to $y_i(\frac{\vec{n}}{||\vec{n}||} \circ \vec{x}_i + \frac{b}{||\vec{n}||}) \geq \frac{1}{||\vec{n}||}$, where $\frac{1}{||\vec{n}||}$ [100] is half the width of the margin (direction to -1 and +1 shown in Figure 2.16). In order to maximize the margin, $||\vec{n}||$ has to be minimized. It leads to the primal formulation in Equation 2.22 [100] using a quadric function, where M is the sample size.

$$\begin{aligned} &\min_{\vec{n},b} \frac{1}{2}||\vec{n}||^2 \\ &y_i(\vec{n} \circ \vec{x}_i + b) \geq 1, \forall i = 1, ..., M \end{aligned} \tag{2.22}$$

It can be reformulated with the Lagrange duality, which introduces weights for the samples.

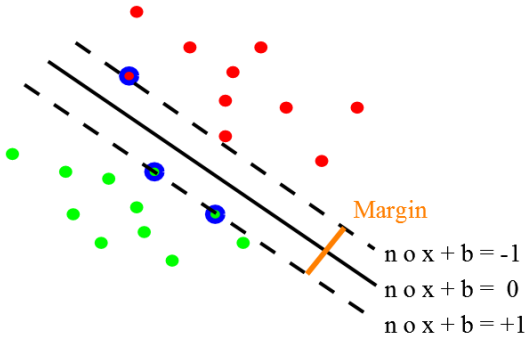


Figure 2.16: Shows the margin (dashed lines) between samples from two classes (red and green dots) and the hyperplane (solid line).

Those weights are called Lagrange multipliers (α_i in Equation 2.23 [100]) and prevent that each sample contributes equally to the solution. Therefore, each alpha has to be greater than or equal to zero ($\alpha_i \geq 0$).

$$L(n, b, \alpha) = \frac{1}{2}||\vec{n}||^2 - \sum_{i=1}^{M} \alpha_i (y_i(\vec{n} \circ \vec{x}_i + b) - 1) \tag{2.23}$$

The Lagrange Equation 2.23 [100] has still to be minimized for $\vec{n}$ and b but maximized for α. Therefore, the gradients for $\vec{n}$ and b has to be zero (quadric function). The first derivation of Equation 2.23 [100] for $\vec{n}$ is $\vec{n} = \sum_{i=1}^{M} \alpha_i * y_i * \vec{x}_i$ (Note:$||\vec{n}|| = (\vec{n}^T \vec{n})^{\frac{1}{2}}$ [100]) and for b it is $0 = \sum_{i=1}^{M} \alpha_i * y_i$. According to the Karush-Kuhn-Tucker conditions, the optimum is $\alpha_i(y_i(\vec{n} \circ \vec{x}_i + b) - 1) = 0$ [100]. Meaning, either $\alpha_i = 0$ or $y_i(\vec{n} \circ \vec{x}_i + b) = 1$ in the optimal solution. Since b is computed from $\vec{n}$ (point normal equation) and $\vec{n} = \sum_{i=1}^{M} \alpha_i * y_i * \vec{x}_i$ (first derivative for $\vec{n}$), it can be seen that only sample points with $\alpha_i > 0$ influence the optimal solution. The points are called support vectors which also gave the name to this method. In Figure 2.16, the support vectors are shown with a blue circle surrounding them. The full

optimization problem formulation can be seen in eqution 2.24 [100].

$$\begin{gathered} \max_{\alpha} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j (x_i \circ x_j) \\ \sum_{i=1}^{M} \alpha_i y_i = 0 \\ \alpha_i \geq 0, \forall i = 1, ..., M \end{gathered} \tag{2.24}$$

It can be obtained by inserting the first derivative $\vec{n} = \sum_{j=1}^{M} \alpha_j * y_j * \vec{x_j}$ in Equation 2.23 [100]. b is not included in the formula because it is not present in its derivative ($\sum_{i=1}^{M} \alpha_i y_i = 0$) but this derivation provides a necessary constraint (side term). After the maximum for Equation 2.23 is found, the normal on the hyperplane can be computed using the derivative with the optimal alphas $\vec{n} = \sum_{j=1}^{SV} \alpha_j * y_j * \vec{x_j}$ and with those $b = y_j - (\vec{n} \circ \vec{x_j})$ [100], where only the support vectors (SV) are used and for b the pair $y_j, \vec{x_j}$ has to be one of those. Therefore, the final decision function $D(x_{neu})$ can be deployed by inserting the derivative as done before.

$$D(x_{neu}) = sgn(\sum_{i=0}^{SV} \alpha_i y_i \vec{x_i} \circ \vec{x_{neu}} + b) \tag{2.25}$$

Equation 2.25 [100] shows the final decision function that is used to compute the class affiliation for a new feature vector x_{neu}.

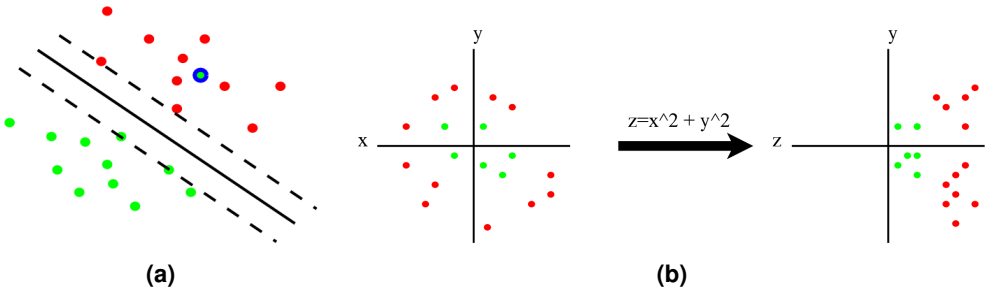


Figure 2.17: (a) a set of points, where one from the green class sample is in the middle of the red class samples (green dot surrounded by a blue circle). (b) introduction of a new feature dimension for linearly not separable data samples.

In the current form, the optimization cannot handle nonlinear problems as shown in Figure 2.17. In (a), a scenario is shown, where a sample has to be misclassified to find an optimal hyperplane, whereas in (b) an example is shown, where the samples have to be transformed. This transformation forms a new dimension z and is computed based on the input vector. This transformation (Θ) raises the problem that in Equation 2.24 [100] and 2.25, the training samples x_i only occure in the scalar product. Therefore, all training samples would have to be transformed into a higher dimensional feature space, which is computationally intensive. Therefore, instead of transforming the entire data to the higher dimensional space, a function is required that reprojects the hyperplane into the feature space of the given data. It is the kernel trick and describes the circumvention of the complex transformation by a suitable core function. The set of these functions has to be continuous, symmetric, and the Mercer conditions [155] have to hold. Since the scalar product also represents such a

function, we can write the kernel function K directly into Equation 2.24 and 2.25, which replaces the scalar product. For example, the polynomial function $(1+a_i \circ b_i)^{ord}$, which is commonly used as kernel, can be computed directly on the input samples. For $ord = 2$ this results in $1+2a_1b_1+2a_2b_21+(a_1b_1)^2+(a_2b_2)^2+2a_1b_1a_2b_2$ [100], for which no transformation into the higher dimensional space of the data is necessary.

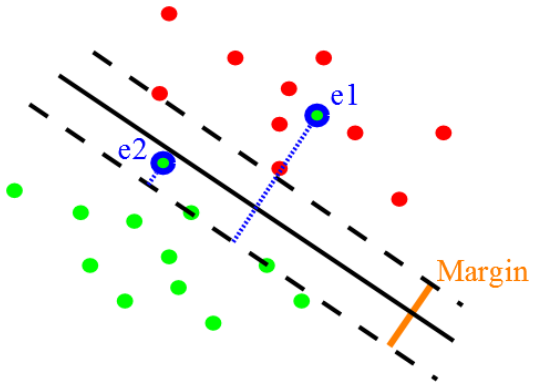


Figure 2.18: An example of linearly separable data with two slack variables $e1$ and $e2$.

The kernel trick and the data transformation are useful in the handling of non-linearly separable data, where the computation costs for the kernel trick only increases linearly. In addition, data that is still not separable based on a transformation as shown in Figure 2.17(a) remains problematic. Therefore, slack variables e, together with a weighting factor C, are inserted into Equation 2.22 [100].

$$\begin{gathered} \min_{\vec{n},b} \frac{1}{2}||\vec{n}||^2 + C\sum_{i=1}^{M} e_i \\ y_i(\vec{n} \circ \vec{x}_i + b) \geq 1 - e_i, \forall i = 1,...,M \\ e_i \geq 0 \end{gathered} \tag{2.26}$$

In Equation 2.26 [100], the extended formula is shown. If e_i is zero, the data point is correctly classified. For $e_i > 1$ as shown in Figure 2.18, $e1$ is on the wrong side of the hyper plane. The blue dotted line represents the correction vector. For $e2$ in Figure 2.18, the slack variable is between zero and one ($0 < e_i \leq 1$). In Equation 2.26, it can be also seen that C weights the sum of misclassified data points and represents a factor which has to be set priorly to the optimization. The final dual optimization problem is formulated as before. In addition, the kernel trick can still be applied. For the formulation of the regression based support vector machine and additional information to the derivation, motivation of single steps and alternative formulations of the slack weighting, the readers are referred to [29], [99], [100], [221].

2.9.2 Artificial neural networks (ANN)

Artificial Neural Networks are a machine learning approach conceptually inspired by the human brain. The first concept was formulated by F. Rosenblatt [199]. In this approach, the

smallest possible net was created, which is a neuron or also called perceptron, Figure 2.19(a).

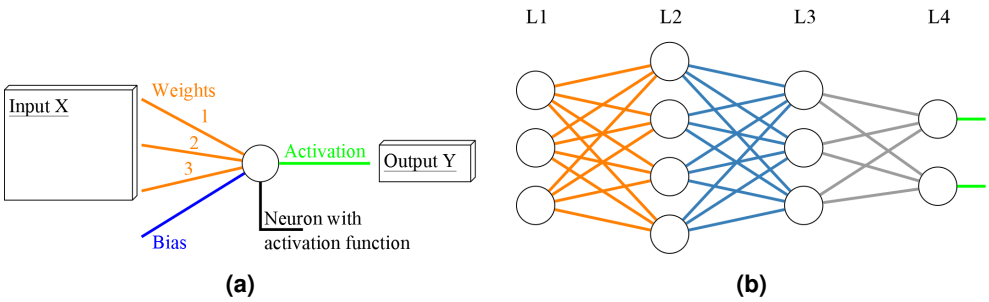


Figure 2.19: (a) a single neuron with three weights (orange), bias (blue) and activation (green). (b) is a parallel and sequential series of neurons and represents an artificial neural network. Each layer in (b) is colored differently, where each neuron is connected to all neurons from the previous layer. The bias term, as it is shown in (a), has been omitted in (b) to give an overview.

Such a neuron consists of weights (represented by orange lines in Figure 2.19(a)), which are multiplied with the input data and a bias term (blue line in Figure 2.19(a)). The activation of a neuron is computed based on a preliminary selected activation function. Any function can be selected as activation function as long as the following conditions apply: (i) Not constant, (ii) Bounded, (iii) Monotonically increasing, and (iv) Continuous [96]. These conditions come from the universal approximation theorem, which proves that any continuous function can be approximated by three layers (input, hidden, and output layer). In practice the sigmoid, hyperbolic tangent, or for deep networks, the rectified linear activation function is used [96]. There are numerous others and, as can be seen for the rectified linear activation function ($rla(x) = max(0,x)$), the conditions to be bounded is not fulfilled. In [220], it is shown that unbounded activation functions still satisfies the universal approximation property. Therefore, the authors in [220] used three reconstruction formulas (Fourier slice theorem, the Radon transform, and the Parseval relation).

$$\begin{aligned} z &= \sum_{i=1}^{3} W_i * X_i + b \\ a(z) &= \frac{1}{1+\exp(-z)} \end{aligned} \tag{2.27}$$

Equation 2.27 shows the computation of the neuron activation ($a(z)$). W_i represents the weight, b is the bias term and X_i are the input values as shown in Figure 2.19(a). The first term to compute z is the sigmoid function. The influence of the weigths and the bias term on the activation can be seen in Figure 2.20. The bias term can be seen as a constant one that is multiplied by a weight. This shifts the activation function (Figure 2.20(b)), whereas the weights influence the steepness (Figure 2.20(a)) [96].
Networks can be constructed as shown in Figure 2.19(b). Therefore, a set of neurons define one layer without connections between the neurons of the same layer. The connections are set only between layers where each neuron is connected to all neurons of the previous layers. For the input layer, we will use the number 1 and the output layer will be referred to by nl. Therefore, in Figure 2.19(b), L1 would be the input layer, where the input data

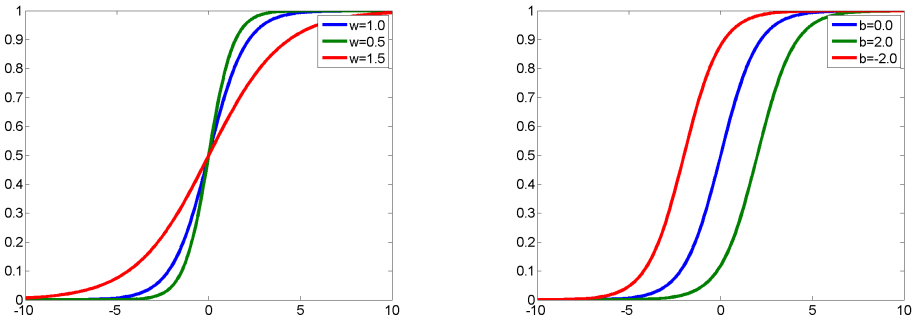


Figure 2.20: (a) the influence of the weights and (n) the impact of the bias term.

represents the activation value and L4 is the output layer with $nl = 4$. This neuron can be denoted by $N_{l,n}$, where n is the neuron number in the layer l. Same for the bias term $b_{l,n}$ and the weights $W_{l,n,k}$ with neuron k of layer $l+1$. Then, we can remove the neuron identifier k from the weights and n from the neuron activation N by arranging them in matrices.

$$forward(l+1,n) = sig(< W_{l,n}, N_l > +b_n) \tag{2.28}$$

Equation 2.28 [96] is the computation formula for an artificial neural network and called forward pass. This computation starts at $l = 1$ and runs to the last layer nl iteratively. W is now a matrix of weights associated to the connections between the node n and the nodes in the previous layer and N is the matrix of the activations. $<,>$ is the Frobenius scalar product, which is a point wise multiplication and summation. The symbol b stands for the bias term. $sig()$ is the sigmoid or logistic function.
The training of such artifical neural networks is usually done using batch gradient decent [96]. Batch, in this context, means that the forward pass is computed over multiple training samples and averaged. This averaging is compensates inaccurate annotations and reduces gradients that direct into less optimal solutions on the error hyperplane [96]. As an error function, usually the half squared distance between the result and the annotations is used, but there exist others, e.g. the softmax function. The training itself can be formulated in four steps:

- Perform feed forward pass
- Compute the error at the output layer nl
- Propagate the error backwards from the last to the first layer
- Update the weights

The first step is described by Equation 2.28 [96]. For the second step, the derivative of the half squared distance has to be computed to minimize the error.

$$Er(nl,n) = -(Y_n - N_{nl,n}) * forward'(nl,n) \tag{2.29}$$

This is shown in Equation 2.29 [96], where $N_{nl,n}$ is the output of node n in the last layer of the CNN, L is the matrix with the labeled correct results, and $'$ indicates the first derivative. The first term $(N_{nl,n} - Y_n)$ is the error and the second term is the first derivative of the forward pass, which gives the gradient.
The back propagation now propagates the error from the last layer to the first layer (Step 3).

$$Er(l,n) = < W_{l,n}, Er_{l+1} > * forward'(l,n) \tag{2.30}$$

In Equation 2.30 [96], n represents the node, l is the layer, and $'$ indicates the first derivative. The first part of the formula $< W_{l,n}, Er_{l+1} >$ multiplies the weights of the connections from node n to all nodes in the next layer with the error Er caused by them. Therefore, this part calculates the impact of this node to the error. In the second part $forward'(l,n)$, the first derivative or gradient is calculated, which represents the direction [96].
The last step is updating the weights. It is done by subtracting the error of the ending node of a connection weighted by the activation of the start node. As an equation: $w_{l,i,j} = w_{l,i,j} - (Er_{l+1,n_j} * N_{l,n_i})$, where n_i is the start node, n_j is the ending node, Er is the back propagated error, N is the activation from the forward pass and $w_{i,j}$ is the weight of the connection. The term $(Er_{l+1,n_j} * N_{l,n_i})$ is the partial derivative and the error reduction is done in the opposite direction of this gradient. In batch learning, this is usually done by using the mean error over all training examples in a batch. Figure 2.21 illustrates the

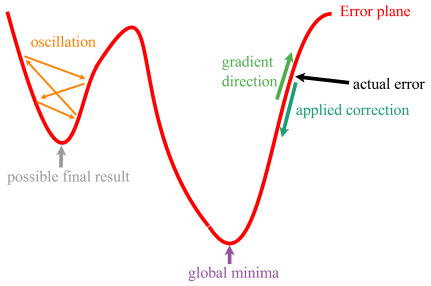


Figure 2.21: Illustrative example of gradient decent and algorithmic challenges.

gradient decent algorithm, where the red line represents the error hyperplane. As can be seen there are two main challenges, the first is that it is possible to find a local minima instead of the wanted global minima. The next challenge is to avoid oscillation, which occurs if the update is either too small or too large. Therefore, the update of the weights is usually combined with a weighting factor that is reduced after iterations.

2.9.3 Convolutional neural network (CNN)

The Convolutional Neural Network (CNN) follows the same concept as the ANN with the difference that it uses layers where a weight is shared between several nodes from the previous layer and layers that reduce the output of previous layers [96]. The former

are called convolution layers and the reduction is named pooling. It follows the idea that natural images have the property of being stationary, which means that image statistics of one part of an image are equal to any another part [96]. Therefore, a learned feature (small weight patch or convolution filter) trained on a small subset of image regions can be applied anywhere on the image with different activation values at each image location [96]. This process is shown in Figure 2.22(a). Afterwards, the pooling is applied, which can be

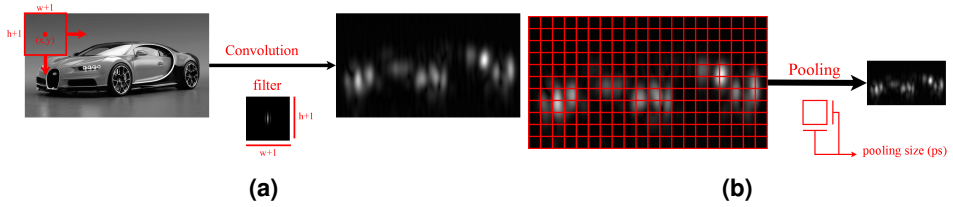


Figure 2.22: (a) a convolution example for one filter and (b) a pooling operation.

averaging over a region or selecting only the maximum (b). The pooling operation is placed as a grid over the image where the CNN is robust against small translations. In a CNN, the

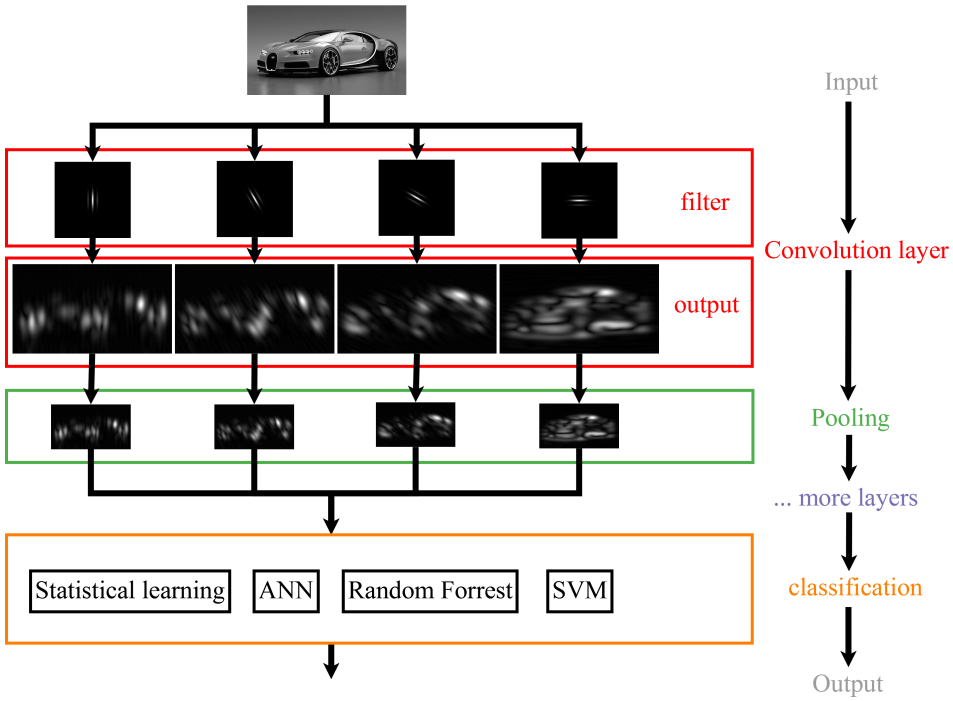


Figure 2.23: Work flow of a CNN with a convolution layer consisting of four filters and a subsequent pooling operation.

convolution layer consists of several filters that are learned. As seen in Figure 2.23, the four filters in the red box are applied to the image and stored separately. Afterwards, pooling is applied to each output and again stored separately. It can be done many times consecutively and will result in a final feature tensor. The last step is a classification, which can be done

either by a support vector machine or by an ANN, which is called fully connected layer in a CNN. The before mentioned consecutive convolution layers in a CNN architecture

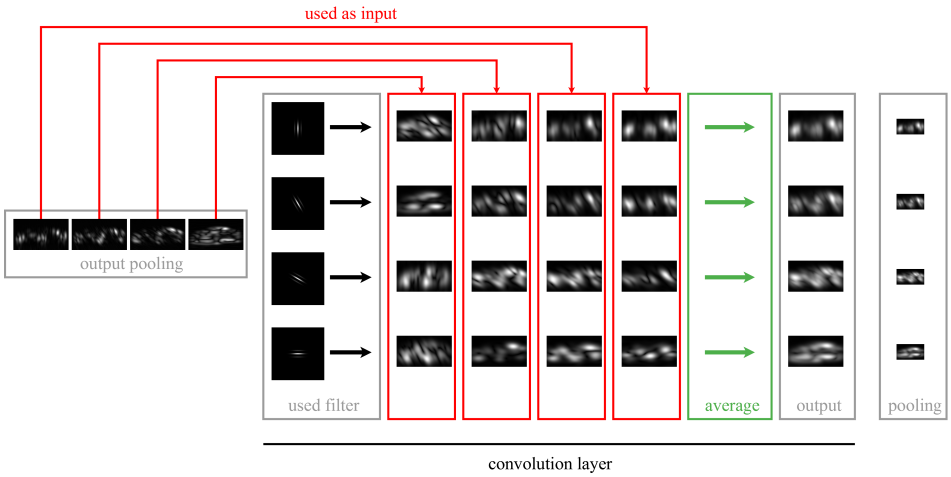


Figure 2.24: Convolution operation for consecutive convolution layers.

have to process volumes. This is shown in Figure 2.24. The operation to process these volumes can either be the averaging of convolution results by separating the volume into two dimensional arrays or the filter itself is a volume also called tensor. The averaging operation is shown in Figure 2.24. The filters from such consecutive convolution layers are also called deeper knowledge.

The forward and backward computation for a CNN is equal to the ANN with some minor differences for the new layers. In the following, we will describe how it is done for the pooling and convolution layer in more detail.

$$forward_conv(l,n) = sig(C_{l,n} \Theta N_l + b_n) \tag{2.31}$$

Equation 2.31 [96] calculates a forward pass for a convolution layer; the symbol meanings are identical to those in Equation 2.28. Θ is the convolution of two functions. In the discrete case, it means shifting the convolution core C over the input matrix N and calculating the Frobenius scalar product of the overlapping elements. Note that Equation 2.31 is identical to Equation 2.28 [96] for a convolution core C with the same size as the input matrix.

$$forward_pool(l) = resize(N_{l-1}, fac_l) \tag{2.32}$$

Equation 2.32 [96] is the forward pass through a pooling layer, which resizes the input by a factor fac_l. Resizing can be done by choosing the maximum, minimum, average, etc. For the CNNs developed in this thesis, we employed averaging in pooling layers.

For pooling layers, there is nothing to update, and therefore the error is propagated through them based on their resizing function. For average pooling, it would be spreading the error equally, and in case of max pooling, the input with the maximum would receive the complete error. For the convolution layer, the update of the filter is the sum of errors

produced by the convolution. Meaning, one weight in a convolution core is updated by the sum of all input activations multiplied by the resulting error from all nodes it connects, which is because it is a shared weight [96]. It is a convolution of the input activation with the error of the convolution layer.
Recent developments in CNNs are multi scale layers [31], [94], the inclusion of transposed convolutional layers (approximated deconvolution) [141], [262] and recurrent CNNs [137], [183]. CNN-based approaches are currently the state-of-the-art in object detection and classification with outstanding robustness and generalization [215], [248]. They, however, require massive parallel processing, which is too expensive for real world applications. Novel approaches reducing the computational costs are, binarization [192] or by employing field programmable gate arrays [164]. The latter one make it possible to produce application specific circuits, which can be integrated into any hardware. This makes them a valuable approach for pupil, iris, and eyelid detection, and therefore, important for eye tracking technology.

3 Pupil detection

The main source of noise in image based eye tracking is a non-robust pupil signal. All steps afterwards, such as the user specific calibration and eye movement classification rely on it. For example, Shippke and Todd reported [209] many problems related to pupil detection, such as changing illumination, motion blur, recording errors, and eyelashes covering the pupil. Those problems still exist, especially when eye-tracking is employed in unconstrained settings, e.g. [116], [119], [140], [217], [224].
The focus of this chapter is robust methods for pupil detection. Sections 3.1 and 3.2 describe the state of the art in pupil detection for head-mounted and remote tracking. Section 3.3 describes two decision based methods, namely ExCuSe [66], ElSe [79], and an approach specifically designed for microscope oculars [85]. In the following Section 3.4, PupilNet [83] is introduced, which is a CNN-based pupil detection method developed during this thesis to increase detection performance. Additionally, multiple datasets with annotated ground truth were released during this work to enable further research in this area. Additionally, the influence of dirt is evaluated for the head mounted algorithms based on a dirt simulation.

3.1 State-of-the-art pupil detection for head mounted eye tracking

Despite various approaches for automated pupil detection in the area of head-mounted eye-tracking, such as [71], [72], [115], [150], [244], the focus of this section will be on the most widely employed approaches.

3.1.1 Starburst

The Starburst pupil detection algorithm is a hybrid algorithm as it integrates feature-based and model-based approaches [136]. First, the image is denoised using a Gaussian filter. Afterwards, adaptive thresholding is applied to localize the corneal reflection. To determine the full extent of the corneal reflection, the algorithm assumes that the intensity profile follows a bivariate Gaussian distribution. The corneal reflection is than removed using radial interpolation.
For pupil center detection, the algorithm sends out rays from a central best guess of the pupil center. Each ray is evaluated using an adaptive threshold to select edges. If an edge point is found, it is selected as contour edge candidate. For each new candidate, another set of rays is generated. This process is iteratively repeated until convergence. The last step

is fitting an ellipse to the candidate points using Random Sample Consensus (RANSAC) paradigm [136].

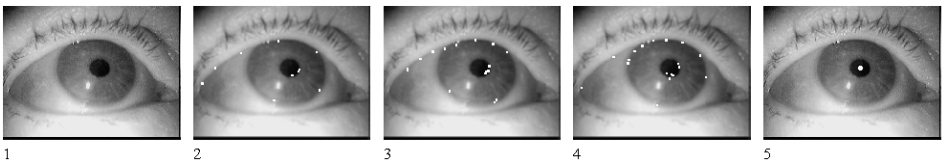


Figure 3.1: The Starburst [136] algorithm. In the first step, the input image (1) is smoothed. Rays are sent out (2) and edge candidates are selected as pupil contour (2). These points serve as new starting points for the next iteration (3). This process is repeated until convergence (4). The pupil center is estimated using a RANSAC ellipse fit (5). [86]

3.1.2 Świrski

The algorithm by Świrski et al. [231] consists of three main steps. In the first step, Haar-like feature are used to estimate a coarse position. This follows the assumption that the pupil is a dark area surrounded by brighter background. The Haar-like features can be computed very efficiently based on the integral image. Therefore, the algorithms uses different sizes of these features in a user specified range.
The strongest response of this set of features is used as coarse pupil center position. In addition, the size of the Haar-like feature is used to determine the region size in which the pupil center is refined. The first refinement is done by segmentation. Therefore, the algorithm applies k-means clustering of the intensity histogram of the selected region. The center of mass of the largest connected component is selected as refined position.

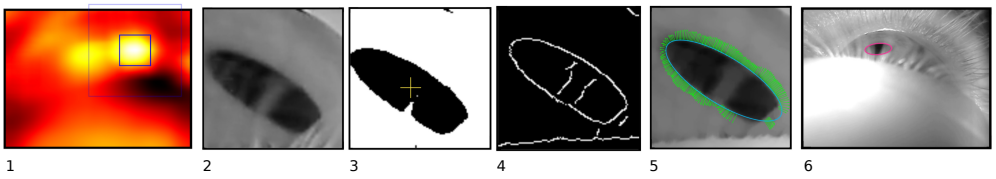


Figure 3.2: The Świrski algorithm [231]. First a coarse pupil position (2) is searched using Haar-like features (1). The region surrounding the coarse position is segmented using k-means clustering of its intensity histogram (3). In addition to the segmentation, a Canny edge detector is applied (4). The final pupil center estimate is obtained using an image-aware Random Sample Consensus (RANSAC) ellipse fitting (5) to detect the pupil (6). [86].

In the last step, the algorithm computes edges using the Canny edge detector. However, before the edge detection is applied, a morphological open is used to remove small gaps in the pupil. The pupil ellipse and the final center are found by fitting an ellipse to the edge points. This is done using Random Sample Consensus (RANSAC) ellipse fitting with an additional support function. The support function ensures that the ellipse lies on a boundary from dark pixels to light pixels, and that they lie along strong image edges.

3.1.3 SET

The SET [113] approach consists of a combination of thresholds and a segment based ellipse fitting. First, the input image is binarized using an intensity threshold (Figure 3.3(2)). Segments which are too small in this binarized image are removed (Figure 3.3(3)). For each segment, the convex hull is computed which results in the segment border. In the last step, an ellipse is fitted to each border segment and the ellipse closest to a circle is selected.



Figure 3.3: The SET algorithm. (1) input image and thresholding result (2). First a region size threshold is applied (3). Afterwards candidate pupil centers are computed (4) and the final center is selected (5). [86]

3.2 State-of-the-art pupil detection for remote eye tracking

In this section, we focus on three mainly used algorithms for pupil detection in the area of remote eye tracking, namely the approach by Droege & Paulus [42], the method by Timm & Barth [236], and by George & Routray [89].

3.2.1 Droege and Paulus

The approach by Droege and Paulus [42] uses the direction and length of the pixel gradients as feature. Due to the low resolution of remote images and the possibility of glints or reflections covering parts of the pupil, the pixel gradients are first filtered based on thresholding their magnitude. Afterwards, each gradient is interpreted as a line e.g. position and direction. Using an M-Estimator, the best intersection point of all lines is found. Afterwards, a pupil template is generated and shifted around the found center position. The best matching position is determined using the least squared error and chosen as pupil center.

3.2.2 Timm and Barth

Timm and Barth [236] also use the image gradients as feature for pupil center detection. In their approach, the idea is that the direction from the pupil center point to any pupil or iris contour point should be equal. Therefore, the algorithm compares the normalized displacement of each image gradient to each image pixel position. The results are accumulated and pupil center pixels which are dark are weighted stronger following thus the assumption that the pupil is usually dark. The final pupil center candidate is the highest accumulated value.

3.2.3 George and Routray

George and Routray [89] began with a coarse positioning using the orientation anulus convolution filter from [9]. The weights in this filter are modified by weighting vertical gradients stronger. This is done to compensate for nearly closed eyes where the eyelids contribute vertically. In addition, an inverted mean filter is applied. The combination of both filter results delivers a probability map, in which each local maxima is selected. The relationship between those local maximas and the responses is computed based on the standard deviation and mean. Based on this relationship, the best local maxima is selected. From this coarse position, all gradients are thresholded based on the angle to the estimated coarse center and their magnitude. The final ellipse fitting is applied using RANSAC.

3.3 Decision-based methods for pupil detection

This section presents two decision-based approaches for pupil detection, which were developed during this thesis. Input to the algorithms are 8-bit gray-scale images. As starting point the general work flow is shown for each algorithm, and afterwards each step is described in detail.

3.3.1 ExCuSe

ExCuSe decides based on a intensity histogram analysis which kind of challenge related to automated pupil detection has to be expected. In case of high intensity values, an edge-based approach is selected, where ExCuSe selects the best curved edge based on multiple evaluations and morphologic filtering. If a dark image is expected, the angular integral projection function [159] is applied for coarse positioning. Afterwards, a threshold is computed and applied. The outline is selected based on rays, which are send out from the coarse position. The final step is a least squares ellipse fitting to the selected edges of the outline. The work-flow of the ExCuSe algorithm for pupil detection is depicted in Figure 3.4. Each step is described in detail in the following.

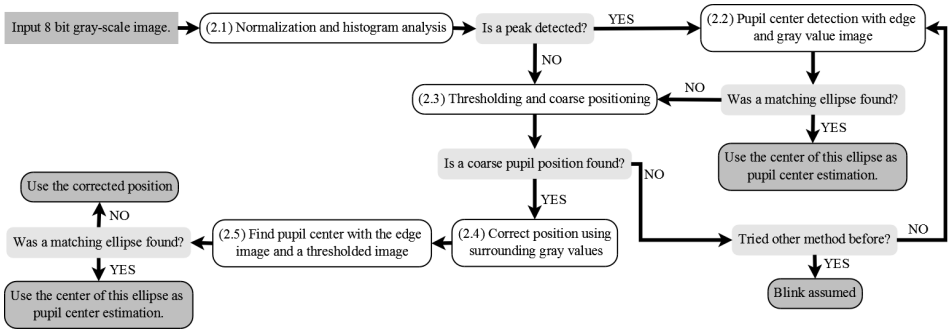


Figure 3.4: The algorithmic work-flow in ExCuSe. Light gray boxes represent decisions, dark gray boxes stand for termination points, and white boxes represent processing steps. [66]

Normalization and histogram analysis

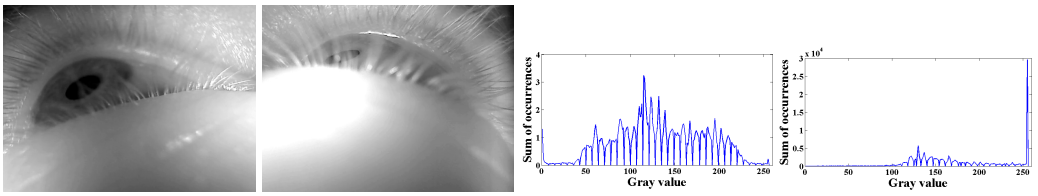


Figure 3.5: Figures 3.5 (a) and (b) show two images from a dataset introduced by Swirski et al. [231] and their corresponding intensity histograms in (c) and (d). Figure 3.5 (b) shows a pupil with a high range of gray values. Eyelashes cover parts of the pupil and reflect the light. [66]

The peripheral regions of the input image (i.e., 10%) are excluded from further processing in order to avoid the frame of eyeglasses. Furthermore, we assume that on images with an overall bright intensity and similar gray values, a reflection on eyeglasses or a bright illumination spot is present. Using an intensity threshold approach to extract the pupil is hard in such cases, since the pupil can not be expected to appear dark and is likely to contain a broad range of intensity values. Thus, in a first step, the input image is normalized (range 0 to 255) and a histogram of the image is calculated. Then the algorithm checks whether the histogram contains a peak in the bright area (Figure 3.5(d)) with a gray value above a threshold th_1 (i.e., $th_1 = 200$ chosen empirically). The peak is detected if a bin in the histogram is higher than a multiple mu_1 of the average image intensity. If such a peak was detected, the pupil can be found based on edge-filtering.

Pupil center detection on edge and gray value image

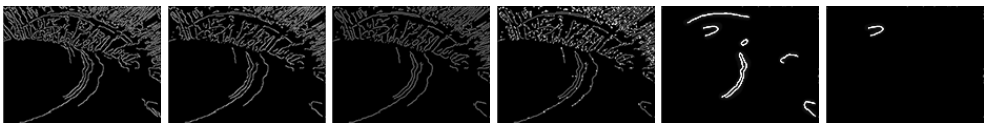


Figure 3.6: (a) A Canny edge filter is applied to the image from Figure 3.5(b). (b) all edge pixels with less than two neighbors and angles between all neighbors $\leq 90°$ are removed. (c) the remaining connected edge pixels represent lines. They are thinned and pixels connecting two lines orthogonally are removed. (d) for each line, the centroid (shown as white point) is inspected and lines close to their centroid are removed (e). (f) the longest line which contains the darkest pixels is assumed to encapsulate the pupil [66].

It is assumed that the pupil appears as a curved edge encapsulating the darkest intensity values of the image. To find such an edge, four processing steps are performed on the Canny-edge-filtered image, Figure 3.6.

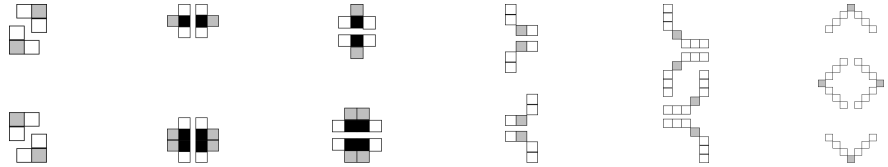


Figure 3.7: Morphologic pixel manipulation patterns. White and gray boxes represent pixels that were detected as edges in the image. If the pattern matches an edge segment, gray pixels are removed and black pixels are added to the edge image. Operand (a) thins lines. Operands (b) and (c) are used to straighten lines. (d), (e) and (f) separate straight parts of a line from curved parts [66].

Filtering the edge image Figure 3.6(a) shows the edge-filtered image from Figure 3.5(b), which appears cluttered and contains many edges that are not relevant for pupil detection. The pupil edges are difficult to detect since they are crossed by the eyelashes. In a first filtering step, thin edge lines (i.e., 1 pixel thickness) and pixels of small rectangular surfaces (2×2 pixels) are removed. More specifically, the above criteria is fulfilled by neighboring pixels which (considered as vectors) have angles greater than 90° between each other.
The remaining edge pixels represent lines which are straight, curved, or consist of both straight and curved parts. The separation step is here of particular interest, e.g., the edge of the eyelashes in the pupil are straight and connected to the curved edge of the pupil. To distinguish between connected line parts that have to be separated and those that do not, the connection point between such parts has to be examined in detail. The assumption is that line parts that have to be separated are orthogonal to each other at the connection point.
To separate lines consisting of curved and straight parts into the corresponding curved and straight segments, the morphologic operations shown in Figure 3.7 are applied. If one of these patterns matches to the edge pixels (shown in gray), that pixel is deleted. Pixels marked black in the figure are added. After thinning, lines can be separated into segments by deletion of just one pixel. However, there are still pixels which prevent the patterns from Figure 3.7(d), (e), or (f) to match. Therefore, lines are straightened using the patterns shown in Figure 3.7(b) and (c). Now the connection points of line parts which are orthogonal to each other can be separated using the patterns in Figure 3.7(d),(e), and (f). The result of this step is shown in Figure 3.6(c).

Remove straight lines The next step is to detect and remove straight lines. Since the pupil is expected to be encapsulated in a curved line, straight lines are of no interest. Therefore, all remaining edge pixels are combined to lines based on their connection to neighboring edge pixels. The steps that are performed to calculate lines from the edge image are the following:

1. Find edge pixels that do not belong to any line yet
2. Create a new line with the edge pixel
3. Add all direct neighbor edge pixels to the line
4. Repeat steps 3 + 4 for all added neighbor pixels

Calculate the line centroid for each line. If the pixel distance between the centroid and at least one point of the line segment is smaller than a threshold di_1, the line is assumed to be straight. Figure 3.6(d) shows all lines with their centroid (white point) and Figure 3.6(e) shows the remaining, curved lines after the removal of straight segments. One of the remaining lines belongs to the pupil.

Choose curved line The pupil is assumed to be a dark spot in the intensity image. Therefore, the pupil candidate with the darkest area contained in it, is most likely to be the pupil. To calculate an intensity value for the contained area, the pixel with a distance of di_2 pixels is selected for each line point which have the smallest Euclidean distance to the line's centroid (i.e., $di_2 = 2$ chosen empirically). For these pixels, the mean gray value is calculated. It is possible that there is more than one curved line belonging to the pupil. The longest line found with the darkest inboard area is selected. To ensure that larger lines are not discarded, a range ra_1 is used in which the mean gray value deemed to be equal (i.e., $ra_1 = 5$). The chosen line is shown in Figure 3.6(f). All points on this line are collected and the center is estimated using ellipse fitting.

Fit ellipse There are basically two ways of fitting an ellipse to a set of points which are described in detail in Section 2.4. For ExCuSe, the direct least squares method for the algebraic ellipse is used. It is fast to calculate and also used as abort criterion on failure for the step (2.2), Figure 3.4.

Thresholding and coarse positioning

If the gray value histogram does not contain a peak (Figure 3.5(c)), the pupil is extracted based on a threshold th_2. Each pixel with a gray value lower than th_2 is set to 255 as shown in Figure 3.8(b). In highly scattered images, the pupil may consist of a range of different intensity values. The threshold th_2 is chosen dependent on the scattering in the image as half the standard deviation of the image intensity. In this step, the goal is to determine the coarse pupil position. It is not necessary to extract the whole pupil. Therefore, a conservative threshold that reduces noise at the potential cost of cutting part of the pupil is preferable. The coarse pupil position is estimated utilizing the Angular Integral Projection Function (AIPF) [159] on the thresholded image. The AIPF allows the calculation of the Integral Projection Function (IPF) for any specified angle and can be understood as a column wise summation in the angular image representation (See also Section 2.6).

$$IPF_h(y) = \int_{x_1}^{x_2} I(x,y)\ \mathrm{d}x \tag{3.1}$$

$$IPF_v(x) = \int_{y_1}^{y_2} I(x,y)\ \mathrm{d}y \tag{3.2}$$

With $I(x,y)$ as the gray value at the location (x,y), Equation 3.1 (as found in [159]) defines the IPF_h (Integral Projection Function horizontally) for the interval $[x_1,x_2]$ and Equation 3.2 define the IPF_v (Integral Projection Function vertically) for the interval $[y_1,y_2]$. The IPF

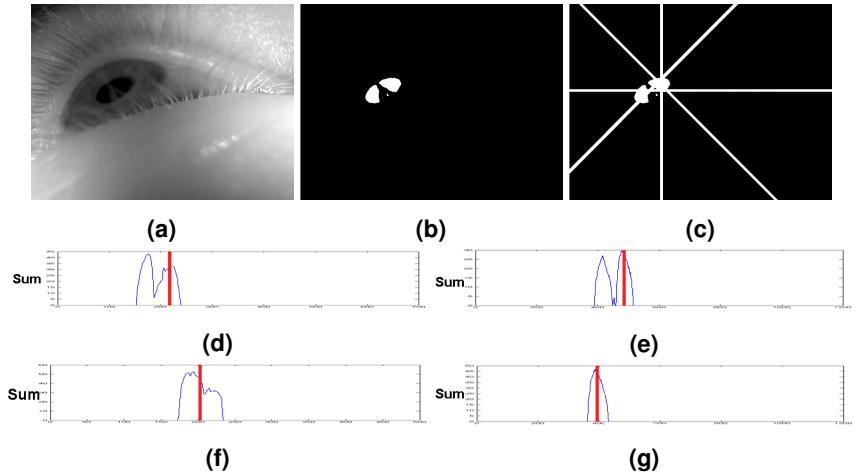


Figure 3.8: (a) An image from the Swirski et al. [231] data set and (b) its corresponding thresholded image. In (c) the coarse positioning (white lines) of the four orientations from the Angular Integral Projection Function (AIPF) [159] are shown. The results of the AIPF calculated on the threshold image for the orientations 0°, 45°, 90° and 135° are shown in the histograms (d), (e), (f), and (g) in corresponding order. The chosen positions are shown as red lines and correspond to the white lines in (c) with (d) defining the vertical white line in (c), (e) the line from the right bottom to the top left corner, (f) to the horizontal line and (g) to the line from the left bottom to the top right corner. [66]

calculates the sum of the intensity values of an image in one direction. For example, outgoing from the x-axes for each row (pixel line from the bottom of the image to the top) the pixel values are summed up and represent one bin in the resulting histogram. Those histograms do not rely on shape and the assumption is that the region with the highest response is the pupil. The AIPF is used because it allows to calculate IPFs for different orientations, it is known to be robust and is fast to calculate. Two well known IPFs are the horizontal (IPF_h) and the vertical (IPF_v) IPF. The IPF_v corresponds to the AIPF with angle 0° and the IPF_h corresponds to the AIPF with angle 90° [159].
With $I(x,y)$ as the gray value at location (x,y), Equation 3.3(as found in [159]) defines the AIPF. Θ is the angle of the line to the x-axis from which the integration rotated by 90° takes place, p is the position on the line or the bin of the corresponding histogram, h is the number of pixels to be integrated and (x_0,y_0) is the position of the start point of the line along which the integration rotated by 90° takes place. The orientations 0°, 45°, 90° and 135° are used for the AIPF to calculate the histograms shown in Figure 3.8(d),(e),(f), and (g).

$$\begin{aligned} AIPF(\Theta,p,h) = \frac{1}{h+1} * \int_{j=-\frac{h}{2}}^{\frac{h}{2}} I\Big(& (x_0 + p\cos\Theta) \\ & +(j\cos(\Theta+90°)), (y_0 + p\sin\Theta) \\ & +(j\sin(\Theta+90°))\Big)\,\mathrm{d}j \end{aligned} \tag{3.3}$$

In these four histograms, the coarse pupil location is assumed at a wide and high response area. The minimum length of the area is specified by ar_1 and the number of consecutive bins allowed to be low is specified by ar_2 ($ar_1 = 7$ and $ar_2 = 5$ are set empirically). This is done to eliminate single high responses in the histogram. Areas of high response are defined by a threshold th_3 which is a percentage of the maximum of the histogram ($th_3 = 0.5$ is estimated empirically). If there is more than one accepted area in a histogram, the assumption is that the pupil can be found at the center of the image. Therefore, the midpoint of the area which is closest to the bin in the histogram corresponding to the center of the image is chosen as the pupil position. The white lines in Figure 3.8(c) represent the angle of the AIPF for each histogram rotated by 90° (angle of the integration) and are the chosen positions. Therefore, these white lines correspond to the red lines from Figures 3.8(d), (e), (f), and (g) drawn to the threshold image shown in Figure 3.8(b). The pupil position is estimated based on the intersection of these lines. The assumption is that the intersection of those lines orthogonal to each other are close to or hit the pupil. This way, up to two intersection points are considered. The pupil position is assumed as the point between these intersections. In the case that no intersection was found, branch *2.2* of the algorithm will take over. If this branch fails to detect the pupil as well, a blink is assumed.

Correct position using surrounding gray values

Once a coarse pupil center estimation has been established, it has to be improved because it is possible that the coarse position lays outside or on the border of the pupil. It can be refined within a small area ar_3 around the estimation without being dependent on the shape or color of the pupil. The only assumption made is that pixels belonging to the pupil are surrounded by brighter or equally bright pixels. This step is important for images in which the pupil is especially hard to detect.

$$PS(x,y) = \sum_{x_i=x_1}^{x_2} \sum_{y_i=y_1}^{y_2} \begin{cases} I(x,y) - I(x_i,y_i), I(x_i,y_i) < I(x,y) \\ 0, I(x_i,y_i) >= I(x,y) \end{cases} \tag{3.4}$$

For each pixel, the sum $PS(x,y)$ of gray value differences to its neighbors is calculated. Only gray values lower than the value of the pixel under consideration are taken into account. For the neighborhood area, the square root of the diagonal of the area specified by ar_3 is used. The mean of the pixel positions with the lowest sum value is the new corrected position. In Equation 3.4, $PS(x,y)$ is the sum calculated for the pixel at position (x,y), $[x_1,x_2]$ is the interval on the x-axis of the neighborhood area, $[y_1,y_2]$ is the interval on the y-axis and $I(x,y)$ is again the gray value at position (x,y).

Find pupil center with the edge and threshold image

This step does not require the corrected position to be the accurate pupil center, however it is required to lie inside of the pupil. The concept of using a threshold image to improve the edge image and refine finding the pupil edges with rays outgoing from this position is described in the following chapter. Only the region ar_4 around the corrected point is of interest for finding the pupil. Only eight rays are used. This is because for too many rays

it is more likely that rays hit edges not belonging to the pupil if the edges belonging to the pupil are not consistently present. Therefore, rays missing the pupil edges can hit other edges that do not belong to the pupil thus making the pupil center detection incorrect.



Figure 3.9: The edge-filtered eye image (a) of the region where the pupil is expected. A threshold image (b) is calculated to determine the threshold border (c). Only edge pixels close to this border are used for further calculations (d). After the edge refinement steps explained in *2.2.1* and *2.2.2*, the remaining edges are used for edge selection (e): rays are sent out from the corrected point (white point in the middle of (f)) into all directions with an angle step of 45°. If a ray hits an edge (white points on the ellipse in (f)) the line belonging to this edge is supposed to belong to the pupil. [66]

Improve edge image with threshold image First, an edge-filter of the image region is calculated, as shown in Figure 3.9(a). The calculated threshold image from step *2.3* is not useful here because the threshold chosen was for coarse positioning and there was no need to extract the whole pupil. In this step, the threshold th_2 (chosen as half the standard deviation) is increased to the full standard deviation to calculate the new threshold image (Figure 3.9(b)). In this step it is important that no part of the pupil gets cut off by an too conservative threshold. Edges of the edge image are preselected by overlay with the threshold image. Only edges close to the border of the threshold region (Figure 3.9(c)) are considered relevant. This border is calculated by accepting only white pixels in the threshold image which have black direct neighbors. Only edges close to the border region are considered, see Figure 3.9(d). To calculate this, the surrounding area ar_5 of each threshold border pixel is inspected. If an edge pixel lies within the ar_5 region of a threshold border pixel it is accepted. Then the border refinement steps are carried out (the result is shown in Figure 3.9(e)).

Find edges that represent the pupil border In the resulting edge image, rays from the corrected position (small white point in the middle of Figure 3.9(f)) are sent in all directions with an angle step of 45° until they hit an edge (white points on the elliptic line in Figure 3.9(f)), similar to the method used by the Starburst algorithm [136]. The intersection points between the rays and the edges are used to collect points. All edge pixels connected to a hit edge pixel and iteratively all that are connected to those pixels are used to fit an ellipse.

3.3.2 ElSe

ElSe is based on ExCuSe, where the edge break up is improved as well as the edge selection. Additionally, the edge break up is formulated in an algorithm instead of only using

morphologic operations. The second step which mainly detects pupil if the contour edges cannot be found is replaced by a weighted blob detection.

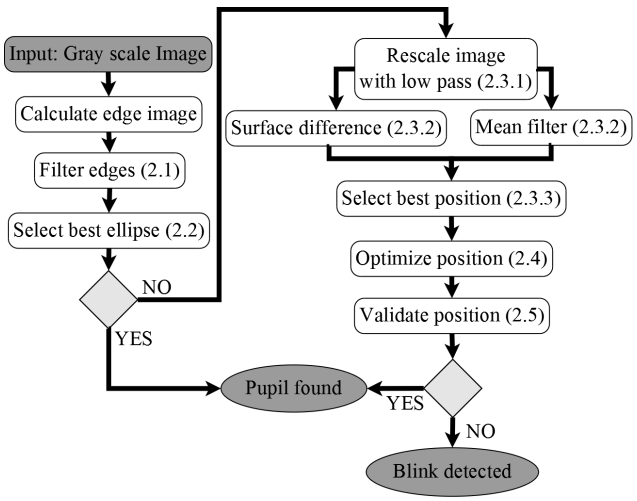


Figure 3.10: Flowchart of the algorithm. Light gray boxes represent decisions, dark gray ellipses termination points, and white boxes represent processing steps. [79]

After normalization, a Canny edge filter is applied to the eye image (Figure 3.10). In the next algorithmic step (Step 2.1 in Figure 3.10), edge connections that could impair the surrounding edge of the pupil are removed. Afterwards, in Step 2.2, connected edges are collected and evaluated based on straightness, inner intensity value, elliptic properties, the possibility to fit an ellipse to it, and a pupil plausibility check. If a valid ellipse describing the pupil is found, it is returned as the result. In case no ellipse is found (e.g., when the edge filtering does not result in suitable edges), a second analysis is conducted. To speed up the convolution with the surface difference (Step 2.3.2) and mean filter (Step 2.3.2), the image is downscaled (Step 2.3.1). After applying the surface difference and mean filter to the rescaled image, the best position is selected (Step 2.3.3) by multiplying the result of both filters and selecting the maximum position. Choosing a pixel position in the downscaled image leads to a distance error of the pupil center in the full scale image. Therefore, the position has to be optimized on the full scale image (Step 2.4) based on an analysis of the surrounding pixels of the chosen position. In the following, each of the above mentioned steps is described in detail.

Filter edges

Edges are split up at positions that do not occur in an ellipse, e.g., orthogonal connectors and edge points with more than two neighbors. Additionally, edges are thinned and straightened in order to improve the breaking procedure based on two approaches (morphologic and algorithmic). Both approaches lead to comparable results. In the provided implementation, ElSe uses the morphologic approach since it requires less computational power.

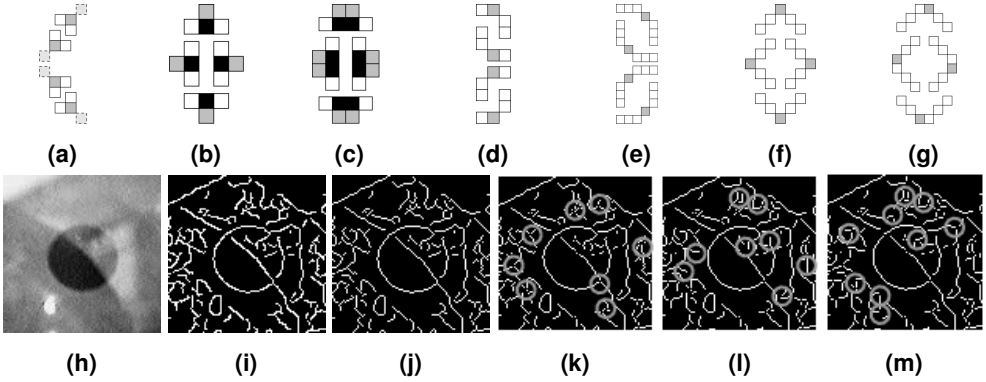


Figure 3.11: Morphologic patterns for edge manipulation. White and dark gray boxes represent pixels that have to remain edge pixels. Light gray boxes with dashed borders (a) represent pixels that have to be removed. If the pattern matches a segment in the edge image, pixels under dark gray boxes are removed, and pixels under black boxes are added to the edge image. The pattern in (a) thins lines, whereas patterns (b) and (c) straightens lines. The patterns (d), (e), (f), and (g) are applied to break up orthogonal connections. (h) input image, (i) edge filtered results, (j) edges after thinning using the morphologic pattern from (a). (k) remaining edges after deleting all edges with too many neighbors. (l) result of edge straightening by applying the operations shown in (b) and (c). (m) result after deleting edge pixels that connect orthogonal by means of the morphologic patterns shown in (d), (e), (f), and (g). [79]

Morphologic approach The employed morphologic operations in Figures 3.11b, 3.11c, 3.11d, and 3.11e are similar to those introduced in ExCuSe [66]. However, in contrast to ExCuSe, no preprocessing based on deletion of edges with low angle is performed. Furthermore, a stable thinning procedure (Figure 3.11a) and deletion of edges with too many neighbors is used. The morphologic processing starts with edge-thinning using the pattern shown in Figure 3.11a. Figure 3.11j presents the result of thinning applied on the Canny edge image from Figure 3.11i. Afterwards, the direct neighborhood of each edge pixel is summed up. If this neighborhood is > 2, the edge pixel is deleted because it has joined more than two lines. Applied to the result from the thinning step, Figure 3.11k shows the remaining edge pixels. Next, a refinement step is performed by applying the straightening patterns in Figure 3.11b and 3.11c, yielding the edges in Figure 3.11l. Then, the patterns shown in Figure 3.11d, 3.11e, 3.11f, and 3.11g are applied; as a result, the orthogonal connections in consecutive edge pixels are separated by deleting the connecting pixel, resulting in Figure 3.11m.

Algorithmic approach The algorithmic approach to filtering the edge image is based on the idea of breaking up lines at positions where the line course cannot belong to a common ellipse. Prerequisites here are edge-thinning, breaking up lines with too many neighbors, and line straightening as described previously. The algorithm starts with calculating the vector orthogonal to the first two points of a line (solid arrow in Figure 3.12a). For each following point, the vector from the starting point is calculated (dashed arrow in Figure 3.12a). Afterwards, the angle and distance between the orthogonal and the calculated vector is

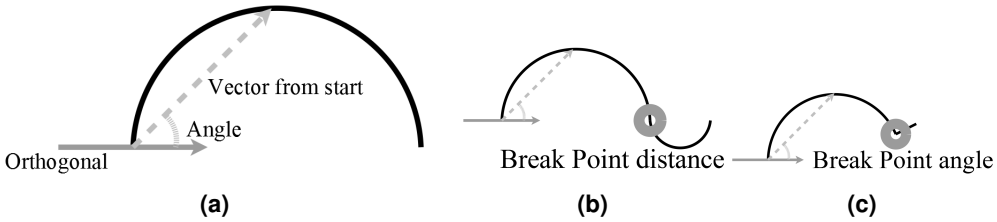


Figure 3.12: In (a), the gray arrow is the calculated orthogonality, whereas the dashed gray arrow is the vector between the starting and the current point. The black line represents the processed edge. As the gray dashed arrow moves along the edge, the angle to the orthogonal decreases, whereas the length of the vector increases. (b) distance breaking and (c) angle breaking condition is triggered. [79]

computed. For an ellipse, this angle has to shrink from 90° to 0°. Once the angle has reached 0°, it has to grow back to 90° whereas the distance has to shrink. If this is not the case in the beginning, the orthogonal vector has to be turned over. In case the shrinking and growing do not apply to the behavior of the line, a point where the edge has to be split is found (Figure 3.12b and 3.12c). This is shown in more detail in the provided pseudocode in Algorithm 1.

Select best ellipse

In this step, all consecutive edge pixels in the edge image are collected. For the morphologic approach, this is done by combining all connected edge pixels into a line. In the algorithmic approach, closed lines can be excluded to decrease runtime. Therefore, open lines (start and end pixel have only one neighbor) and closed lines have to be separated. Open lines are collected by starting new lines only on pixels with one neighbor and closed lines are collected by starting at any pixel not accessed in the first step. These lines are evaluated based on their shape, the resulting shape after an ellipse fit, and the image intensity enclosed by the ellipse.



Figure 3.13: (a) input image and (b) edge-filtered result. For each line, it is analyzed whether it is curved based on the centroid of its pixels. The result is shown in (c) with the mean positions as bright dots. Then the algorithm fits an ellipse to the line. In case of success, the ellipse is further analyzed. Remaining lines after this fitting step are shown in (d). The first evaluation of the ellipse filters stretched ellipses by comparing the ratio of the two ellipse radii. The result is shown in (e). For the pupil area restriction a maximum and minimum percentage of pixels in the image is used as parameters. Picture (f) shows the remaining contour after this step [79].

Algorithm 1 Separation of lines collected from the edge image. [79]

Require: $DIR = 0, START, ORTHO, ANGLE, DIST$
  function *BreakLines*($LINE$)
    for IDX *to Size*($LINE$) **do**
      $VEC = CALC(START, LINE(IDX))$
      if $DIR = 0$ **then**
        if $ANGLE >= ANGLE(ORTHO, VEC) AND DIST <= |VEC|$ **then**
          $ANGLE = ANGLE(ORTHO, VEC)$
          $DIST = |VEC|$
        else
          $ADD_BREAK_POINT()$
        end if

        if $ANGLE = DIRECTION$ **then**
          $DIR = 90$
        end if
      else
        if $ANGLE <= ANGLE(ORTHO, VEC) AND DIST >= |VEC|$ **then**
          $ANGLE = ANGLE(ORTHO, VEC)$
          $DIST = |VEC|$
        else
          $ADD_BREAK_POINT()$
        end if
      end if
    end for
    return $BREAK_POINTS$
  end function

Remove straight lines Since pupil contours exhibit a round or elliptical shape, straight lines have to be removed. It is analyzed whether each line is straight or curved based on the mean position of all pixels that belong to the line. If the shortest distance of a line pixel to the mean position is below an empirically set threshold *min_mean_line_dist*, the line is straight. Note that this decision is taken for both x and y dimensions. An example of this step is shown in Figure 3.13c, where the mean position is represented by a white dot.

Ellipse fitting To fit an ellipse, we employed a direct least squares ellipse fit as in [59]. An example result is shown in Figure 3.13d.



(a) **(b)**

Figure 3.14: Calculation of the difference between the inner and outer area of an ellipse [79].

Ellipse evaluation In this step, ellipses that are unlikely to describe the pupil are excluded. The first restriction pertains the shape of the pupil by restricting the ratio between the two ellipse radii. The rationale is that the pupil position relative to the eye tracker camera can only distort the pupil ellipse eccentricity to a certain point. The second restriction regards the pupil area in relation to the image size, since the eye tracker camera has to be positioned at a restricted distance from the eye (neither to close nor to far), which is reflected on the ratio of the image area occupied by the pupil. Two thresholds are used, namely $min_{area} = 0.5\%$ and $max_{area} = 10\%$ of the total image area [79]. Due to eye physiology, the last evaluation step expects the pupil to be darker than its surroundings. Figure 3.14a shows the calculated pattern based on the radius of the ellipse. To reduce computation time, only a portion of the minimum enclosing, unrotated rectangle is considered, as shown in Figure 3.14a. Pixels within the gray box in Figure 3.14a contribute to the pupil intensity value and those within the black box contribute to the surrounding intensity. The size of the gray box is $\frac{1}{2}$ of the width and height of the enclosing rectangle [79]. The white box in Figure 3.14a has the size of the enclosing rectangle and the surrounding black box has $\frac{3}{2}$ of this size.
To evaluate the validity of an ellipse, the surface difference of the pupil box and the surrounding box is calculated (as shown in Figure 3.14a). This difference is compared against a threshold [79].

Rate ellipse All found ellipses have to be compared against each other. For this, the inner gray value of each ellipse is computed by calculating a vector between each point of the line and the center of the ellipse [79]. This vector is shortened by multiplying it stepwise from 0.95 to 0.80 with a step size of 0.01. Figure 3.14b shows the line pixels in gray and all

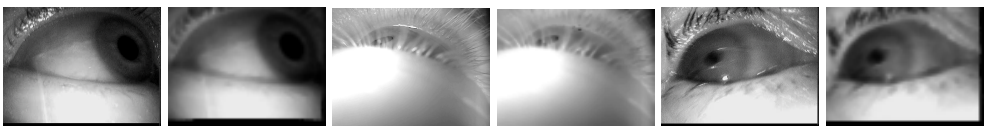
pixels contributing to the inner gray value ($gray_{value}$ in Equation (3.5)) in white. Note that each pixel can only contribute once to the inner gray value [79]. This value is normalized by the sum of all contributing pixels.

$$eval(el) = gray_{value} * (1 + |el_{width} - el_{height}|) \tag{3.5}$$

The best of all remaining ellipses is chosen by selecting the ellipse with the lowest inner gray value and the roundest shape. Equation (3.5) shows the formula for calculating the rank of an ellipse, where el is the ellipse, and el_{width}, el_{height} are the radii of the ellipse. If el_{width} and el_{height} are equal the ellipse is round [79]. The variable $gray_{value}$ in Equation (3.5) is the calculated inner gray value as specified before. The ellipse with the lowest value calculated based on Equation (3.5) is chosen. If there is more than one ellipse with the lowest value, the one with the most edge points and therefore clearest contour is chosen [79].

Coarse positioning

A different approach is chosen if the algorithm cannot find a good pupil edge – e.g., due to motion blur, the pupil being located in a dark spot, or the pupil being hidden by eyelashes. More specifically, an additional method that tries to find the pupil by first determining a likely location candidate and then refining this position is applied. Since a computationally demanding convolution operation is required, the image is downscaled to keep run-time traceable (see Section 3.3.2). This rescaling process contains a low pass procedure to preserve dark regions and to reduce the effect of blurring or eyelashes. Afterwards, the image is convolved with two different filters separately: 1) a surface difference filter to calculate the area difference between an inner circle and a surrounding box, and 2) a mean filter. The results of both convolutions are multiplied, and the maximum value is set as the starting point of the refinement step.



(a) (b) (c) (d) (e) (f)

Figure 3.15: (a), (c) and (e) show input images taken from the data set proposed by [66] and [231]. The respective results of the downscaling operation by a factor of six using the mean between zero and the mean of the input image region influencing a pixel are shown in (b), (d) and (f) [79].

Rescale image with low pass There are several methods to downscale an image, e.g., based on nearest neighbor, bilinear or bicubic interpolations, based on Lanczos kernel or more advanced downscaling operations like content adaptive [125] or clustering based [90] downscaling. In case the edge detection part of the algorithm could not find a good edge because of motion blur (Figure 3.15e) or eyelashes (Figure 3.15c), a downscaling operation that weights dark pixels stronger would be preferable. However, considering that the pupil could also be in a dark region of the image (as in Figure 3.15a), weighting dark pixels too

strong can lead to a merging of the pupil and the surrounding dark region. A fast method to calculate the intensity histogram and the mean (Equation (3.6)) of the pixels influencing the new pixel is applied. Afterwards, the mean of the lower part of the histogram (defined as the part smaller than the mean of the whole histogram) is computed (Equation (3.7)). The resulting value is used as the intensity of the new pixel. This method weights dark pixels stronger based on the intensity distribution of the influencing area (specified by x_1, y_1, x_2 and y_2). $I(x_i, y_i)$ denotes the intensity value of a pixel.

$$Mean(x_1, y_1, x_2, y_2) = \frac{\sum_{x_i=x_1}^{x_2} \sum_{y_i=y_1}^{y_2} I(x_i, y_i)}{|x_1 - x_2| * |y_1 - y_2|} \tag{3.6}$$

$$MUM(x_1, y_1, x_2, y_2) = \frac{\sum_{x_i=0}^{Mean(x_1, y_1, x_2, y_2)} IH(x_i) * x_i}{\sum_{x_i=0}^{Mean(x_1, y_1, x_2, y_2)} IH(x_i)} \tag{3.7}$$

Equation 3.7 yields the mean neighborhood intensity of the dark neighborhood regions, where darkness is defined by the mean calculated in Equation (3.6). Therefore, it uses the intensity histogram of the region which is denoted as $IH(x_i)$ and the intensity index denoted as x_i. Overlapping regions are used with a window $radius_{scale} = 5$ (Figure 3.16a). The overlapping regions do not include the center of the other boxes, and therefore, $radius_{scale} = 5$ downscales an image by a factor of six (Figure 3.16b).

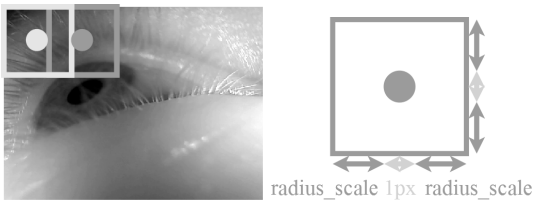


Figure 3.16: (a) shows how neighborhood regions of pixels close to each other in the downscaled image can overlap (light gray box and dark gray box). Each gray box represents the pixels influencing the intensity of a pixel in the downscaled image. The circles represent the center of a region. In (b) the construction of the window based on the parameter $radius_{scale}$ is shown [79].

Convolution filters The convolution filters used are a mean (Figure 3.17a) and a surface difference filter (Figure 3.17b). Because of the unknown shape and expected roundness of the pupil, both filters contain the shape of a circle. The algorithm expects the input image to contain the complete eye, and therefore, the expected pupil size depends on image resolution. To calculate the parameter $radius_{filter}$, the resolution is divided in the x and y dimension of the image by 100. Afterwards, the maximum of these two values is rounded up and used as the parameter $radius_{filter}$. The construction of the filters based on this value ($radius_{filter} = radius$) is shown in Figure 3.17c.
The diameter of such a circle in the real image is

$$(radius_{scale} + 1) * (radius_{filter} * 2 + 1), \tag{3.8}$$

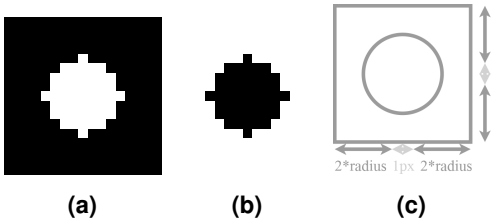


Figure 3.17: (a) mean filter, where the white region's sum is one and the black region's is zero. (b) surface difference filter, where the black inner circle sums up to minus one and the surrounding white to one. Both kernels have the same size. (c) filter construction, where the radius is calculated based on the image resolution. [79]

which is expected to be larger than the real pupil. This is important for the surface difference filter (Figure 3.17b) because, on larger pupils, the result in the middle would be lower than the result closer to the border of the pupil.

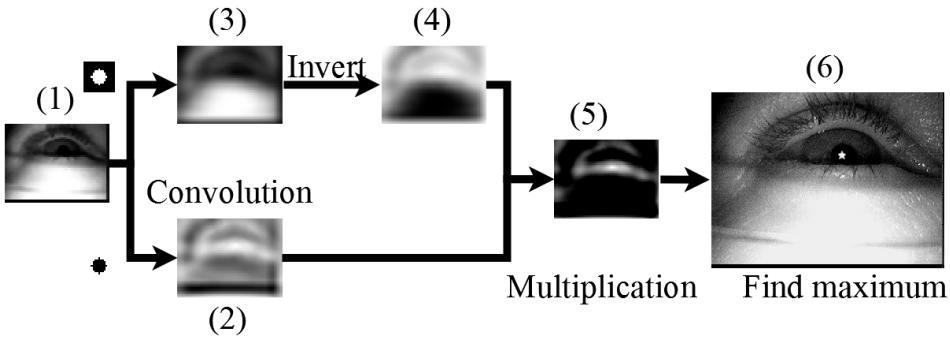
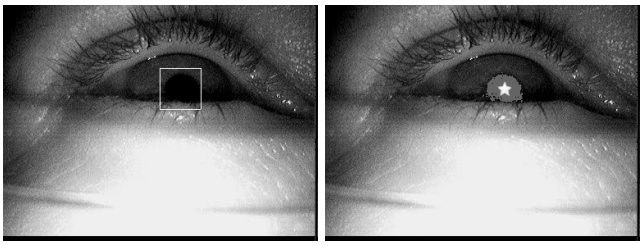


Figure 3.18: Workflow after downscaling of the coarse positioning [79]. The input (1) is the downscaled image. (2) result of the convolution with the surface difference filter (Figure 3.17a). (3) convolution result of the mean filter (Figure 3.17b) and (4) inverted image. The result (5) is the point-wise multiplication of (2) and (4). The absolute maximum of (5) is represented by a white cross in the real image (6) taken from the data set proposed in [66].

Select best position To find the best fitting position of the pupil, first the downscaled image is convolved with the surface difference filter (Figure 3.17b). All areas with low intensity values in the inner circle of the filter (black in Figure 3.17b) and high values in the surrounding area will have positive results (white in Figure 3.18(2)). The bigger this difference is, the higher the convolution response. The idea behind this is that the pupil is surrounded by brighter intensity values. Problems with this filter are that other areas respond also with positive values and the filter response does not include intensity information of the inner area (black in Figure 3.17b). To find the pupil, which is expected to be dark, the mean filter (Figure 3.17a) is used to include the intensity response of the inner area (Figure 3.18(3)). To achieve this, the result of the convolution with the mean filter has to be inverted (Figure 3.18(4)). This is because the response of areas with low intensity is low, and, to use it as weight for the result of the surface difference filter, it has to be high.

The weighting is done by pointwise multiplication of the two convolution responses of the inverted mean (Figure 3.18(4)) and the surface difference filter (Figure 3.18(2)). In the result of the weighting (Figure 3.18(5)) the maximum is searched and used as coarse position (white cross in Figure 3.18(6)).
By inverting the surface difference and not inverting the mean filter, this algorithm searches for a white blob. Additionally, by reducing the filter size and operating only on a small area surrounding the pupil center position, it can be used for cornea reflection detection.

Optimize position



(a) (b)

Figure 3.19: In (a), the area, in which the optimization takes place is enclosed by a white box. (b) shows the pixels below the calculated threshold (dark gray area) and the resulting position (white cross) [79].

The coarse position is based on the downscaled image, and, therefore, one pixel error relative to the pupil center represents a distance of six pixels in the original image. For the optimization step, the coarse position is expected to be contained within the pupil and calculate a pupil intensity threshold using the neighborhood of the coarse position in the real image. The mean of this box is calculated, and the absolute difference to the pixel value of the coarse position is computed. This difference is added to the coarse position pixel value and used as a threshold. To optimize the position, a small window (Figure 3.19a, white box) surrounding the coarse position in the real image is thresholded. The dark gray area in Figure 3.19b shows this thresholded region. The window size is chosen to be $radius_{filter} * radius_{filter}$ in each direction empirically. Afterwards, the center of mass of the thresholded pixels is calculated and used as pupil center position (white cross in Figure 3.19b).

Validate position

The second method will always find a pupil location, even if the eye is currently closed. Therefore, the candidate location has to be validated. This is done in the same way as for the ellipse validation shown in Figure 3.14a. For the two diameters of the ellipse $radius_{filter} * radius_{filter} * 2 + 1$ is used. The parameter $validity_{threshold}$ is set to the value previously defined (i.e., 10).



(a) Body **(b)** Mouth piece **(c)** Pedal

Figure 3.20: Control inputs from the Opmi Pentero 900 and 800 operating microscopes (source Carl Zeiss AG https://www.zeiss.com) [85].

3.3.3 Pupil detection on microscopy images

Whereas eye tracking is already applied in the field of microsurgery (e.g., incorporated into LASIK's[1] delivery systems to lessen the effect of patient eye movement [234]), little to no attention has been given to the opposite perspective of the microscope (i.e., the surgeon's). From this point of view, there is much to be gained. Throughout microsurgery, typically all of the surgeon's extremities are busy; a modern operating microscope includes features for directional movements, zooming, focusing, and illumination, which the surgeon manages through several controls in the microscope body, foot pedal, or mouth pieces (Figure 3.20). Thus, eye tracking as an additional input method would be immensely convenient and potentially yield multiple benefits. By enabling the surgeon to move the system faster and effortlessly, fatigue and operation time are diminished. Naturally, a less fatigued surgeon is less likely to perform harmful mistakes [177], [242] and faster operation times decrease the risk of infection [28], [47]. Moreover, surgeon fatigue could also be assessed from the eye tracking data by means of pupillometric information. Additionally, the gaze information can be easily shared with other co-observers (e.g., co-surgeon), this also eliminates the need for verbal cues (e.g., "see the nerve at 3 o'clock"), which may be misleading due to different image orientations between surgeon and co-observer [212]. Furthermore, scanpaths of expert surgeons can be integrated into educational systems, thus speeding up the learning process for students and novice surgeons [130]. Tracking the eyes of a microscope user differ significantly from head-mounted or remote eye tracking as described previously. The size and placement of the microscope impedes the utilization of these traditional eye tracking approaches. Instead, an imaging device can be coupled within the microscope optics in order to obtain images from the user's eyes, resulting in images from a perspective inside the microscope eyepiece (Figure 3.21a). From this perspective, the main challenge is that only a small part of the eye (and often pupil) are visible due to occlusion by the eyepiece. Although this issue can be attenuated by a redesign of the optics, such a procedure is expensive and time consuming. Since this type of recording and the images are drastically different from conventional systems, the first step is to explain the recording system.

[1] Laser-assisted in situ keratomileusis

Microscopy imaging system

Figure 3.22 shows the setup design of the imaging system. The red box surrounds the pupil monitoring. The light reflecting back from the surgeons eyes into the ocular is projected on the two cameras, which are on the left and right ends. This projection is done using beam splitters. The white box in Figure 3.22 represents the path of the surgeons vision onto a digital display. Limitations of the system are the field of view onto the eyes of the surgeon. This area is limited to 4x4 mm^2 and caused by the small openings in the ocular itself. Another challenge which arises for surgical microscopes is the movement of the surgeons head. The surgeon has to move his head to increase his field of view without moving the microscope which is also caused by the small openings in the ocular. Due to this only a small portion of the pupil is visible. The third limitation of the system is an restricted depth of field (1.4mm). This comes from the lens setup internally. Additional challenges are changing illumination conditions and blurred images.

3.3.4 Pupil detection methodology

For pupil detection on microscopy images, we developed a decision based approach similar to ExCuSe and ElSe. This had to be used due to the fact that there is not enough ground truth data available. In fact, the process of annotating data is a laborious and time consuming task [85].
The workflow of the pupil detection algorithm for microscopes is shown in Figure 3.23. In the following subsections, each step is described in detail.

Preprocessing

The preprocessing consists of three steps. In the first step, the image is smoothed using a Gaussian filter to reduce noise. Then, the image is inverted and squared, serving as a high pass filter. Because of the high variations in the input images, the third step consists of averaging the result of the second step. This assigns larger weights to intensity values higher than the Gaussian function and is, therefore, more robust to intensity fluctuations that can origin from the high pass calculation [85].



(a) Microscope **(b)** Remote **(c)** Remote eye box **(d)** Head-mounted

Figure 3.21: Sample images from remote, head-mounted, and microscope-integrated eye tracking. Note that only a small part of the eye and pupil are visible in the microscope image (the dashed yellow line represents the contour of the eyepiece) [85].

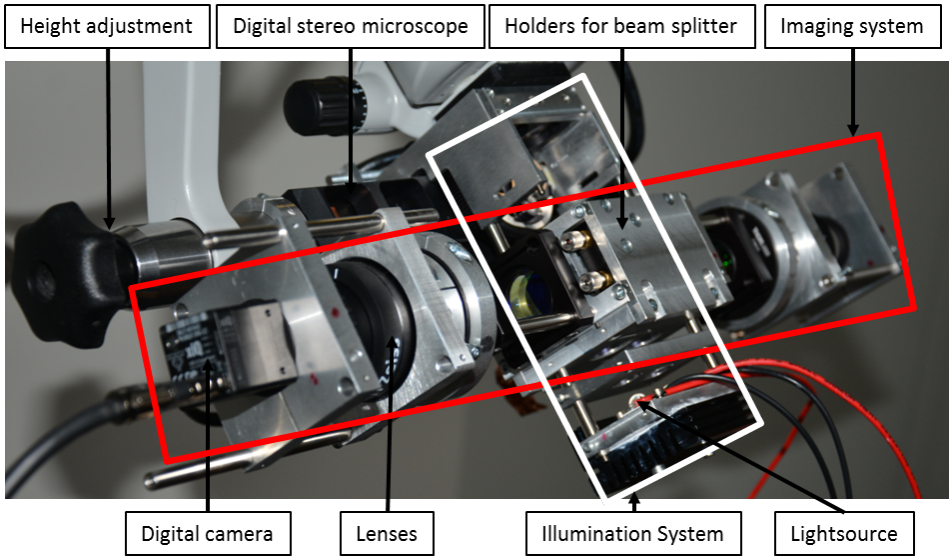


Figure 3.22: Pupil monitoring and imaging system for a surgical microscope ocular [85].

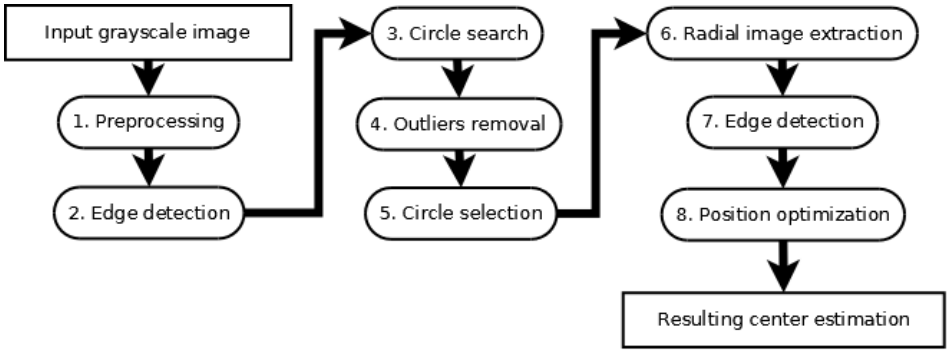
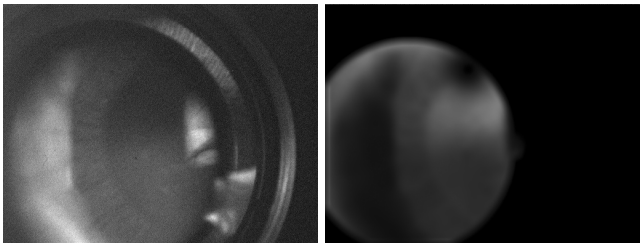


Figure 3.23: The algorithmic workflow for pupil center detection. Rounded boxes represent processing steps and squared boxes are the input and output. [85]

In Equations (3.9), (3.10) and (3.11), X and Y represent image coordinates, X_0 and Y_0 are the coordinates of the center position, σ is the standard deviation. The first step represents the Gaussian convolution $I(X,Y) * Gauss(X,Y,\sigma)$, where $I(X,Y)$ is the inverted input image. The second step is described by Equation (3.11), in which the first step can be seem in the denominator [85]. The operation conducted in the last step is represented by 3.11, in which t represents the size of the box for the averaging function.

$$Gauss(X,Y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(X-X_0)^2+(Y-Y_0)^2}{2\sigma^2}} \tag{3.9}$$

$$Highpass(X,Y) = \frac{1}{(1+(I(X,Y) * Gauss(X,Y,\sigma)))^2} \tag{3.10}$$

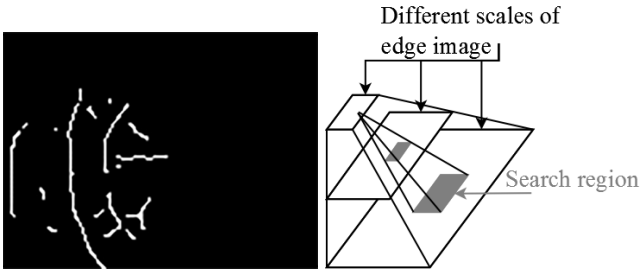


(a) (b)

Figure 3.24: (a) shows the input grayscale image (recorded from the imaging system in Figure 3.22). In (b) the image after preprocessing is shown [85].

$$Box(X,Y,t) = \begin{cases} 0, |X - X_0| > t \\ 0, |Y - Y_0| > t \\ 1, otherwise \end{cases} \tag{3.11}$$

Edge detection



(a) (b)

Figure 3.25: (a) shows the result of the edge detection (combination of all results from the pyramid in (b)). (b) shows the pyramid of edge images, where those are represented as black boxes. The dark gray boxes represent search regions for edge filtering in a higher resolution scale, meaning that edges have to be present in all gray boxes to be valid [85].

To extract edges, a Canny operator [32] was applied and combined with a multi scale enhancement technique inspired by edge focusing as presented in [16]. First, the image shown in Fig. 3.24b is downscaled to different scales as in [43]. Afterwards, the Canny edge detector was applied to each scale. A valid edge pixel has to be present in each scale; otherwise, it is discarded from further processing. This process is shown in Figure 3.25b, where the black boxes represent the different scales of the edge image, and the dark gray boxes represent search regions in higher scales to decide if a edge pixel is valid or not [85].

$$Edge(e) = \begin{cases} 1, \forall i \exists a \in S_i, a = e \\ 0, otherwise \end{cases} \tag{3.12}$$

In Equation (3.12), S_i represents the edge image in scale i and e the extracted edge pixel. The result of this step for Fig. 3.24b can be seen in Figure 3.25a.

Circle search

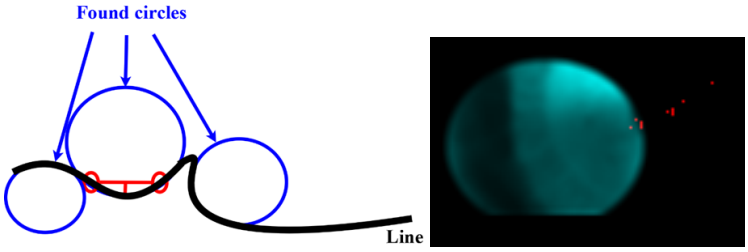


Figure 3.26: (a) shows the workflow of the circle search. The black line represents edge pixels, while the red circles represent two selected edge pixels. The red lines show an exemplary circle tendon and circle segment height which are used for circle radius calculation (blue circles). In (b), valid circle votes are drawn as red dots (those are the centers of all valid blue circles) [85].

To estimate the pupil center, all possible circles are searched since the pupil has to be round and not elliptical. To accomplish this, each line is inspected and, for all possible pairs of line points, search for a possible circle segment height outgoing from the center of both selected points. The segment height is searched orthogonally to the vector between both points [85].

$$Radius(tendon,h) = \frac{4*h^2+tendon^2}{8*h} \tag{3.13}$$

Equation (3.13) represents the circle radius calculation based on tendon and segment height. *tendon* is the length of the tendon of the circle segment and h the height of this segment (red lines in Figure 3.26a). All radii that are too high or too low to be a possible pupil radius are discarded. Found valid pupil centers are shown in Figure 3.26b as red dots.

Outliers removal

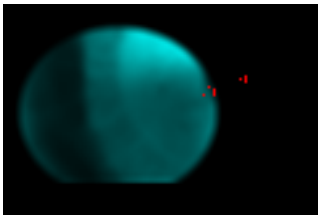


Figure 3.27: All remaining pupil center candidates (red dots) after outliers removal [85].

Outliers detection is done under the assumption that most of all votes belonging to a line are valid and that the variation between those votes follows a normal distribution [85].

Therefore, the outliers removal is done based on the 95% quantile on the standard deviation of all votes of a line. Fig. 3.27 shows the remaining votes from Figure 3.26b after outlier removal.

$$\tau = \frac{q_{0.95} * (n-1)}{\sqrt{n} * \sqrt{(n-2) * (q_{0.95}^2)}} \tag{3.14}$$

$$\phi = std(centers) * \tau \tag{3.15}$$

$$Outlier(i) = \begin{cases} 1, |centers(i) - mean(centers)| > \phi \\ 0, else \end{cases} \tag{3.16}$$

Equations (3.14), (3.15) and (3.16) describe this iterative process. After each outliers removal step, the parameters τ and ϕ have to be recalculated. In Equation (3.14), $q_{0.95}$ represents the 95% quantile, whereas n represents the number of found centers for a line. In Equations (3.15) and (3.16), std represents the standard deviation and i the index of a found pupil center. This step can also remove correct center estimates if an image is blurred and the edges are inaccurate [85]. Therefore, an upper bound of votes per line was used for which no outlier selection has to be performed.

Circle selection

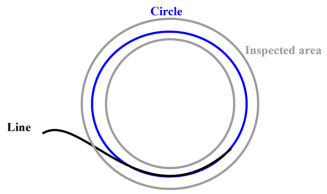


Figure 3.28: Recalculation of the edge value outgoing from a possible pupil center. The black line represents a found edge in Step 2. The gray area is the inspected area for edge value calculation and the blue line represents the calculated circle [85].

To select the pupil center, all remaining votes are inspected in an inverse manner. Outgoing from the estimated center and the calculated radius, the edge value is recalculated. The idea behind this step is that non-connected edges can vote for the same center [85]. The gray circular ring in Figure 3.28 represents the area that is inspected for one center vote. In this area, the difference between the closest and the furthest part is calculated and summed up. This is only done if an edge pixel is present. The closest part is defined as the intensity values before this edge pixel, and the furthest part is defined by the intensity values after the edge pixel outgoing from the center [85].

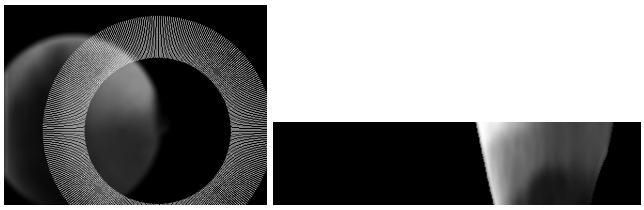
$$RI(c, \alpha, r) = I(c_x + cos(\alpha) * r, c_y + sin(\alpha) * r) \tag{3.17}$$

$$V(c,\alpha) = \begin{cases} \frac{1}{e-1-r_{min}} \sum_{i=r_{min}}^{e-1} RI(c,\alpha,i) - \\ \frac{1}{r_{max}-e+1} \sum_{i=e+1}^{r_{max}} RI(c,\alpha,i), \\ \exists e \in RI(c,\alpha,r_{min} : r_{max}) \\ \\ 0, else \end{cases} \tag{3.18}$$

$$EV(c) = \frac{1}{2*\pi} \sum_{i=0}^{2*\pi} V(c,i) \tag{3.19}$$

Equation (3.17) describes the transformation from a center point c to its radial intensity value specified by the radius r and the angle α. In Equation (3.18), these values are used to calculate the intensity difference between two areas in Figure 3.28: 1) the area between the inner ring (given by r_{min}) and the the edge pixel e (*Line*) and 2) the area between e and the outer ring (given by r_{max}). Equation (3.19) uses this function to calculate the complete intensity difference of the circular ring. The center with the highest edge value is selected as pupil center. To correct for small pupil center jitter, the found position is stabilized in the subsequent steps.

Radial image extraction



(a) **(b)**

Figure 3.29: In (a), the white lines represent the extracted image parts. (b) is the resulting image segment where the height is the radius, and the widths are the different angles. Therefore lining up the white lines in (a) result in (b) [85].

The first step for position stabilization is the extraction of a radial image using Equation (3.17). The selected image lines are shown as white lines in Figure 3.29a. The resulting radial image is shown in Figure 3.29b. The height of the image shown in Figure 3.29b is the length of the white lines shown in Figure 3.29a and represents therefore the radius. For each angle, one line is added to Figure 3.29b; thus, the widths represents the angles [85].

Edge detection

For edge detection in the radial image, only vertical intensity differences have to be calculated. Therefore, the image is calculated by subtracting the intensity value of the inspected



(a) (b)

Figure 3.30: In (a) the magnitude image of the radial image is shown, which is used for edge detection (intensity differences calculated on Figure 3.29(b)). (b) shows the found maximal responses as white dots, which are the detected edges [85].

pixel from the subsequent value (vertically) [85].

$$ERI(X,Y) = |I(X,Y) - I(X,Y+1)| \tag{3.20}$$

In Equation (3.20), this calculation is shown with X and Y representing the coordinates, and $I(X,Y)$ the intensity value in the radial image. The result is shown in Figure 3.30a. For each vertical line in this image, the maximum is searched and marked as a white dot in Figure 3.30b. These white pixels are the maximal votes for each angle, whereas each white dot represent a radius [85].

Position optimization

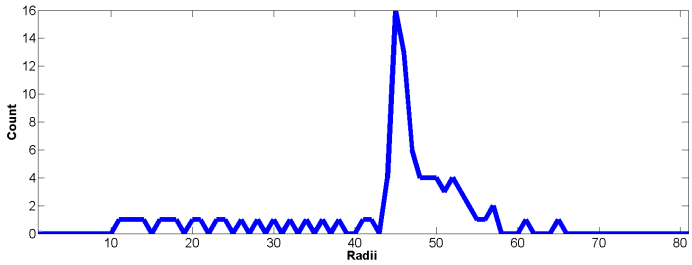


Figure 3.31: All found radii (white dots in Figure 3.30b) as depicted in a histogram (downscaled radii), where the x axis corresponds to the radii and the y axis the amount of occurrences of this radii [85].

All of the maximas found in the edge detection step are collected in a histogram of all found radii (Figure 3.31). In this histogram, the smallest segment with at least one third of all radi is searched [85]. All edge values in the selected segment are collected, and a least squares circle fit is applied. The resulting circle center is used as pupil center estimation.

$$a*X + b*Y + c = -(X^2 + Y^2) \tag{3.21}$$

Equation (3.21) shows the linear equation system that has to be solved, where X and Y represent all collected circle border coordinates. The center of the circle is given by $(-\frac{a}{2}, -\frac{b}{2})$, and $sqrt(\frac{a^2+b^2}{4} - c)$ is the radius [85].

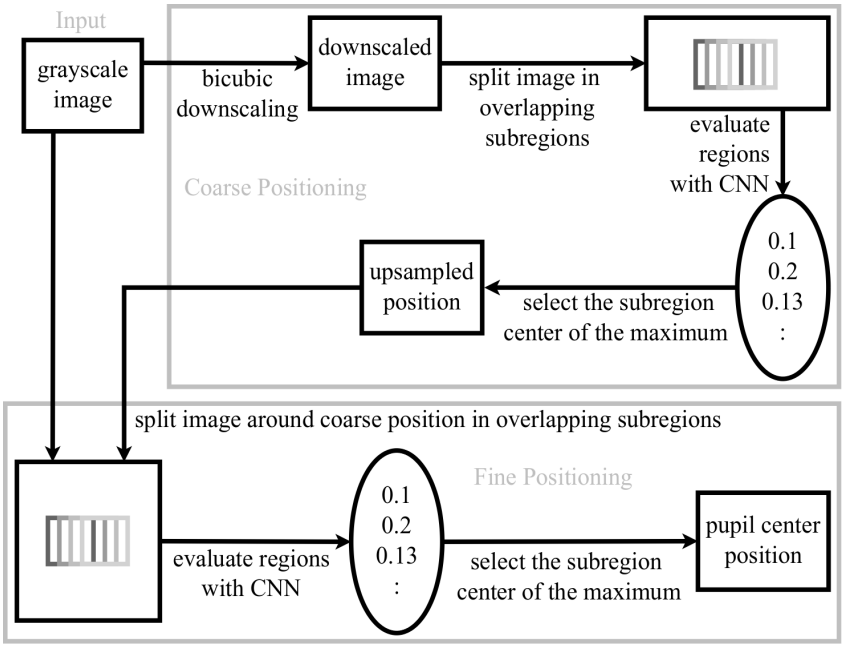


Figure 3.32: Workflow of PupilNet. First a CNN is employed to estimate a coarse pupil position based on subregions from a downscaled version of the input image. This position is then refined using subregions around the coarse estimation in the original input image by a second CNN [83].

3.4 PupilNet: Pupil detection based on CNNs

As mentioned previously throughout this chapter, image-based pupil detection for the purpose of gaze tracking faces various challenges, such as motion blur, out of focus images, and nearly closed eyes, hanging eyelids, poor image quality, or low resolution. In such images, edge-based approaches will very likely fail. Convolutional neural networks promis to be superior to conventional approaches due to their robustness. With this motivation, PupilNet, a CNN-based approach for pupil detection, was developed during this thesis and will be described in this section. Although PupilNet proved to be superior to other approaches (Section 3.5) a typical disadvantage of CNNs are high computational costs. In a revised version of PupilNet, we further introduced an approach which runs in real-time on only one CPU core.

The overall workflow for PupilNet is shown in Figure 3.32. In the first stage, the image is downscaled and divided into overlapping subregions. These subregions are evaluated by the first CNN, and the center of the subregion that evokes the highest CNN response is used as a coarse pupil position estimate. Afterwards, this initial estimate is fed into the second pipeline stage. In this stage, subregions surrounding the initial estimate of the pupil position in the original input image are evaluated using a second CNN. The center of the subregion that evokes the highest CNN response is chosen as the final pupil center location. This two-step approach has the advantage that the first step (i.e., coarse positioning) has to handle less noise because of the bicubic downscaling of the image and, consequently,

involves less computational costs than detecting the pupil on the complete upscaled image. In the following subsections, these pipeline stages and their CNN structures are described in detail, followed by the training procedure employed for each CNN.

3.4.1 Coarse positioning stage

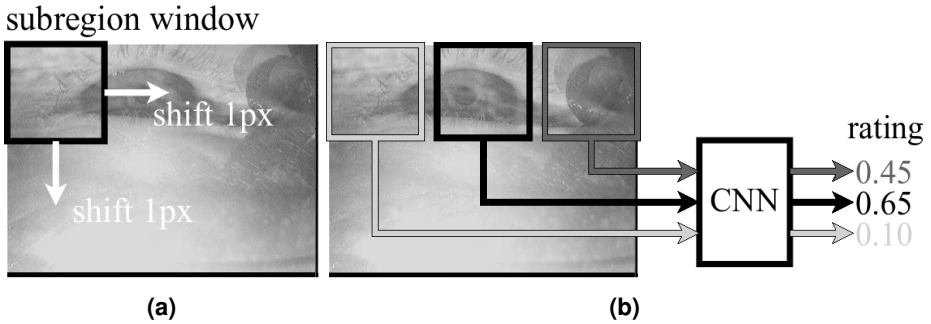


Figure 3.33: The downscaled image is first divided in subregions (24×24) with a stride of one pixel (a), which are then rated by the first stage CNN (b) [83].

Directly employing CNNs on images of this size would demand a large amount of resources and, thus, would be computationally expensive, impeding their usage in state-of-the-art mobile eye trackers. Thus, one of the purposes of the first stage is to reduce computational costs by providing a coarse estimate that can in turn be used to reduce the search space of the exact pupil location. However, the main reason for this step is to reduce noise, which can be induced by different camera distances, changing sensory systems between head-mounted eye trackers [20], [45], [193], movement of the camera itself, or the usage of uncalibrated cameras (e.g., out of focus, unbalanced white levels). To achieve this goal, first the input image is downscaled (four times) using a bicubic interpolation, which employs a third order polynomial in a two dimensional space to evaluate the resulting values. Given that these images contain the entire eye, the CNN input size of 25×25 pixels has to guarantee that the pupil is fully contained within a subregion of the downscaled images.
Subregions of the downscaled image are extracted by shifting a 25×25 pixels window with a stride of one pixel (see Fig. 3.33a) and evaluated by the CNN, resulting in a rating within the interval [0,1] (see Fig. 3.33b). These ratings represent the confidence of the CNN that the pupil center is within the subregion. Thus, the center of the highest rated subregion is chosen as the coarse pupil location estimation.
The core architecture of the first stage CNN is summarized in Fig. 3.34. The first layer is a convolutional layer with kernel size 6×6 pixels, one pixel stride, and no padding. The convolution layer is followed by an average pooling layer with window size 4×4 pixels and four pixels stride, which is connected to a fully-connected layer with depth one. This layer is approximated using again a convolution stage. The reason behind this is that the averaging of the convolutions is more robust and can be learned more easily than a fully connected neural network. The output is then fed to a single perceptron, responsible for yielding the final rating within the interval [0,1]. The main idea behind the selected architecture is

that the convolutional layer learns basic features, such as edges, approximating the pupil structure. The average pooling layer makes the CNN robust to small translations and blurring of these features (e.g., due to the initial downscaling of the input image). The fully connected layer incorporates deeper knowledge on how to combine the learned features for the coarse detection of the pupil position by using the logistic activation function to produce the final rating.

3.4.2 Fine positioning stage

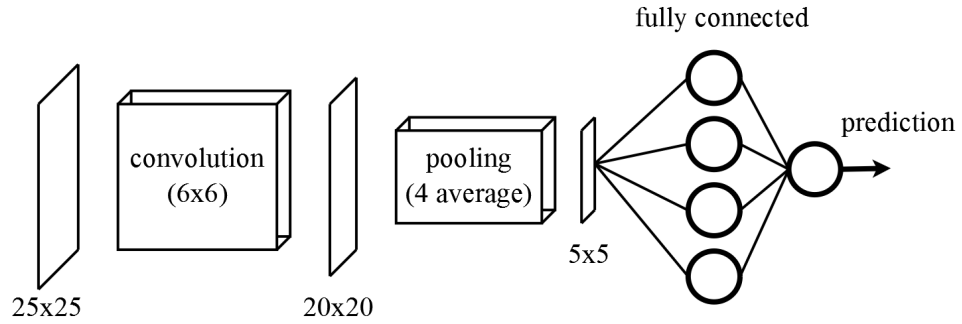


Figure 3.34: The coarse position stage CNN. The first layer consists of the shared weights or convolution masks, which are summarized by the average pooling layer. Then a fully connected layer combines the features forwarded from the previous layer and delegates the final rating to a single perceptron [83].

Although the first stage yields an accurate pupil position estimate, it lacks precision due to the inherent error introduced by the downscaling step. Therefore, it is necessary to refine this estimate. This refinement could be attempted by applying methods similar to those described in Section 3.1 to a small window around the coarse pupil position estimate. However, since most of the previously mentioned challenges are not alleviated by using this small window, a second CNN that evaluates subregions surrounding the coarse estimate in the original image is applied.
The second stage CNN employs the same architecture pattern as the first stage (i.e., convolution $\Rightarrow$ average pooling $\Rightarrow$ fully connected $\Rightarrow$ single logistic perceptron) since their motivations are analogous. Nevertheless, this CNN operates on a larger input resolution to increase accuracy and precision. Intuitively, the input image for this CNN would be 100×100 pixels: the input size of the first CNN input (25×25) multiplied by the downscaling factor (4). However, the resulting memory requirement for this size was larger than available on the test device; the closest working size possible: 89×89 pixels is used. The size of the other layers were adapted accordingly. The convolution kernels in the first layer were enlarged to 20 pixels to compensate for increased noise and motion blur. The dimension of the pooling window was increased by one pixel on each side, leading to a decreased input size on the last convolution layer and reduced runtime. This CNN uses eight convolution filters and eight perceptrons due to the increased size of the convolution filter and the input region size. Subregions surrounding the coarse pupil position are extracted based on

a window of size 89×89 pixels centered around the coarse estimate, which is shifted in a radius of 10 pixels (with a one pixel stride) horizontally and vertically. Analogously to the first stage, the center of the region with the highest CNN rating is selected as fine pupil position estimate.

3.4.3 CNN training methodology

Both CNNs were trained using supervised batch gradient descent [132] with a dynamic learning rate from 10^{-1} to 10^{-6}. The learning rate was dropped after each ten epochs by 10^{-1}. In the first round, the training was set to 50 epochs and afterwards the best performing CNN on the validation set was selected. This was repeated four times and in each new round the starting learning rate was decreased by a factor of 10^{-1}. After the last round, fine tuning by inspecting each iteration additionally was performed. In each round a new training set was generated. The batch size for one iteration was 100 and all CNNs' weights were initialized using a Gaussian with standard deviation of 0.01.

Training set creation

The coarse position CNN was trained on subregions extracted from the downscaled input images that fall into two different data classes: containing a valid ($label = 1$) or invalid ($label = 0$) pupil center. Training subregions were extracted by collecting all subregions with center distant up to twenty four pixels from the hand-labeled pupil center. In the first round of training, only half of the distance was used to reduce the amount of invalid examples. Subregions with center distant up to three pixels were labeled as valid examples, while the remaining subregions were labeled as invalid examples. This procedure results in an unbalanced set of valid and invalid examples, therefore only samples from both diagonals (top left to bottom right and reverse) are used, where every second was discarded in case of the invalid samples. This reduces the amount of samples per frame. Due to the huge size difference of the data sets, the amount of samples per set was reduced to 20,000 for the first round, and 40,000 for the others. Therefore, randomly two thousand images per data set were selected.

Fast fine accuracy improvement

The main idea here is to use the response of the coarse CNN surrounding the maximum value for estimating the best pupil center location directly. For a fast accuracy improvement of all CNNs, the response surrounding the maximum position is converted into a probability distribution. Such a response of a CNN is shown in Figure 3.35 on the top left. The converted area is surrounded by a green square. The resulting distribution is shown in Figure 3.35 on the bottom right. To convert the response into a distribution, each value is divided by the sum of all values in the square (Equation 3.22).

$$D(x,y) = \frac{R(x,y)}{\sum_{i=0}^{n}\sum_{j=0}^{n} R(i,j)} \tag{3.22}$$

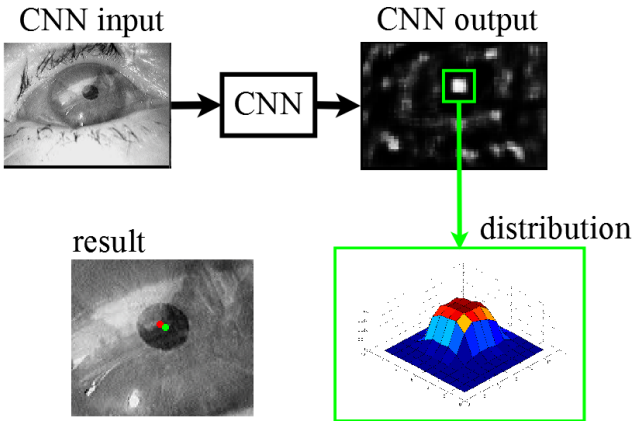


Figure 3.35: The workflow of the accuracy improvement. On the top left the input image is shown. Right represents the output of the CNN. For accuracy improvement, the surrounding area of the maximum position is converted to a distribution and a shift vector is computed. This distribution is shown in the green box on the bottom right. On the bottom left, the maximum position (red dot) and the shifted position (green dot) are shown [83].

In Equation 3.22, $D(x,y)$ represents the distribution value at location x,y, and $R(x,y)$ the CNN response at location x,y. Each value in this distribution is weighted by the displacement vector to the maximum position. The calculation is shown in Equation 3.23.

$$\overrightarrow{SV} = \sum_{i=-\frac{n}{2}}^{\frac{n}{2}} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}} D(\frac{n}{2}+i, \frac{n}{2}+j) * \begin{pmatrix} i \\ j \end{pmatrix} \tag{3.23}$$

In Equation 3.23 $\overrightarrow{SV}$ is the vector shifting the initial maximum position (red dot in Figure 3.35 on the bottom left) to the new more accurate position (green dot in Figure 3.35 on the bottom left). $D(i,j)$ is the result of Equation 3.22 at location i,j and $\begin{pmatrix} i \\ j \end{pmatrix}$ is the displacement vector to the center.

3.5 Evaluation of pupil detection algorithms on head mounted eye-tracking images

3.5.1 Data sets

The proposed algorithms were evaluated in the realm of head-mounted eye-tracking on five data set. The main characteristics of these data sets are summarized in Table 3.1 and will be briefly described in the following:

Świrski

The data set introduced by Świrski et al. [231] provides 600 manually annotated eye images. The data was collected during indoor experiments with 2 subjects and different angles. The main challenges are highly off-axial camera position and eyelid occlusion.

ExCuSe

This data set was introduced with the ExCuSe [66] algorithms. It consists of 38,401 manually annotated eye images from 17 different subjects. Exemplary images are shown in Figure 3.36. The first nine data sets in ExCuSe were recorded during an on-road driving experiment [118]. The remaining eight data sets were recorded during a supermarket search task [217]. The main challenge in this data set are changing illumination conditions, reflections on glasses, and contact lenses.

ElSe

The data set introduced with the ElSe [79] algorithm consists of 55,712 eye images. All of those recording where obtained during an on-road driving experiment [118] except XXIII and XXIV. Those two data sets where recorded indoor from two asian subjects, where XXIV contains additional reflections. The challenges in the eye images include motion blur, reflections, and low pupil contrast.

PupilNet

Together with PupilNet [83], 41,212 eye images were published. These images stem from an on-road driving experiment [118] and were recorded using a Dikablis head-mounted eye tracker. The main challenge in those data sets are reflections and changing illumination conditions.

Labeled pupils in the wild (LPW)

Labeled Pupils in the Wild (LPW) [240] is a data set which contains 66 eye region videos from 22 subjects. For recording the Pupil Labs [120] eye-tracker was used. Each video consists of about 2,000 frames recorded with 95 fps. This results in 130,856 labeled eye images. The data set is the largest data set and covers a wide range of realistic indoor and outdoor illumination conditions. The challenges regarding image processing are glasses, make-up, variable skin tones, eye colors, and face shapes.

3.5.2 Experimental results

The results will be reported in terms of detection rates (percentage of image frames where the pupil was detected) in relation to the pixel distance between the detected and annotated pupil. Table 3.2 summarizes the performance of the evaluated algorithms on each data set. The average detection rates (for a pixel distance of five) per data set of the evaluated

algorithms are shown in Figure 3.37. For the algorithm from Świrski, we replaced the random number generator to make the algorithm more stable and it also improved the results. As can be seen, PupilNet outperforms all the other algorithms. For the data set from LPW PupilNet was not evaluated since the focus is on demonstrating feasibility and a further twenty-two training sessions and evaluation phases represent a high computational effort. In addition, each training phase is more than doubled, even for the already evaluated data sets. This is because the training data set is doubled with the additional use of LPW [240]. A detailed performance analyses on each data set is visualized in Figure 3.37. The highest detection rates are achieved either by PupilNet or ElSe. The highest overall detection rates are achieved on the data set by Świrski et al. [231] data set. Since this data set was collected in a laboratory setting, it is however the least challenging, although most of the contained eye images are highly off-axial. For this data set, the algorithms ExCuSe, ElSe, and Swirksi reach a detection rate far beyond 70% at a pixel distance of 5. With a detection rate of 86.17% (Table 3.2), ExCuSe is the best performing algorithm among the state-of-the-art. The data sets ExCuSe, ElSe, and LPW provide a large corpus of eye images collected in outdoor scenarios and represent the various challenges that have to be faced when head-mounted eye trackers are employed in such settings. Figure 3.37b shows the evaluation results on the ExCuSe data set. For this data set, PupilNet is the best performing algorithm with a detection rate of 80% at a pixel error of 5. The ElSe and ExCuSe algorithms achieved also good detection rates of about 75% and 55% respectively, whereas the remaining algorithms show detection rates that are lower than 30%. Due to the many sources of noise summarized in Table 3.1, the ElSe and PupilNet data set contain the most challenging eye images. The overall performance of the pupil detection algorithms is therefore quite poor. The best detection rate was also achived by PupilNet (70%) for a pixel error of 5. ElSe (50%) and ExCuSe (35%) are the best decision-based approaches, while the remaining algorithms show detection rates of at most 10%. According to the evaluation results on the LPW data set (Figure 3.37d), ElSe proves to be the most robust algorithm when employed in outdoor scenarios. At a pixel error of 5, ElSe shows a detection rate of 70%. Good detection rates (approximately 50%) are also achieved by the algorithms ExCuSe and Swirksi, whereas the remaining approaches have detection rates below 40%.

Table 3.1: Five publicly available data sets containing 266,781 ground-truth eye images were employed for the evaluation of pupil detection algorithms. Explicit challenges associated with each data set are mentioned shortly [86].

Data set		Images	Description
Świrski [231]		600	Highly off-axis, pupil occluded by eye lashes
ExCuSe [66]	I	6.554	Reflections
	II	505	Reflections, changing illumination conditions
	III	9,799	Reflections, recording errors, bad illumination conditions
	IV	2,655	Contact lenses, bad illumination
	V	2,135	Shifted contact lenses
	VI	4,400	Bad illumination, mascara
	VII	4,890	Bad illumination, mascara, eyeshadow
	VIII	630	Bad illumination, pupil occluded by eyelashes
	IX	2,831	Reflections, additional black dot on iris
	X	840	Bad illumination, highly of-axis (pupil at image boarder)
	XI	655	Reflections, bad illumination, additional black dot on iris
	XII	524	Bad illumination
	XIII	491	Bad illumination, eyelashes covering pupil
	XIV	469	Bad illumination
	XV	363	Shifted contact lenses
	XVI	392	Mascara, eyelashes
	XVII	268	Bad illumination, eyelashes covering pupil
ElSe [79]	XVIII	10.794	Reflections, changing illumination conditions
	XIX	13,474	Reflections
	XX	10,344	Reflections, changing illumination conditions
	XXI	9,133	Bad illumination, Reflections
	XXII	10,370	Reflections, changing illumination conditions
	XXIII	636	Asian subject
	XXIV	961	Asian subject, reflections
PupilNet [83]	NEW I	12,169	Reflections, additional black dot on iris
	NEW II	7,031	Bad illumination, Reflections, changing illumination conditions
	NEW III	8,779	Reflections, changing illumination conditions
	NEW IV	8,690	Reflections, changing illumination conditions
	NEW V	4,543	Dark, nearly closed eyes
LPW [240]	1	5,999	Changing illumination conditions, eyelashes
	2	6,000	Bad illumination
	3	6,000	Reflections, changing illumination conditions
	4	6,000	Bad illumination, off-axis, reflections, pupil at border
	5	6,000	Border of glasses covering pupil, blurred images
	6	5,999	Eyelashes covering pupil, pupil at border
	7	6,000	Reflections, small pupil, bad illumination, pupil at border
	8	6,000	Reflections, small pupil, bad illumination, pupil at border
	9	6,000	Eyelashes covering pupil, pupil at border
	10	6,000	Highly off-axis, bad illumination, pupil at border
	11	6,000	Eyelashes covering pupil, pupil at border
	12	6,000	Bad illumination, pupil at border
	13	5,127	Reflections, bad illumination, eyelashes covering pupil
	14	6,000	Highly off-axis, eyelashes covering pupil
	15	6,000	Reflections
	16	5,731	Reflections, bad illumination
	17	6,000	Highly off-axis, eyelashes covering pupil, bad illumination
	18	6,000	Reflections, bad illumination
	19	6,000	Highly off-axis, pupil at border, upside down, reflections
	20	6,000	Bad illumination, pupil at border
	21	6,000	Pupil at image border
	22	6,000	Mascara, pupil at border

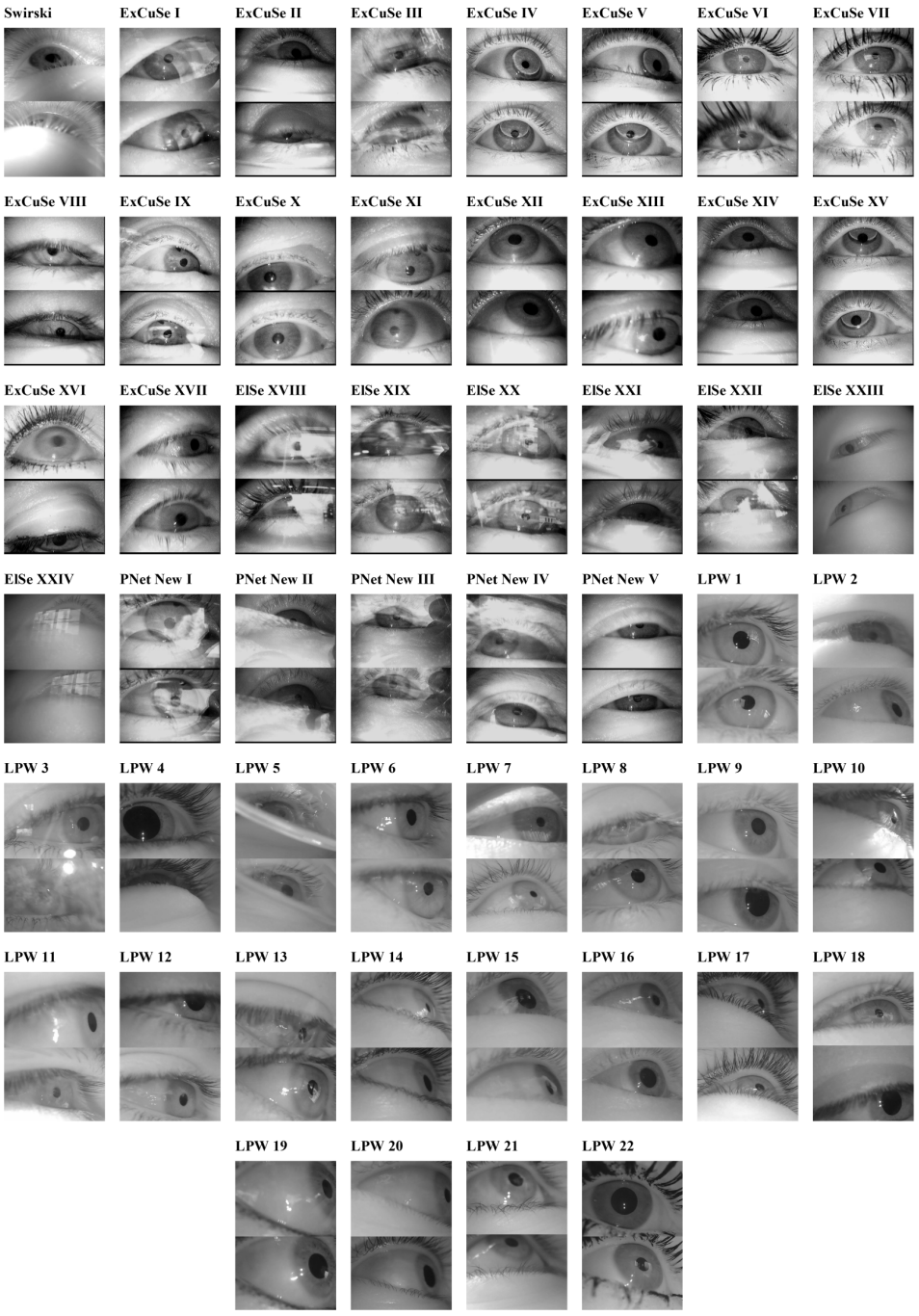


Figure 3.36: Example images of the data sets provided by [231],[66], [79], [83] and, [240].

Table 3.2: Performance comparison of SET, Starburst, Świrski, ExCuSe, ElSe and PupilNet in terms of detection rate up to an error of five pixels (in (%)). The best performance on each data set is shown in bold.

Data set		*ELSE*	*EXCUSE*	*PNET$_{fast}$*	*PNET*	*SET*	*STARBURST*	*SWIRSKI*
Świrski [231]		82	***87***	–	–	59	20	78
	I	***86***	72	77	82	7	5	5
	II	65	40	***80***	79	45	3	23
	III	64	38	62	***66***	12	2	7
	IV	83	80	90	***92***	2	4	35
	V	85	76	91	***92***	18	14	78
	VI	78	60	73	***79***	5	18	19
	VII	60	49	73	***73***	2	2	39
	VIII	68	55	***84***	81	38	8	41
ExCuSe [66]	IX	***87***	76	86	86	10	12	24
	X	79	79	80	***81***	57	51	31
	XI	75	58	85	***91***	24	22	20
	XII	79	80	***87***	85	56	64	71
	XIII	74	69	79	***83***	22	45	61
	XIV	84	68	91	***95***	47	19	52
	XV	57	56	***81***	81	39	9	63
	XVI	60	35	80	***80***	53	8	19
	XVII	90	79	***99***	97	91	1	66
	XVIII	57	24	55	***62***	0	3	15
	XIX	33	23	34	***37***	5	5	9
	XX	78	58	79	***79***	4	5	23
ElSe [79]	XXI	47	52	81	***83***	3	2	8
	XXII	53	26	50	***58***	2	7	2
	XXIII	94	93	86	90	0	7	***96***
	XXIV	53	46	46	***55***	0	2	44
	new I	62	22	69	***69***	5	8	7
	new II	26	16	44	***45***	2	1	1
PupilNet [83]	new III	39	34	45	***49***	2	1	4
	new IV	54	48	***83***	82	6	2	5
	new V	75	59	78	***81***	42	0	2
	1	81	67	–	–	39	41	***84***
	2	39	26	–	–	0	20	***41***
	3	***42***	38	–	–	0	7	31
	4	24	28	–	–	***37***	10	17
	5	***23***	18	–	–	0	0	8
	6	56	57	–	–	59	13	***63***
	7	65	***69***	–	–	36	8	66
	8	***79***	78	–	–	35	36	78
	9	***60***	56	–	–	0	31	56
	10	58	62	–	–	62	3	***71***
LPW [240]	11	***49***	49	–	–	22	18	31
	12	***81***	71	–	–	32	26	72
	13	44	***44***	–	–	27	16	27
	14	69	74	–	–	52	24	***76***
	15	***53***	44	–	–	2	9	38
	16	***82***	73	–	–	2	20	74
	17	63	40	–	–	48	2	***68***
	18	***76***	68	–	–	32	35	61
	19	24	21	–	–	***27***	3	25
	20	17	13	–	–	34	17	***41***
	21	53	44	–	–	***57***	27	56
	22	***71***	63	–	–	0	13	6

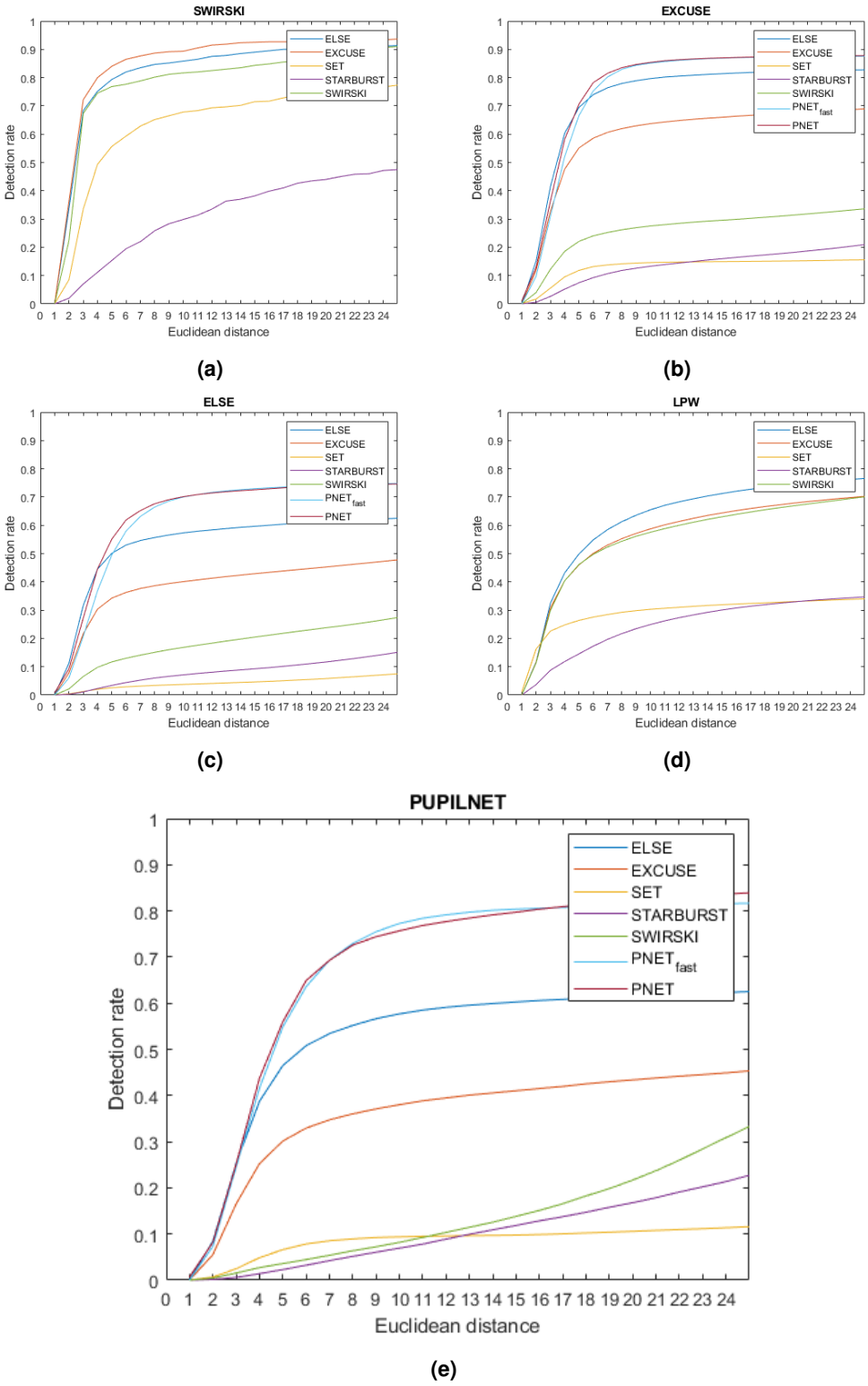


Figure 3.37: Detection rates of the algorithms ElSe, ExCuSe, PupilNet, SET, Starburst, and Świrski for each of the data sets described in Table 3.1.

3.6 Evaluation of pupil detection algorithms on remote eye-tracking images

3.6.1 Data sets

For this evaluation we employed three data sets which will be described in the following.

BioID



Figure 3.38: Example images from the BioID data set [73].

The BioID data set consists of 1521 grayscale images from 23 subjects with a resolution of 384x286. It was recorded in an office environment with varying illumination. The challenges in the data set are different camera distances, blinks, and small head movements. Figure 3.38 shows example images from the BioID data set.

GI4E

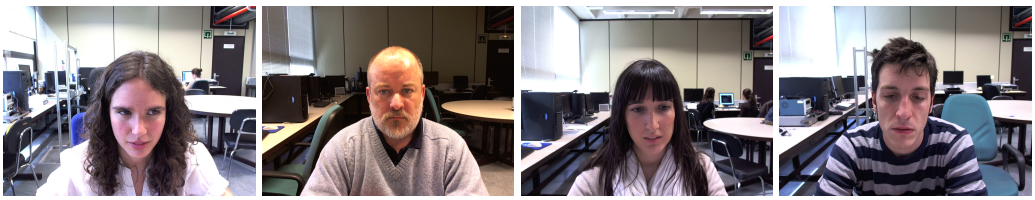


Figure 3.39: Example images from the GI4E data set [73].

The GI4E [245] data set provides 1236 RGB images from 103 subjects with a resolution of 1280x720. The data set was recorded using a web cam. Each subject was recorded looking at different screen locations. Figure 3.39 shows four example images.

Own data set [73]

In [73], a data set consisting of 445 manually labeled images was proposed. Each image has a resolution of 1280x720 pixels. The recordings in this data set consist of RGB and near infrared images. The used camera was a pan tilt zoom camera for observation (FOSCAM FL9826P). The challenges in the data set are head and eye movements as well as blinks and reflections. Examples of this data set are shown in Figure 3.40. In Figure 3.41 exemplary challenging frames are shown.



Figure 3.40: Example images from the data set [73]. The left two images are grayscale converted RGB images. The remaining ones are infrared recorded [73].



(a) Pupil covered by reflection

(b) Off-axial camera perspective

(c) Pupil indistinguishable from iris

(d) Bright pupil effect

Figure 3.41: Challenges posed in the data set [73] [73].

3.6.2 Evaluation procedure

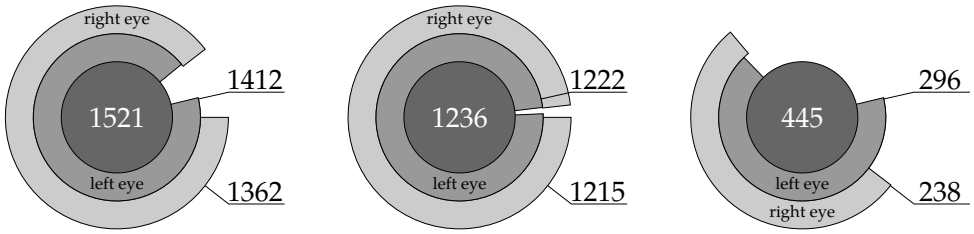
Table 3.3: The eye region resolutions in pixel for all data sets including the detected eye boxes [73].

	BioID		GI4E		New	
	Man. labeled	HC detected	Man. labeled	HC detected	Man. labeled	HC detected
Minimum	20x40	12x18	20.5x20.9	26x39	22x30	35.1x49.7
Maximum	20x40	35x52	26.7x44.9	42x62	47x99	79x118
Mean	20x40	21.2x31.8	22x31.2	30.1x45.2	24.5x60.8	49.8x74.5
Median	20x40	21x32	21.5x30.7	30x45	22x61	49x74

The evaluation includes two scenarios. The first is an evaluation on manually annotated eye boxes and the second an evaluation on detected eye boxes. This is due to the varying shape, size, and position occurring in eye detection, which is a non-trivial task. The different resolutions for the detected and annotated eye boxes are shown in Figure 3.3 (minimum, maximum, mean and median). For detection, the Haar Cascade [247] classifier together with KLT-feature tracking [238] was used (from openCV [24]). As shown in Figure 3.42, the Haar Cascade achieved 2774 successful detections on the BioId data set, 2437 on the GI4E and 534 on [73], respectively.

3.6.3 Results

Figure 3.43 shows the evaluation results. Plots on the left side provide the detection rate based on the Euclidean distance to the ground truth. On the right side the relative error is presented, which is the Euclidean distance divided by the eye image diagonal. The relative



(a) BioID **(b)** GI4E **(c)** Own data set

Figure 3.42: Proportion of correctly detected eye regions by means of the Haar Cascade classifier togther with the KLT tracking [73].

error compensates for effects from differently sized regions. It has to be noted that each row in Figure 3.43 corresponds one data set.
The results show that algorithms designed for remote images do not achieve better results than algorithms for head-mounted tracking. The best result was achieved by the second step of the ElSe [79] algorithm. In addition, the algorithm from Timm and Barth [236] has a stable detection rate on all data sets. Therefore, those algorithms are robust against various sources of noise, such as illumination or off-axial camera position. In Table 3.4, the detection rate for a relative error up to 20% is shown.

Table 3.4: Detection rate of all algorithms with a relative error of 20% [73].

	BioID		GI4E		[73]	
	Man. labeled	HC detected	Man. labeled	HC detected	Man. labeled	HC detected
George	0.772	0.884	0.794	0.760	0.479	0.703
Droege	0.096	0.739	0.304	0.283	0.066	0.316
Timm	0.696	0.823	0.806	0.867	0.891	0.838
SET	0.599	0.638	0.189	0.395	0.288	0.323
Starburst	0.273	0.915	0.904	0.835	0.066	0.802
Swirski	0.759	0.799	0.729	0.815	0.605	0.814
ElSe	0.907	0.939	0.983	0.898	0.927	0.933
ExCuSe	0.011	0.796	0.484	0.311	0.037	0.065

The best performing algorithms on the BioID data set are ElSe and Starburst (Table 3.4) but the difference between the manually labeled and detected eye box is high for Starburst. The same was observed for the GI4E data set, where ElSe has very high detection rates. The difference between the data sets BioID and GI4E is that for BioID the detections on the detected eye boxes are better, whereas in GI4E it is exactly the opposite. For the data set from [73], ElSe again outperformed all the other algorithms (Table 3.4) reaching detection rates of more than 89%.

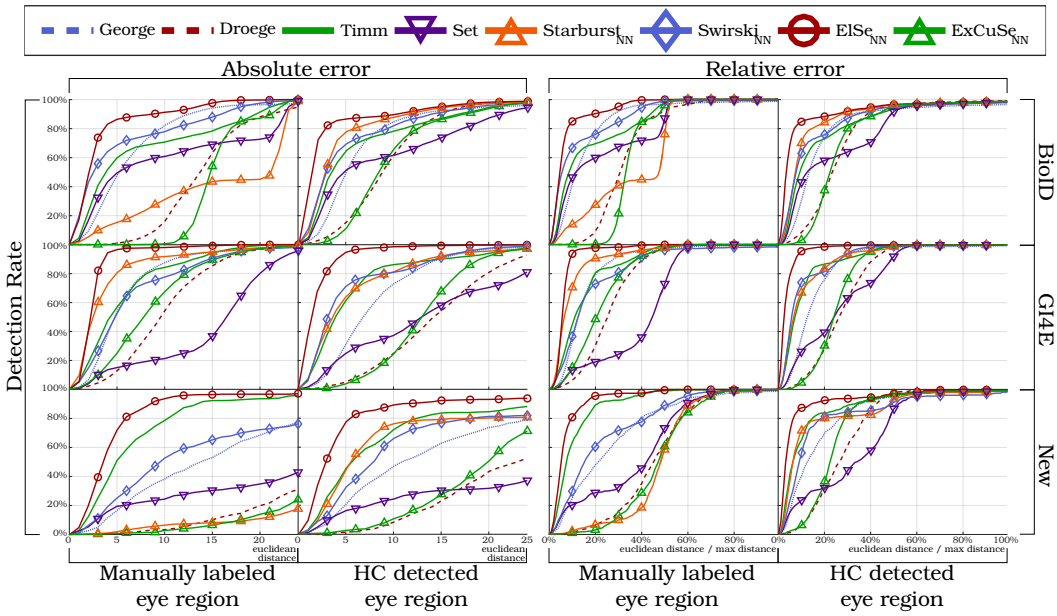


Figure 3.43: The left sided six plots (title absolute error) show the Euclidean distance in pixels, whereas the right sided six plots (title relative error) show it normed with the eye box diagonal. In the top box, the membership of the color to the algorithm is defined. The first and third column show the results for the labeled eye boxes and the secound and fourth column the results for the detected eye boxes [73].

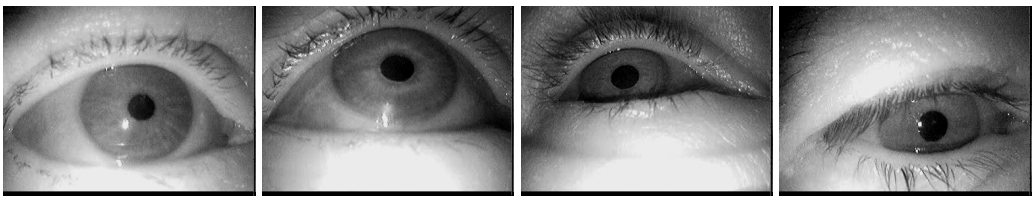
3.7 Evaluation on dirt simulation images

As most of today's eye-trackers are video based, dirt and smudges, both on the device as well on the subject's eyeglasses, are a further potential source of error that may be less common in a well maintained laboratory, but become relevant in real-world applications. Just think of a remote tracking setup in an automotive driver monitoring system. Since it is hard to objectively quantify the amount and nature of dirt in a real experiment, an image synthesis method was employed on top of real eye-tracking videos recorded during a driving experiment.

3.7.1 Data sets

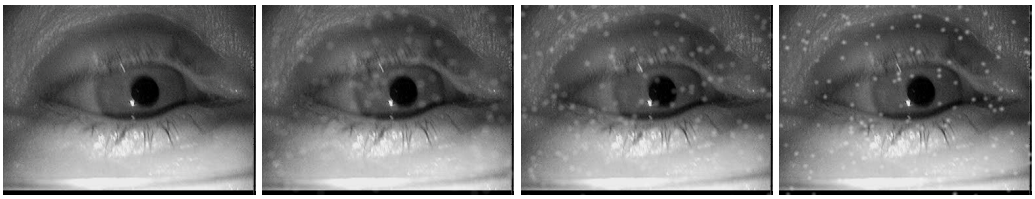
The evaluation was done on a subset of the data set by Fuhl et al. [66], namely data set X, XII, XIV, and XVII (Figure 3.44). Based on visual inspection, these data sets were found to be mostly free of dust particles and provided thus good baseline results for all of the evaluated algorithms. A total of 2,101 images from four different subjects were extracted. These images do not contain any other challenges to the pupil detection such as make-up or contact lenses, to investigate the isolated influence of dust and dirt. However, all subjects wore eyeglasses and the ambient illumination changed. Furthermore, these are not completely synthetic images as comparable results can only be achieved within strict laboratory conditions, where dust would usually simply be removed from the recording

devices.



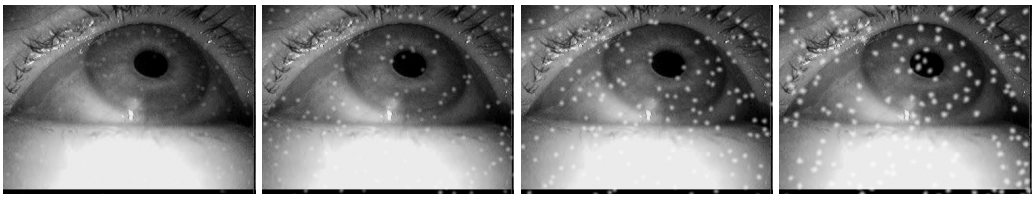
(a) X **(b)** XII **(c)** XIV **(d)** XVII

Figure 3.44: Example images selected from the respective data sets published by [66], [74].



(a) Original image. **(b)** focal 2.8 mm **(c)** focal 4.0 mm **(d)** focal 5.6 mm

Figure 3.45: Simulation results for different focal lengths on one image. 200 particle of size group 2 were inserted [74].

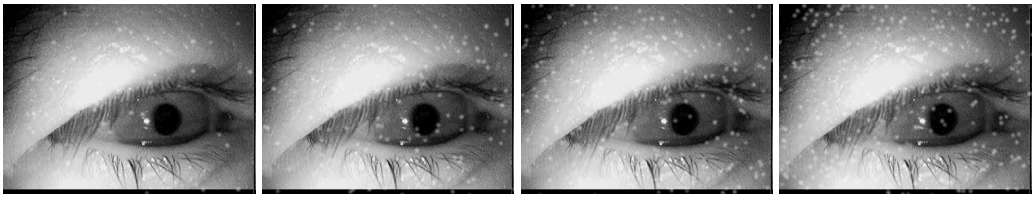


(a) Size group 1 **(b)** Size group 2 **(c)** Size group 3 **(d)** Size group 4

Figure 3.46: Simulation results for different particle size groups on one image. The particle amount is set to 200 and the focal length is 5.6 [74].

Figure 3.45 shows the influence of the focal length on the final image. It should be noted here that in a realistic scenario, the focal length would also have an influence on the image of the eye, not just on the particles. This effect was omitted here (visible for example at the eyelashes). For the images in Figure 3.45, a focal length of 5.6 puts the dust particles in focus. For real dust particles this is based on their distance to the camera and depends mainly on the design of worn glasses. Most eye cameras do not employ an autofocus mechanism but provide a possibility of adjusting the focus. However, it is rarely adjusted with dust on the eyeglasses in mind. In the following, the value of 5.6mm focus is used as a reference.
Another important aspect of dirt is the size of different particles. This effect is shown in Figure 3.46. The amount and focal length is fixed to 200 and 5.6, respectively. For real

world recordings, dust can occur in different sizes for which six size groups are evaluated. As can be seen in Figure 3.46(d) they are not simple dots but varying polygons. The effect



(a) 50 particles **(b)** 150 particles **(c)** 250 particles **(d)** 350 particles

Figure 3.47: Simulation results for different amounts of particles on one image. The particle size group is set to 2 and the focal length is 5.6 [74].

of the amount of particles rendered can be seen in Figure 3.47. The particles are spread uniformly over the image. This is one limitation of the current simulation, as realistic dust distributions would include the lens lenticular buckle of the camera and the curvature of the glasses of a subject.

3.7.2 Results

Figure 3.48 shows the detection rate of the evaluated algorithms over all data sets. The detection rate is reported based on the difference in pixels between the manually labeled and the automatically detected pupil center (pixel error). The red vertical line marks the results (i.e., detection rate) for a pixel error of 5, which is considered as an acceptable pixel error for the given image resolution. More details regarding the detection rates are provided in Table 3.5. For the evaluation, the data sets with all combinations of focal length

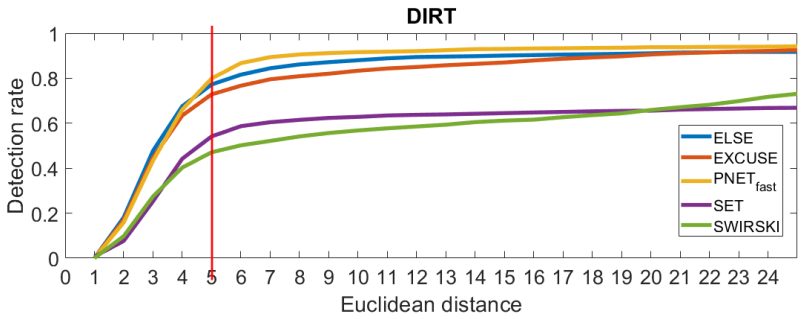


Figure 3.48: Results on all data sets without dust simulation. The detection rate is shown with regard to the Euclidean distance error in pixels. The red line represents the 5 pixel error.

Table 3.5: The 5 pixel error detection rate on the original, non-modified data sets for all algorithms.

SET	*Swirski*	*ExCuSe*	*ElSe*	$PNet_{fast}$
59.04%	51.07%	74.49%	83.02%	86.72%

(i.e., 2.8, 4.0, and 5.6), size groups (1-6) and particle amount (50-500) are simulated. Dirt particle placement is calculated based on a uniform distribution. To ensure the same dirt placement for each algorithm, the simulation results were stored. Figure 3.49 shows the

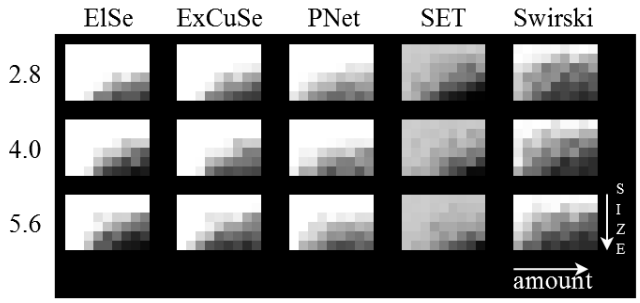


Figure 3.49: Results of all algorithms for different focal length (2.8, 4.0 and 5.6), changing amount of dirt particles (50 to 500) and altering size groups (1 to 6).

detection results for different dirt simulation results. The bright intensity means that the detection rate is above or equal to 50% of correctly detected pupils with less than 5 pixels distance. This normalization to 50% was done to equalize the visualization between all algorithms (Table 3.5). As can be seen in Figure 3.49, the algorithms ExCuSe, ElSe, and the fast version of PupilNet are the most robust. For the very challenging images, large amount of dirt particles with a high size group, PupilNet and SET are more robust since they are not solely based on edge detection. In addition, it has to be mentioned that the higher but still low detection rates of PupilNet are still not satisfactory. For an even more robust detection, the dirt simulation has to be integrated into the training of PupilNet. This allows an automatic data augmentation and will result in a more robust and general CNN.

3.8 Evaluation of pupil detection algorithms for microscopy images

This evaluation compares the decision-based approach for microscopy pupil center detection against three state-of-the-art remote pupil center detection algorithms, three head-mounted ones and the Hough Transform. Images were recorded from six subjects with the ocular based recording system. The challenges for pupil detection are also shown and described.

3.8.1 Data sets

Figure 3.50 shows challenges which occur in the pupil monitoring. The lighting conditions (b,c) can change heavily due to changes in the display brightness. These changes also lead to low contrast images (c) and with the limitation of the field of depth many images are blurred (a,f). Reflections on the cornea and from internal lenses occur (c) because the digital display emits light with wavelengths between 390 and 700 nm. These reflections are present in all data sets.

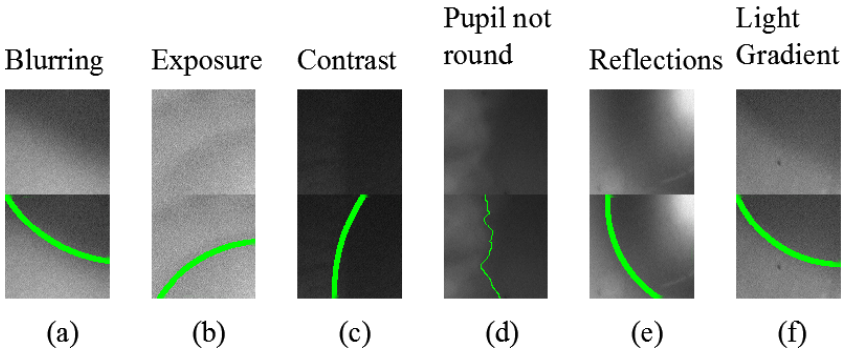


Figure 3.50: Challenges for pupil center detection arise in the data sets. The green line in the images below show the pupil border [85].

3.8.2 Results

The used evaluation metric is the relative error, i.e. the Euclidean distance to the ground truth divided by the image diagonal. All algorithms were adapted as good as possible to be able to detect pupil centers in these microscope images. The results for each data set are shown in Fig. 3.51. In data set VI (Fig. 3.51f), the Hough transform would be a good alternative approach for pupil center detection. In comparison to all other data sets, it is obvious that this is the only data set where it could have been used. The images from data set I are very noisy, which has a high impact on the accuracy (Fig. 3.51a). For data set IV, the main challenge is that only a small part of the pupil is visible.

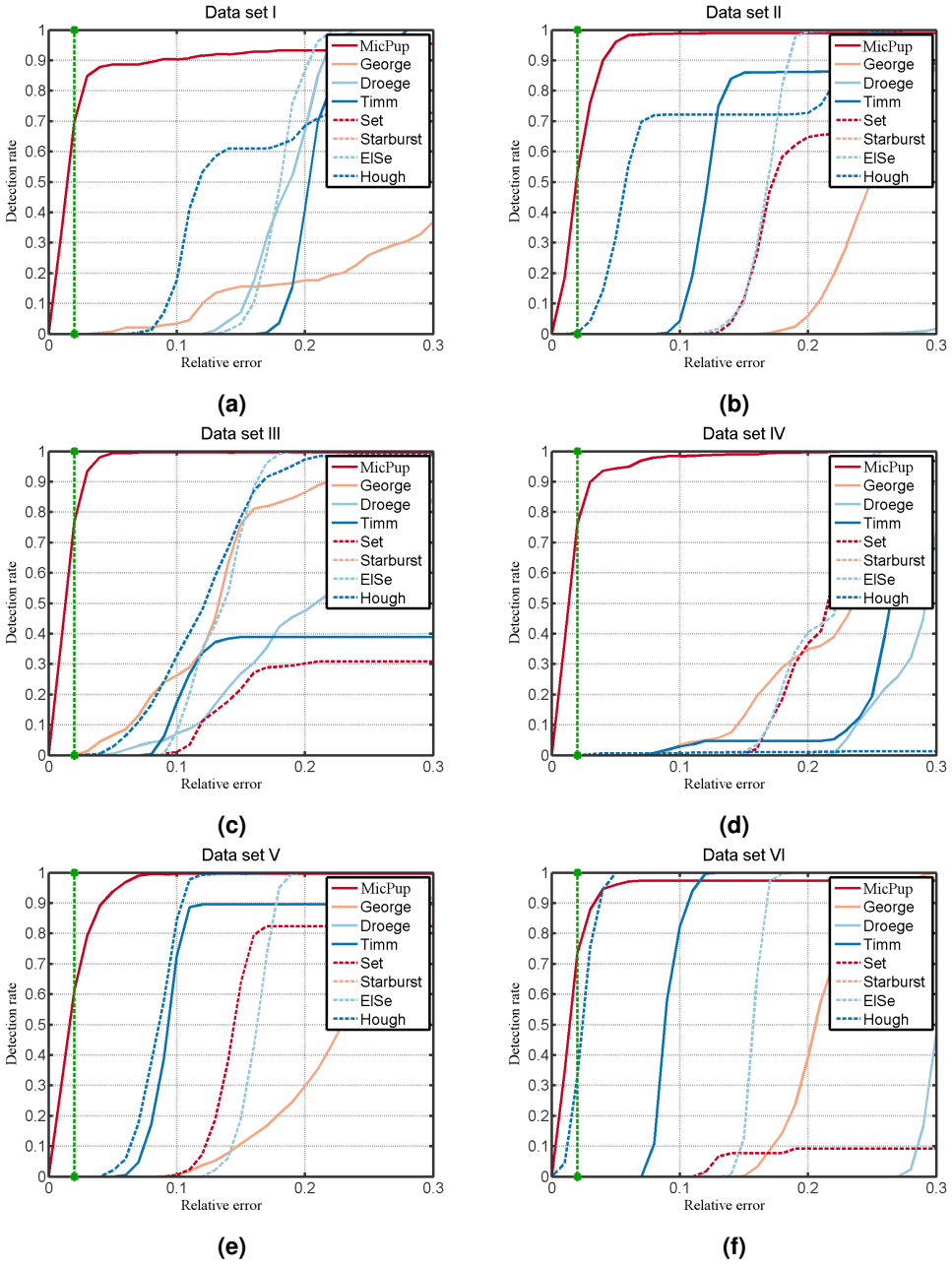


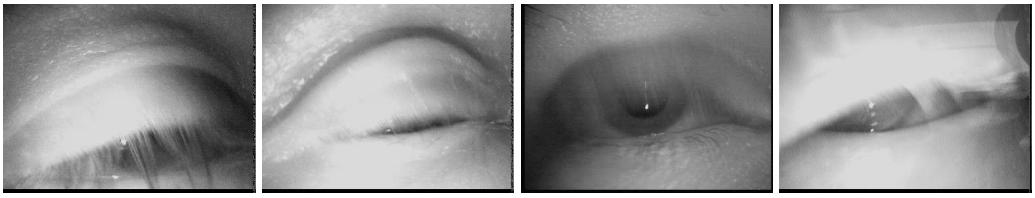
Figure 3.51: Results for all evaluated algorithms on the microscope data sets. The bottom axis shows the euclidean pixel distance relative to the hand labeled position divided by the maximal error, and the left axes show the percentage of correct detected pupil centers. The vertical green dashed line corresponds to an error of 20 pixels [85]. The microscopy pupil center detection algorithm is denoted by MicPup.

3.9 Conclusion

This chapter introduced multiple methods for robust and real-time pupil detection. Our evaluation showed that - by the time they were published - these methods improved the state-of-the-art by a large margin. In the following chapter, we will introduce and discuss novel methods for the automated extraction of another important information source of the eye, namely the eyelids.

4 Eyelid detection

While pupil center detection is crucial for gaze estimation, the eye lids may be used to infer information about cognitive states, such as vigilance, fatigue, and drowsiness [154], [229], [263] of a person. In addition, the eye lids protect the eye from particles and limit the amount of light entering the pupil [61]. Therefore, eyelashes, blinking, and squinting are essential mechanisms to ensure eye healthiness; however, these mechanisms also create several challenges for computer vision based algorithms in video-based eye tracking. A collection of such challenges are shown in Figure 4.1. The first challenge are eye lashes, which occlude the pupil (Figure 4.1a), or the occlusion comes from the eye lids itself (Figure 4.1b). Due to the fast movement of the eye lids, motion blur can also occur (Fig. 4.1c). Therefore, a robust and accurate detection of eyelids can not only help in extracting information about vigilance, fatigue, and, drowsiness, but it can also improve pupil detection by restricting the area in which the pupil has to be found.



(a) (b) (c) (d)

Figure 4.1: Some of the challenges caused by the eyelids, such as occlusion and motion blur.

This chapter introduces two methods for eyelid extraction. Section 4.1 first gives an overview of the current state-of-the-art. In Section 4.2, the novel methods are described, with the first being rule-based and the second being an optimization approach. The last Section 4.3 covers the evaluation and discussion of the proposed algorithms.

4.1 State-of-the-art algorithms for eye lid detection

Eyelid extraction methods originated as a byproduct of attempting to improve iris recognition due to occlusion by the eyelids [40]. The Hough transform for example was used by Wildes [253] to detect the eyelids. In [40], the iris and pupil is searched first. Within the iris region, the upper and lower eyelid are searched as curvilinear edges. For example, Daugmann [40] uses a statistical spline fitting method for outline estimation.
In [229], the input image is partitioned into vertical regions. In those regions, candidates for the upper and lower eyelid are selected based on the intensity distribution of the region. The results are grouped into upper and lower eyelid candidates and outliers are removed.

As eye opening, the mean distance between all remaining upper and lower eye lid candidates is used. Adam et al. [2] applies first anisotropic diffusion to refine the input image. Afterwards, a Canny edge detector is applied. Outgoing from the iris center, edges are selected, in which are either above or below. Edges which are shorter than the mean edge length are ignored. Edges with the highest horizontal response are selected and parabolic curves are fitted to them. Another work by Yang et al. [263] proposed a four point fitting approach for eyelid detection. First, a likelihood map is generated based on the texture and color between the current and a reference frame. The four points are selected based on the highest likelihood and two parabolas are fitted to estimate the eye lid outline.
Radman et al. [189] radially search for candidate points outgoing from the iris center. Therefore, the input image is converted into the HSI (Hue Saturation Intensity) color space. Outgoing from the center, eyelid contour points are computed based on a fixed threshold on the intensity channel. The two highest responses above and below the iris center are selected and connected using the live wire method [161]. As cost function, a weighted combination of the Canny edge detector, gradient orientation, gradient magnitude, and Laplacian zero-crossing are used. The resulting connection path represents the eyelid.
In [30], the iris center is detected first. Afterwards a morphological closing-opening operation is applied to remove eyelashes and punctual bright light spots. For each column, the intensity distribution is analyzed. In each distribution, the minimum is selected as eyelid point. This results in a set of candidate points, where the part above and below the iris is separated. A parabola is fitted then to those points in a least squares sense.
VASIR [135] is a state-of-the-art iris recognition tool. It is developed by the National Institute of Standards and Technology and starts with the detection of the iris and pupil using the Hough transform. Afterwards, the Hough transform is used to detect the eye lids in the iris area. The last step is fitting a third order polyniomal to these eye lid points in a least squares sense. The resulting polyniomal is used as eyelid.

4.2 Methods

In the following, we will describe two rule-based algorithms for eye lid detection and eye aperture estimation. In the first method, the edge image is refined using additional image statistics. For the second method, a optimization is described. This formulation is implemented as rule-based algorithm to reduce the computational costs.

4.2.1 Gradient map refinement method

Let $I[r,c]$ be a digital close-up image of the eye in the near-infrared spectrum with r rows and c columns. The eyelid detection task consists of selecting two sets of pixels P_l and P_u in I that lie respectively on the *lower* and *upper* eyelids, which are then used to fit functions representing the outline of each eyelid.
The method consists mainly of I) rescaling the image preserving dark regions to reduce noise and computation costs, II) filtering the image according to a combination of local features to generate a likelihood map for the eyelids, III) detecting edges on the likelihood map, and

selecting two edges to represent the eyelids based on their orientation and horizontal shift in respect to one another, enclosed intensity value, and accumulated likelihood. These steps are described in detail in the following, followed by a graphical representation exemplifying the output of each stage in the algorithm (Figure 4.4).

Rescaling

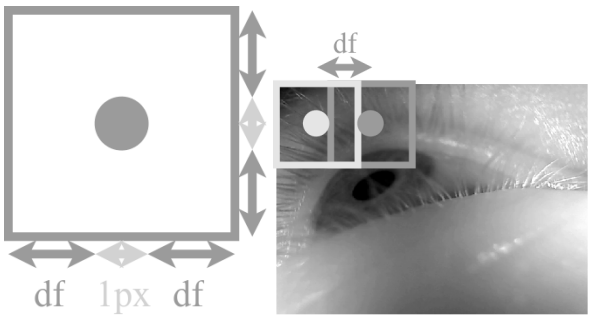


Figure 4.2: Downscaling window size (on the top) and stride (on the bottom) – not in scale in relation to each other [81].

To preserve thin dark structures that usually lie close to the eyelid, such as eyelashes, a downscaling operation that favors lower intensity pixels is used (as proposed in [79]). Let df be a downscaling factor (five in the implementation). The values of the downscaled image D are calculated from the pixels $p_i \in I$ based on a square sliding window W with sides of $l = 2 * df + 1$ pixels and stride of df pixels (see Fig. 4.2). For each position of W, the mean intensity in the window is calculated as

$$\mu = \frac{1}{l^2} \sum_{p_i \in W} p_i, \tag{4.1}$$

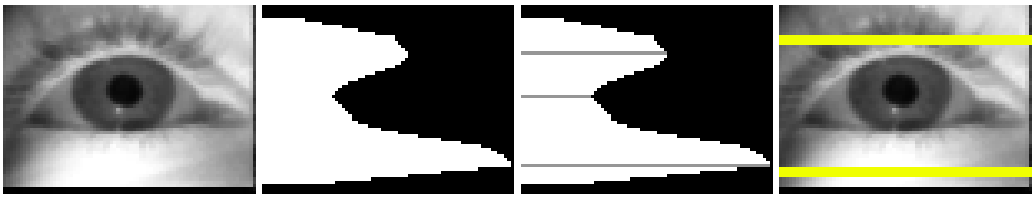
the window intensity histogram H is computed, and the corresponding pixel value d in the downscaled image is then evaluated as

$$d = \frac{\sum_{j=0}^{\mu} H(j).j}{\sum_{j=0}^{\mu} H(j)} \tag{4.2}$$

Likelihood Map Generation

Four different local features are exploited to generate the likelihood map, namely the mean, standard deviation, skewness, and horizontal edges. For each pixel in D, these features are derived from a neighborhood centered on that pixel. The mean, standard deviation, and skewness are computed over a local square window sized 7×7 pixels and act as rotation invariant sparse edge filters; additionally, these filters also respond to edges partially covered

by eyelashes. The mean is calculated as the first moment (m_1) and uses the complement of the downscaled image as input (to weight small shadows close to the eyelids higher). The standard deviation is calculated as the second moment around the mean (m_2), whereas the skewness is evaluated as the third moment around the mean (m_3). The horizontal edge



(a) Rescaled (D) **(b)** Smoothed distribution **(c)** Local maxima and minima **(d)** Boundaries

Figure 4.3: The input (a), and its smoothed mean horizontal intensity values distribution (b), from which local maxima and minima (c) are identified and employed to determine the plausible eyelid boundaries – yellow lines in (d) [81].

response is calculated using the Prewitt operator [186] and serves as a reinforcement of horizontal edges in the likelihood map; it is worth noticing that even if the eye is not precisely aligned horizontally with relation to the camera, some parts of the eyelid evoke a response due to the arcuate nature of the eyelid. Each feature produces an associated activation map: the mean (M_1), standard deviation (M_2), skewness (M_3), and horizontal edge (E) maps. These maps are point-wisely[1] combined to generate the likelihood map L as

$$L = E^2 \odot M_1 \odot M_2 \oslash M_3, \tag{4.3}$$

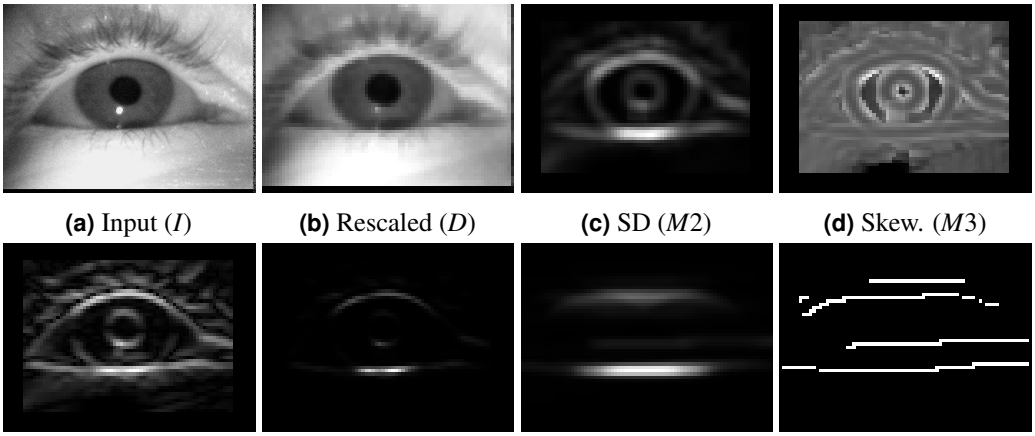
effectively resulting in a high pass filter for E, M_1 and M_2 and in a low pass one for M_3. An averaging filter with a height of three pixels and covering $\approx 30\%$ of L's width is applied to L in order to connect horizontally disjointed high likelihood regions and increase the response of straight horizontal edge parts. This operation introduces some noise, which is partially removed by setting negligible values (smaller than one cent of one percent of the maximum value) to zero. Additionally, the likelihood map values outside plausible eyelid regions are set to zero. The boundaries of this region are determined based on the mean horizontal intensity values distribution, considering that the pupil and iris exhibit lower intensity values relative to the skin patches above and below it. Prior to the analysis, the distribution is smoothed with an averaging filter of seven pixels to remove high peeks and holes (Figure 4.3b). Afterwards, all local maxima and minima are determined (Figure 4.3c); the tuple of two distinct maxima and one minimum $\{max_a, max_b, min\}$ that maximizes the distance

$$max_a + max_b - 2 \cdot min \tag{4.4}$$

is used to determined the boundaries, which are set to max_a and max_b (Figure 4.3d). This yields the refined likelihood map L_r.

[1] $\odot$ and $\oslash$ denote point-wise multiplication and division, respectively.

Edge Detection and Selection



(a) Input (I) **(b)** Rescaled (D) **(c)** SD ($M2$) **(d)** Skew. ($M3$)

(e) Prewitt (E) **(f)** Likelihood (L) **(g)** Refined (L_r) **(h)** Edges (E)

Figure 4.4: Function performed by each stage in the eyelid detection algorithm – normalized per image [81].

Edge detection is applied to L_r by means of non-maximum suppression, followed by a thinning morphological operation, resulting in a set of edges. Let E_i and E_j be two distinct edges, and the mean position for an edge be the mean position of all pixels belonging to the edge. For each possible pair of edges (E_i, E_j), four metrics are calculated:

$\Sigma_L(E_i, E_j)$ **:** The *accumulated likelihood* is based on the values $v_k \in L_r$ and defined as

$$\Sigma_L(E_i, E_j) = \sum_{v_k \in E_i} \times \sum_{v_k \in E_j} \tag{4.5}$$

$\delta_h(E_i, E_j)$ **:** The *horizontal shift* is defined as the distance between the horizontal components of the edges mean position.

$\alpha(E_i, E_j)$ **:** The *relative angle* is defined as the normalized angle between the mean position of the edges (e.g., $90^\circ \overset{\alpha}{\mapsto} 1$ and $0^\circ \overset{\alpha}{\mapsto} 0$).

$\iota(E_i, E_j)$ **:** The *enclosed intensity* is defined as the mean intensity enclosed by the area generated by the seven pixels orthogonal to each pixel in the vector that connects the mean position of the edges.

Figure 4.5 shows a graphical representation of these metrics for an edge pair. For each pair, these metrics are then combined to form a total score

$$\tau(E_i, E_j) = \Sigma_L \times \delta_h \times \alpha \times \iota, \tag{4.6}$$

and the pair with highest score is selected as eyelids. Afterwards, the selected edge points with values in L_r smaller than one third of their maximum value are removed to attenuate the effect of spurious edges introduced by the filters in previous steps.

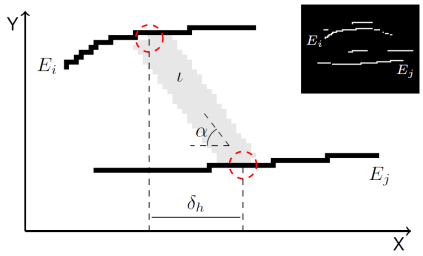
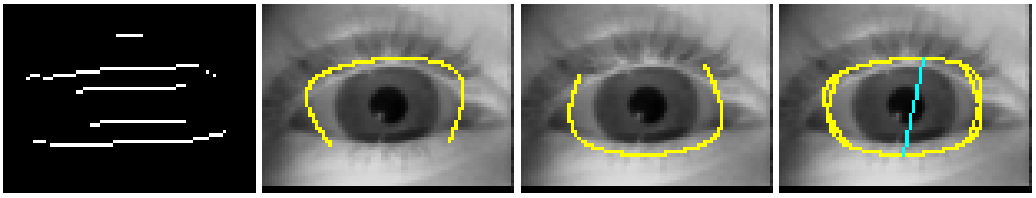


Figure 4.5: Graphical representation of edge selection metrics for an edge pair (E_i, E_j). Red dashed circles present the mean position of the edges, and the gray area represents the area considered when evaluating ι. This is the edge pair selected to represent the eyelids given the edge image on the top right corner [81].

Eyelid Aperture Estimation



(a) Edges (E) **(b)** Upper Bézier **(c)** Lower Bézier **(d)** Aperture

Figure 4.6: Edge (E), upper and lower eyelid Bézier curves, and the resulting ellipsis with the aperture estimation (minor axis, in cyan) [81].

In order to estimate the eyelid aperture, the ending points of the upper and lower eyelids are used to fit two Bézier curves. One curve uses the upper eyelid ending points as first and last control points, whereas the other uses those from the lower eyelid. The combination of these two curves result in an ellipsis that approximates the eye outline. The major and minor ellipsis axis are determined based on the two orthogonal point pairs with maximal distance, and the minor axis is used as estimate for the eyelid aperture. To compensate for the small vertical smearing introduced by the box filter in the previous step, two pixels are substracted (one for each eyelid) from the estimated distance before upscaling it to the input image scale. Figure 4.6 shows the resulting procedure for three distinct situations. The main advantage of the Bézier-curve-based approach is that fitting the Bézier curves is a stable procedure, whereas commonly polynomial fit approaches employed in related work are unstable. As a result, the algorithm performs more uniformly across different scenarios. It is important to notice that this approach does not model the eye canthi region accurately; however, these regions can be safely disregarded without loss of information since they are not pertinent to estimating the eyelid aperture and features of interest (e.g., pupil) do not lay in these regions.

4.2.2 Optimal oriented edge response method

The general idea behind the approach are oriented edges; in other words, the upper and bottom eyelids are approximated by two functions that define a path along which the sum of orthogonal edge values is maximized. Let $e1, e2$ be the positions of the eye corners, $u_{up}(x), b_{bp}(x)$ be the polynomials representing the upper and lower eyelid with parameters up, bp, respectively. The task can then be formulated as the general optimization problem

$$\underset{e1,e2,up,bp}{\operatorname{argmax}} \int_{e1}^{e2} |\Delta u_{up}(x)| + |\Delta b_{bp}(x)| \, dx, \tag{4.7}$$

where Δ is the difference between the inner and outer intensity values orthogonal to the respective polynomial gradient. Let p be a polynomial line of the form $p(x) = ax^2 + bx + c$, then delta is $inner - outer$ intensity, where inner/outer depend on the eyelid orientation (lower/upper) along p.
This optimization problem is not convex and, thus, approximations with the Levenberg-Marquardt method can yield wrong maxima [160]. Therefore, either all combinations of $e1, e2, up, bp$ have to be evaluated or a good initialization has to be found. Since the former is prohibitively expensive, a heuristic is implemented that tries to approximate this optimization problem by searching for suitable starting positions that restrict the polynomials parameters, thus requiring only partially solving the overall problem. The work flow of the algorithm can be seen in Figure 4.7, which will be described in detail in the following. It is worth noticing that a downscaling of the input image is ommited in this figure. This process downscales the input image based on one side of the image, preserving its aspect ratio; this allows us to optimize algorithm parameters and computational costs indepdently of the input image resolution. In the first step, the algorithm searches for a possible bottom

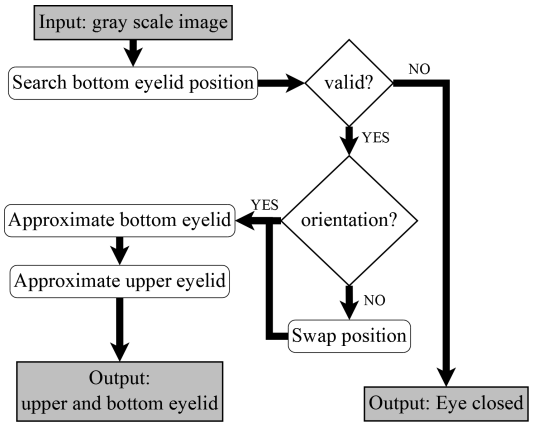


Figure 4.7: General overview of the algorithm work flow [67].

eyelid location as initial position, assuming the bottom eyelid to be easier to locate. The rationale behind this step is that the bottom eyelid tends to exhibit less variability than its upper counterpart (since its eyelashes are less pronounced) as well as present an overall

straighter outline. The position is then validated based on its surrounding intensity; notice that if the eye is slightly open, the upper eyelid assumes a similar form, thus requiring us to analyze if the valid position belongs to the upper or lower eyelid based on its orientation. If the position is valid and correctly oriented, bottom eyelid points are searched through a vertical position optimization paired with an outlier removal mechanism. Polynomials are then iteractively fitted to the left and right side of these points, removing extremeties points when invalid fits are found. The algorithm then approximates the upper eyelid area restriction using the intensity distribution orthogonal to the lower eyelid orientation as well as coarse eye corners locations; from these, the upper eyelid approximation is derived.

Bottom Eyelid Search

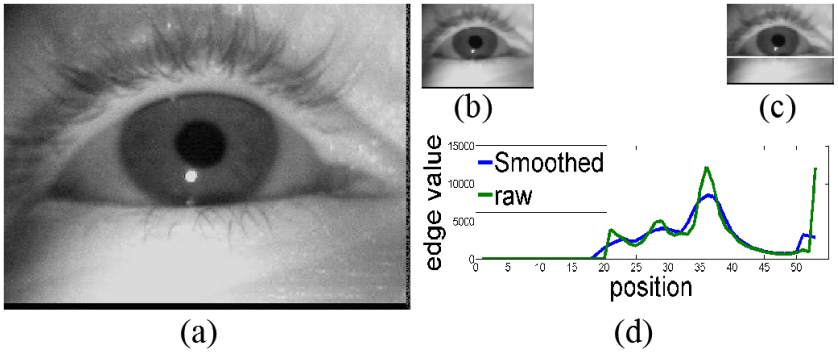


Figure 4.8: The input image (a) and its downscaled version (b). The histogram of horizontally oriented edge values is shown as a green line in (d), whereas the smoothed version is shown as a blue line. Based on the peak in the smoothed histogram, the selected starting position is shown as the white line in (c) [67].

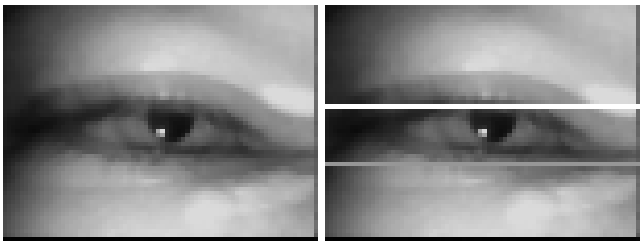
The search of the bottom eyelid is performed solely in the lower two thirds of the input image to avoid eyebrows and reduce computational costs. This step first looks for the maximal row wise summed horizontal edge value (i.e., a straight line) to be used as initial position, based on the bottom eyelid invariability assumption. This values are computed as

$$HE_y = \sum_{1}^{X} |I_{x,y-2} + 2I_{x,y-1} - 2I_{x,y+1} + I_{x,y+2}|, \tag{4.8}$$

where $I_{x,y}$ is the intensity at position x, y and X the image width. It is worth noticing that a deviation of ± 2 pixels is employed to compensate for recording skewness and slight bottom eyelid curvature; this approach also translates into less computational costs relative to a employing a vector as indexing orientation. The raw result of this evaluation is shown in green in Figure 4.8d. However, directly employing these values may yield wrong results due to outliers or recording artifacts, such as the black border at the bottom of Figure 4.8a, which result in artifical peaks. Hence, the raw signal is smoothed through a mean filter following the range from Equation (4.8), shown in blue in Figure 4.8d. The highest peak

in the smoothed histogram ($smooth(HE_y)$) is then selected as starting position (shown as a white line in Figure 4.8c).

Bottom Eyelid Validity and Orientation



(a) (b)

Figure 4.9: The downscaled input image (a), and an overlay showing the first (and wrong) selected position (white line), which violates the method's assumptions. Thus, the next (and correct) maximum is searched, yielding the correct selection (gray line) [67].

In particular cases, the previous step may wrongly select the upper eyelid instead of the lower one; for instance, if both eyelids are relatively straight (see Figure 4.9a) or the eye is shut. The mean intensity above and below the eyelid position for validity and orientation, assuming that the intensity above the lower eyelid must be lower than below it due to the low intensity of iris and dark pupil – the opposite being true for the upper eyelid. If the validity or orientation assumptions are violated, the starting position is then moved to the next $smooth(HE_y)$ maximum; this procedure is exemplified in Figure 4.9b and evaluated as

$$Ori(mp_y) = \frac{\sum_{i=1}^{X} \sum_{j=mp_y-1}^{mp_y-10} I_{i,j}}{\sum_{i=1}^{X} \sum_{j=mp_y+1}^{mp_y+10} I_{i,j}}, \tag{4.9}$$

where mp_y is the position of the selected eyelid on the y axis and X is the width of the image. If $Ori(mp_y) > 1$ the next maximum has to be searched; if $1 \geq Ori(mp_y) > 0.9$ the eye is considered closed.

Bottom Eyelid Approximation

As can be seen in Figure 4.9b and Figure 4.8c, the starting position does not necessarily lay on the lower eyelid. Unfortunately, a complete search over possible lower eye courses is computationally too expensive, and occlusions limit the discovery of pareto-optimal courses. The point lying on the selected mp_y line (white line in Figure 4.10b) that maximizes an area difference[2] is selected as starting position (white circle in Figure 4.10b) to have an entry point for shape evaluation of the later fitted polygons. In practice, this area difference acts as a coarse edge detector, with higher robustness but lower accuracy. Afterwards,

[2]This area difference is defined by a square of five pixels above and below the line.

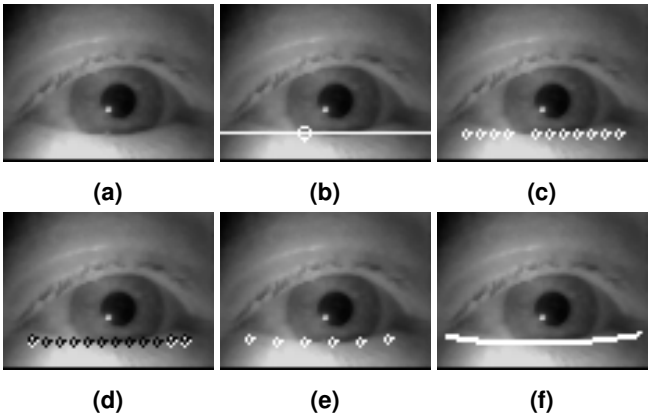


Figure 4.10: The lower eyelid approximation procedure. The input image (a), starting point (b), and possible candidate points (c). The black dots in (d) are the corrected positions; the remaining points after outliers removal is shown as white dots in (e). The resulting two parabolas are shown in (f) [67].

candidates for optimization on the line are selected starting from the entry point, using a stride of five pixels, and limited to the inner ninety percent of the image width (white circles in Figure 4.10c). For each candidate, its vertical position is optimized (black dots in Figure 4.10d). This is done by shifting each point vertically until the maximum area difference value is reached (again square of five pixels above and below the actual position). Because the position is already expected to be close to the eyelid and many higher but wrong positions are likely, only continuously increasing maximums are selected in this way. The next step is the first outliers removal step, which is performed bidirectionally and outgoing from the selected starting position. This is done by inspecting the gradient between consecutive optimized candidate positions. The vector $v = p_2 - p_1$ between two points p_1, p_2 is calculated, which yields the gradient $g = \frac{v_y}{v_x}$. Outliers are filtered based on a gradient threshold of one quarter, thus removing consecutive positions with low changes (compare Figure 4.10c to Figure 4.10e). In other words, let g_1 and g_2 be consecutive gradients along the polynomial; g_2 is considered an outlier if $\frac{g_2}{g_1} < \frac{1}{4}$ or $\frac{g_1}{g_2} < \frac{1}{4}$.
This is a consequence of the five pixel stride between candidates, which should result in more than a single pixel vertical shift. The second outliers removal step is based on the convexity of the resulting least squares polynomial fit (Figure 4.11). If the resulting polynomial is concave (i.e., curved downwards, red line in Figure 4.11) the outermost point is dropped (red dots in Figure 4.11), and a new least squares polynomial fit is performed. This is done until the resulting polynomial is convex and removes candidates that are out of the range of the eyelid. The resulting two polynomials (i.e., for the left and right directions) are used as bottom eyelid approximation (shown in Figure 4.10f).

Upper Eyelid Approximation

For the fast approximation of the upper eyelid, first the search region has to be specified, and the coarse positions of upper and bottom eyelid intersections have to be estimated.

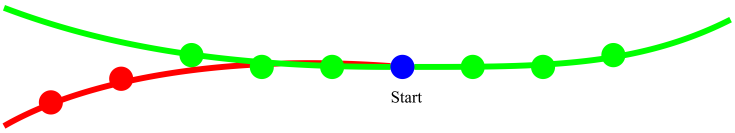


Figure 4.11: Second outliers removal step illustration. The blue point is the starting point; green and red dots are candidate points. Lines in green have the correct convexity, whereas the red line is concave. Dots in red are removed iteratively until the green line results from the least squares polynomial fit [67].

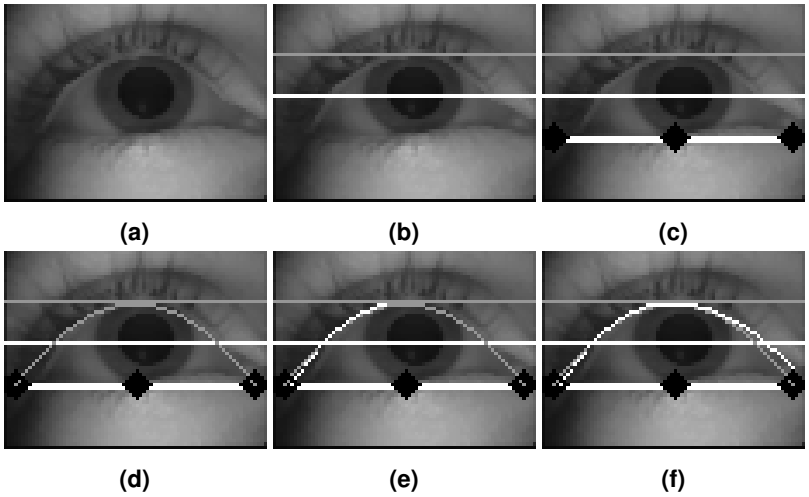


Figure 4.12: Upper eyelid approximation procedure. The input images is shown in (a). In (b) the area in which the search takes place is shown through the two lines. The preliminary intersection points and the center are shown as black diamonds in (c). (d) shows the result of the hight approximation (light gray polynomial). In (e) and (f) the approximation for the left and right side of the upper eyelid is shown (white polynomials) [67].

Therefore, the algorithm starts by selecting a second maximum in HE_y, but this time over the complete image and starting from the last maximum position. Due to the invalid responses produced by skin folds above the upper eyelid, only the next local maximum is searched. As can be seen in Figure 4.8d, there is another local maximum between both eyelids, which is caused by the pupil and the cornea reflection (white dot below the pupil in Figure 4.8a). The result of this step can be seen as the white line in Figure 4.12b. The area between this white line and the second gray line in Figure 4.12b is the search region. For future use, the distance between those lines is defined as ΔWG. This second gray line is calculated by doubling the distance between the bottom eyelid and the white line (next local maximum).
After the search region has been found, the start, passage, and ending points of the upper eyelid polynomial have to be found. In Figure 4.12c, the used initial positions are marked as black diamonds. The center diamond is the center of the bottom eyelid on the x axis, around which the algorithm searches for the upper eyelid high position (between the white

and the gray line in Figure 4.12b). The left and right black diamonds in Figure 4.12c are the estimated eye corners. Those are set to the position, where the bottom eyelid polynomial intersects the starting line of the search region (white line in Figure 4.12b) or to the outmost position on the bottom eyelid. Those initial positions are only coarse, and will be refined in the following three steps.
Due to the computational costs for estimating all three variables at the same time and the non convexity of the problem, each variable is estimated separately. For each variable the algorithmic steps for optimization are as following:

1. Shift position of variable
2. Fit polynomial[3] to three points
3. Evaluate polynomial with Equation 4.10

$$OEV(f(x)) = \sum_{i=1}^{X} |I(i, f(i) + | \perp f'(i)|) \\ -I(i, f(i) - | \perp f'(i)|)| \tag{4.10}$$

Equation 4.10 describes the valuation of a polynomial based on its oriented edge value. $f(x)$ is the polynomial, $I(x,y)$ the intesity value at location (x,y) and X the width of the image. $| \perp f'(i)|$ is the normed orthogonal of the tangent of the polynomial at position i. $OEV(f(x))$ is therefore the summed difference between opposite intensity values along the polynomial. In addition to Equation 4.10, square single differences are squared if the previous difference has the same sign. In other words, if d_1 and d_2 are positive, then $d_2 = d_2^2$, where d_1 and d_2 are consecutive deltas along the polynomial.
The first refined position is the high point. Therefore, the initial x axis position from the center of the bottom eyelid (centered black diamond in Figure 4.12c) is shifted vertically in the search region and horizontally between $-\Delta WG$ and $+\Delta WG$ (step 1 in the enumeration list). The other left and right eye corner stay fixed, and, for each shift of the high position, a polynomial is fitted to the three points (step 2 in the enumeration list). This polynomials are evaluated with Equation 4.10 (step 3 in the enumeration list) and the maximum is selected. The result of this step can be seen in Figure 4.12d as the gray polynomial.
For the eye corners, the same procedure is carried out. The shift region around each initial eye corner position is $\frac{\Delta WG}{2}$ in each direction. For the left eye corner the result can be seen in Figure 4.12e and the final result in Figure 4.12f.

4.3 Evaluation

In this section we evaluate our methods against the state-of-the-art eye lid detection algorithms. For evaluation real data from a driving study is used. The challenges encountered by individual subjects as well as the results over the entire data are further discussed.

[3] The used polynomial is $ax^2 + bx + c$

4.3.1 Data sets

Most public available data sets containing eyelid annotations stem from biometric related research [81]. Those are collected under laboratory conditions and follow guidelines to

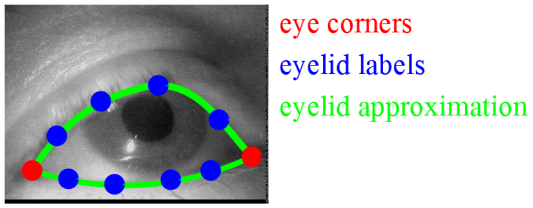


Figure 4.13: Labels for eye corners (red) and eyelid points (blue) labels. The eyelid outline can be accurately approximated through Bezier splines fit to the hand-labeled points [67].

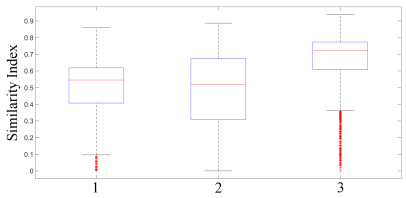
ensure a certain noise-to-signal ratio. Therefore, they are not realistic compared to real-world scenarios. With [81] and [67] we introduced a realistic data set containing 5100 hand-labeled images, which are employed for evaluation. Each frame was annotated with ten equally-spaced points: one point on each eye corner (canthus), four points lying on the lower eyelid, and four points lying on the upper eyelid (see Figure 4.13). These images were collected from 22 subjects in an driving experiment [118] under real world conditions. This data contains challenges like strong skin wrinkles, blinks, half blinks, blurry images, reflections, and changing illumination. Examples are shown in Figure 4.14.



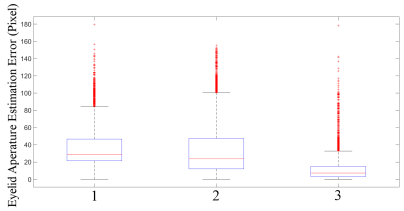
Figure 4.14: Examples from the data sets [67].

4.3.2 Results

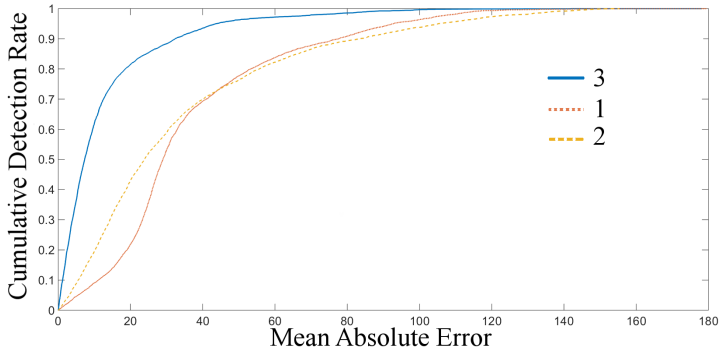
As metric for the similarity between the estimated eyelid outline and the ground truth, the Jaccard index is used. This index is given by $\frac{A \cap B}{A \cup B}$, where A and B are the areas defined by the estimated and ground truth eyelid outlines. Additionally, the eyelid aperture error is evaluated through the Hausdorff distance, $max(min(d(C,D)))$, C and D are the sets of points from the upper and lower eyelid. $d()$ represents the Euclidean distance. For the Hausdorff distance first all minmal distances are calculated. Those are the vertical connections between the upper and lower eyelid. In the second step, the maximum of those connections is selected, which is the eye lid opening. As a measure of error the absolut distance between the ground truth opening and the estimated opening is used. The results are reported using boxplots: the central mark is the median, edges of the box are the 25th and 75 percentiles, and whiskers extend to the extreme non-outliers data points (Figure 4.15). Figure 4.15a shows the overall similarity results (higher is better). In general



(a) Outline similarity (higher better).



(b) Eyelid aperture estimation (lower better).



(c) Cumulative detection rate (top left is better).

Figure 4.15: Overall results in terms of outline similarity, eyelid aperture estimation, and cumulative detection rate [67].

a score of > 0.5 is considered similar, our approach in [67] reaches a mean similarity score of 0.66. Our rule based scores 0.5, and VASIR [135] 0.47. For the eyelid aperture estimation error (Figure 4.15b), the best method reached 12.93 pixels [67]. The pure rule based approach [81] reached 36.84 and VASIR [135] 34.65. This means that our method improves the eyelid aperture estimation by $\approx$21.7 pixels. The cumulative detection error up to ten pixels is shown in Figure 4.15c. In this data, our method reached 61.94% detection rate, whereas [81] reached 9.08% and VASIR [135] 19.53%, respectively.
In terms of average run time, our approach was the fastest (3.4ms). The best performing method required 7.1ms and VASIR [135] 3305.3ms. The timing evaluation was performed on an Intel i5-4570 (@3.2GHz) processor.

4.4 Conclusion

In this chapter the methods and the resulting improvement to the state-of-the-art in eyelid detection were shown. In the following we will discuss automatic data annotation and transfer learning for future improvements in the area of pupil and eyelid detection.

5 Transferlearning for pupil and eyelid detection

As introduced in the previous chapters, eye-tracking methodology based on deep learning approaches requires training on a huge amount of annotated data. In fact, large ground-truth data is a typical problem associated with deep-learning methods such as CNNs. A state-of-the-art approach to cope with this problem is to generate simulated data. For example, the authors in [230] used rendered images for gaze position estimation. In other works such as [127], [216], rendering was applied to measure the effect of eyeglasses on the gaze signal quality. [256] applied a *k*- nearest neighbor estimator on rendered images for gaze estimation. This approach was further improved by [267] using a CNN. Rendering itself is also a challenging task if it needs to simulate realistic data with all its challenges. Therefore, models need to be trained on both synthetic and real images. The annotation of real data is a tedious task, especially if high accuracy is required. To tackle this issue, the algorithm *Multiple Annotation Maturation (MAM)* was developed during this thesis, which is a self-training algorithm based on a grid of detectors. Unlabeled data is clustered based on the detection, iteration, and recognition. MAM is capable of annotating large amounts of data without the need of human intervention. In addition, specialized object detectors are created in each iteration, which can be used for new data annotations or online detection. MAM is the focus of this chapter. Section 5.1 gives first an overview on the state-of-the-art in the area of transfer learning to provides a context for MAM. MAM is then introduced in Section 5.2. Evaluation results are presented in Section 5.3. Finally, Section 5.4 introduces an eye-tracking data annotation tool (EyeLad), which was developed during this thesis to facilitate data annotation based on the MAM algorithm.

5.1 State-of-the-art algorithms for transfer learning

Transfer learning itself refers to the problem of adapting a classifier to a new challenge or enhancing its general performance on unknown data [63]. The three main categories to solve such problems are inductive, transductive, and unsupervised learning [36], [63].

5.1.1 Inductive learning

In the inductive case, annotated data is provided in both domains (target and source). For the inductive case, two main approaches exists, namely self-thought and multi-task learning. Self-thought learning uses unlabeled data to improve the classification performance. For example in [190], a two step approach was proposed. The unlabeled data is first analyzed using sparse coding [171]. This information is used in form of the obtained basis vectors

to transform the annotated data. The transformed data represents the new training set and is used to train a new classifier like a support vector machine (SVM). Multi-task learning in contrast, improves the classification based on the information gain from other tasks or classes. It has been shown experimentally in [12], [35], [53], [235] that multi-task learning outperforms individual task learning. This is especially true if multiple tasks are related to each other. In [12], a Gaussian Mixture Model on a general Bayesian statistics-based approach as developed by [7], [14] was used. [53] developed a nonlinear kernel function, which couples multi-task parameters to two regularization parameters and separated slack variables per task. In another work [33], the authors inspected pedestrian detection in different datasets, where the recording system differs (DC [162] and NICTA [165]). A nearest neighbor search was used to adapt the distribution between the training sets.

5.1.2 Transductive learning

In the transductive case, available labeled data in the source domain is used to adapt the model to a new (related) domain. For most cases the domain is equal and therefore the problem is reduced to the sample selection bias, i.e. finding a weighting for the training data to obtain a better classifier for example in [106]. In [225] an alternative approach was proposed using a covariance shift. Here the re-weighting of data is computed based on the importance of samples in a cross-validation scenario. If the domain between the training and the target set differs, it is usually known as domain adaption. In this realm, [103] proposed a Large Scale Detection through Adaptation (LSDA). The idea behind this approach is to learn the difference between a classification and detection task to transform classification data into detection data. For example, [191] proposed a way to adapts a recurrent convolutional neural network detector trained on labeled data to unlabeled data. The first step of this approach is to normalize the data by applying a principal component analysis. Based on the first principle components, a transformation matrix is computed which aligns the annotated and unlabeled data. Afterwards, the annotated data is transformed and used for training. In [112], the authors used a Gaussian process regression to reclassify uncertain detections of a Haar Cascade classifier [247]. [55] proposed a pipeline for domain adaption which starts with maximum mean discrepancy (same as in [142], [175], [226]). Therefore, the dimensionality of the training data is reduced and the distance of the target and source domain based on the used features is minimized. Gaussian Mixture Models are trained as transformation and applied on the source domain. The same is done for different classes to adjust the class-conditional distribution as proposed in [54]. [142] did the same with the difference of applying a modified version of the maximum mean discrepancy. [226] learned a nonlinear transformation kernel as proposed by [176]. The difference in the approach from [226] was to use the eigendecomposition to avoid the need for semidefinite programming (SDP) solvers. [261] proposed to incrementally improve the target classifier. Therefore, their model needs some ground truth in the target domain. For new data, the training data is updated based on the detections and the detectors are retrained. This step is repeated until there is no update to the training set.

5.1.3 Unsupervised learning

This is the most challenging section of transfer learning. The most famous representer of this group is the Principal Component Analysis [255]. The main application of unsupervised learning is the feature extraction [175] based on the principal component analysis or autoencoders. In an autoencoder, the signal itself is the target label and the internal weights are learned as a sparse representation. This representation serves as an easier and understandable structure of input data for machine learning algorithms. Such features enable the use of one-shot object classification, as proposed by [56] or one-shot gesture recognition by [250]. [56] initialized a multi-dimensional Gaussian Mixture Model on already learned object categories. The model was retrained on a small set of new object classes using Expectation Maximization. Other approaches try to develop robust features which enable unsupervised learning. [258] for example proposed a new feature extractor, which is the extended motion history images. It is based on gait energy information to compensation for pixels with low optical flow. This means that pixels are also weighted based on the motion information, which also compensates for the loss of the initial frames (scene change). Other features are the 3D enhanced motion scale-invariant feature transform (3D EMoSIFT) and motion scale-invariant feature transform (MoSIFT).

5.2 General idea of MAM

Our algorithm belongs to the self-training approaches. It improves itself by extracting information out of a given data base with only a tiny fraction of samples or an given initial detector. In each iteration, it adds new annotations to its training set based on how sure it is that these are correct. The general idea behind the MAM algorithm is that an object in a video can be considered as a function, which has at different locations the same response. Figure 5.1a shows an example of such a continuous function represented by the orange line. This function represents the object under certain conditions like occlusions or different deformations (hereafter referred to as a *challenge*). Given some sample points (gray dots), it is possible to either approximate the function by fitting known models to these given points, or following the gradient and iteratively improving the approximation. The first approach is more often based on Bayesian statistics on other known functions with similar behavior. For rare cases or new scenarios, which might be underrepresented in the statistics, such approaches would usually fail. The gradient-based approach, also known as tracking, extracts features from the object and searches for them in the next image frame. A disadvantage of such an approach is given in the case of discontinuities of the function as shown in Figure 5.1b.
Formal definitions:

- Video: $V = \{1, 2, ...N\}$
- Known: $K \subseteq V$
- Labels: L
- Ages: A
- Detector: $D^{Age}_{Iter,Feat}$

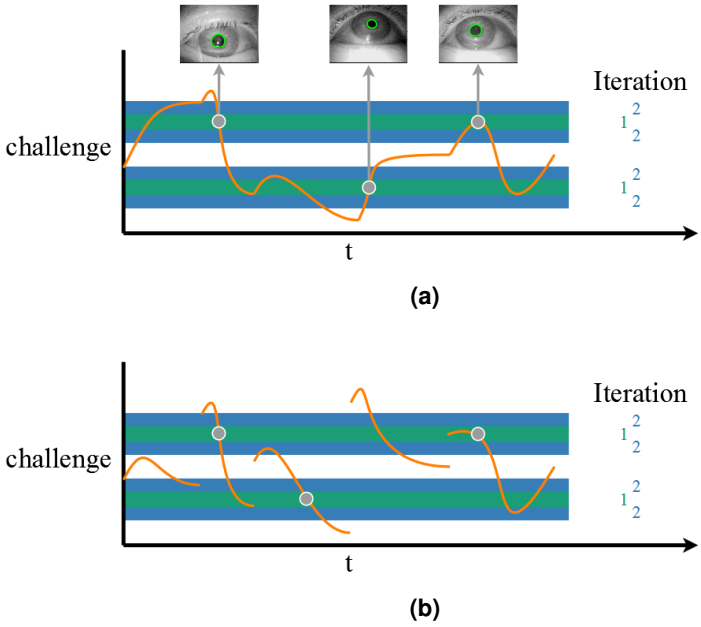


Figure 5.1: MAM tries to extend its knowledge of the object. The orange line represents the object to be detected in the video (in this case it is the pupil on eye images). The x-axis represents the timeline of the video, whereas gray dots represent the initially given labels. The green bar represents the detected objects representing similar challenges. Blue is the detection state after the second iteration [70].

The idea is to detect similar images over a video (as visualized in Figure 5.1) and let the detector cluster the datasets based on their age (A). Outgoing from the initial labels (gray dots), the algorithm searches over the complete video for the object in a similar state (orange line) based on its current knowledge. For iteration one, the extension is represented by the green bar, where the intersection between the orange line and the green bar represent new knowledge (new part of K with a detection box in L and an age in A). This new information is then used in the next iteration to further extend the knowledge; which is shown by the blue bars representing the knowledge extension after the second iteration.

Such an iteratively trained detector has the disadvantage that multiple instances of the same challenge in the training set (K,L) lead to a reduced capability of generalization. This means that the knowledge extension in each iteration (blue and green bar in Figure 5.1) would reduce with each iteration until it saturates. To cope with this problem, one could re-weight the training set as proposed in [106], which could be a further extension of the MAM approach. However, in this scenario, it can neither be guaranteed that the training data is clean nor is it possible to rely on a more generalized detector making fewer mistakes. Therefore, the described algorithm uses the detector itself to cluster the data by assigning each detection an age (A), which is updated by an aging function (Equation 5.2). Based on this age, it is possible to build subsets and train multiple detectors (Equation 5.1). One detector which guarantees the validity and, thereby, is also used as termination criteria is

given in Equation 5.3. It is trained over the entire knowledge and if it results in a weak classifier it is expected that there is too much invalid data in the knowledge base. The other detectors are trained on subsets of the knowledge. These subsets are created based on newly found locations together with already found ones. This avoids the saturation (stopping to detect new locations), which would happen if the method would always train on the entire knowledge.

$$\begin{gathered} D^{Age}_{Iter,Feat} = \frac{1}{2}||w||^2 \sum_i^{|A<Age|} \alpha_i \\ (y_i \in L_{A<Age}(\langle x_i \in Feat(K_{A<Age}), w\rangle + b) - 1) \end{gathered} \tag{5.1}$$

Equation 5.1 shows the simplified optimization of an SVM for the age subsets. w is the orthogonal vector to the hyperplane, α is the Lagrange multipliers, and b is the shift. In this optimization, α has to be maximized and b, w minimized. With $L_{A<Age}$, the subset of L, which has a lower age than Age is addressed. The same applies for $K_{A<Age}$. $Feat()$ represents a transformation of the input data. In the implementations, only the raw and histogram equalized images are used. The detector $D^{Age}_{Iter,Feat}$ can be of any kind of machine learning algorithm, e.g. CNN, random forest, neural net, etc.

$$A(i) = \begin{cases} A(i) += a & ,K(i) \in D^{Age}_{Iter,Feat}(V) \\ 0 & ,else \end{cases} \tag{5.2}$$

Equation 5.2 specifies the aging function. If the detector D^{Age}_{Iter} detects a previously found object on an image, the age of this object is increased by a constant factor a.

$$STOP = \begin{cases} 1 & \frac{TP}{TP+FP} < TH \\ 1 & \frac{TP}{TP+FN} < TH \end{cases} \tag{5.3}$$

Equation 5.3 specifies the termination criteria, where $\frac{TP}{TP+FP}$ represents the precision and $\frac{TP}{TP+FN}$ the recall.
Both criteria give an insight of the performance on a data set, which is in this case, the knowledge ($K_{A<Age}$). If the detector performs below a previously specified threshold (TH), it can be excluded from the actual iteration or the entire algorithm can terminate.

5.2.1 The MAM Algorithm

Figure 5.2 shows the workflow of the algorithm, where either a previously labeled set or a detector can serve as input. The input represents the initial knowledge of the algorithm. In the first iteration, only one detector can be trained (since only one age group exists). After n iterations, there can be theoretically n age groups, though it does not happen in practice. Nonetheless, it is useful to restrict the number of age groups for two reasons. First, it reduces the computational costs in the detection part (since each detector has to see the entire video). Second, it packs together similar challenges, which would generate more stable detectors. In the implementations, always three age groups were used. The first group ($G1$), which trains on the entire knowledge for validation (Equation 5.3) or correction. In the second group ($G2$), all objects detected twice are selected. Then, in the last group

($G3$), only objects detected once are selected. After detection, the age is updated, where each group belongs to a different a as specified in Equation 5.2.

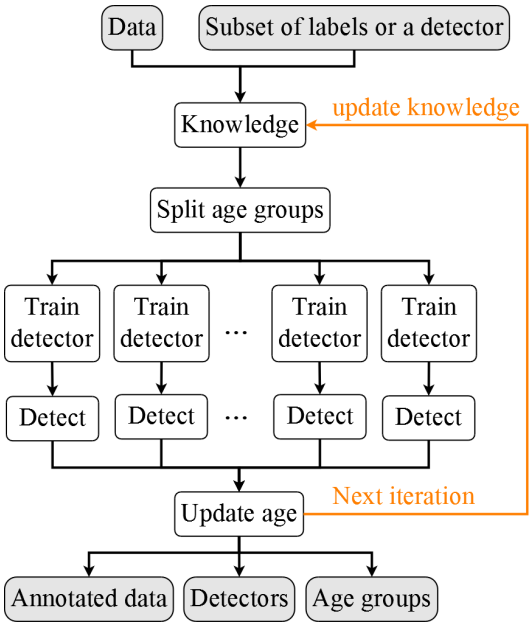


Figure 5.2: Workflow of the MAM algorithm. The gray boxes on top represent the input and those on the bottom, the output for each iteration. The algorithm starts by splitting its knowledge into age groups and trains detectors for each of them. Afterwards, knowledge and age are updated and a new iteration starts (orange arrow) [70].

For implementation, the histogram of oriented gradients (HOG) together with an SVM as proposed by [57] was used. More specifically, the DLIB implementation from [123] is used. The HOG features rely on cells which make them either inaccurate (on pixel level) or consume large amounts of memory (overlapping cells). In the implementation, the computed gradients are shifted below the cell grid in x and y directions by one and up to eight pixels (used cell size). For each shift, a detection is performed and the results are collected. In total, there are 64 possible detections per image and object.

$$PD^{Age}_{Iter,Feat}(V(i)) = \{D^{Age}_{Iter,Feat}(V(i)_{0,0}), \ldots \\ , D^{Age}_{Iter,Feat}(V(i)_{7,7})\} \tag{5.4}$$

In Equation 5.4, this collection process is shown where $PD^{Age}_{Iter,Feat}(V(i))$ represents the detections for frame $V(i)$. The idea is that the average of all detections is accurate. For some of those detections, the counterpart is missing (no detection on the opposite shift). Therefore, outliers removal for two times standard deviation is performed. The shift procedure does not only improve the accuracy, but also increases the detection rate.
Another issue with accuracy is when it comes to deformable objects in addition to moving occlusions (Figure 5.3b, 5.3c, 5.3f, 5.3g, 5.3h), changing lightning conditions (Figure 5.3b,

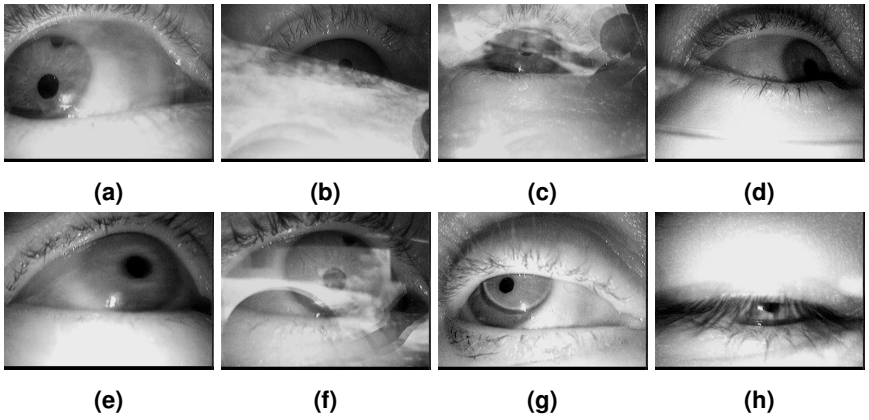


Figure 5.3: Subset of challenges which arise in pupil center detection. Deformations, reflections, motion blur, nearly closed eyes and contact lenses are shown.

5.3d) and distractors (Figure 5.3a, 5.3f). For example, this is the case when the task is about pupil center detection. The circular pupil deforms to an ellipses as shown in Figure 5.3. In addition, the pupil changes size and many people use makeup or need eyeglasses, all of which lead to reflections in the near infrared spectrum. To adapt to those challenges, the described algorithm uses a grid of detectors and an average over the deformation. This averaging is dependent on the combination possibilities for different types of success patterns of the grid. In the implementation, the minimal grid consisting of nine detectors with a shift of gs pixels is selected. For the averaging operation, all possible symmetric means are calculated as shown in Equation 5.6. Here, $SD^{Age}_{Iter,Feat}(V(i))$ is defined as the set of successful detections of all grid detectors $D^{Age,GP_{x,y}}_{Iter,Feat}(V(i))$, where $GP_{x,y}$ stands for the grid position and $x,y \in \{-1,0,1\}$. In this case, the set of values for x,y would be multiplied by 5;

$$Comb(C) = \begin{cases} 1 & ,\forall (x,y) \in C | D^{Age,GP_{x,y}}_{Iter,Feat}(V(i)) \\ & \in SD^{Age}_{Iter,Feat}(V(i)) \\ & ,\sum((x,y) \in C) = 0 \\ 0 & ,else \end{cases} \tag{5.5}$$

$$DC_j = \frac{\sum_{b=1}^{\binom{9}{j}} Comb(CP(b)) \sum_{d=1}^{j} D^{Age,GP_{CP(b,d)}}_{Iter,Feat}(V(i))}{\sum_{b=1}^{\binom{9}{j}} Comb(CP(b))\, j} \tag{5.6}$$

Equation 5.5 evaluates if the actual set of grid positions is symmetric and if every grid position has a detection. C is a set of (x,y) tuples. In Equation 5.6, the mean value over a set of sets (CP) is calculated, where each subset has a tuple amount of j. j is evaluated from one to nine which is the chosen grid size (3×3 grid). The average over these nine means is taken as the box center. $CP(b)$ is the combination b of grid positions from $\binom{9}{j}$ possible grid position combinations. d in $CP(b,d)$ specifies one (x,y) tuple in the actual

set b. Equation 5.5 and 5.6 formalize the grid averaging in the implementation.

5.3 Evaluation

The algorithm was evaluated on different publicly available data sets ([66], [73], [79], [114], [245]). The first evaluation is without the grid of detectors to demonstrate the performance of the aging approach. The subsequent evaluations refer to the grid based object detection.

5.3.1 Remote driving datasets

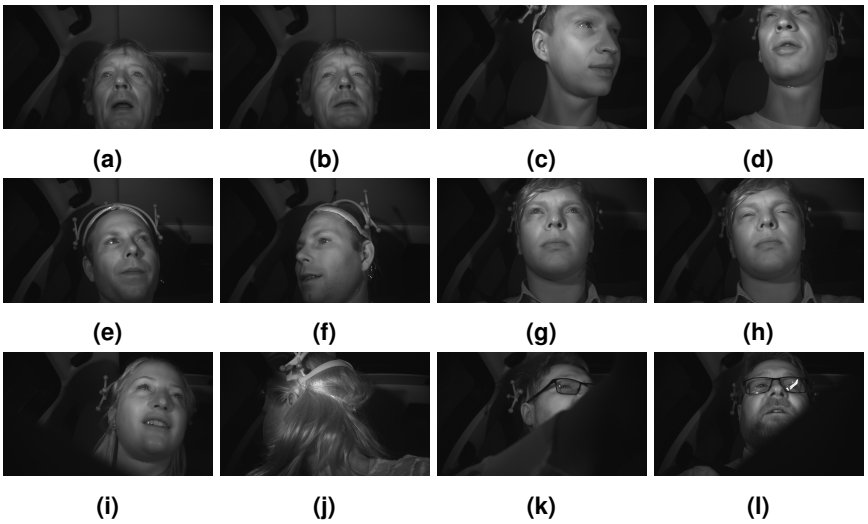


Figure 5.4: Exemplary images of the dataset from [70], where two consecutive pictures represent the same subject.

The dataset from [70] contains more than 16,200 hand-labeled images from six different subjects. These images were recorded using a near-infrared remote camera in a driving simulator setting at Bosch GmbH, Germany. As exemplarily shown in Figure 5.4l, the subjects drove in a naturalistic way, meaning, when turning the steering wheel, eyes or head are occluded. All eyes on these images are annotated using a modified version of EyeLad from [80]. Eyes which are occluded by approximately 50% were not annotated. The smallest enclosing eye boxes, the pupil outline with five points, and for the eye corners and the upper and lower eyelid, three points each were labeled. The pupil annotation consists of five points on the outline with sub-pixel accuracy (Figure 5.5). This new data contains different kind of occlusions like reflections (Figure 5.5d), the nose of the subject (Figure 5.5f), occlusion due to steering (Figure 5.5e), and occlusion of the pupil or eyelids due to eyelashes (Figure 5.5b).

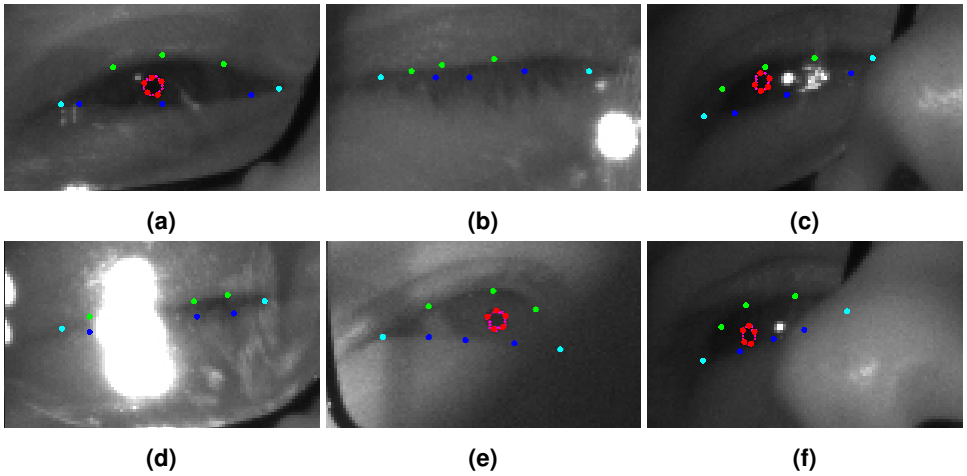


Figure 5.5: Exemplarily eyelid and pupil annotations. The red dots are on the pupil boundary, green dots represent the upper eyelid, blue dots the lower eyelid, and the turquoise dots are on the eye corners [70].

5.3.2 Evaluation without grid of detectors

Two types of experiments were conducted. The first one is the selection of an initial set of ten images and the second experiment was the usage of an initial detector. This detector was trained on the remaining data in the evaluation.
Table 5.1 shows the results for the eye detection task for different iteration stages. The MAM algorithm was evaluated for a maximum of 15 iterations. Most of the error in the data set stems from unlabeled images (annotation criteria was to annotated only eyes where more than 50% is visible). This can be seen especially for subject 6. Here the error reaches 28% in relation to all possible correct detections. In Figure 5.6, examples from subject 6

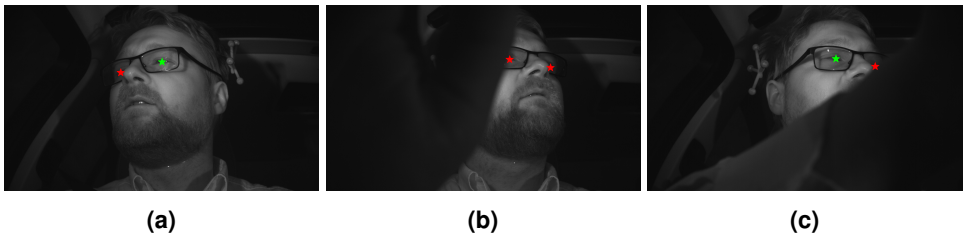


Figure 5.6: Exemplary eye detections that are valid but not annotated in the data set. The red star represents a detection by the MAM algorithm that was not annotated and the green star represents an annotation that was successfully found [70].

are shown. As can be seen, the MAM algorithm detected an eye that was not annotated (red star). The error of 28% is nearly to 100% from such examples. The same applies for subject 2 and 5.

Table 5.1: Overview of the eye detection results for each data set with preliminary annotated images or a used detector in advance. T stands for a correct detection and F for an invalid detection. Both percentages are calculated in relation to the amount of possible correct detections. It should be mentioned that for subject 6 from the data set, there are many unannotated frames, since eyes are occluded by approximately 50% (100% of the error is on non-annotated locations). First, Mid and Last stand for the results after the first, middle and last iteration, respectively [70].

		Detector						10 annotations					
Dataset	Subject	First		Mid		Last		First		Mid		Last	
		T	F	T	F	T	F	T	F	T	F	T	F
From this work	Sub1	.99	0	1	0	1	0	.95	0	1	0	1	0
	Sub2	.94	0	1	.01	1	.01	.59	0	.90	.01	1	.01
	Sub3	.71	.01	.96	.02	.97	.02	.30	0	.85	.06	.95	.02
	Sub4	.99	0	.99	0	.99	0	.78	0	.99	0	.99	0
	Sub5	.60	0	.93	.03	.98	.02	.46	.01	.82	.03	.97	.03
	Sub6	.59	.01	.91	.03	.98	.09	.73	.01	.99	.14	1	.28
GI4E [245]		.36	0	.95	0	.96	0	.22	0	.58	0	.92	0
[73]		.43	0	.55	.01	.92	.03	.48	0	.84	.02	.93	.04

5.3.3 Evaluation with grid (3×3) of detectors

Three experiments for deformable and fast moving objects were conducted. Those are pupil center detection for head-mounted and remote eye trackers, and eye lid detection. The challenges arise from deformation, motion blur, reflections on glasses, eye lashes, make-up, illumination changes etc. for the head-mounted experiment. The main challenge in the remote experiment is the low resolution in addition to the aforementioned interference factors. For the eye lid experiment, the main challenges are the deformation and appearance changes of the point annotations through eye lashes and head movements. Additionally, there are occlusions through the steering wheel, reflections etc.

Pupil center detection for images from a head mounted eye-tracker

In Table 5.2 the results are shown. As can be seen, the MAM approach, had the highest detection performance for all datasets. The maximum of iterations was set to 15. For initialization of the MAM algorithm, ten annotations were selected. The distance between the selected annotations was again ten frames ($i \mod 10 = 0$). Though the MAM algorithm outperforms all the competitors, the results provide a basis for even further improvement. The input to the algorithm was the entire data set, except for data set XIX. This data set had to be separated since the memory consumption exceeds the limits of our server. The same selection of ten frames from 13,473 images was performed but the MAM algorithm was started on three different instances with two times 5,000 images and once with 3,473 images.
Parameters: detection window size 65×65*, cell size* $cs = 8$*, the grid shift* $gs = 5$*,* $\varepsilon = 0.01$

Table 5.2: Overview of the head mounted pupil center detection results for each individual data set. All values are rounded down to two decimal places. The percentages below 5 represents all detections with an error up to five pixels as suggested by [79] and below 10 all with an error up to ten pixels. [70]

	ID	Swirski		ExCuSe		ElSe		MAM	
		5	10	5	10	5	10	5	10
ExCuSe [66]	I	.05	.07	.71	.78	.85	.92	**.89**	**.97**
	II	.23	.42	.39	.59	.65	.81	**.81**	**.95**
	III	.06	.08	.37	.40	.63	.67	**..79**	**..83**
	IV	.34	.38	.79	.84	.83	.86	**.93**	**.98**
	V	.78	.82	.75	.79	.84	.89	**.93**	**.99**
	VI	.19	.21	.59	.63	.77	.82	**.89**	**.96**
	VII	.39	.46	.48	.57	.59	.68	**.82**	**.96**
	VIII	.41	.47	.55	.62	.68	.79	**.88**	**.94**
	IX	.23	.28	.75	.80	.86	.92	**.90**	**.97**
	X	.30	.35	.78	.85	.78	.85	**.93**	**.98**
	XI	.20	.21	.58	.61	.75	.79	**.94**	**.95**
	XII	.71	.77	.79	.88	.79	.89	**.88**	**.98**
	XIII	.61	.78	.69	.82	.73	.89	**.84**	**.98**
	XIV	.51	.66	.68	.76	.84	.90	**.91**	**.94**
	XV	.62	.79	.55	.72	.57	.73	**.69**	**.86**
	XVI	.18	.37	.34	.47	.59	.79	**.92**	**.99**
	XVII	.66	.72	.78	.85	.89	.94	**.98**	**.99**
ElSe [79]	XVIII	.15	.18	.23	.25	.56	.59	**.62**	**.66**
	XIX	.9	.14	.23	.29	.33	.39	**.53**	**.61**
	XX	.22	.24	.57	.61	.78	.80	**.89**	**.91**
	XXI	.08	.18	.52	.58	.47	.55	**.82**	**.86**
	XXII	.02	.04	.26	.29	.52	.56	**.73**	**.77**
	XXIII	.96	.96	.93	.97	.94	.99	**.98**	**1**
	XXIV	.43	.54	.45	.50	.52	.59	**.69**	**.83**

(SVM) and $C = 1$ *(SVM).*

Pupil center detection on remote eye-tracking images

For comparison in remote pupil detection, the best competitor in [73] was chosen (ElSe [79]). It outperformed all competitor algorithms on all data sets.
For the data sets GI4E [245], BioID [114], and [73], the labeled eye boxes were used with an increase of box size by twenty pixels. This was done to increase the challenge for pupil center detection. In the data set from [70], the box size was set to 161×161. For MAM, again ten images were initially selected with a fixed distance of ten ($i \mod 10 = 0$). As shown in Table 5.3, the approach surpasses the state-of-the-art clearly.
Parameters: detection window size 31×31*, cell size* $cs = 4$*, the grid shift* $gs = 2$*,* $\varepsilon = 0.01$ *(SVM) and* $C = 1$ *(SVM).*

Table 5.3: Overview of the remote pupil center detection results for each individual data set. All values are rounded down to two decimal places. The percentages below 3 represents all detections with an error up to three pixels, below 6 up to six pixels and below 10 up to ten pixels. [70]

		ElSe			MAM		
		3	6	9	3	6	9
	GI4E [245]	.07	.50	.70	**.94**	**.98**	**.99**
	BioID [114]	.16	.43	.64	**.85**	**.93**	**.95**
	[73]	.26	.63	.82	**.64**	**.81**	**.84**
From this work	Sub1	0	.04	.04	**.93**	**.98**	**.99**
	Sub2	.45	.67	.86	**.83**	**.99**	**.99**
	Sub3	.01	.14	.42	**.82**	**.90**	**.93**
	Sub4	0	.06	.46	**.95**	**.96**	**.96**
	Sub5	.01	.14	.52	**.92**	**.98**	**.98**
	Sub6	0	0	0	**.61**	**.71**	**.77**

Eye lid detection using MAM

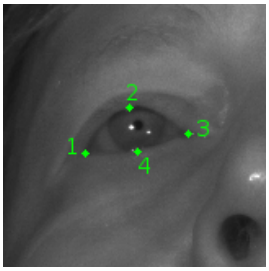


Figure 5.7: Points used for eyelid evaluation. One marks the left eye corner, two the upper eyelid center, three the right eye corner and four the lower eyelid center.

For the eyelid experiment, the approach was evaluated against the shape detector from [121]. This predictor was trained on all data sets except the one for evaluation. The defined eyelid shape is constructed by four points (Figure 5.7). The eye image size was the same as in section 5.3.3. Again ten points were used with distance ten ($i \mod 10 = 0$). As can be seen in Table 5.4, the MAM algorithm is more often the most accurate, even under the condition to detect each point separately without any global optimization between the points.

Runtime

Table 5.5 provides the average runtime of the state-of-the-art and the resulting grid detector. The created detectors have a higher runtime in comparison to the competitors but they are still applicable in real time. Since they can be adapted automatically in comparison to the other approaches, their main advantage lies in the reduction of the development effort together with the increased detection rates.

Table 5.4: Overview of the remote eyelid point detection results for each individual subject. All values are rounded down to two decimal places. The percentages below 3 represents all detections with an error up to three pixels, below 6 up to six pixels and below 9 up to nine pixels [70].

		Left			Right			Upper			Lower		
		3	6	9	3	6	9	3	6	9	3	6	9
MAM	Sub1	**.91**	.98	.99	**.87**	**.97**	**.98**	**.31**	**.50**	**.59**	**.80**	**.99**	**1**
	Sub2	.75	.95	.98	**.66**	**.89**	.94	**.43**	**.70**	**.79**	**.69**	**.96**	.99
	Sub3	**.64**	**.90**	**.95**	**.59**	**.88**	**.95**	**.35**	**.61**	**.74**	**.68**	.93	.95
	Sub4	**.46**	**.93**	**.99**	**.82**	**.98**	**.99**	**.34**	**.72**	**.87**	**.80**	**.98**	**.99**
	Sub5	**.46**	**.73**	.86	**.58**	**.91**	**.97**	**.30**	**.61**	**.76**	**.63**	.79	.88
	Sub6	.29	.58	.74	**.31**	**.53**	.72	**.30**	**.60**	**.75**	.40	.68	.78
[121]	Sub1	.88	**.99**	**1**	.01	.21	.76	.29	.48	**.59**	.32	.94	**1**
	Sub2	**.77**	**.99**	**.99**	.28	.82	**.96**	.10	.27	.46	.56	**.96**	**1**
	Sub3	.28	.77	.93	.32	.57	.76	.18	.45	.64	.64	**.94**	**.99**
	Sub4	.39	.76	93.	.39	.73	.87	.03	.12	.28	.66	.94	.98
	Sub5	.43	**.73**	**.88**	.41	.65	.79	.18	.45	.66	.54	**.85**	**.93**
	Sub6	**.34**	**.69**	**.81**	.24	.52	**.73**	.23	.54	.73	**.52**	**.83**	**.94**

Table 5.5: Overview of the runtime of the algorithms for each experiment. For the method, the resulting detector as grid of five detectors were evaluated [70].

Head mounted				Remote		Eye lid	
Swirski	ExCuSe	ElSe	Prop.	ElSe	Prop.	[121]	Prop.
8ms	6ms	7ms	8ms	4ms	7ms	1ms	14ms

5.3.4 Limitations

The evaluation showed that the MAM method performs successfully for accurate point detection. Apparent limitations are the configuration of the window size and the SVM. The latter results limits the capability to learn all possible deformations and occlusions. Therefore, large videos have to be split into separate files, running the approach multiple times. Additionally, the size of the data set is also limited by the memory due to the high memory consumption for SVM training. Another limitation can be derived by Equation 5.3, which is introduced to avoid the usage of weak classified detectors. It is possible that an age group is flooded with wrong detections. Therefore, the detector will fill the knowledge with wrong representations. Equation 5.3 still holds for the age group learning on the entire knowledge. However, due to the initial set is very small (as in the evaluation), it is possible that the correct object gets understated. Therefore, the algorithm would improve its knowledge of the wrong object.

Eye-tracking is automated by MAM. In case of new challenges or application areas, the data can be automatically annotated and new detectors are generated. As a result, only the detector in the eye-tracking software needs to be replaced to make the software usable. For

the application of MAM additional software is necessary, which supports the user in the data annotation process. This is described in the following.

5.4 EyeLad: Eye-tracking data annotation tool

To facilitate the data annotation process, an annotation tool was developed during this thesis and will be described in the following. First, the EyeLad GUI is described in detail based on the annotation process for remote eye-tracking data. We also introduce point tracking as a useful feature of EyeLad.

EyeLad GUI

The global settings are adjusted in the top left red box (Figure 5.8A). These settings allow to change tracking features to the intensity of the surrounding region, the gradient of this image region, or the binary result of the Canny edge detector [32]. The Canny edge detector

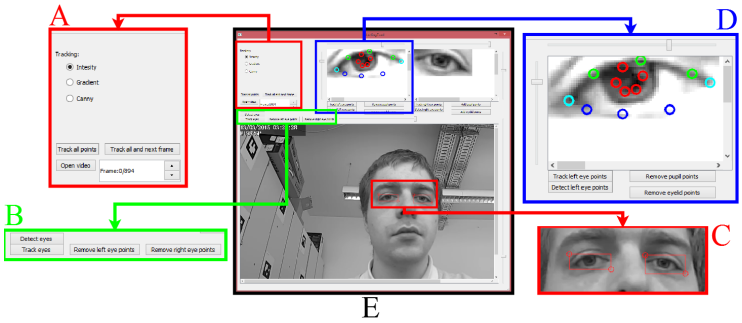


Figure 5.8: The graphical user interface of our labeling tool (E). The red box on the left (A) shows the general adjustment settings and the frame counter with position. The right red box (C) shows the labeled eyes, where the circles are movable by mouse. The green box (B) shows the main window buttons and the blue box (D) the buttons needed for eye feature labeling. The slider above the main window is for normalization and the slider on the right side for zooming in the same [80].

implementation from [66] is used due to the automatic low-high threshold selection. This section also allows to select a video, choose between tracking all points from the previous frame or switching to the next frame and track everything. The selection box on the bottom shows the total frame count, current frame, and allows navigating through the video.
The main visualization area (Figure 5.8E) shows the current frame from the video. This region is used for eye region annotation. In the green box on the bottom left (Figure 5.8B), the controls buttons are shown. These controls allow the automatic detection of the eyes, tracking or remove/add eye annotations. The slider above the current frame is to adjust the normalization of the image. Similarly, sliders can be found above the left and right eye boxes to adjust the normalization for the eye boxes separately (Figure 5.8D). Examples of this normalization are shown in Figure 5.9. As can be seen in the first row, the normalization allows the user to more accurately identify the pupil border and the eyelid points. The second row is an example where the illumination was insufficient due to the head rotation.

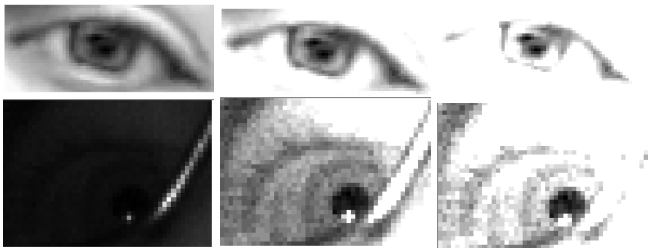


Figure 5.9: Each row shows the original input image (first column) and normalized images (second and third columns) [80].

In addition, the reflection (small white dot and glass frame) forces the imaging sensor to compress the eye intensity values. This leads to a large gap in the intensity histogram. As a result, the image is perceived as mostly dark by the annotator. The second image in this row shows the same image normalized. The slider on the right side of the current frame visualization is used to adjust the zooming (25% to 500%). This feature is also present in the eye boxes (see right side of Figure 5.8D).
In Figure 5.8, the red box on the bottom right highlights the eye boxes (Figure 5.8C). These boxes are defined by two points marked by red circles, which can be dragged and dropped with the mouse. This region is automatically updated on slider change. The blue box on the top right (Figure 5.8D) shows a detailed view for the controls and visualization of the right eye feature annotation area. Here the pupil, eyelid and eye corner points can be tracked using information from the previous frame based on the features set in the global settings (Figure 5.8A). For pupil detection, ElSe [79] which was presented in Section 3.3, is integrated with the modifications to work on remote eye tracking images [73]. For automatic eyelid detection a modified version of the method in [81] is integrated which was discussed in Section 4.2. It returns equally distant points from the Bézier splines. The two points with maximal distance are used as eye corners.
The red circles in the blue eye box (Figure 5.8D) represent the pupil outline points. Green circles are upper eyelid points, the dark blue circles are the lower eyelid points, and the cyan points are the eye corners. The saving and loading of labeling points is done in the background as to not interrupt the user. For each change, everything is saved and, if a video is loaded, an equally named txt file is created. For eye detection, the Haar Cascade [246] face and eye detection from OpenCV [24] is integrated. The first step is the face detection. In the resulting face region, the eye detection is applied. The resulting squares are than converted into a two point format and visualized as shown in Figure 5.8C.

EyeLad: Point Tracking

For tracking, the surrounding region of each labeled point is used. In the new image the position that minimizes the function 5.7 is searched

$$T(x,y,I,N) = \sum_{i=-\frac{n}{2}}^{\frac{n}{2}} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}} |I(i,j) - N(x+i,y+j)|, \tag{5.7}$$

Where I is the image patch of the labeled image, and N is the new image in which the position has to be searched. x, y is the current candidate location and n the patch window size. Different tracking features are integrated, which means that I and N in Equation 5.7

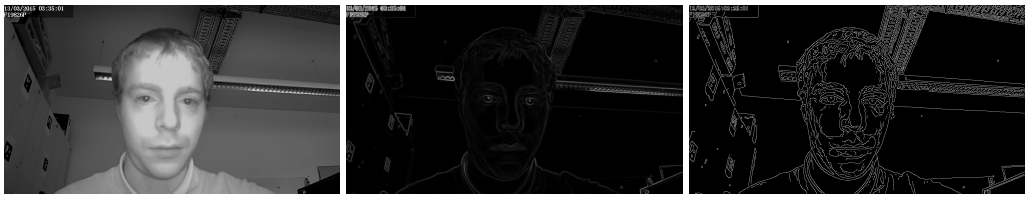


Figure 5.10: The three kinds of features used for tracking. The first image is the input image and represents raw intensity values. The second image is the gradient magnitude, whereas the last image is the result from the Canny edge detection [80].

can be intensity, gradient, or binary images. In Figure 5.10, examples of all three features are shown on the complete image.

5.5 Conclusion

In this chapter a method for automatic data annotation and detector creation was described and compared to the state-of-the-art. Additionally, a annotation tool which can be used in combination with the method is explained. In the following we will discuss the visualization of eye tracking data which serves to extract information and uses the results of the previously described algorithms.

6 Visualization of eye-tracking data

Despite methodological issues as discussed in the previous chapters, the development of eye-tracking devices with higher sampling rates creates a large amount of data needed to be handled, especially for longer experiments. In order to gain insights from these enormous data collections, eye-tracking data has to be analyzed and visualized carefully. Usually, data structuring and analysis processes are conducted with the aim of finding such relationships between eye-movement patterns and subject behavior. Such patterns consist of a sequence of fixations, where the eye is held still and directed towards the perceived areas, and saccades, or fast eye movements, during which the visual input is suppressed. A common method to further simplify the data is performed by defining regions of interest (ROI) or areas of interest (AOI), i.e., areas with a specific semantic meaning. The semantic meaning of ROIs are usually determined as having special interest to the researcher. For instance, studies on web page organization and design [93], [174] as well as graph reading [222], product design [152], and dwell time on facial areas for children with autism [10] have relied on ROI definitions to interpret eye movement behavior. For data analysis and statistics, information for each ROI can be performed separately: It can be, though is not limited to, average dwell time, or the number of fixations on the ROI. Also, connections between ROIs, such as transition probabilities, can be investigated. Manual ROI annotation is naturally a subjective step. Given the large amount of data and the desire to analyze changes in ROIs and ROI shapes associated with specific time segments, it quickly becomes laborious. Therefore, automatic ROI generation based on the data of different viewers and variable time segments is a useful and supportive automation process. It not only supports the researcher, but also allows for determining ROIs in a data-driven, objective way: For instance, ROI difference comparison between subjects, groups of subjects, or different time slices. Analysis techniques based on ROIs can be extended by the automation, such as the development of circular ROI transitions [19], that can be inspected over time and visualized as a video. This chapter introduces novel methods for eye-tracking data visualization which were developed during this thesis and demonstrates their applicability to expertise classification in a historical art viewing experiment.

6.1 State-of-the-art for automated AOI generation from eye-tracking data

In this section, existing approaches for creating areas of interest(AOI) or regions of interest (ROI) in an automated way are descibed. These methods are grouped into three categories (shape, stimulus and data based). Shape-based methods are preliminary defined regions that are not related to the data nor the stimulus. Stimulus-based methods use only the image for

for ROI creation. The main attention for ROI generation in this category goes to saliency maps. While the field of saliency maps is very vivid and hundreds of different approaches do exist, only the most commonly methods will be described in this work. For further reading an saliency maps the reader is referred to [22].

Shape-based ROI generation Shape-based ROIs are often applied for their simplicity. Their application is justified if the content of the stimuli is equally distributed over the whole area. The main disadvantage is that the borders of such ROIs will not correspond to meaningful objects. This leads to the problem that the interpretation of results can be difficult because shape sizes implies some assumptions on the data. These can be quantified by reporting the shape size, as contrary to manual annotation at different detail levels in different stimulus regions.

Data-based ROI generation Those methods utilize the semantic information contained within the eye-tracking data itself in order to generate ROIs. For example, [257] and [169] proposed thresholding approaches for fixation heat maps. In [76] a gradient-based segmentation is proposed to avoid local heatmap maxima that would result in less viewed ROIs being omitted. A mean-shift clustering on fixation locations was proposed in [187] and [203] with the same goal of not omitting less viewed ROIs. The advantage of these methods is that small calibration errors can be compensated. Human knowledge about possible stimulus segmentations can be extracted. However, enough data for a fully converged heatmap needs to be available. Furthermore, those approaches cannot cope with spatially overlapping ROIs. Therefore, they are fused to one large ROI.

Stimulus-based ROI generation Privitera et al. [188] proposed to segment a stimulus image into coherent regions. They evaluated ten different algorithms for image feature extraction and clustering and compared against human fixations. Their goal was to improve image compression quality through incorporate the regions relevant to a human to the compression. Other approaches originate from image segmentation [6], [214]. The main interest here are the exact regions of objects. As there are stimulus materials that do not contain objects, this approach is not always applicable. Therefore, a more general extraction of features motivated by the low-level processes of human vision is of interest. Most prominent approaches for such saliency maps are [105], [111].

Frequency-based saliency maps Other approaches are applied in the frequency domain of an image (Fourier transform [23]). Two representatives are frequency-tuned salient region detection [1] and spatio-temporal saliency detection [98]. The amplitude and phase spectrum is used to determine the saliency of image regions. The advantage is that the high level image structure is preserved but with the drawback of overemphasizing object boundaries.

Local and global saliency maps Color based methods can be categorized into local and global models. In local models the surrounding region of each pixel is investigated. Based

on local color contrast, a saliency value for each pixel is computed [110], [147]. The advantage of local methods is that they produce less blurry saliency maps but miss global relations between regions and textures. They are usually sensitive to edges, and therefore, prone to noise. Global models in contrast include the correlations and juxtapositions of different image regions [92], [139], [251]. Global methods are consistent in terms of image structures but come with a higher computational cost due to the involved combination possibilities. Therefore, those approaches are usually applied only to low resolution images which results in a loss of small but salient content. The approach in [1] for example uses both local and global relations to create a per pixel saliency map. The global part is the dissimilarity to the average image color and the local contrast computation is done by Differential of Gaussian filters with a preliminary blurring. In [145], a Conditional random fields (CRF) was used for saliency detection. This machine learning approach can learn to extract local features and to position them in a global context. Therefore, nodes are distributed over the image, where each node influences its neighboring nodes. Local neighborhoods represent local features and the global relationships is learned based on the layer structure. This means that higher layers learn to assess different combinations of local features. An extension of this approach was proposed in [202] and [64], where image segmentation is used additionally to group regions more accurately.

6.2 Data-driven methods for ROI generation

In the following subsections, different methods for the automated generation of ROIs in a data-driven way are explained: local maxima thresholding, heatmap gradients and overlap clustering. These methods were developed during this thesis and already published in [76], [77] and [78].

6.2.1 Threshold-based ROI algorithm

Figure 6.1(a) shows the workflow of our algorithm. It starts with pre-thresholding the fixation heatmap. Areas with a density smaller than the pre-threshold are irrelevant for further computations. For example, for all images in Figure 6.1, a pre-threshold of 1% of the maximum density of the heatmap was used.
In the next step, a local maxima in the density of the heatmap is searched. This step is motivated by the observation that two high density ROIs that are spatially close to each other get easily fused to one bigger ROI when a simple density threshold is applied now: the spread of the Gaussian applied at every fixation location makes the heatmap smooth and fusing of ROIs easier. Considering local maxima as candidate ROIs enables us to treat close-by maxima as separate ROIs and to fuse them afterwards, if applicable. Depending on the data recorded (and the measurement error of the eye-tracker), the heatmap may contain many local maxima. To determine relevant maxima, a sliding window of a user-defined size W is applied. Only the largest of all local maxima within the window is considered a valid candidate. Based on the size of the window, candidate ROIs can be fused and smaller ROIs discarded. The size of the window is highly stimulus dependent and determines the

desired detail level of the analysis.

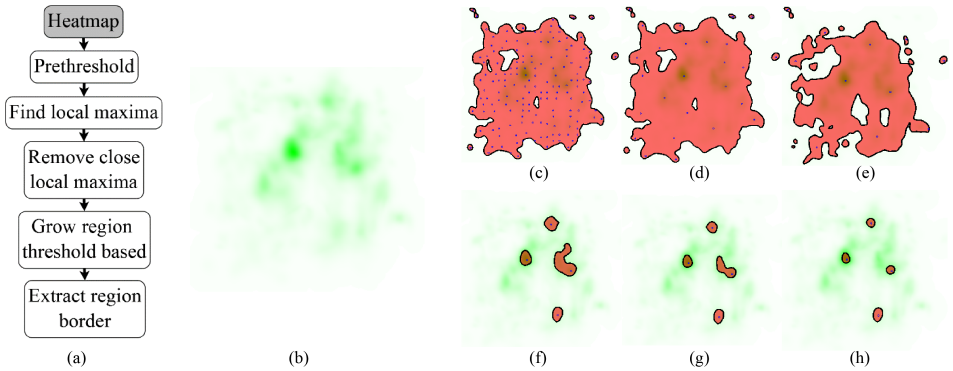


Figure 6.1: (a) shows the workflow of the threshold based ROI algorithm. (b) shows the fixation heatmap, which is the foundation of the following calculations. (c,d,e) visualize local maxima in the heatmap density as blue dots when using different window sizes (50px, 150px, 250px). The area highlighted in red is the pre-threshold of 1% of maximum density. The bottom row contains the results for different values of the threshold, where the red areas are extracted ROIs at (f) 50%, (g) 60% and (h) 70% of the density at the local maximum from (e) [77].

$$LocMax(x_i, y_i, W) = \begin{cases} 1, & I(x_i, y_i) > I(x_i + x_k, y_i + y_k) \\ & \forall x_k, y_k \in W \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

Equation 6.1 calculates the candidate selection with x_i, y_i being the pixel position in the heatmap, W the set of allowed deviations in the search window and $I(x_i, y_i)$ the density at location x_i, y_i in the heatmap. The equation assigns the value 1 to valid candidate local maxima, 0 otherwise. In the next step, a threshold is applied to each region, based on

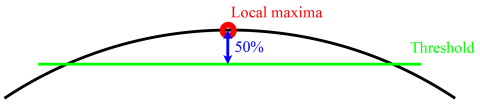


Figure 6.2: Calculation of the cutoff threshold for one local maximum. The black line represents the density distribution within the heatmap, the red dot the local maximum and the green line the calculated threshold based on 50% of the density at the local maximum [77].

a percentage of the heatmap density at the local maximum. This step is similar to the second iteration in the related approach [169] where a 50% is applied. All pixels above the threshold and connected to the local maximum are assigned to a new ROI. An example of the thresholding and how the percentage parameter influences the size of the generated ROI is shown in Figure 6.2.
Offshoots as shown in Figure 6.1(f)(red area on the right side, close to the center) or very small local maxima can combine ROIs. Polygons borders are then calculated for all ROIs. The pseudocode in Algorithm 2 shows the threshold procedure that is based on a region

Algorithm 2 Thresholding algorithm. *ROI* holds the ROI region and is initialized as the point of the local maximum. More pixels are added during the iterations. TH is the threshold density, calculated from a percentage of the density at the local maximum. a and b are 2d pixel locations within the heatmap.

Require: ROI, TH, I
  function $Grow_{region}(ROI, TH, I)$
    while $\exists a \in I | distance(a,b) < 2 \; \forall b \in ROI$ **do**
      if $TH < (I(a))$ **then**
        $add(a, ROI)$
      end if
    end while
    return ROI
  end function

growing approach. The algorithm searches for pixels neighboring the current ROI and adds them to the ROI if their density value is above the threshold. This step is repeated to convergence.

6.2.2 Gradient-based ROI algorithm

The workflow of the gradient-based ROI algorithm is shown in Figure 6.3(a). The first step is a pre-thresholding just as in the threshold-based algorithm. The impact of the pre-thresholding parameter is shown in Figure 6.3. Generally, ROIs get smaller and low density ROIs are discarded for an increasing pre-threshold. In a second step the density gradient is calculated (Figure 6.4(b)).

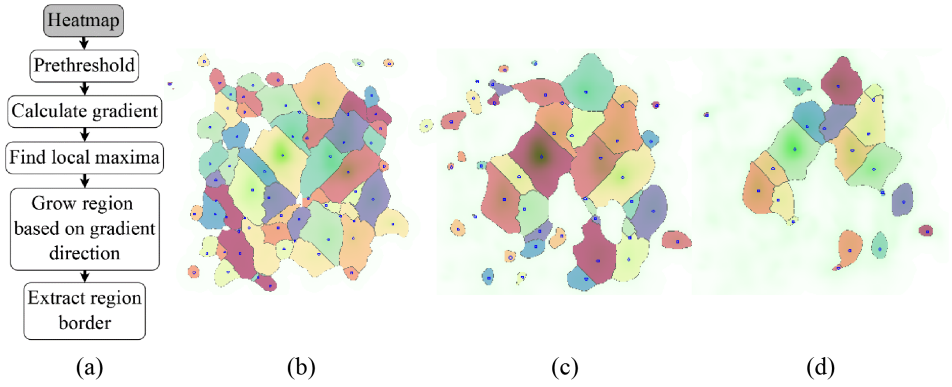


Figure 6.3: (a) shows the workflow of the gradient based ROI algorithm. In (b) the pre-threshold is applied at 1% of the maximum density level. (c) and (d) at 5% and 10%, respectively. Individual ROIs are highlighted in different colors [77].

When the gradient (the first derivative of the density) crosses zero, there is no slope in the heatmap (Figure 6.4(b) red and green circles). A derivative of zero implies a local maximum or minimum or a saddle point in the original function (see Figure 6.4). The algorithm progresses from each local maximum towards the next point. Where the gradient

crosses zero, the borders of the ROI are found. At the bottom of Figure 6.4 the assignment to ROIs is shown.

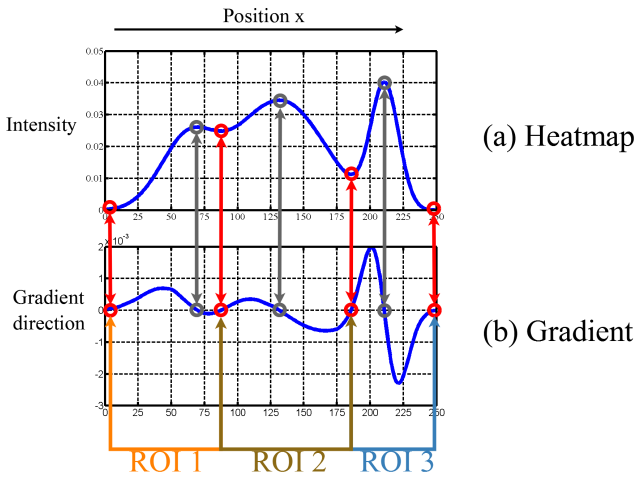


Figure 6.4: Shows the heatmap density in (a) and its first derivative, the gradient, in (b). Red circles mark breaking points of ROIs and the green circles mark are the starting points for growing a new ROI [77].

In the implementation, directional gradients were used which are pointing to the highest value in their 8-connected neighborhood and the position itself. Afterwards the algorithm starts at those values that do not point to any neighbor but have the highest density within their neighborhood. All neighbor pixels with a gradient pointing towards this location are added to the new ROI. This step is repeated until convergence.
This procedure allows ROIs to be completely contained within larger ROIs. Those contained ROIs can easily be identified by their enclosing polygon outline being completely contained within a larger polygon (considering the enclosing border polygon $ROI_i \subset ROI_j | i \neq j$). If such a ROI is found, it is joined with the enclosing ROI (considering the pixel position values $ROI_i \cup ROI_j | i \neq j$).

$$Dir(x_i, y_i, W) = \begin{cases} (x_k, y_k), & Max(I(x_i + x_k, y_i + y_k)) \\ & \forall x_k, y_k \in W \end{cases} \tag{6.2}$$

Equation 6.2 describes the gradient calculation. I is the intesity value, W contains all position shifts [-1, 0, 1] pixel in each direction (8 neighbors and pixel itself) and x_i, y_i is the starting location. The formula returns the vector to a neighboring maximum or $(0,0)$ if the position itself is the maximum in its neighborhood. The algorithm for growing the region via the gradient is shown in Algorithm 3 [77].

6.2.3 Overlap clustering

Overlap clustering is based on geometrical observations. Fixations for example are areas which are approximated in a very simplified fashion by a circle. Based on these geometric

Algorithm 3 Gradient-ased region growing. *ROI* holds the ROI region and is initialized as the point of the local maximum. More pixels are added during the iterations from neighboring positions a [77].

Require: ROI, I
  function $Grow_{region}(ROI, I)$
    while $\exists a \in I \mid Dir(a) + a \in ROI$ **do**
      $add(a, ROI)$
    end while
    return ROI
  end function

structures, their relationship can be determined. In [129], the authors describe how to fit ellipses to samples recorded during a fixation and how the ellipsoid shape can be used for data quality assessment, as the actual spatial extent of the fixation is represented by the ellipse axes. In this chapter the elliptic shape is used for the clustering of fixations.

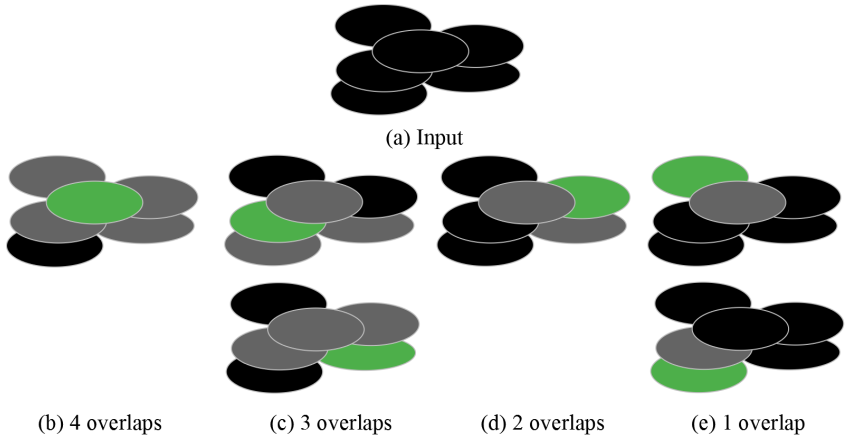
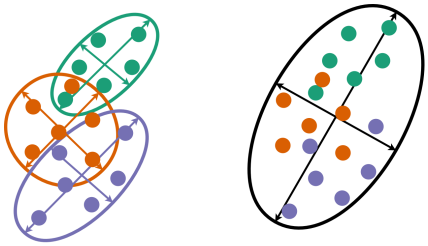


Figure 6.5: Overlap clustering procedure. (a) Ellipse representation of a set of fixations. In (b,c,d,e) All fixations that overlap the ellipse of the fixation currently under consideration (green) are highlighted (in gray). Each such set of a fixation and its overlaps (green plus gray ellipses) is considered a cluster. Fixations that do not participate in the cluster are shown in black. The clusters are ordered from left to right in descending order of cluster size [77].

Therefore, intersections between the ellipses are calculated and overlapping ellipses are merged to clusters. The process is visualized in Figure 6.5.
In the second step, overlapping clusters are merged together. This starts from the largest cluster, i.e., the one shown in Figure 6.5(b), and searches for other clusters that overlap with it. If two clusters overlap, they are merged together. This process is repeated until convergence. Afterwards the process is repeated with the largest of the remaining clusters that can still be extended. This is done by calculating the mean minor and major axis using the principal component analysis on all gaze points belonging to the fixation ellipses included in the cluster. This is shown in Figure 6.6, where in (a) the fixations and their gaze points are drawn in the same color and in (b) the resulting cluster shape is shown. The axis

Algorithm 4 First clustering step: F is a list containing all fixations, AC will contain all found clusters after the algorithm is run.

Require: F, AC
  function $Init_{Cluster}(F, AC)$
    for $a \in F$ **do**
      $add(a, C)$
      for $b \in F$ *and* $a \neq b$ **do**
        if $a \subseteq b$ **then**
          $add(b, C)$
        end if
      end for
      $add(C, AC)$
    end for
    return AC
  end function



(a) Fixations (b) Overlap Cluster

Figure 6.6: (a) Fixations are the outlines of ellipses, whereas the dots represent the gaze points. Dots and ellipses of the same color belong together. (b) The black ellipses represents the overlap cluster calculated based on all contained points. Arrows outgoing from the center of the ellipses are the axis calculated from the principal component analysis [77].

of all ellipses (indicated by arrows) are the vectors calculated with the principal component analysis.

Figure 6.7 visualizes this procedure. The largest cluster of the first step is chosen as a starting point and successively enlarged by overlapping clusters. In the example, one minor cluster is merged with the larger one ((d) new dark grey ellipses). Once the cluster cannot grow anymore, the next cluster is processed which has not been merged yet (c). The two final clusters are shown in (d).

The algorithm is adjustable by specifying a minimum count of overlapping fixations required to create a cluster. This minimum fixation threshold can either be applied to all data together or to the data of each subject separately. This way the algorithm can also cope with large and dense data. Figure 6.8 shows each of the explained methods applied to the same data. Obviously, the generated ROIs are not identical. Every method has its advantages and disadvantages; for example the gradient and mean shift approach generate a lot of ROIs, partially based on the nature of the methods, but also on the parameter choice. As described previously, this algorithms are integrated into EyeTrace, where ROIs can be

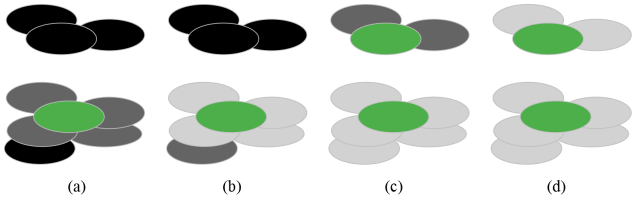


Figure 6.7: (a) is the starting cluster, which is the largest possible cluster from the first step. The green ellipses is the starting point of a cluster and the dark gray clusters belong to it. In (b), the first merge is applied, visualized here by the new dark gray ellipses and the light gray ellipses already contained in the cluster. (c) shows that the bottom cluster cannot grow any more and is therefore finished. The next starting cluster in (c) is the top cluster visualized by the new green ellipses and the ellipses belonging to it (dark gray). (d) is the result of the overlap clustering for the given ellipses [77].

Algorithm 5 Cluster merging step: *AC* contains the clusters found by the first step, merged clusters are stored in *GC*.

Require: AC, GC
  function $Merge_{Cluster}(AC, GC)$
    while $AC > 0$ **do**
      $C = max(AC)$
      $remove(C, AC)$
      while $\exists a \in C | a \in AC$ **do**
        $C_{sub} = get(a, AC)$
        $add(C_{sub}, C)$
        $remove(C_{sub}, AC)$
      end while
      $add(C, GC)$
    end while
    return GC
  end function

removed, manually added and modified in case that the automatically generated ROIs are too large/small or unavailable.
The usefulness of a ROI however, depends always on the quality of the recorded data and the goal of the study or task. The data quality refers to the size of the ROI needed to capture the gaze attention of a subject sufficient enough to see similarities or gaze behavior and excluding unrelated behavior. For example, data with low precision needs larger ROIs, whereas large ROIs could induce an error in high-precision data. In case of low accuracy data, which induces a localization problem, automatic ROI generation can help visualizing the offset.



(a) Heatmap **(b)** Meanshift **(c)** Overlap **(d)** Threshold **(e)** Gradient

Figure 6.8: (a) shows the fixation heatmap where red is the highest value. In (b,c,d,e) calculated regions of interest are shown using meanshift clustering (b), overlap clustering (c), threshold based (d) and gradient based methods (e) [77].

6.3 ROI experiments and method comparison

In the following, cumulative clusters will be refered to as clusters. In research literature on visual behavior, clusters are calculated on the data of a single subject. For cumulative clusters, data of more than one subject is combined. This is an important prerequisite for ROI generation, because otherwise the clusters would overlap. In the following evaluation, the aim is to compare the explained methods with the state-of-the-art meanshift algorithm [187]. The first part of the evaluation is related to the impact of automatic ROI generation on statis-



(a) Original **(b)** Manually **(c)** Meanshift **(d)** Overlap **(e)** Threshold GP **(f)** Threshold Fix **(g)** Gradient GP **(h)** Gradient Fix

Figure 6.9: Shows the used clusters for experiments in 6.3.1. (a) is the original image and in (b) the manually annotated ROIs with labels are shown. In (c) the clusters found by meanshift, (d) overlap clustering, (e) threshold and (g) gradient based gaze point heatmap ROIs, (f) threshold and (h) gradient based fixation heatmap ROIs [77].

tical values. Therefore, the generated ROIs are compared based on the results of common ROI statistics. Additionally, the area differences between manually annotated and ROIs generated in an automated way are compared. In the second evaluation part, the generated ROIs are evaluated in an classification task. The aim of that task was to distinguish between expert and novice art viewers. Afterwards, advantages and disadvantages of each method are summarized and applications to abstract art are discussed.

6.3.1 Automatic vs. annotated statistics

The recordings used for evaluation stem from an art viewing experiment with nine subjects which was conducted at the Department of Computer Science at the University of

Table 6.1: The parameters used to calculate the images are shown in Figure 6.9. Letters behind the method name refer to Figure 6.9 [77].

Parameter	Meansh.(c)	Over.(d)	Th. GP(e)	Th. Fix(f)	Gr. GP(g)	Gr. Fix(h)
Radius	35px	-	-	-	-	-
Minimum Fix	30	15	-	-	-	-
Gradient	1	-	-	-	-	-
Iterations	100	-	-	-	-	-
Threshold	-	-	50%	50%	-	-
Prethreshold	-	-	20%	20%	25%	25%
Window	-	-	250px	170px	-	-

Tübingen. The subjects viewed for one minute an image from the "The Art of Painting". Eye movements were recorded by means of an EyeTribe eye-tracker at 25 Hz sampling rate. Therefore, almost one thousand fixations were recorded. The ROIs evaluated for these recordings are shown in Figure 6.9.

Table 6.2: Averaged statistic results over all nine subjects (mean gaze position of both eyes) for generated ROIs and annotated ones. The first column specifies the annotated object and the second one the used method. Columns three through six contain statistics, e.g. amount of gaze points (GP), average gaze point duration (GP dur) in ms, amount of fixations (Fix) and average fixation duration (Fix dur) in ms [77].

ROI	Method	GP	GP dur	Fix	Fix dur	ROI	Method	GP	GP dur	Fix	Fix dur
	Manually	83.3	61.2	5	455.1		Manually	48.2	32	1.8	480.4
	Meanshift	96.3	140.4	7	357.1		Meanshift	194.7	359.1	11.2	546
	Overlap	89.6	121.5	6.3	372.4		Overlap	141.7	252.7	7.8	542.5
Face	TH GP	175.2	295.3	10	549.6	Mask	TH GP	101	393	5.8	614.8
	TH Fix	168.4	267.8	9.7	554.8		TH Fix	93.3	213.4	5.3	612.1
	Grad	232.7	371	14.2	489.3		Grad	103.4	435.8	6	604.1
	Grad Fix	242.6	412.3	14.6	490.8		Grad Fix	98.1	229.3	5.5	605
	Manually	63.1	90.3	5.3	329		Manually	84.4	150.2	6.3	454.2
	Meanshift	79	219.5	5.5	454.1		Meanshift	112.8	368.7	6.3	477.3
	Overlap	46.4	135	3.4	446.5		Overlap	104.6	411.2	5.5	485.8
Head	TH GP	175	264.1	10.8	497.7	Chand.	TH GP	59.7	159.3	4.1	324.7
	TH Fix	156.3	203.7	10.3	454.4		TH Fix	62.4	157.8	4.4	328.2
	Grad	196.7	298.1	12.3	481.8		Grad	61.7	171.1	4.2	336.6
	Grad Fix	92.8	105.8	7.2	334.3		Grad Fix	65.4	184.7	4.6	325.1

Where applicable, the parameters to get the best ROI representations for the face of the woman, the painter's head, the mask and the chandelier, are chosen because it is expected that these regions attract most of the gaze (as can be seen in Figure 6.8(a)). The used parameters for the ROI computation are shown in Table 6.1 (letters refer to Figure 6.9). Fixations are identified with the Bayesian Mixture Model proposed in [233]. Furthermore, a measure of how similar the segmented areas are to the manual annotations is calculated by the equation $\frac{A \cap B}{A \cup B}$, where A is the manually annotated ROI area and B the generated one.

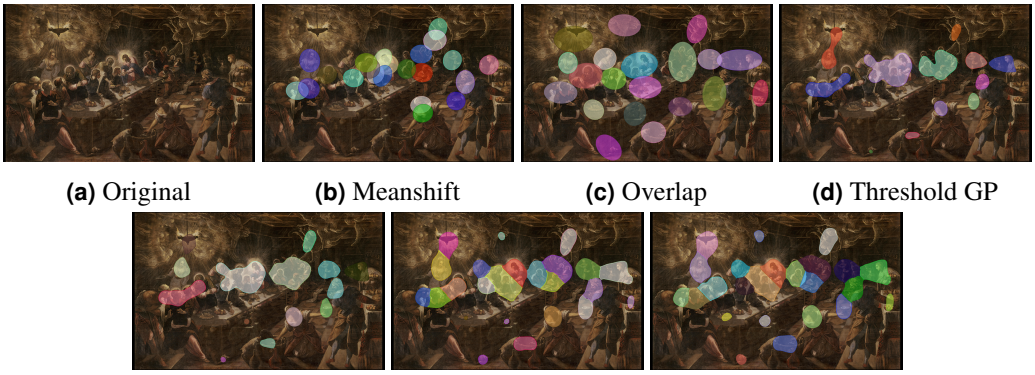
Table 6.3: Area similarity of the generated ROI and the manually annotated one using the formula $\frac{A \cap B}{A \cup B}$. Values closer to one represent a better area similarity. The analysis was performed separately for different ROIs given in the column heading [77].

Method/ROI	chandelier	Head	Face	Mask
Meanshift	0.29	0.49	0.29	0.21
Overlap	0.19	0.48	0.45	0.30
Threshold	0.15	0.16	0.34	0.31
Threshold fix.	0.17	0.20	0.41	0.35
Gradient	0.16	0.18	0.17	0.30
Gradient fix.	0.18	0.36	0.15	0.34

In Table 6.2, the resulting fixation and gaze point statistics are shown. As key metrics for the statistics, the amount of gaze points (GP), the average gaze point duration (GP dur), the amount of fixations, and the average fixation duration are used. All values are averaged over all nine subjects. As can be seen in Table 6.2, the mean statistical values between the meanshift approach and overlap clustering are similar, except for the head ROI, where the shape is rather vertically stretched in the case of overlap clustering and rather horizontally (overlapping the painting ROI) for the meanshift approach (Figure 6.9(d)). The same effect can be observed for the gradient approach on the fixation heatmap, which also separates the head ROI vertically (Figure 6.9(h)). In terms of area overlap, the effect is negligible (Table 6.3). For the woman's face, in terms of segmentation quality, there is a large difference between meanshift and overlap clustering. This is due to the overlap clustering partitioning the ROI into two subregions (Figure 6.9(d)). Statistically, there is only a difference between the gradient approach and all the others, which is due to the larger ROIs (Figure 6.9(g,h)). Table 6.3 shows that the chandelier ROI metrics are highest for the meanshift method. This effect is small when compared to the overlap clustering, but the threshold and gradient approach yield different results for all metrics. This is due to the smaller area resulting from the pre threshold step.
For the mask ROI, the gradient and threshold based approach have the best segmentation results (Table 6.3). In this case the effect can also be seen in the statistics Table 6.2. The largest amount of gaze points and fixations are in the meanshift approach, which is due to the large cluster, as can be seen in Figure 6.9(c).

6.3.2 ROI generation in a classification task

Aim of this evaluation was to showcase the usefulness of the ROI generation approach in a classification task. For this evaluation the data recorded in [198] at the University of Vienna is used. It contains eye movement data from 40 subjects, 20 experts and 20 novices. The used eye-tracker model was "IViewX RED 120" at a sampling rate of 120 Hz. The recordings were performed while the subject was sitting in front of an 30 inch monitor with a screen resolution of 2560x1600 pixels. Each subject viewed the artwork for 2 minutes at a head distance of 0.9 meters ($\tilde{3}$ feet). To perform a classification of expertise based on eye movement data, ROIs were extracted and a support vector machine (SVM) classifier was trained. The evaluation was done using a 20 folds cross validation based on the Matlab



(a) Original **(b)** Meanshift **(c)** Overlap **(d)** Threshold GP

(e) Threshold Fix **(f)** Gradient GP **(g)** Gradient Fix

Figure 6.10: Shows the used clusters for experiments in 6.3.2. (a) is the original image. In (b) the clusters found by meanshift, (c) overlap clustering, (d) threshold and (e) gradient based gaze point heatmap ROIs, (f) threshold and (g) gradient based fixation heatmap ROIs [77].

2015b's SVM implementation. Common praxis with SVM classification is to evaluate for different parameters and select the best performing result. The kernel scale parameter is evaluated in the range $1-10$ with a step wide of 0.1. The evaluated kernels are *'linear'*, *'Gaussian'* and *'polynomial'*. Each kernel function is evaluated with data standardization from Matlab. The used statistical values for each ROI are given in the following list where X stands for gaze point (G) or fixation (F).

S_{X1} time of first entry
S_{X2} amount
S_{X3} per minute
S_{X4} share
S_{X5} total time
S_{X6} minimal consecutive time
S_{X7} maximal consecutive time
S_{X8} average consecutive time

From those 16 statistical values ($S_{G1-8} + S_{F1-8}$) all possible subsets of $1-4$ are evaluated and the best result was selected for classification.

Table 6.5 shows the results for each method in combination with a specific kernel. As can be seen the highest score is achived by the meanshift clustering. This is due to the overlapping clusters and the more centered localization in the image. The worst results are obtained by the gradient-based ROIs for this scenario. Overall, it can be seen that all methods can be used to achieve results above chance level (50%). Higher classification results can be achived by evaluating more combinations of statistical values, including transitions, increasing the kernel scale and calculating ROIs with different parameters.

Table 6.4: The parameters used to calculate the images shown in Figure 6.10. Letters behind the method name refer to Figure 6.10 [77].

Parameter	Meansh.(c)	Over.(d)	Th. GP(e)	Th. Fix(f)	Gr. GP(g)	Gr. Fix(h)
Radius	35px	-	-	-	-	-
Minimum Fix	100	80	-	-	-	-
Gradient	1	-	-	-	-	-
Iterations	100	-	-	-	-	-
Threshold	-	-	50%	50%	-	-
Prethreshold	-	-	20%	20%	20%	20%
Window	-	-	250px	170px	-	-

Table 6.5: The classification results for all ROI generation algorithms with different kernels [77].

Method	Linear	Gaussian	Polynomial
Meanshift	72.5%	85%	77.5%
Overlap	75%	80%	75%
Threshold GP	75%	75%	75%
Threshold Fix	80%	77.5%	77.5%
Gradient GP	70%	72.5%	72.5%
Gradient Fix	70%	72.5%	67.5%

6.3.3 Discussion

In Figure 6.11 some strong and weak points mentioned in Table 6.6 for each method, are shown. (a) shows the over segmentation of the gradient approach. This happens if the prethreshold is set very low. The threshold approach can segment large ROIs as shown in Figure 6.11(b). This effect can also be disadvantageous if the ROIs, which should be separated, intermingle, as shown in Figure 6.11(e) (face and mask). Figure 6.11(c,d) show the overlapping occuring for the meanshift and overlap approach. This is due to the result of the principle component analysis and the region approximation as an ellipses. Another weak point of the overlap clustering is shown in Figure 6.11(c) and (f), where it can be seen that the cluster size is fixed. This is due to the fixed size of the fixation ellipses.



(a) Gradient **(b)** Threshold **(c)** Overlap **(d)** Meanshift **(e)** Threshold **(f)** Overlap

Figure 6.11: Exemplary weak and strong point visualizations for the different methods. (a) over segmentation, (b) larg segment ROI, (c,d) overlapping example, (d) large clusters, (e) intermingle ROIs and (f) fixed size [77].

When it comes to abstract paintings, a top-down definition of ROIs is difficult, since there

Table 6.6: Advantages and disadvantages for each implemented method [77].

Method	Pros	Cons
Meanshift	size adjustable finds clusters with low gaze activity	four parameters relies on fixations clusters can overlap
Overlap	one parameter finds clusters with low gaze activity minimum fixations can be applied	size fixed relies on fixations clusters can overlap
Threshold	size adjustable different input data segmentation non overlapping	three parameters problems with low gaze activity clusters
Gradient	one parameter size adjustable finds clusters with low gaze activity different input data segmentation non overlapping	tends to over segmentation ROIs are in close contact to each other



Figure 6.12: Jackson Pollock's "Convergence" with cumulative clusters (red ellipses). Clusters where calculated using overlap clustering and a minimum of 250 fixations per cluster [77].

are no semantically meaningful objects (e.g. persons) depicted. In such cases automated ROI generation is very helpful. One example is the famous work `"Convergence"` from Jackson Pollock, which is a very dynamic and agitated painting [37]. Since the colors are smeared on the canvas, it is almost impossible to predict, where the eyes of an observer will fixate. With cumulative clusters it is possible to investigate regions in the painting that were most fixated by the observers. In such cases, an automated ROI generation can be helpful. In Figure 6.12, ROIs computed with cumulative clusters and using overlap clustering are shown. Those regions seem to have a high contrast and attracted the observer's attention with salient red colors. The second cluster on the left is an exceptional case. Here the salient color is blue. The network of fined tossed lines surrounding it supporting the saliency of the region.

Another application for ROIs are the transitions between regions. Here saccades and scan-path can be analyzed. Figure 6.13 shows the transitions calculated between clusters. The

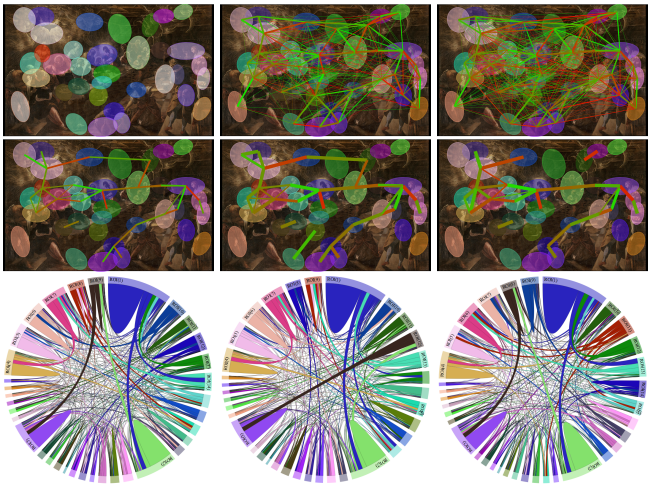


Figure 6.13: Jacopo Tintoretto's "The Last Supper" with cumulative clusters calculated using the overlap clustering. In (a) the calculated clusters are shown. (b) and (c) show the transitions (saccades and scanpath) between those clusters as overlay. In the second row the saccade transitions are shown using a normed threshold for all, novices and experts. The last row shows chord diagrams, where the ROIs are the outline and the saccades build the connecting polygonal shapes between them [77].

first row shows the clusters and transitions based on saccades and scanpath as overlay on the image. Those images look very confusing without filtering. The second row in Figure 6.13 shows the saccades as transitions for all subjects and separated by groups (experts and novices). For thresholding, a fixed value (0.4) was used on the transitions normed by time (one minute) and subject count. In this visualization, it can be seen that both groups differ significantly.

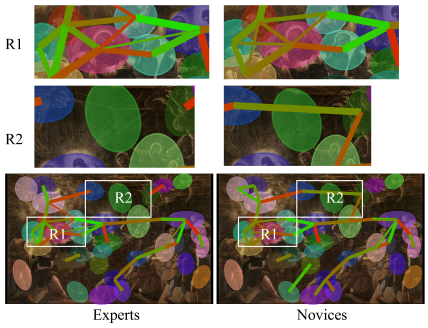


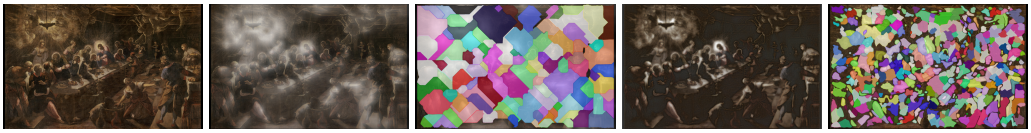
Figure 6.14: Shows the clear difference between experts and novices for images (e) and (f) from Figure 6.13. R1 and R2 are the region identifiers [77].

For a better visualization, two regions from Figure 6.13(e) and (f) are highlighted in Figure 6.14. It can be seen that Region 1 (R1) is different due to the strength difference of the vertical line on the left side. It has to be noted that the width of a line indicates the global proportion of a connection. In addition, the thin horizontal line on the left is not present for the experts. The reverse case for the thin horizontal line on the right side. In Region 2 (R2)

the difference is obvious.
Another impressive visualization technique is shown by the chord diagrams in Figure 6.13. Without filtering, the main differences and similarities between experts and novices are visible. One example is the skewed dark line only visible for novices and the dark curve on the left side of the diagram, which is only visible for experts. A clear similarity is the blue vertical curve, present for both groups.

6.4 Saliency-based ROI generation

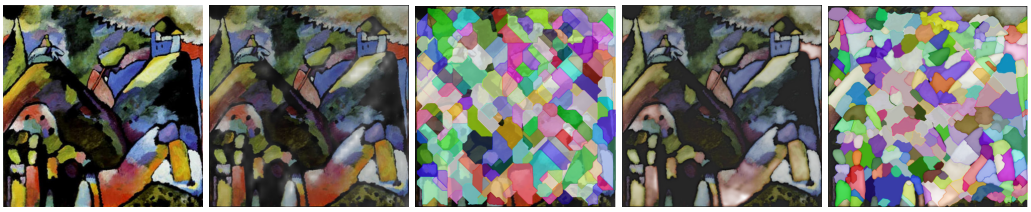
The previously described and evaluated methods for ROI creation utilize the semantic knowledge of the viewer about the stimulus material. Distinct fixation targets correspond to distinct ROIs. However, there are also cases, where this association fails, for example due to low eye tracking accuracy compared to the resolution of the stimulus material. Another scenario where this association fails are ROI regions which overlap, such as a face partially occluding another face.
Due to those limitations, the idea is to integrate early features of the human visual system into determining ROI boundaries. Such a method generally over-segments the image e.g. Figure 6.8(e). In contrast, it allows to distinguish ROIs even when one region is very dominant. The over-segmentation can easily be resolved once the actual eye-tracking data is applied to the ROIs. Many of them will contain only very few fixations and can therefore be removed. This means we (1) Calculate the saliency map based on the stimulus image, and (2) Compute ROIs based on the saliency heat map.
Figure 6.15 shows some stimulus images with generated ROIs. The first artwork (Figure 6.15(a)) is the famous painting "The Last Supper" by Jacopo Tintoretto. It illustrates a complex dark scene with bright spots at the gloriole and the hanging oil lamp. Those two regions are the most salient areas for both algorithms. The gradient-based ROI generation algorithm enables to extract other regions as well (Figure 6.15(c,e)). The second representative for artworks is "Paradise" by Lukas Cranach shown in Figure 6.15(f). The generated ROIs (Figure 6.15(h,j)) of both algorithms separate the persons in the center of the image well. For the method from Itti and Koch [111] they are more coarse, which comes due to the downscaling. This artwork is a perfect example, where the stimulus-based approach outperforms the data-driven ROI generation.
In addition to these classical paintings, some examples for abstract art are shown in (Figure 6.15(k,p)), namely "Improvisation 9" by Vasilii Kandinsky and "Shimmering Substance" by Jackson Pollock. Figure 6.15(m,o) and (r,t) show that the ROI generation based on the saliency over-segments the scene. In case of a data-driven approach it would be possible to extract only larger ROIs as shown in Figure 6.12. Since abstract art usually does not contain objects, it would be also impossible to refine the large ROIs. Therefore, the stimulus based approach with the over-segmentation outperforms again the data-driven ROI generation. In addition, enables new ways of analyzing abstract art.
In the following, the approach is compared to fixation mean-shift clustering [203], gaze heatmap ROI generation based on the gradient [76], and based on thresholding [169].



(a) The Last Supper, Jacopo Tintoretto **(b)** Saliency Itti **(c)** ROI Itti **(d)** Saliency Frequency-tuned **(e)** ROI Frequency-tuned



(f) Paradise, Lukas Cranach **(g)** Saliency Itti **(h)** ROI Itti **(i)** Saliency Frequency-tuned **(j)** ROI Frequency-tuned



(k) Improvisation 9, Vasilii Kandinsky **(l)** Saliency Itti **(m)** ROI Itti **(n)** Saliency Frequency-tuned **(o)** ROI Frequency-tuned

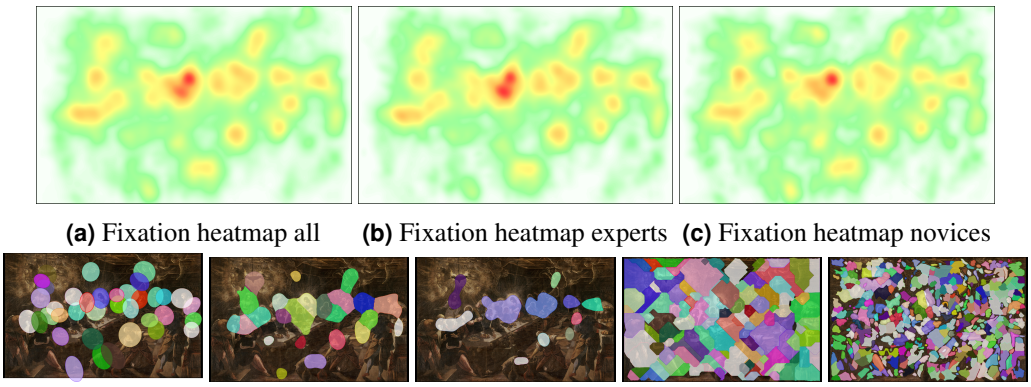


(p) Shimmering Substance, Jackson Pollock **(q)** Saliency Itti **(r)** ROI Itti **(s)** Saliency Frequency-tuned **(t)** ROI Frequency-tuned

Figure 6.15: (a,f,k,p) show the stimulus images. In (b,g,l,q) the [111] saliency map for the stimulus image is shown, and in (c,h,m,r) the ROIs generated based on these. (d,i,n,s) show the frequency-tuned [1] saliency map for the stimulus image and (e,j,o,t) the ROIs generated based on these [78].

6.4.1 Evaluation procedure

For evaluation,we employ the previously described data set, which was recorded by [198] at the University of Vienna (Section 6.3.2).



(a) Fixation heatmap all **(b)** Fixation heatmap experts **(c)** Fixation heatmap novices

(d) ROIs from clusters **(e)** ROIs from heatmap gradient **(f)** ROIs from heatmap threshold **(g)** ROIs from Itti **(h)** ROIs from frequency-tuned

Figure 6.16: The first row shows fixation heatmaps for (a) all, (b) only the experts and (c) only novices. In the second row the generated ROIs for (d) the fixation clustering, (e) heatmap, (f) Itti saliency map and (g) frequency-tuned saliency map are shown [78].

ROIs were generated jointly on data of all subjects. A gaze heatmap of all subjects is shown in Figure 6.16(a). A visual comparison of Figures 6.16(b) and (c) show a strong overall similarity with only small differences. For the evaluation, the following key metrics for each ROI and subject are calculated:

time to first gaze (S_1), amount of gaze points (S_2), gaze points per minute (S_3), share of gaze points (S_4), total time of gaze points (S_5), minimal consecutive time of gaze points (S_6), maximal consecutive time of gaze points (S_7), average consecutive time of gaze points (S_8), time of first fixation (S_9), amount of fixations (S_{10}), fixations per minute (S_{11}), share of fixations (S_{12}), total time of fixations (S_{13}), minimal consecutive time of fixations (S_{14}), maximal consecutive time of fixations (S_{15}), average consecutive time of fixations (S_{16})

For the following classification, the accuracy could be increased by using metrics of consecutive ROIs. However, we did not undertake such optimizations. In addition, the transitions would increase the number of features exponentially since they are also applicable in different kinds (global, relative to ROI, incoming or outgoing, transitions as saccade or scan path etc.). All statistics were collected based on the averaged gaze position between both eyes.

Figure 6.16(d,e,f,g,h) shows the generated ROIs for all compared methods. As can be seen in Figure 6.16(d) and (e,f), clustering and generating ROIs from the heat map produces similar results. Figure 6.16(g) and (h) show the results of the two saliency maps. As can be seen they are over-segmented.

6.4.2 Results of the saliency based ROI evaluation

As classificator the Matlab 2015b's support vector machine (SVM) was used and configured as *'Standardize'=false, 'KernelFunction'='linear', 'KernelScale'='auto', 'OutlierFraction'=0.0, 'Nu'=0.5*. 100 was the initial random seed. We performed a 20-fold cross-validation. The features $\{S_1,\ldots,S_{16}\}$ were evaluated in different combinations, where k represents the amount of combined features. This means that for $k=3$ all possible triple combinations of features are evaluated. For automatic ROI generation, five methods were applied and compared to each other.

CLU Fixation mean-shift clustering [187], [203].
$HEAT_G$ Heatmap Gradient [76].
$HEAT_T$ Heatmap Threshold [169], [257].
SAL_I Saliency maps generated with Itti et al.'s method [111].
SAL_{FT} Saliency maps generated with the Frequency-Tuned method [1].

The simplest and most intuitive approach would be the comparison based on the best classification results. Those are shown in Table 6.7. $HEAT_T$ has the best linear classification result for $k=1-4$ mainly due to feature S_12 (*share of fixations*) with the tightly fitted ROIs (shown in Figure 6.16(f)).

Table 6.7: Maximal classification result per k for each ROI generation algorithm (using all ROIs) [78].

k	*CLU*	$HEAT_G$	$HEAT_T$	SAL_I	SAL_{FT}
1	0.60	0.5500	**0.7500**	0.6750	0.6250
2	0.6750	0.5750	**0.7500**	0.6750	0.6250
3	0.6750	0.5500	**0.7500**	0.6750	0.6250
4	0.6750	0.5500	**0.7500**	0.6750	0.5750
5	**0.6750**	0.5500	**0.6750**	**0.6750**	0.5750
6	**0.6750**	0.5250	0.6500	**0.6750**	0.5250

While the heatmap threshold method reaches superior performance in Table 6.7, it should be noted that the saliency-based methods can keep up with the other data-driven methods. Another important fact is that the heatmaps were generated based on all data sets. For a real scenario, the evaluated data set would not have been contributed to this heatmap. Therefore, a better approach is to evaluate the robustness of the classification for different feature sets. This is an important consideration, as ROIs that show us as many inter-group effects as possible are preferable, not only the strongest ones.

Another way to evaluate the quality of the ROIs is therefore their stability across different feature sets. This means that the capability of the ROIs to extract information out of the statistical values is analyzed. Therefore, the entire set of combinations possibilities per k is evaluated and the mean and standard deviation of all results is computed. The mean score

for a method is then given by

$$S_\mu(ROI_m, CS, k) = \frac{\sum_{i=1}^{\binom{|S|}{k}} CVS(ROI_m, CS_i)}{\binom{|S|}{k}}, \tag{6.3}$$

where m represents the ROI generation method, $|S|$ is the amount of statistical values, CS_i the Combination Set of statistical values, $\binom{|S|}{k}$ the binomial coefficient and $CVS()$ the Cross-Validation Score for the classification. With the mean $S_\mu(ROI_m, CS, k)$ the standard deviation can be defined as in Equation 6.4.

$$\sqrt{\frac{\sum_{i=1}^{\binom{|S|}{k}} (CVS(ROI_m, CS_i) - S_\mu(ROI_m, CS, k))}{\binom{|S|}{k} - 1}} \tag{6.4}$$

The best result for one feature (S_1) using Equation 6.3 is reached by $HEAT_T$, which

Table 6.8: The calculated score (mean) for each method using Equation 6.3 (using all ROIs) [78].

k	CLU	$HEAT_G$	$HEAT_T$	SAL_I	SAL_{FT}
1	0.4328	0.4516	**0.5391**	0.5016	0.4438
2	0.4290	0.4525	0.5073	**0.5198**	0.4148
3	0.4250	0.4508	0.4885	**0.5283**	0.4085
4	0.4252	0.4505	0.4780	**0.5375**	0.4072
5	0.4284	0.4502	0.4696	**0.5470**	0.4068
6	0.4335	0.4500	0.4608	**0.5564**	0.4060

are the most restrictive ROIs (first row Table 6.8). For the other feature combinations (S_{2-16}), this method is outperformed by SAL_I. It can also be observed in Table 6.8 that the only method continuously improving its score is SAL_I, while the others decrease in linear classification performance. This means that SAL_I is the overall most robust ROI set, with results constantly over chance level. In Table 6.9, the standard deviations are shown.

Table 6.9: The calculated standard deviation for each method using Equation 6.4 (using all ROIs) [78].

k	CLU	$HEAT_G$	$HEAT_T$	SAL_I	SAL_{FT}
1	**0.0884**	0.0716	0.1208	0.1192	0.0892
2	0.0849	0.0645	0.1077	0.1002	**0.0520**
3	0.0820	0.0531	0.0925	0.0837	**0.0328**
4	0.0790	0.0454	0.0835	0.0716	**0.0227**
5	0.0769	0.0413	0.0791	0.0641	**0.0189**
6	0.0753	0.0394	0.0764	0.0603	**0.0182**

Those values hold information about the reliability of the results from Table 6.8, which is indicated by a low standard deviation. As can be seen, the reliability for higher feature

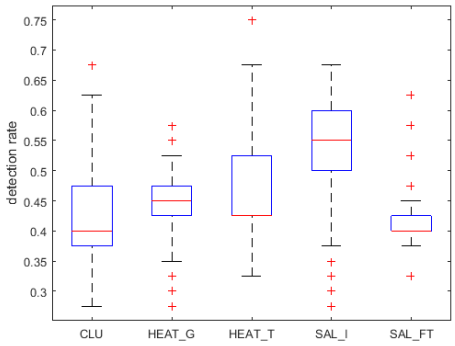


Figure 6.17: Whisker plot over all feature combinations S_{1-16}. The red line is the median. The blue box represents the 25% and 75% percentiles. Black horizontal lines are the minimum and maximum of the evaluated data and the red crosses represent outliers. It has to be mentioned that higher feature set combinations are over-represented due to the higher combination possibilities [78].

combination increases for all ROI generation algorithms. In addition, box plots are shown in Figure 6.17. Here the higher feature combinations dominate the result. Again it can be seen that SAL_I outperforms the other approaches especially if the 25% and 75% percentiles are considered.
These results indicate that ROIs generated on the saliency map proposed by [111] can be applied to art viewing experiments. As they can be computed even before the data is recorded, they could be used for online classification scenarios without the need of recording large amounts of data per piece of art.

6.5 Conclusion

In this chapter methods for automatic AOI generation were described as well as their usage in visualizations and the analysis of eye-tracking data. In the next chapter an online system for automatic focus adjustment will be described.

7 All in practice: Gaze-based focus adaption

Modern digital microscopes and cameras share a similar problem in the sense that the auto focus is only applied to the center of the field of view of the camera. For various applications, such as microsurgery, microscopical material inspection, and human-robot collaborative settings, it would be a significant usability improvement to allow users to adjust the focus in the image to their point of interest without reorienting the camera or requiring manual focus adjustments. This improvement would not only generate a benefit for the user of the optical system, but also to non-users: For instance, patients would benefit from a faster surgery and a less strained surgeon. Applied to different monitors, the efficiency gain would be even greater. This chapter combines different methods presented during this thesis to develop a gaze-based autofocus technique and to thus showcase both the applicability and real-time capability of the proposed algorithms. We propose a gaze-based autofocus technique based on a commercial eye tracker that captures the subject's gaze. This gaze information is then mapped to a screen where the camera images are presented. The gaze imnformation position on the image is mapped to the estimated depth map, from which the focal length of the camera is automatically adjusted. This adaption enables the user to quickly explore the scene without manually adjusting the camera's focal length.
Section 7.1 discusses related work in this realm. The proposed hardware setup is presented in Section 7.2. Workflow and the focus estimation method is described in Section 7.3. Finally, a comparison to the state-of-the-art is presented in Section 7.4.

7.1 Related work

Different approaches exist to compute a 3D representation of a given a set of images produced with different camera focal lengths. Therefore, two main steps have to be applied. The first step is measuring how focused each pixel in this set is. Due to plain surfaces and noise in those images, not all measures are correct. Therefore, an interpolation is applied, which smooths the depth over the entire image. In the following, the shape-from-focus measuring operators are described briefly, grouping then similarly to Pertuz et al. [181].

- **Gradient-based** measure operators are first or higher order derivatives of the Gaussian and are commonly applied for edge detection. The idea here is that unfocused or blurred edges have a lower response than sharp or focused edges. The best performing representatives according to [181] are first order derivatives [91], [200], second central moment on first order derivativs [179], and the second central moment on a Laplacian (or second order derivatives) [179].
- **Statistics-based** measurements are based on calculated moments of random variables. These random variables are usually small windows shifted over the image.

The idea behind statistics for focus measurements is that the moments (especially the second central moment) reach their maximum at focused parts of the image. According to [181], the best representatives are [265] using Chebyshev moments, second central moment of the pricipal components obtained from the covariance matrix [252], second central moment of second central moments in a window [179], and second central moment from the difference between the image and a blurred counterpart [97], [108], [208], [227].

- **Frequency-based** measures transform the image to the frequency domain, which is usually used in image compression. These transformations are Fourier, wavelet, or curvelet transforms. Afterwards, the coefficients of the base functions are summed [107], [260], [264] or the statistical measures are applied on the coefficients [181]. The idea behind frequency-based focus measure is that the need of many base functions (or non zero coefficients) to describe an image is a measure of complexity or structure in the image. This amount of structure or complexity is the measure of how focused the image is.

- **Texture-based** measures use recurrence of intensity values [101], [208], [227], computed locally binarie patterns [144], or the distance of orthogonally computed gradients in a window [108]. The idea here is equivalent to the frequency based approaches, meaning that the amount of texture present (complexity of the image) is the measure of how focused the image is.

For 3D reconstructions, common methods are:

- **Gaussian and polynomial fit**: These techniques fit a Gaussian [168] or polynomial [223] to the set of focus measures. To accomplish this, samples are collected outgoing from the maximum response of a pixel in the set of measurements (for each image, one measurement) in both directions. The maximum of the resulting Gaussian or polynomial is then used as depth estimate.

- **Surface fitting**: Here the samples for the fitting procedure are volumes around a pixel of focus measures. The surface is fitted to those samples, and the value aligned (in direction of the set of measurements) to the pixel is used as new value. The improvement to the Gaussian or polynomial fit is that the neighborhood of a pixel influences its depth estimation also. This approach together with a neural network for final optimization has been proposed by [8].

- **Dynamic programming**: In this technique the volume is divided in sub volumes. For each sub volume an optimal focus measure based on the result of a least squares optimization technique is computed. These results are combined and used as depth estimation [3], [4], [158], [228].

- **Surface fitting**: Here the samples for the fitting procedure are volumes around a pixel of focus measures. The surface is fitted to those samples, and the value aligned (in direction of the set of measurements) to the pixel is used as new value. The

improvement to the Gaussian or polynomial fit is that the neighborhood of a pixel influences its depth estimation too. This approach together with a neural network for final optimization has been proposed by [8].

7.2 Hardware setup

As shown in Figure 7.1, our setup consists of a Dikablis Professional eye tracker [51], a desktop computer visualizing the image from the camera, and an optotune lens. The eye-tracker can however be easily replaced other devices. The optotune lens has a focal tuning range of 50mm to 120mm [172]. It can be adjusted online over the lens driver, since the reaction time of the lens is 2.5ms [172]. The XIMEA mq013mge2 digital camera shown in Figure 7.1 was used with a frame rate of 60 Hz and resolution 1280x1024. For estimating

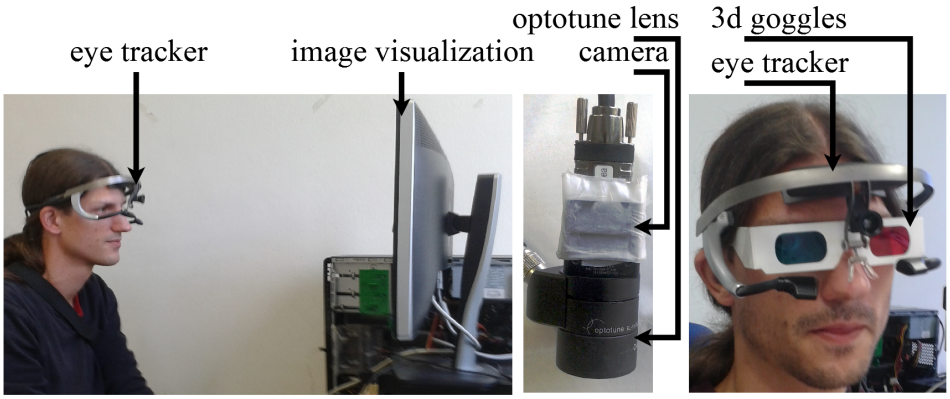


Figure 7.1: The system for image recording consisting of a digital camera (XIMEA mq013mge2) and an optotune lens (el1030). On the left side the subject with eye tracker looking at the image visualization is shown. The same subject with 3D googles is shown on the right [82].

the subjects gaze, the EyeRec [205] and the pupil center detection algorithm ElSe [79], which was presented previously in Section 3.3.2, were employed. The calibration was performed using a nine point grid with a second order polynomial fit in a least squares sense after data collection. Figure 7.2 shows the GUI of the system. The subjects gaze is mapped to the central screen. This is done by detecting the four surrounding markers and computing a transformation. This compensates for head movements of the subject. In the top row, the two images of both cameras are shown. For each, a depth map is created and visualized on the right side in Figure 7.2. Based on a slight displacement of both cameras, it is possible to use the red cyan technique, to achieve a 3D impression for the user. The focal length of both cameras is automatically set to the depth at the users gaze position in real time.

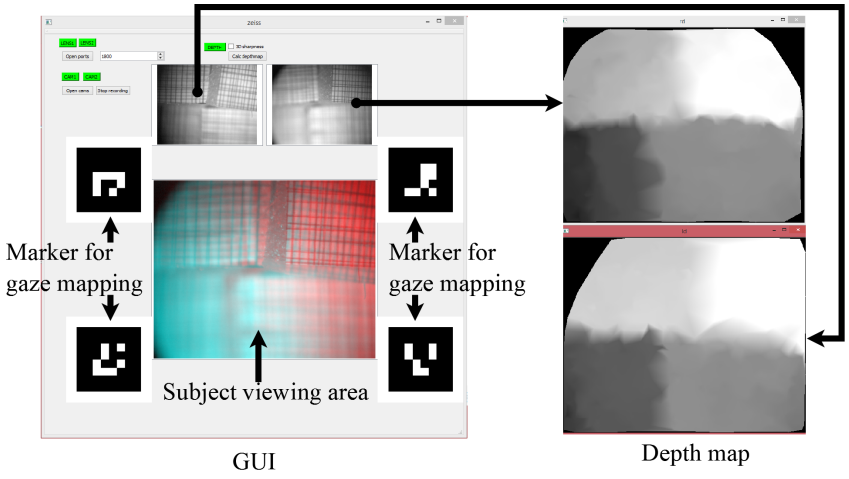


Figure 7.2: The GUI of our system. In the top row, the images from the two cameras with optotune lenses are shown. The depth map for those are on the right. The correspondence is marked by an arrow. Markers on the left side are used to map the gaze coordinates from the head mounted eye tracker to the subject's view area. For a 3D representation to the user the images from both cameras were overlapped in red and cyan, which can be seen in the subject's viewing area [82].

7.3 Method

All steps of the algorithm are shown in Figure 7.3. The input to the algorithm is a set of gray scale images recorded with different focal length. The images have to be in the correct order, otherwise the depth estimation will assign wrong depth values to focused parts of the volume. The main idea behind the algorithm is to estimate the depth map only based on

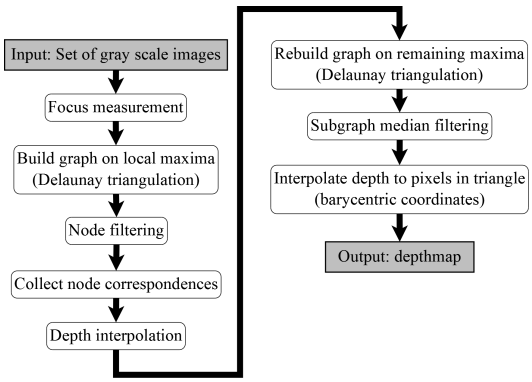
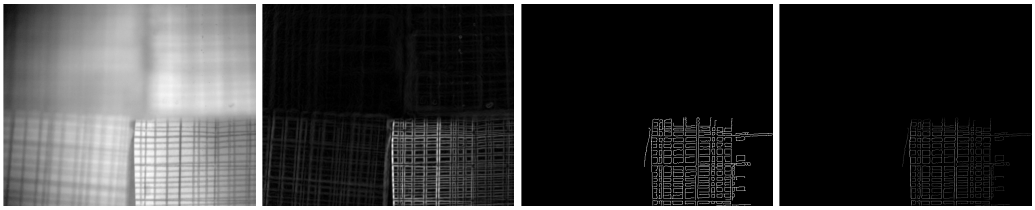


Figure 7.3: The algorithmic workflow. The gray boxes are in and output of the algorithm. White boxes with rounded corners are algorithmic steps. [82]

parts of the image, in which the focus is measurable, and interpolate it to the surrounding pixels if possible. Regions, where the focus is measurable are clear edges or texture in an image. Plain regions, for example, usually induce erroneous depth estimations, which have to be filtered out afterwards, typically using a median filter in classical shape-from-focus



(a) Input **(b)** Magnitude **(c)** Edges **(d)** Filtered magnitude

Figure 7.4: Canny edge based in focus estimation for one input image 7.4a. In 7.4b and 7.4c the output of the canny edge filter is shown and the filtered magnitude image in 7.4d. [82]

methods. Therefore, the Canny edge detector [32] is used as focus measure. The applied filter is the first derivative of a Gaussian. The resulting edges are used to filter the magnitude response of the filter, allowing only values with assigned edges to pass. For each filtered pixel magnitude, a maximum map through the set of responses is collected. In this map most of the pixels have no value assigned. Additionally, it has to be noticed that the same edge can be present in this map multiple times because the changing focal length influences the field of view of the camera. This leads to tracing edges in the maximum map.

After computing and filtering the focus measures of the image set, they have to be separated in parts. Therefore, candidates representing a strong edge part and their corresponding edge trace are needed to interpolate the depth estimation for the candidate pixel. The candidate selection is performed by only selecting local maxima using a eight connected neighborhood in the maximum map. These local maxima are used to build a graph representing the affiliation between candidates. This graph is build using the Delaunay triangulation, connecting candidates without intersections.

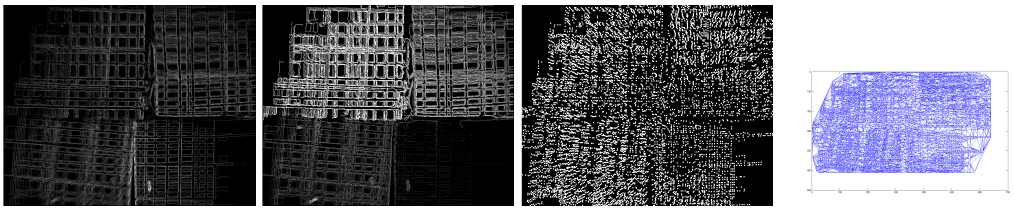
The separation of this graph into maximum response and edge trace responses is performed by separating nodes that are maximal in their neighborhood from those which are not. For interpolation of the depth value of maximal nodes, non maxima nodes are assigned based on their interconnection to the maxima and to an additional non maxima node. Additionally. the set of responses is searched for values at the same location. since the influence of the field of view does not affect all values in the image, and, as a result, centered edges stay at the same location. The interpolation is performed fitting a Gaussian (as in [168]) to all possible triple assignments, and using the median of all results.

The graph spanned by the maxima nodes and the corresponding interpolated depth values is now the representation of the depth map. For further error correction, an interdependent median filter is applied to each node and its direct neighbors in a non-iterative way to ensure convergence. The last part of the algorithm is the conversion of this graph into a proper depth map. It has to be noticed that this graph is a set of triangles spanned between maximal nodes. Therefore, each pixel in the resulting depth map can be interpolated using its barycentric coordinates between the three assigned node values of the triangle it is assigned to. Pixels not belonging to a triangle have no assigned depth value. All steps are described in the following subsections with more detail.

7.3.1 Focus measurement

Figure 7.4 shows the first step of the algorithm for one input image 7.4(a). The canny edge filter [32] is used with the first derivative of a Gaussian as kernel ($N'(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \frac{\partial}{\partial x} \frac{\partial}{\partial y}$). The response (magnitude) of the convolution with this kernel is visualized in the Figure 7.4(b). For σ, which is the standard deviation of the Gaussian. After adaptive threshold selection (95% are not edges) and non maximum suppression of the canny edge filter, the resulting edges (Figure 7.4(c)) are used as filter mask. In other words, only magnitude values assigned to a valid edge are allowed to pass. The stored magnitude responses for the input image are shown in Figure 7.4(d). The idea behind this step is to restrict the amount of information gathered per image, consequently reducing the impact of noise on the algorithm. These two parameters (σ and non-edge ratio) are the only variables of the method.

7.3.2 Graph representation



(a) Magnitude of maximum values **(b)** Depth of maximum values **(c)** Local maxima **(d)** Graph representation

Figure 7.5: Maximum responses in the set of images. In (a) the maximum magnitude for each location collected in the image is shown (black means no measurement collected) and the corresponding depth values in (b). (c) shows the local maxima (pixels increased for visualization) of the maximum magnitude map 7.5a on which a Delaunay triangulation is applied resulting in a graph representation (d) [82].

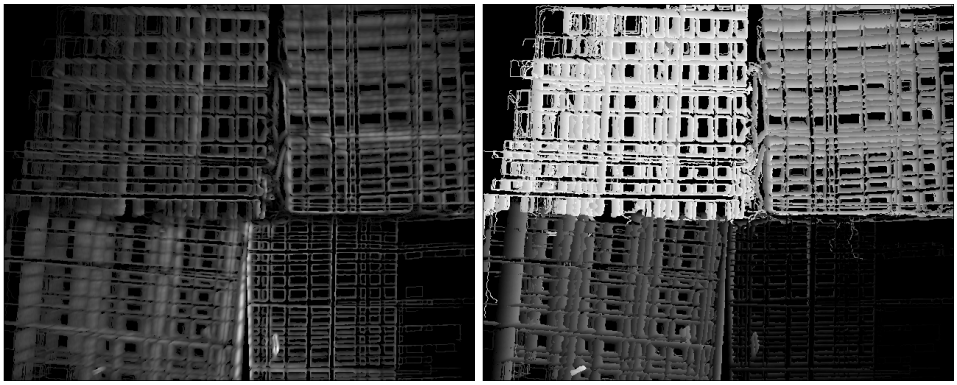
After in each image the focus measure was applied and filtered, the maximum along z of each location is collected in a maximum map (Figure 7.5(a), Equation 7.1).

$$M(x,y) = max_z(V(x,y,z)) \tag{7.1}$$

$$D(x,y) = \begin{cases} z & \text{M(x,y)} \in \text{V(x,y,z)} \\ 0 & \text{M(x,y)=0} \end{cases} \tag{7.2}$$

Equation 7.1 calculates the maximum map M (Figure 7.5(a)), where V represents the volume of filtered focus measures (one for each image). The coordinates x,y correspond to the image pixel location, and z is the image index in the input image. Equation 7.2 is the corresponding depth or z-index map D (Figure 7.5(b)), where an image set position is assigned to each maximum value.
In Figure 7.5(a) and its corresponding depth map 7.5(b), it can be seen thata depth estimation

is not optained for each pixel. Additionally, most of the collected edges have traces, meaning that the edge was collected in images recorded with different focal length. The trace occurs because changes in focal length induce a scaling of the field of view of the camera. In



(a) Magnitude trace **(b)** Depth trace

Figure 7.6: Maximum magnitude responses (7.6a) and the assigned depth index (7.6b) in the set of images. In comparison to Figure 7.5 were only 19 images in the input set where used, here the set consist of 190 images to show the traces more clear [82].

Figure 7.6, these traces and their occurrences are shown more clear due to the increased amount of the input image set (190 images). For Figure 7.5, 19 images in the input set are used. The bottom right part of Figure 7.6(a) shows that the occurrence of those traces is not present as strongly as in the other parts. This is due to the lens center (in the setup bottom right) from which the field of view scale impact increases linearly in distance.

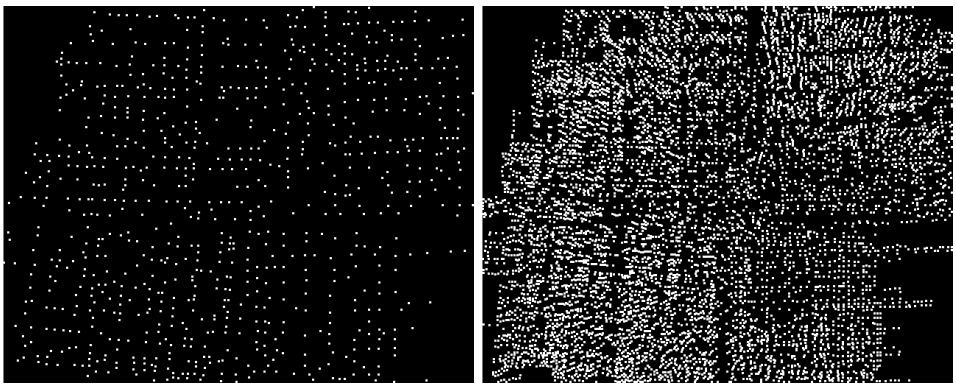
The next step of the algorithm is the computation of local maxima (Figure 7.5(c)) and, based on those, setting up a graph by applying the Delaunay triangulation (Figure 7.5(d)). The idea behind this step is to abstract the depth measurements, making it possible to estimate the depth of plain surfaces (as long as their borders are present) without the need of specifying a window size. Additionally, this graph is used to assign a set of depth estimations to one edge by identifying connected traces. These values are important because the set of input images does not have to contain the optimal focus distance of an edge. Therefore, the set of depth values belonging to one edge are used to interpolate its depth value.

The local maxima (Figure 7.5) are computed based on a eight connected neighborhood on the maximum magnitude map (Figure 7.5(a)). Based on those points, the Delaunay triangulation (Figure 7.5(d)) sets up a graph, where each triple of points creates a triangle if the circumcircle does not contain another point. This graph G_{all} (Figure 7.5) contains multiple maxima from the same edge on different depth plains. To separate those, two types of nodes: a maximal response set G_{max} (Figure 7.7(a)) and a non maximal response set G_{nonmax} (Figure 7.7(b)) are used.

$$G_{max} = \forall i \in G_{all}, \forall j \in CN(G_{all}, i), \\ V(j) \leq V(i) \tag{7.3}$$

$$G_{nonmax} = \forall i \in G_{all}, i \notin G_{max} \tag{7.4}$$

Equation 7.3 is used to build the maximal response set G_{max} (Figure 7.7a), where i is a Node in G_{all} and $CN(G_{all}, i)$ delivers all connected neighbors of i. Therefore, only nodes with an equal or higher magnitude value compared to their connected neighbors belong to G_{max}. G_{nonmax} (Figure 7.7b) consists of all nodes in G_{all}, which are not in G_{max} and specified in Equation 7.4.



(a) Nodes G_{max} **(b)** Nodes G_{nonmax}

Figure 7.7: White dots represent node locations (pixels increased for visualization). In 7.7a the nodes which have an equal or larger magnitude value compared to their connected neighbors in G_{all} are shown. 7.7b show the remaining non maximal nodes of G_{all} after removing those in G_{max} [82].

7.3.3 Node correspondence collection

Algorithm 6 performs the candidate selection. Candidates are possible node correspondences and marked by $CAN(a)$, where a is the index node for the assignment. For each node in G_{max}, connected nodes in G_{nonmax} with a different depth value are collected. The connected nodes to the node from G_{nonmax} has to be collected too, because it is necessary to collect all nodes that could possibly build a line over a trace and the maximum could be the last or first measurement.In case the node from G_{max} is close to the lens center, where the scaling has low to no impact, the volume of responses (V) has to be searched as well. After all candidates are collected, each pair has to be inspected to be a possible line trace or, in other words, a valid pair of corresponding focus measures.

$$\begin{aligned} COR(a) = \forall b, c \in CAN(a), \\ \begin{cases} b \neq c \\ D(a) \neq D(b) \neq D(c) \\ \vec{ab} \measuredangle \vec{ac} = \pi \end{cases} \end{aligned} \tag{7.5}$$

Equation 7.5 describes the correspondences collection based on the collected candidates ($CAN(a)$) belonging to node a. The equations after the large bracket are the conditions,

where $D(a)$ is the depth index of node a and $\vec{ab} \measuredangle \vec{ac}$ is the angle between the two vectors $\vec{ab}$ and $\vec{ac}$.

Algorithm 6 Algorithm for candidate selection where $CAN(a)$ are all candidates for node a, $CN(a)$ are the connected neighbors to node a, V the set of focus measure responses for each input frame, z the frame index, $D(a)$ the depth index of node a, G_{all} all local maxima, G_{max} all maximal nodes and G_{nonmax} all not maximal nodes [82].

Require: $G_{all}, G_{max}, G_{nonmax}, V$
  function $Select_{correspondences}(G_{all}, G_{max}, G_{nonmax})$
    for $a \in G_{max}$ **do**
      for $b \in CN(G_{all}, a)$, $b \in G_{nonmax}$ **do**
        if $D(a) \neq D(b)$ **then**
          $add(CAN(a), b)$
        end if
        for $c \in CN(G_{all}, b)$ AND $c \in G_{nonmax}$ **do**
          if $D(a) \neq D(c)$ **then**
            $add(CAN(a), c)$
          end if
        end for
      end for
      for $z \in V(a)$, $V(a,z) > 0$ **do**
        if $D(a) \neq z$ **then**
          $add(CAN(a), z)$
        end if
      end for
    end for
    return CAN
  end function

7.3.4 Interpolation

For estimating the real depth of a node in G_{max}, the three point Gaussian fit technique proposed by Willert and Gharib [254] is used. The assumed Gaussian is $M = M_{peak} e^{-0.5 \frac{D-\bar{D}}{\sigma}}$, where M is the focus measure response, D the depth, and σ the standard deviation of the Gaussian. This can be rewritten with the natural logarithm $ln(M) = ln(M_{peak}) - 0.5 \frac{D-\bar{D}}{\sigma}$. $\bar{D}$ is the depth value, where the Gaussian has the highest focus measure (mean) and obtained using Equation 7.6.

$$\begin{aligned} M^{+}(a,b) &= ln(M(a)) - ln(M(b)) \\ M^{-}(a,b,c) &= M^{+}(a,b) + M^{+}(a,c) \\ D^{2-}(a,b) &= D(a)^2 - D(b)^2 \\ \Delta D(a,c) &= 2|D(a) - D(c)| \\ \bar{D}(a,b,c) &= \frac{M^{+}(a,c) * D^{2-}(a,b)}{\Delta D(a,c) * M^{-}(a,b,c)} \end{aligned} \tag{7.6}$$

In Equation 7.6, a, b, c are node triples obtained from $COR(a)$, where M is the focus measure, and D is the depth value (the same letters as in Equation 7.1 and 7.2 are used for simplification and it has to be noted that it is not valid for nonmembers of G_{all}, which are obtained through the response volume (6)).
Since $COR(a)$ can have more than one pair of possible interpolations, the median over all possible interpolation values ($D(a) = Median(\{\bar{D}(a,b,c)\}), \forall b,c \in COR(a)$) is used.

7.3.5 Rebuild graph

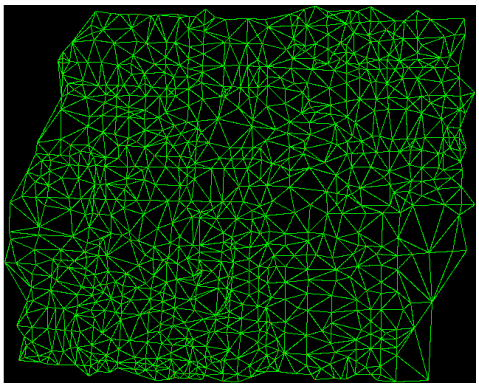
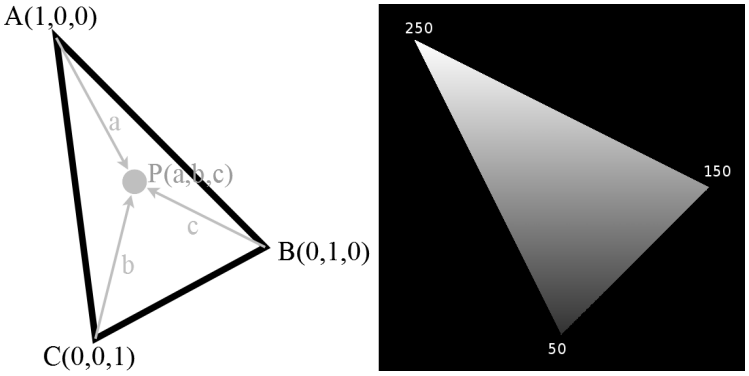


Figure 7.8: The graph build on G_{max} using Delaunay triangulation [82].

For using those interpolated nodes in G_{max} as image representation, the graph has to be rebuild. Again the Delaunay triangulation is used with the result shown in Figure 7.8. Due to possible errors from the interpolation or the focus measurement, a median filter is applied on the depth of the node and its neighborhood ($D(a) = Median(\{D(CN(a)), D(a)\})$). This median interpolation is performed interdependently; in other words, the values are stored directly into the depth map D, influencing thus the median filtering of its neighbors. This way of median filtering is used because it delivered slightly better results. A more time consuming approach would be to determine the median iteratively. However, such an iterative approach could lead to oscillation and therefore to non convergence.

7.3.6 Depth map creation

For depth map creation, the graph in Figure 7.8 has to be transformed into a surface. This is done by assigning each pixel in a triangle the weighted value of the depth estimations from the corner nodes. The weights are determined using the distance to each node. The idea behind this is to have linear transitions between regions with different depth values. This makes it more comfortable for the subject to slide over the scene with their gaze, without having an oscillatory effect of the focal length close to region borders. This can be achieved very fast using barycentric coordinates (Figure 7.9a) to linearly interpolate (Figure 7.9(b)) those three values, which is usually applied in computer graphics to 3D



(a) Barycentric **(b)** Interpolation

Figure 7.9: In 7.9a the barycentric coordinates of a triangle spanned by nodes A,B and C is shown. The gray dot P in the middle of this triangle has coordinates a,b and c which is related to its distance to A, B and C. 7.9b shows an exemplary interpolation in such a triangle, where the numbers next to each corner are the intesity value of the corner pixel [82].

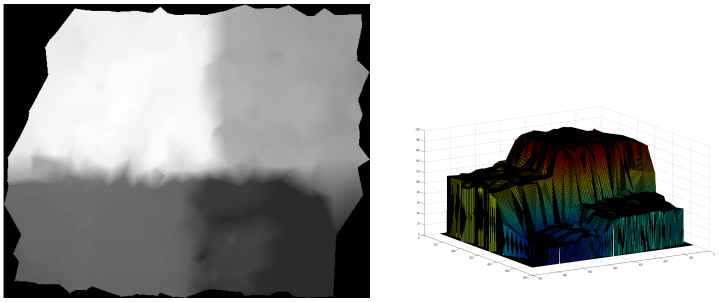
models. See section 2.6.2 for a detailed description of barycentric coordinates.

$$\begin{gathered} P(a,b,c) \\ a = \frac{\Delta PBC}{\Delta ABC}, b = \frac{\Delta PAC}{\Delta ABC}, c = \frac{\Delta PAB}{\Delta ABC} \end{gathered} \tag{7.7}$$

Equation 7.7 describes the transformation from Cartesian coordinates to barycentric coordinates, where A,B and C are the corner nodes of a triangle (Figure 7.9a), Δ is the area of the spanned triangle and a,b and c are the barycentric coordinates. An exemplary interpolated triangle can be seen in Figure 7.9(b).

$$D(P) = a * D(A) + b * D(B) + c * D(C) \tag{7.8}$$

For depth assignment to point P, Equation 7.8 is used, where D is the depth value. The resulting depthmap after linear interpolation of all triangles can be seen in Figure 7.10a. In this depth map, white is closer to the camera and dark is further away. Comparing Figure 7.10a to Figure 7.8 it can be seen that areas for which no enclosing triangle exists are threated as not measurable (black regions in Figure 7.10a). If an estimation for those regions is wanted, it is possible to assign those pixels the depth value of the closest pixel with depth information or to interpolate the depth value using barycentric coordinates of the enclosing polygon. The 3D reconstruction based on the depth map from Figure 7.10a can be seen in Figure 7.10b.



(a) Depth map **(b)** 3D model

Figure 7.10: In 7.10a (normalized) white is closer, dark gray is further away and black means that the depth measure could not estimate a depth value. The 3D model in 7.10b is generated with the depth map from 7.10a using matlab [82].

7.4 Evaluation

In this section, we present evaluation results of the proposed method on publicly available and our own datasets. The method is compared to the state-of-the-art and also capable of detecting areas were no depth estimation can be made.

7.4.1 Data sets

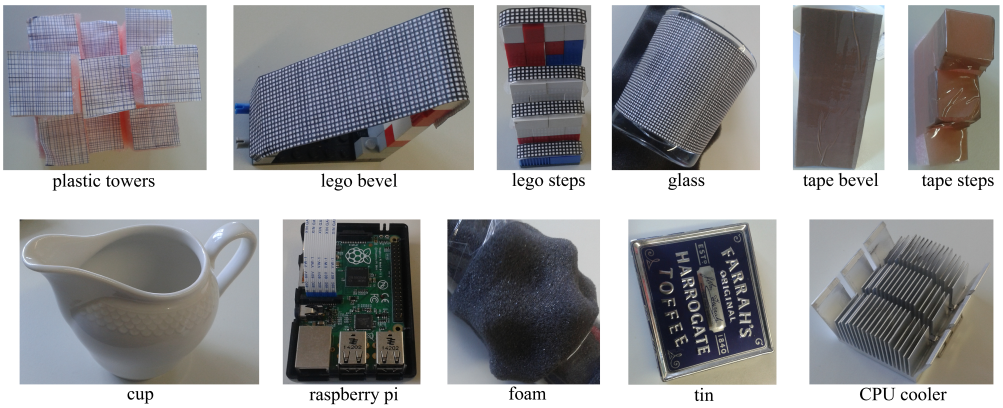


Figure 7.11: Shows all objects used to generate the data sets. Below each object image stands the title which will be used further in this document [82].

Each object in Figure 7.11 was scanned with 191 steps over the complete range (focal tuning range of 50mm to 120mm [172]) [82]. The objects plastic towers, lego bevel, lego steps and glass are coated with a grid of black lines which should simplify the depth estimation. For the objects tape bevel and tape steps package tape is used to reduce the focus information. Objects cup, raspberry pi, foam, tin and CPU cooler are real objects where tin and raspberry pi are scanned in an oblique position. All objects except for the cpu cooler are used for evaluation, whereas the said object is used in limitations. For evaluation, zeromotion from

Suwajanakorn et al. [228] and balcony, alley and shelf from Möller et al. [158] were used additionally.

7.4.2 Results

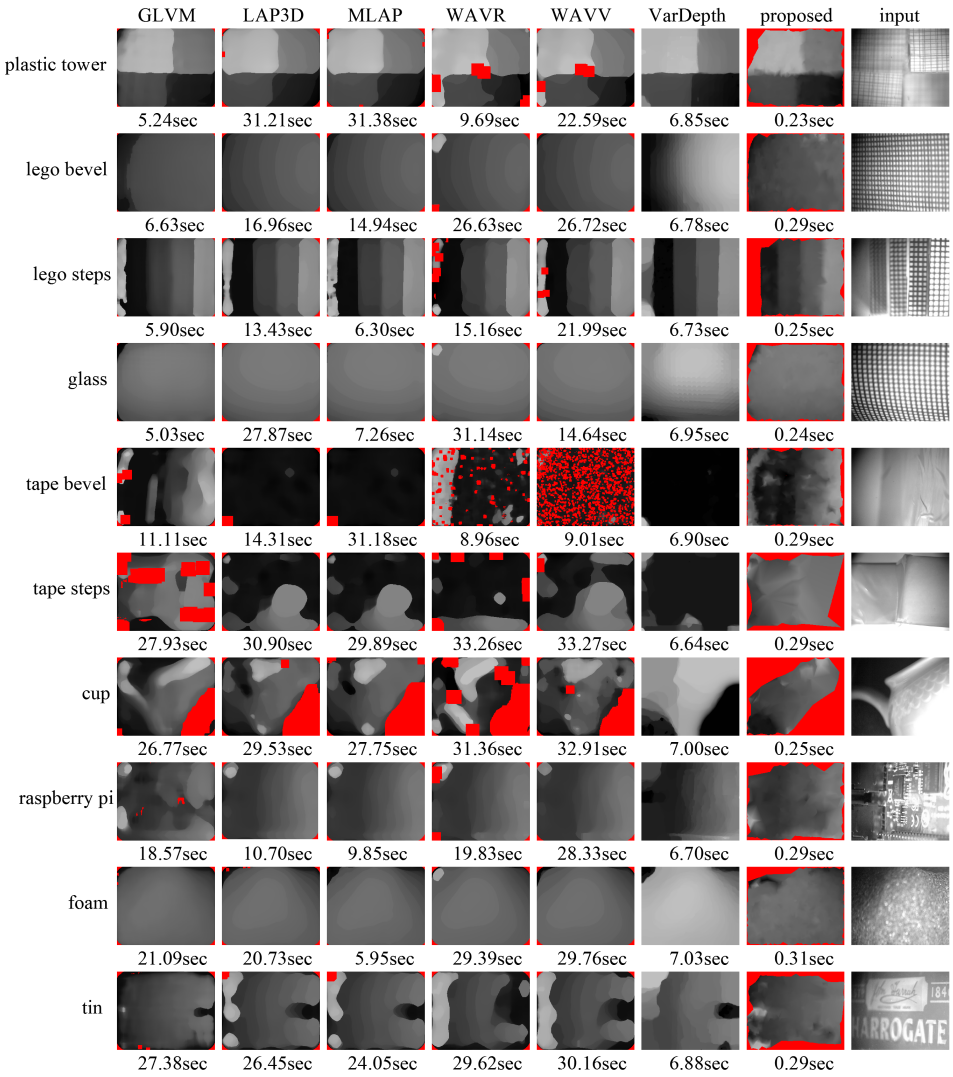


Figure 7.12: The results on all data sets from [82] are shown. On the left side the data set is named and in the top row the algorithm is named, modified gray level variance (GLVM), Laplacian in 3D window [5] (LAP3D), modified Laplacian [167] (MLAP), ratio of wavelet coefficients [260] (WAVR), variance of wavelets coefficients [264] (WAVV), variational depth (VarDepth) [158] and the algorithm. Below each depth map the processing time of the inoput image set and the depth estimation is shown. Red regions are marked by the algorithm to be not measurable. Brighter means closer to the camera. Red regions are marked by the algorithm to be not measurable. Brighter means further away from the camera [82].

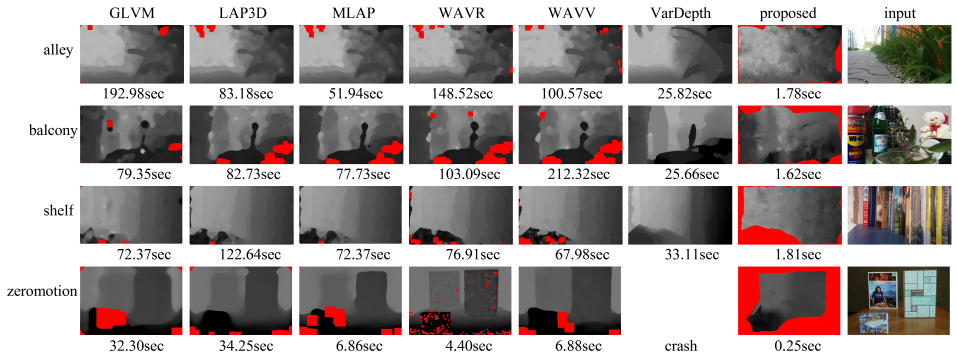


Figure 7.13: The results on the data sets alley [158], balcony [158], shelf [158] and zeromotion [228] are shown. The first three data sets have a resolution of 1920x1080 and the last one has 774x518. In the top row each algorithm is named as in Figure 7.12. Below each depth map, the complete processing time is shown. Red regions are marked by the algorithm to be not measurable. Brighter means further away from the camera [82].

For evaluation, 20 images per object are used from the set of 191. Those images were equally spread, meaning that the change in focal length between consecutive images is constant. The algorithm was than applied as previously described. For the evaluation no parameter from the algorithm in [82] was changed. The algorithm variational depth [158] is evaluated with the parameters as specified by the authors on a GeForce GT 740 GPU. The shape from focus (SFF) measures which are evaluate are modified gray level variance (GLVM), modified Laplacian [167] (MLAP), Laplacian in 3D window [5] (LAP3d), variance of wavelets coefficients [264] (WAVV) and ratio of wavelet coefficients [260] (WAVR) as implementation from Pertuz et al. [181], [182]. Those are chosen because they were reported to be the best performing ones [181]. For optimal parameter estimation, all focus measure filter sizes and median filter sizes are considered. For dpeth interpolation, the Gauss fitting was used.

7.4.3 Algorithm evaluation

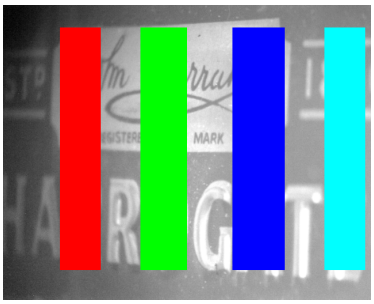
It is very difficult to make exact depth measurements for all objects and manually labeling those. Therefore, the depth map for each algorithm and the timings are shown. In Figure 7.12 the results from the method and the state-of-the-art is shown. The first four rows are the results of objects, where the surface is marked with a grid. The purpose of those is to have a comparison based on more easy objects. For plastic tower, the irregular transitions between the four regions are due to the triangle interpolation and a wanted result. The 3D map can be checked in Figure 7.10b. In the third row (lego steps), it can be seen that the method correctly detects the not measurable region. For tape bevel, GLVM performs best, but the method is closest to its result in comparison to the others. For the tape steps object the method outperforms the others in a fraction of runtime. The most difficult object in the data set is the cup, which contains a big reflection and only a small part is textured. The other methods evaluate the rim closer to the camera as the body of the cup which is not true. The method labels big parts as not measurable (red regions), which is correct. The valid

region is estimated appropriate with a small error (white spot). Raspberry pi is estimated correctly by all methods except GLVM. For foam, all methods perform well. The last object of the data set [82] is tin. The two bright spots on the left side of the result of the method are due to dust particles on the lens which get sharp on a closer focal length. The best performing algorithm for tin is GLVM.
In Figure 7.13 the results on the data sets provided by Suwajanakorn et al. [228] and Moeller et al. [158] are shown. In comparison to the other algorithms the results are not as smooth. For alley, all algorithms except the one from [82] smooth out the left house and variational depth, which was proposed with the data set [82], has two invalid regions at the top left. For the data set balcony [158] the algorithm in [82] did not get the centered leaf correctly but the depth approximation of the remaining area is comparable to the state of the art. For the shelf [158] and zeromotion [228] data sets the algorithm [82] performs better because it detects the non measurable regions. For the algorithm variational depth [158], it was not possible to provide a depth map for zeromotion from [228] because the algorithm crashes.

7.4.4 Best index evaluation

For evaluation against the best index in the image stack, a region and an index was selected. For the depth map reconstruction only 19 equally spaced images are used and evaluated against the 191 recorded indexes. For evaluation, the image stacks from "lego steps", "plastic tower", "tape steps" and "tin" (see Figure 7.12) are used. The Figures 7.14 7.15 7.16 7.17 show the mean absolute error ($\frac{1}{n}|v_i - v_{gt}|$). Over each table, the specific data set with the evaluated regions marked by different colors is shown. Regions without depth estimation (marked red in Figure 7.12) are excluded in the calculation of the mean absolute error. As can be seen, the method performed similar to the state-of-the-art with less computational time.

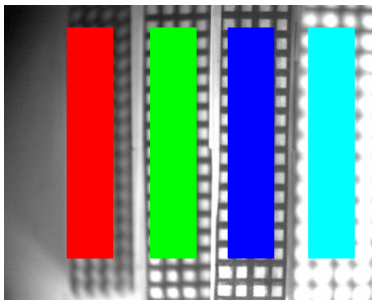
Figure 7.14: Shows the results for the data set tin. The values are the mean absolute errors over the marked regions. The regions are named R1-4 and visualized as red is R1, green is R2, blue is R3 and cyan is R4 [82].



Method	R1	R2	R3	R4
GLVM	3.40	3.88	6.76	17.33
LAPM	6.06	3.80	11.48	21.92
LAP3	12.01	3.87	13.47	22.02
WAVV	19.10	4.33	19.87	30.87
WAVR	27.15	4.59	32.53	57.54
VARDEPTH	28.25	18.14	10.29	35.22
[82]	12.96	4.07	6.96	5.61

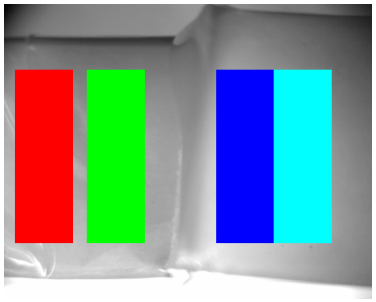
We applied eye tracking for automatically adapting focus to the presented image. Due to the real-time capability, our methods can be beneficial in various applications where autofocus facilitates interaction (e.g. surgery, security, human-robot collaboration, etc.). The algorithm proposed here shows similar performance as the state of the art but requires minimal computational resources and requires no parameter adjustment.

Figure 7.15: Shows the results for the data set lego steps. The values are the mean absolute errors over the marked regions. The regions are named R1-4 and visualized as red is R1, green is R2, blue is R3 and cyan is R4 [82].



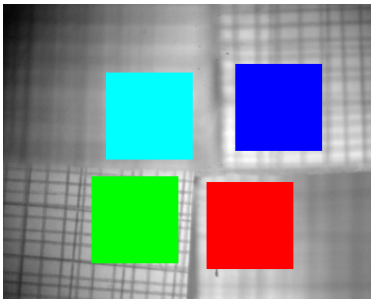
Method	R1	R2	R3	R4
GLVM	8.26	4.65	3.78	14.86
LAPM	12.85	5.28	3.75	11.68
LAP3	12.77	5.27	3.61	11.80
WAVV	12.98	5.33	3.57	11.77
WAVR	14.13	4.92	3.09	11.76
VARDEPTH	19.61	0.72	2.68	12.98
[82]	3.67	4.14	5.28	12.90

Figure 7.16: Shows the results for the data set tape steps. The values are the mean absolute errors over the marked regions. The regions are named R1-4 and visualized as red is R1, green is R2, blue is R3 and cyan is R4 [82].



Method	R1	R2	R3	R4
GLVM	21.56	14.05	25.83	2.18
LAPM	45.86	62.75	39.90	63.64
LAP3	45.86	62.75	39.90	63.64
WAVV	34.84	59.93	24.33	43.74
WAVR	56.53	65.62	95.51	112.41
VARDEPTH	52.89	61.42	104.43	109.43
[82]	12.39	19.14	6.27	15.93

Figure 7.17: Shows the results for the data set plastic tower. The values are the mean absolute errors over the marked regions. The regions are named R1-4 and visualized as red is R1, green is R2, blue is R3 and cyan is R4 [82].



Method	R1	R2	R3	R4
GLVM	6.26	2.08	2.37	8.06
LAPM	4.70	2.29	3.20	10.82
LAP3	4.74	2.20	3.22	10.73
WAVV	9.74	2.96	3.75	13.35
WAVR	7.57	3.79	3.95	13.31
VARDEPTH	8.35	2.06	4.24	12.79
[82]	3.27	4.02	2.57	2.70

7.5 Conclusion

In this chapter a real time system for automatic focus adjustment based on the gaze information was presented and compared to the state-of-the-art. In the following we will summarize this thesis and all presented approaches.

8 Conclusion

Video-based eye tracking has the advantages that it is not intrusive and can be applied in a variety of scenarios. The main challenge associated with this technology is however to cope with the different environmental settings, lighting conditions, individual properties of users, different camera perspectives and, additional equipment like eyeglasses or contact lenses which lead to a non robust eye-tracking signal. In this thesis, four novel pupil detection algorithms are presented. Among these methods, ExCuSe [66] and ElSe [73], [79], [86] are both decision-based and are build on the assumption that the pupil is dark or an edge of the pupil corner can be extracted. The main problem with this is that edges are not always present and heavily dependent on the image quality. However, both proved robustness for a variety of use-cases, run in real time are capable of handling high frame rates. Although initially designed for head-mounted eye-trackers, ElSe is also applicable to remote scenarios as shown throughout this thesis. In addition, together with both algorithms, a large data set of manually annotated images was published which opened the way for a machine learning approaches. The third algorithm for pupil detection is PupilNet [83], which is based on Convolution Neural Networks. This machine learning approach requires large amounts of data and training time but is capable of learning robust features together with a classifier. This was done to handle problems which exceed the possibilities of the above decision-based approaches. PupilNet runs also in real time and was published together with an additional data set for further research. The fourth pupil detection algorithm was specifically designed for surgical microscopes [85]. As the data recorded through an ocular is different to the head mounted scenario, the pupil surrounding is jagged, which renders the ellipse fitting to edges useless. The edge extraction itself is also impeded by the small opening area of the ocular, which only allows a part of the pupil to be captured on an image. In addition, the sharp focus distance of the camera is limited to a tiny zone, making most of the images blurred.

The second important feature in eye-tracking images of a subject are the eyelids. While little research has been done in this area, the importance of this feature rises. Through novel upcoming use cases, such as autonomous driving, extracting eyelids is not only important for eye-tracking signal validation, but might also be helpful to estimate the cognitive state of the subject. Therefore, this thesis introduced two novel algorithms for eyelid extraction. All of the before mentioned algorithms deal with the problem of accurate point detection. The main limiting factor for further progress is the lack of available annotated data. Therefore, Multiple Annotation Marturation (MAM) was developed, which automatically annotates accurately identical point locations even under deformations. As input MAM needs only a small subset of samples or an initial detector. As shown in the evaluation section, it outperforms all the other algorithms by only giving MAM ten examples of the data. It is based on HOG features together with an SVM classifier and capable of annotating

large videos without a human intervention. This is achieved by clustering the video into age groups based on their detection state. This algorithm, originally from a collaboration with an industrial partner, was published together with a remote data set in an car environment. MAM does not only annotate the data set, it also outputs specialized detectors for a data set which could be used afterwards. Therefore, it can be considered as a summary algorithm of the already mentioned methods.
The above mentioned algorithms have already found applications in various settings. In addition to the application fields mentioned in this thesis, the methods were applied to publicly available tools which were also developed at the university Tübingen. The first tool to be mentioned is EyeLad [80], which is a supportive annotation tool for eye regions in head mounted and remote scenarios. It is possible to use the before mentioned algorithms for pupil and eye lid detection and in addition to track each point. It visualizes regions separately and annotations can be tracked or copied to new frames.
Another application scenario, where ElSe was applied for pupil center detection, is the focus adjustment of a camera based on the gaze location. Therefore, a software was developed which records the gaze of a person and maps it to the camera scene [82]. The mapping was performed by detecting markers surrounding the presented camera image. To map the gaze position of the user to the camera image on the monitor, a projection had to be computed based on four markers surrounding this image (mapping between the eye tracker field camera image location and the monitor coordinate system). This allows the user to move her head freely. Afterwards, the focus was set for the point the user looked at on the presented image. The depth map was computed based on a novel algorithm which runs in real time and was published together with a new data set to encourage further research in this topic.
Another important aspect of gaze is its information content. To get a better insight into humans gaze behavior and what it reveals about them, several novel visualizations were developed, which support researchers in extracting information and finding correlations. In a cooperation with the University Wien, the software EyeTrace was additionally developed which integrates most of the algorithms presented in this thesis. It hold a variety of algorithms for fixation extraction and visualizations. Modern visualization techniques are however based on regions of interest (ROI) to reduce the noise in eye tracking data. Therefore, it is necessary to manually annotate those regions. In this thesis, three novel algorithms were proposed to extract those regions automatically [76]–[78].
In summery, all methods described in this thesis are applicable to real world scenarios and run in real time. While every algorithm can be further improved, the presented work surpassed the state-of-the-art at the time it was published. All algorithms were released together with annotated data to fasten the progress of future research and the code was made publicly available for reproduce ability and usage. Future advances will come through the field of machine learning by reducing the computational costs which is already done by binary neural networks or cascaded conditional distributions (Random Ferns [71], [72]). In addition to the selective feature extraction and polynomial mapping, the appearance-based approach will increase in popularity by its simple applicability. Simulation for data generation and the simulated data enhancement using techniques like [216] will further speed up this progress.

Bibliography

[1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection", in *Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1597–1604. DOI: 10.1109/CVPR.2009.5206596.

[2] M. Adam, F. Rossant, F. Amiel, B. Mikovikova, and T. Ea, "Eyelid localization for iris identification", *Radioengineering*, vol. 17, no. 4, pp. 82–85, 2008.

[3] M. B. Ahmad and T.-S. Choi, "A heuristic approach for finding best focused shape", *Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 4, pp. 566–574, 2005.

[4] M. B. Ahmad and T.-S. Choi, "Shape from focus using optimization technique", in *International Conference on Acoustics Speech and Signal Processing Proceedings*, IEEE, vol. 2, 2006, pp. II–II.

[5] Y. An, G. Kang, I.-J. Kim, H.-S. Chung, and J. Park, "Shape from focus through Laplacian using 3D window", in *Second International Conference on Future Generation Communication and Networking*, IEEE, vol. 2, 2008, pp. 46–50.

[6] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.161.

[7] N. Arora, G. M. Allenby, and J. L. Ginter, "A hierarchical Bayes model of primary and secondary demand", *Marketing Science*, vol. 17, no. 1, pp. 29–44, 1998.

[8] M. Asif and T. S. Choi, "Shape from focus using multilayer feed forward neural networks", *Transactions on Image Processing*, vol. 10, no. 11, pp. 1670–1675, 2001, ISSN: 1057-7149. DOI: 10.1109/83.967395.

[9] T. J. Atherton and D. J. Kerbyson, "Size invariant circle detection", *Image and Vision Computing*, vol. 17, no. 11, pp. 795–803, 1999.

[10] B. Auyeung, M. V. Lombardo, M. Heinrichs, B. Chakrabarti, A Sule, J. B. Deakin, R. Bethlehem, L Dickens, N. Mooney, J. Sipple, *et al.*, "Oxytocin increases eye contact during a real-time, naturalistic social interaction in males with and without autism", *Translational psychiatry*, vol. 5, no. 2, e507, 2015.

[11] H. Bahmani, W. Fuhl, E. Gutierrez Mlot, E. Kasneci, and S. Wahl, "Feature-based attentional influences on the accommodation response", vol. 16, p. 680, Sep. 2016.

[12] B. Bakker and T. Heskes, "Task Clustering and Gating for Bayesian Multitask Learning", *Journal of Machine Learning Research*, vol. 4, no. May, pp. 83–99, 2003.

[13] S. Baluja and D. Pomerleau, "Non-intrusive gaze tracking using artificial neural networks", in *Advances in Neural Information Processing Systems*, 1994, pp. 753–760.

[14] J. Baxter, "A Model of Inductive Bias Learning", *J. Artif. Int. Res.*, vol. 12, no. 1, pp. 149–198, Mar. 2000, ISSN: 1076-9757. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622248.1622254.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)", *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[16] F. Bergholm, "Edge focusing", *Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 726–741, 1987.

[17] A. Bergua, *Das menschliche Auge in Zahlen*. Springer-Verlag, 2017.

[18] E Bezzel. and R. Prinzinger, "Sinnesorgane Auge", *Ornithologie. Stuttgart: Eugen Ulmer*, pp. 148–54, 1990.

[19] T. Blascheck, M. Raschke, and T. Ertl, "Circular heat map transition diagram", in *Proceedings of the Conference on Eye Tracking South Africa*, ACM, 2013, pp. 58–61.

[20] R. A. Boie and I. J. Cox, "An analysis of camera noise", *Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[21] C. F. Borges and T. Pastva, "Total least squares fitting of Bézier and B-spline curves to ordered data", *Computer Aided Geometric Design*, vol. 19, no. 4, pp. 275–289, 2002.

[22] A. Borji and L. Itti, "State-of-the-art in visual attention modeling", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 185–207, 2013.

[23] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*. McGraw-Hill New York, 1986, vol. 31999.

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

[25] C. Braunagel, W. Stolzmann, E. Kasneci, T. C. Kübler, W. Fuhl, and W. Rosenstiel, "Exploiting the potential of eye movements analysis in the driving context", in *Internationales Stuttgarter Symposium: Automobil- und Motorentechnik*, M. Bargende, H.-C. Reuss, and J. Wiedemann, Eds. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, pp. 1093–1105, ISBN: 978-3-658-08844-6. DOI: 10.1007/978-3-658-08844-6_74. [Online]. Available: https://doi.org/10.1007/978-3-658-08844-6_74.

[26] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer Science & Business Media, 2013, vol. 31.

[28] A. A. den Broeder, M. C. Creemers, J. Fransen, E. de Jong, D.-J. R. de Rooij, A. Wymenga, M. de Waal-Malefijt, and F. H. van den Hoogen, "Risk factors for surgical site infections and other complications in elective surgery in patients with rheumatoid arthritis with special attention for anti-tumor necrosis factor: a large retrospective study.", *The Journal of rheumatology*, vol. 34, no. 4, pp. 689–695, 2007.

[29] C. J. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[30] P. Cai and C. Wang, "An eyelid detection algorithm for the iris recognition", *International Journal of Security and its Applications*, vol. 9, no. 5, pp. 105–112, 2015.

[31] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection", in *European Conference on Computer Vision*, Springer, 2016, pp. 354–370.

[32] J. Canny, "A computational approach to edge detection", *Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[33] X. Cao, Z. Wang, P. Yan, and X. Li, "Transfer learning for pedestrian detection", *Neurocomputing*, vol. 100, pp. 51–57, 2013.

[34] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression", in *Computer Vision and Pattern Recognition*, 2012. [Online]. Available: https://www.microsoft.com/en-us/research/publication/face-alignment-by-explicit-shape-regression/.

[35] R. Caruana, "Multitask learning", in *Learning to Learn*, Springer, 1998, pp. 95–133.

[36] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]", *Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[37] L. Commare and H. Brinkmann, "Aesthetic Echoes in the Beholder's Eye? Empirical evidence for the divergence of theory and practice in the perception of abstract art.", in *M. Zimmermann (Ed.): Vision in Motion. Streams of Sensation and Configurations of Time*, Diaphanes, 2016.

[38] H. D. Crane and C. M. Steele, "Generation-V dual-Purkinje-image eyetracker", *Applied Optics*, vol. 24, no. 4, pp. 527–537, 1985.

[39] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 886–893.

[40] J. Daugman, "How iris recognition works", *TCSVT*, vol. 14, no. 1, pp. 21–30, 2004.

[41] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society*, pp. 1–38, 1977.

[42] D. Droege and D. Paulus, "Pupil center detection in low resolution images", in *Eye Tracking Research and Applications*, ACM, 2010, pp. 169–172.

[43] C. E. Duchon, "Lanczos filtering in one and two dimensions", *Journal of Applied Meteorology*, vol. 18, no. 8, pp. 1016–1022, 1979.

[44] A. T. Duchowski, *Eye Tracking Methodology: Theory and Practice*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007, ISBN: 1846286085. DOI: 10.1007/978-1-84628-609-4.

[45] D. Dussault and P. Hoess, "Noise performance comparison of ICCD with CCD and EMCCD cameras", in *Optical Science and Technology, the SPIE 49th Annual Meeting*, International Society for Optics and Photonics, 2004, pp. 195–204.

[46] H. Edelsbrunner and N. R. Shah, "Incremental topological flipping works for regular triangulations", *Algorithmica*, vol. 15, no. 3, pp. 223–241, 1996.

[47] D. L. Edmonston and G. D. Foulkes, "Infection rate and risk factor analysis in an orthopaedic ambulatory surgical center.", *Journal of Surgical Orthopaedic Advances*, vol. 19, no. 3, pp. 174–176, 2009.

[48] S. Eivazi, W. Fuhl, and E. Kasneci, "Towards Intelligent Surgical Microscope: Micro-surgeons' Gaze and Instrument Tracking", in *Proceedings of the 22Nd International Conference on Intelligent User Interfaces Companion*, ser. IUI '17 Companion, Limassol, Cyprus: ACM, 2017, pp. 69–72, ISBN: 978-1-4503-4893-5. DOI: 10.1145/3030024.3038269. [Online]. Available: http://doi.acm.org/10.1145/3030024.3038269.

[49] S. Eivazi, A. Hafez, W. Fuhl, H. Afkari, E. Kasneci, M. Lehecka, and R. Bednarik, "Optimal eye movement strategies: a comparison of neurosurgeons gaze patterns when using a surgical microscope", *Acta Neurochirurgica*, pp. 1–8, 2017. DOI: 10.1007/s00701-017-3185-1.

[50] S. Eivazi, M. Slupina, W. Fuhl, H. Afkari, A. Hafez, and E. Kasneci, "Towards Automatic Skill Evaluation in Microsurgery", in *Proceedings of the 22Nd International Conference on Intelligent User Interfaces Companion*, ser. IUI '17 Companion, Limassol, Cyprus: ACM, 2017, pp. 73–76, ISBN: 978-1-4503-4893-5. DOI: 10.1145/3030024.3040985. [Online]. Available: http://doi.acm.org/10.1145/3030024.3040985.

[51] Ergoneers, *Dikablis Glasses*, 2016.

[52] *Ergoneers*, http://www.ergoneers.com/, Accessed: 2019-01-01.

[53] T. Evgeniou and M. Pontil, "Regularized multi–task learning", in *Proceedings of the tenth ACM International Conference on Knowledge Discovery and Data Mining*, ACM, 2004, pp. 109–117.

[54] N. FarajiDavar, T. De Campos, J. Kittler, and F. Yan, "Transductive transfer learning for action recognition in tennis games", in *International Conference on Computer Vision Workshops (ICCV Workshops)*, IEEE, 2011, pp. 1548–1553.

[55] N. Farajidavar, T. E. de Campos, and J. Kittler, "Adaptive Transductive Transfer Machine.", in *British Machine Vision Conference*, 2014.

[56] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[57] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[58] R. A. Fisher, "The use of multiple measurements in taxonomic problems", *Annals of Human Genetics*, vol. 7, no. 2, pp. 179–188, 1936.

[59] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476–480, 1999.

[60] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[61] F. Fralick, "Anatomy and physiology of the eyelid.", *Transactions on American Academy of Ophthalmology and Otolaryngology*, vol. 66, p. 575, 1962.

[62] J. H. Friedman, "Stochastic gradient boosting", *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[63] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, 10. Springer series in statistics New York, NY, USA: 2001, vol. 1.

[64] S. Frintrop, M. Klodt, and E. Rome, "A real-time visual attention system using integral images", in *International Conference on Computer Vision Systems*, vol. 25, 2007.

[65] A. Fuchs, C. Kaneko, and C. Scudder, "Brainstem control of saccadic eye movements", *Annual Review of Neuroscience*, vol. 8, no. 1, pp. 307–337, 1985.

[66] W. Fuhl, T. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci, "ExCuSe: Robust Pupil Detection in Real-World Scenarios", in *16th International Conference on Computer Analysis of Images and Patterns*, G. Azzopardi and N. Petkov, Eds., Springer International Publishing, 2015, pp. 39–51, ISBN: 978-3-319-23192-1. DOI: 10.1007/978-3-319-23192-1_4. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23192-1_4.

[67] W. Fuhl, T. Santini, and E. Kasneci, "Fast and Robust Eyelid Outline and Aperture Detection in Real-World Scenarios", in *Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 1089–1097. DOI: 10.1109/WACV.2017.126.

[68] W. Fuhl, N. Castner, and E. Kasneci, "Histogram of oriented velocities for eye movement detection", in *Workshop on Modeling Cognitive Processes from Multimodal Data (MCPMD18)*, Sep. 2018.

[69] W. Fuhl, N. Castner, and E. Kasneci, “Rule based learning for eye movement type detection”, in *Workshop on Modeling Cognitive Processes from Multimodal Data (MCPMD18)*, Sep. 2018.

[70] W. Fuhl, N. Castner, L. Zhuang, M. Holzer, and E. Kasneci, “MAM: Transfer learning for fully automatic video annotation and specialized detector creation”, in *Egocentric Perception, Interaction and Computing Workshop (EPIC at ECCV)*, Sep. 2018.

[71] W. Fuhl, S. Eivazi, B. Hosp, A. Eivazi, W. Rosenstiel, and E. Kasneci, “BORE: Boosted-oriented edge optimization for robust, real time remote pupil center detection”, in *Proceedings of the Symposium on Eye Tracking Research and Applications*, Jun. 2018.

[72] W. Fuhl, D. Geisler, T. Santini, T. Appel, W. Rosenstiel, and E. Kasneci, “CBF:Circular binary features for robust and real-time pupil center detection”, in *Proceedings of the Symposium on Eye Tracking Research and Applications*, Jun. 2018.

[73] W. Fuhl, D. Geisler, T. Santini, W. Rosenstiel, and E. Kasneci, “Evaluation of State-of-the-art Pupil Detection Algorithms on Remote Eye Images”, in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, ser. UbiComp ’16, Heidelberg, Germany: ACM, 2016, pp. 1716–1725, ISBN: 978-1-4503-4462-3. DOI: 10.1145/2968219.2968340. [Online]. Available: http://doi.acm.org/10.1145/2968219.2968340.

[74] W. Fuhl, D. Hospach, T. Kübler, W. Rosenstiel, O. Bringmann, and E. Kasneci, “Ways of improving the precision of eye tracking data: Controlling the influence of dirt and dust on pupil detection”, *Journal of Eye Movement Research*, vol. 10, no. 3, 2017, ISSN: 1995-8692. [Online]. Available: https://bop.unibe.ch/index.php/JEMR/article/view/3657.

[75] W. Fuhl and E. Kasneci, “Eye movement velocity and gaze data generator for evaluation, robustness testing and assess of eye tracking software and visualization tools”, in *Egocentric Perception, Interaction and Computing Workshop (Poster in EPIC at ECCV)*, Sep. 2018.

[76] W. Fuhl, T. C. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci, “Arbitrarily shaped areas of interest based on gaze density gradient”, in *European Conference on Eye Movements*, Vienna, Austria, Aug. 2015.

[77] W. Fuhl, T. Kuebler, H. Brinkmann, R. Rosenberg, W. Rosenstiel, and E. Kasneci, “Region of interest generation algorithms for eye tracking data”, in *Third Workshop on Eye Tracking and Visualization (ETVIS), in conjunction with ACM ETRA*, Jun. 2018.

[78] W. Fuhl, T. Kuebler, T. Santini, and E. Kasneci, “Automatic generation of saliency-based areas of interest”, in *Symposium on Vision, Modeling and Visualization (VMV)*, Sep. 2018.

[79] W. Fuhl, T. C. Santini, T. Kübler, and E. Kasneci, "ElSe: Ellipse Selection for Robust Pupil Detection in Real-world Environments", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '16, Charleston, South Carolina: ACM, 2016, pp. 123–130, ISBN: 978-1-4503-4125-7. DOI: 10.1145/2857491.2857505. [Online]. Available: http://doi.acm.org/10.1145/2857491.2857505.

[80] W. Fuhl, T. Santini, D. Geisler, T. Kübler, and E. Kasneci, "EyeLad: Remote Eye Tracking Image Labeling Tool", in *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Feb. 2017.

[81] W. Fuhl, T. Santini, D. Geisler, T. Kübler, W. Rosenstiel, and E. Kasneci, "Eyes wide open? eyelid location and eye aperture estimation for pervasive eye tracking in real-world scenarios", in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 1656–1665.

[82] W. Fuhl, T. Santini, and E. Kasneci, "Fast camera focus estimation for gaze-based focus control", *CoRR*, vol. abs/1711.03306, 2017. arXiv: 1711.03306. [Online]. Available: http://arxiv.org/abs/1711.03306.

[83] W. Fuhl, T. Santini, G. Kasneci, and E. Kasneci, "PupilNet: Convolutional Neural Networks for Robust Pupil Detection", *arXiv preprint arXiv:1601.04902*, 2016.

[84] W. Fuhl, T. Santini, T. Kübler, N. Castner, W. Rosenstiel, and E. Kasneci, "Eye movement simulation and detector creation to reduce laborious parameter adjustments", *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1804.00970.

[85] W. Fuhl, T. Santini, C. Reichert, D. Claus, A. Herkommer, H. Bahmani, K. Rifai, S. Wahl, and E. Kasneci, "Non-intrusive practitioner pupil detection for unmodified microscope oculars", *Computers in Biology and Medicine*, vol. 79, pp. 36–44, 2016. DOI: 10.1016/j.compbiomed.2016.10.005.

[86] W. Fuhl, M. Tonsen, A. Bulling, and E. Kasneci, "Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art", *Machine Vision and Applications*, vol. 27, no. 8, pp. 1275–1288, 2016, ISSN: 1432-1769. DOI: 10.1007/s00138-016-0776-4. [Online]. Available: http://dx.doi.org/10.1007/s00138-016-0776-4.

[87] W. Gander, G. H. Golub, and R. Strebel, "Least-squares fitting of circles and ellipses", *BIT Numerical Mathematics*, vol. 34, no. 4, pp. 558–578, 1994.

[88] D. Geisler, W. Fuhl, T. Santini, and E. Kasneci, "Saliency Sandbox-Bottom-up Saliency Framework.", in *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2017, pp. 657–664.

[89] A. George and A. Routray, "Fast and Accurate Algorithm for Eye Localization for Gaze Tracking in Low Resolution Images", *arXiv preprint arXiv:1605.05272*, 2016.

[90] T. Gerstner, D. DeCarlo, M. Alexa, A. Finkelstein, Y. Gingold, and A. Nealen, "Pixelated image abstraction", in *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, 2012, pp. 29–36.

[91] J.-M. Geusebroek, F. Cornelissen, A. W. Smeulders, and H. Geerts, "Robust autofocusing in microscopy", *Cytometry*, vol. 39, no. 1, pp. 1–9, 2000.

[92] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1915–1926, 2012.

[93] J. H. Goldberg, M. J. Stimson, M. Lewenstein, N. Scott, and A. M. Wichansky, "Eye tracking in web search tasks: design implications", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ACM, 2002, pp. 51–58.

[94] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features", in *European Conference on Computer Vision*, Springer, 2014, pp. 392–407.

[95] S. Goni, J. Echeto, A. Villanueva, and R. Cabeza, "Robust algorithm for pupil-glint vector detection in a video-oculography eyetracking system", in *Proceedings of the 17th International Conference on Pattern Recognition*, IEEE, vol. 4, 2004, pp. 941–944.

[96] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[97] F. C. Groen, I. T. Young, and G. Ligthart, "A comparison of different focus functions for use in autofocus algorithms", *Cytometry*, vol. 6, no. 2, pp. 81–91, 1985.

[98] C. Guo, Q. Ma, and L. Zhang, "Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform", in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–8.

[99] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines", *Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[100] M. Heinert, "Support Vector Machines–Teil 1: Ein theoretischer Überblick", *Zeitschrift für Geodäsie, Geoinformation und Landmanagement*, vol. 3, pp. 179–189, 2010.

[101] V. Hilsenstein, "Robust autofocusing for automated microscopy imaging of fluorescently labelled bacteria", in *Digital Image Computing: Techniques and Applications (DICTA'05)*, IEEE, 2005, pp. 15–15.

[102] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[103] J. Hoffman, S. Guadarrama, E. S. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko, "LSDA: Large scale detection through adaptation", in *Advances in Neural Information Processing Systems*, 2014, pp. 3536–3544.

[104] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[105] X. Hou and L. Zhang, "Saliency Detection: A Spectral Residual Approach", in *Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383267.

[106] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data", in *Advances in Neural Information Processing Systems*, 2007, pp. 601–608.

[107] J.-T. Huang, C.-H. Shen, S.-M. Phoong, and H. Chen, "Robust measure of image focus in the wavelet domain", in *International Symposium on Intelligent Signal Processing and Communication Systems*, IEEE, 2005, pp. 157–160.

[108] W. Huang and Z. Jing, "Evaluation of focus measures in multi-focus image fusion", *Pattern Recognition Letters*, vol. 28, no. 4, pp. 493–500, 2007.

[109] T. Imai, K. Sekine, K. Hattori, N. Takeda, I. Koizuka, K. Nakamae, K. Miura, H. Fujioka, and T. Kubo, "Comparing the accuracy of video-oculography and the scleral search coil system in human eye movement analysis", *Auris Nasus Larynx*, vol. 32, no. 1, pp. 3–9, 2005.

[110] L. Itti and P. F. Baldi, "Bayesian surprise attracts human attention", in *Advances in Neural Information Processing Systems*, 2006, pp. 547–554.

[111] L. Itti, C. Koch, and E. Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov. 1998, ISSN: 0162-8828. DOI: 10.1109/34.730558. [Online]. Available: http://dx.doi.org/10.1109/34.730558.

[112] V. Jain and E. Learned-Miller, "Online domain adaptation of a pre-trained cascade of classifiers", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2011, pp. 577–584.

[113] A.-H. Javadi, Z. Hakimi, M. Barati, V. Walsh, and L. Tcheang, "SET: a pupil detection method using sinusoidal approximation", *Frontiers in Neuroengineering*, vol. 8, p. 4, 2015, ISSN: 1662-6443. DOI: 10.3389/fneng.2015.00004. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fneng.2015.00004.

[114] O. Jesorsky, K. J Kirchberg, and R. W. F., "Robust face detection using the hausdorff distance", in *Audio and Video-based Biometric Person Authentication*, Springer, 2001, pp. 90–95.

[115] N. Kan, N. Kondo, W. Chinsatit, and T. Saitoh, "Effectiveness of Data Augmentation for CNN-Based Pupil Center Point Detection", in *2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, IEEE, 2018, pp. 41–46.

[116] E. Kasneci, "Towards the Automated Recognition of Assistance Need for Drivers with Impaired Visual Field", eng, PhD thesis, University of Tübingen, Wilhelmstr. 32, 72074 Tübingen, 2013. [Online]. Available: http://tobias-lib.uni-tuebingen.de/volltexte/2013/7033.

[117] E. Kasneci, G. Kasneci, T. C. Kübler, and W. Rosenstiel, "Online Recognition of Fixations, Saccades, and Smooth Pursuits for Automated Analysis of Traffic Hazard Perception", in *Artificial Neural Networks*, Springer International Publishing, 2015, pp. 411–434, ISBN: 978-3-319-09903-3. DOI: 10.1007/978-3-319-09903-3_20.

[118] E. Kasneci, K. Sippel, K. Aehling, M. Heister, W. Rosenstiel, U. Schiefer, and E. Papageorgiou, "Driving with binocular visual field loss? A study on a supervised on-road parcours with simultaneous eye and head tracking", *PloS ONE*, vol. 9, no. 2, e87470, 2014. DOI: 10.1371/journal.pone.0087470.

[119] E. Kasneci, K. Sippel, M. Heister, K. Aehling, W. Rosenstiel, U. Schiefer, and E. Papageorgiou, "Homonymous Visual Field Loss and Its Impact on Visual Exploration: A Supermarket Study", *Translational Vision Science and Technology*, vol. 3, no. 6, 2014.

[120] M. Kassner, W. Patera, and A. Bulling, "Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction", in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, ACM, 2014, pp. 1151–1160. DOI: 10.1145/2638728.2641695.

[121] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees", in *Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867–1874. DOI: 10.1109/CVPR.2014.241.

[122] A. Keil, G. Albuquerque, K. Berger, and M. Magnor, "Real-time gaze tracking with a consumer-grade video camera", in *Proceedings of the Conferences on Computer Graphics, Visualization and Computer Vision*, V. Skala, Ed., 2010.

[123] D. E. King, "Dlib-ml: A machine learning toolkit", *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1755–1758, 2009.

[124] R. Klein, *Algorithmische Geometrie: Grundlagen, Methoden, Anwendungen*. Springer-Verlag, 2006.

[125] J. Kopf, A. Shamir, and P. Peers, "Content-adaptive image downscaling", *Transactions on Graphics*, vol. 32, 2013.

[126] T. C. Kübler, E. Kasneci, and W. Rosenstiel, "SubsMatch: Scanpath Similarity in Dynamic Scenes Based on Subsequence Frequencies", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '14, Safety Harbor, Florida: ACM, 2014, pp. 319–322, ISBN: 978-1-4503-2751-0. DOI: 10.1145/2578153.2578206. [Online]. Available: http://doi.acm.org/10.1145/2578153.2578206.

[127] T. C. Kübler, T. Rittig, E. Kasneci, J. Ungewiss, and C. Krauss, "Rendering Refraction and Reflection of Eyeglasses for Synthetic Eye Tracker Images", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '16, Charleston, South Carolina: ACM, 2016, pp. 143–146, ISBN: 978-1-4503-4125-7. DOI: 10.1145/2857491.2857494. [Online]. Available: http://doi.acm.org/10.1145/2857491.2857494.

[128] T. C. Kübler, K. Sippel, W. Fuhl, G. Schievelbein, J. Aufreiter, R. Rosenberg, W. Rosenstiel, and E. Kasneci, "Analysis of Eye Movements with Eyetrace", in *8th International Joint Conference on Biomedical Engineering Systems and Technologies*, A. Fred, H. Gamboa, and D. Elias, Eds. Cham: Springer International Publishing, 2015, pp. 458–471, ISBN: 978-3-319-27707-3. DOI: 10.1007/978-3-319-27707-3_28. [Online]. Available: https://doi.org/10.1007/978-3-319-27707-3_28.

[129] T. C. Kübler, K. Sippel, W. Fuhl, G. Schievelbein, J. Aufreiter, R. Rosenberg, W. Rosenstiel, and E. Kasneci, "Analysis of Eye Movements with Eyetrace", in *8th International Joint Conference on Biomedical Engineering Systems and Technologies*, A. Fred, H. Gamboa, and D. Elias, Eds. Cham: Springer International Publishing, 2015, pp. 458–471, ISBN: 978-3-319-27707-3. DOI: 10.1007/978-3-319-27707-3_28. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-27707-3_28.

[130] T. Kübler, S. Eivazi, and E. Kasneci, "Automated Visual Scanpath Analysis Reveals the Expertise Level of Micro-neurosurgeons", in *International Conference On Medical Image Computing and Computer Assisted Intervention Workshop*, Oct. 2015.

[131] T. Kübler, W. Fuhl, R. Rosenberg, W. Rosenstiel, and E. Kasneci, "Novel Methods for Analysis and Visualization of Saccade Trajectories", in *European Conference on Computer Vision Workshops*, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 783–797, ISBN: 978-3-319-46604-0. DOI: 10.1007/978-3-319-46604-0_54. [Online]. Available: https://doi.org/10.1007/978-3-319-46604-0_54.

[132] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient back propagation", in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 9–48.

[133] Y. LeCun, K. Kavukcuoglu, C. Farabet, *et al.*, "Convolutional networks and applications in vision.", in *Proceedings of IEEE International Symposium on Circuits and Systems*, 2010, pp. 253–256. DOI: 10.1109/ISCAS.2010.5537907.

[134] D.-T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation", *International Journal of Computer and Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.

[135] Y. Lee, R. J Micheals, J. J Filliben, and P. J. Phillips, "VASIR: an open-source research platform for advanced iris recognition technologies", *Journal of research of the National Institute of Standards and Technology*, vol. 118, p. 218, 2013.

[136] D. Li, D. Winfield, and D. J. Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches", in *Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, IEEE, 2005, pp. 79–79.

[137] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3367–3375.

[138] L. Lin, L. Pan, L. Wei, and L. Yu, "A robust and accurate detection of pupil images", in *3rd International Conference on Biomedical Engineering and Informatics (BMEI)*, IEEE, vol. 1, 2010, pp. 70–74.

[139] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum, "Learning to detect a salient object", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, pp. 353–367, 2011.

[140] X. Liu, F. Xu, and K. Fujimura, "Real-time eye detection and tracking for driver observation under various light conditions", in *Intelligent Vehicle Symposium, 2002. IEEE*, IEEE, vol. 2, 2002, pp. 344–351.

[141] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[142] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation", in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2200–2207.

[143] X. Long, O. K. Tonguz, and A. Kiderman, "A high speed eye tracking system with robust pupil center estimation algorithm", in *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2007, pp. 3331–3334.

[144] J Lorenzo, M Castrillon, J. Méndez, and O Deniz, "Exploring the use of local binary patterns as focus measure", in *International Conference on Computational Intelligence for Modelling Control and Automation*, IEEE, 2008, pp. 855–860.

[145] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[146] D. G. Lowe, "Object recognition from local scale-invariant features", in *The proceedings of the seventh IEEE International Conference on Computer Vision*, Ieee, vol. 2, 1999, pp. 1150–1157.

[147] Y.-F. Ma and H.-J. Zhang, "Contrast-based image attention analysis by using fuzzy growing", in *Proceedings of the eleventh ACM International Conference on Multimedia*, ACM, 2003, pp. 374–381.

[148] S. Martinez-Conde, S. L. Macknik, and D. H. Hubel, "The role of fixational eye movements in visual perception", *Nature Reviews Neuroscience*, vol. 5, no. 3, p. 229, 2004.

[149] F. Martinez, A. Carbone, and E. Pissaloux, "Gaze estimation using local features and non-linear regression", in *19th IEEE International Conference on Image Processing*, IEEE, 2012, pp. 1961–1964.

[150] I. Martinikorena, R. Cabeza, A. Villanueva, I. Urtasun, and A. Larumbe, "Fast and robust ellipse detection algorithm for head-mounted eye tracking systems", *Machine Vision and Applications*, pp. 1–16, 2018.

[151] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions", *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[152] F. Mawad, M. Trías, A. Giménez, A. Maiche, and G. Ares, "Influence of cognitive style on information processing and selection of yogurt labels: Insights from an eye-tracking study", *Food Research International*, vol. 74, pp. 1–9, 2015.

[153] R. K. McConnell, *Method of and apparatus for pattern recognition*, US Patent 4,567,610, 1986.

[154] L. K. McIntire, R. A. McKinley, C. Goodyear, and J. P. McIntire, "Detection of vigilance performance using eye blinks", *Applied Ergonomics*, vol. 45, no. 2, pp. 354–362, 2014.

[155] J. Mercer, "Functions of positive and negative type, and their connection with the theory of integral equations", *Philosophical Transactions of the Royal Society of London*, vol. 209, pp. 415–446, 1909.

[156] Metrovision, *Metrovision Electro-OculoGraphy*, 2016.

[157] R. S. Michalski, J. G. Carbonell, and T. M. M. Learning, "An Artificial Intelligence Approach", *Understanding the Nature of Learning*, vol. 2, pp. 3–26, 1983.

[158] M. Moeller, M. Benning, C. Schönlieb, and D. Cremers, "Variational depth from focus reconstruction", *Transactions on Image Processing*, vol. 24, no. 12, pp. 5369–5378, 2015.

[159] G. J. Mohammed, B. R. Hong, and A. A. Jarjes, "Accurate pupil features extraction based on new projection function", *Computing and Informatics*, vol. 29, no. 4, pp. 663–680, 2012.

[160] J. J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory", in *Numerical Analysis*, Springer, 1978, pp. 105–116.

[161] E. Mortensen, B. Morse, W. Barrett, and J. Udupa, "Adaptive boundary detection using 'live-wire' two-dimensional dynamic programming", in *Proceedings of Computers in Cardiology*, IEEE, Nov. 1992, pp. 635 –638, ISBN: 0-8186-3552-5. DOI: 10.1109/CIC.1992.269378.

[162] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1863–1868, 2006.

[163] J. L. Mundy and A. Zisserman, "Projective geometry for machine vision", 1992.

[164] Z. Nagy and P. Szolgay, "Configurable multilayer CNN-UM emulator on FPGA", *Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 6, pp. 774–778, 2003.

[165] S. T. Namin, M. Najafi, M. Salzmann, and L. Petersson, "A multi-modal graphical model for scene analysis", in *Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2015, pp. 1006–1013.

[166] N. M. Nasrabadi, "Pattern Recognition and Machine Learning", *Journal of Electronic Imaging*, vol. 16, no. 4, p. 049 901, 2007.

[167] S. K. Nayar, K. Ikeuchi, and T. Kanade, "Surface reflection: physical and geometrical perspectives", Defense Technical Information Center Document, Tech. Rep., 1989.

[168] S. K. Nayar and Y. Nakagawa, "Shape from focus", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 8, pp. 824–831, 1994.

[169] M. Nyström, "Off-line Foveated Compression and Scene Perception: An Eye-Tracking Approach", PhD thesis, Lund University, 2008.

[170] T. Ohno, "One-point calibration gaze tracking method", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ACM, 2006, pp. 34–34.

[171] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images", *Nature*, vol. 381, no. 6583, pp. 607–607, 1996.

[172] Optotune, "Datasheet: EL-10-30-Series", 2016.

[173] M. Ozuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code", in *Conference on Computer Vision and Pattern Recognition*, Ieee, 2007, pp. 1–8.

[174] B. Pan, H. A. Hembrooke, G. K. Gay, L. A. Granka, M. K. Feusner, and J. K. Newman, "The determinants of web page viewing behavior: an eye-tracking study", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ACM, 2004, pp. 147–154.

[175] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis", *Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[176] S. J. Pan and Q. Yang, "A survey on transfer learning", *Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[177] C. Payne, "Tired Eyes from Microsurgery—It's Blinking Obvious!", *Journal of Reconstructive Microsurgery*, vol. 29, no. 01, pp. 067–068, 2013.

[178] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space", *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[179] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study", in *15th International Conference on Pattern Recognition*, IEEE, vol. 3, 2000, pp. 314–317.

[180] A. Peréz, M. Cordoba, A Garcia, R. Méndez, M. Munoz, J. L. Pedraza, and F Sanchez, "A precise eye-gaze detection and tracking system", in *Conferences on Computer Graphics, Visualization and Computer Vision*, UNION Agency-Science Press, 2003.

[181] S. Pertuz, D. Puig, and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus", *Pattern Recognition*, vol. 46, no. 5, pp. 1415–1432, 2013.

[182] S. Pertuz, D. Puig, and M. A. Garcia, "Reliability measure for shape-from-focus", *Image and Vision Computing*, vol. 31, no. 10, pp. 725–734, 2013.

[183] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling", in *International Conference on Machine Learning*, 2014, pp. 82–90.

[184] D. K. Prasad, M. K. Leung, and C. Quek, "ElliFit: An unconstrained, non-iterative, least squares based geometric Ellipse Fitting method", *Pattern Recognition*, vol. 46, no. 5, pp. 1449–1465, 2013.

[185] V. Pratt, "Direct least-squares fitting of algebraic surfaces", in *Special Interest Group on Computer Graphics and Interactive Techniques*, ACM, vol. 21, 1987, pp. 145–152.

[186] J. M. Prewitt, "Object enhancement and extraction", *Picture Processing and Psychopictorics*, vol. 10, no. 1, pp. 15–19, 1970.

[187] C. M. Privitera and L. W. Stark, "Algorithms for Defining Visual Regions-of-Interest: Comparison with Eye Fixations", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 9, pp. 970–982, Sep. 2000, ISSN: 0162-8828. DOI: 10.1109/34.877520. [Online]. Available: http://dx.doi.org/10.1109/34.877520.

[188] C. M. Privitera and L. W. Stark, "Evaluating Image Processing Algorithms That Predict Regions of Interest", *Pattern Recognition Letters*, vol. 19, no. 11, pp. 1037–1043, Sep. 1998, ISSN: 0167-8655. DOI: 10.1016/S0167-8655(98)00077-4. [Online]. Available: http://dx.doi.org/10.1016/S0167-8655(98)00077-4.

[189] A. Radman, N. Zainal, and M. Ismail, "Efficient iris segmentation based on eyelid detection", *Journal of Engineering Science and Technology*, vol. 8, no. 4, pp. 399–405, 2013.

[190] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data", in *Proceedings of the 24th International Conference on Machine learning*, ACM, 2007, pp. 759–766.

[191] A. Raj, V. P. Namboodiri, and T. Tuytelaars, "Subspace alignment based domain adaptation for rcnn detector", *arXiv preprint arXiv:1507.05578*, 2015.

[192] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks", in *European Conference on Computer Vision*, Springer, 2016, pp. 525–542.

[193] Y Reibel, M Jung, M Bouhifd, B Cunin, and C Draman, "CCD or CMOS camera noise characterisation", *The European Physical Journal Applied Physics*, vol. 21, no. 01, pp. 75–80, 2003.

[194] A. Richards, *Alien vision: exploring the electromagnetic spectrum with imaging technology*. SPIE press, 2001, vol. 9.

[195] D. Robinson, "The mechanics of human saccadic eye movement", *The Journal of Physiology*, vol. 174, no. 2, pp. 245–264, 1964.

[196] D. A. Robinson, "A method of measuring eye movemnent using a scieral search coil in a magnetic field", *Transactions on Bio-medical Electronics*, vol. 10, no. 4, pp. 137–145, 1963.

[197] D. A. Robinson, "The mechanics of human smooth pursuit eye movement.", *The Journal of Physiology*, vol. 180, no. 3, pp. 569–591, 1965.

[198] R. Rosenberg, "Blicke messen: Vorschläge für eine empirische bildwissenschaft", *Jahrbuch der Bayerischen Akademie der Schönen Künste*, vol. 27, pp. 71–86, 2014.

[199] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.", *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[200] M. J. Russell and T. S. Douglas, "Evaluation of autofocus algorithms for tuberculosis microscopy", in *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2007, pp. 3489–3492.

[201] S. Russell, P. Norvig, and A. Intelligence, "A modern approach", *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, p. 27, 1995.

[202] U. Rutishauser, D. Walther, C. Koch, and P. Perona, "Is bottom-up attention useful for object recognition?", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, vol. 2, 2004, pp. II–II.

[203] A. Santella and D. DeCarlo, "Robust Clustering of Eye Movement Recordings for Quantification of Visual Interest", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '04, San Antonio, Texas: ACM, 2004, pp. 27–34, ISBN: 1-58113-825-3. DOI: 10.1145/968363.968368. [Online]. Available: http://doi.acm.org/10.1145/968363.968368.

[204] T. Santini, W. Fuhl, T. Kübler, and E. Kasneci, "Bayesian Identification of Fixations, Saccades, and Smooth Pursuits", in *Eye Tracking Research and Applications*, Charleston, South Carolina: ACM, 2016, pp. 163–170, ISBN: 978-1-4503-4125-7. DOI: 10.1145/2857491.2857512. [Online]. Available: http://doi.acm.org/10.1145/2857491.2857512.

[205] T. Santini, W. Fuhl, T. Kuebler, and E. Kasneci, "EyeRec: An Open-Source Data Acquisition Software for Head-Mounted Eye-tracking", in *Proceedings of the 11th International Conference on Computer Vision Theory and Applications*, Feb. 2016.

[206] T. Santini, W. Fuhl, D. Geisler, and E. Kasneci, "EyeRecToo: Open-source Software for Real-time Pervasive Head-mounted Eye Tracking.", in *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2017, pp. 96–101.

[207] T. Santini, W. Fuhl, and E. Kasneci, "CalibMe: Fast and Unsupervised Eye Tracker Calibration for Gaze-Based Pervasive Human-Computer Interaction", in *Proceedings of the Conference on Human Factors in Computing Systems*, ser. CHI '17, Denver, Colorado, USA: ACM, 2017, pp. 2594–2605, ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025950. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025950.

[208] A. Santos, C Ortiz de Solórzano, J. J. Vaquero, J. Pena, N. Malpica, and F Del Pozo, "Evaluation of autofocus functions in molecular cytogenetic analysis", *Journal of Microscopy*, vol. 188, no. 3, pp. 264–272, 1997.

[209] S. K. Schnipke and M. W. Todd, "Trials and Tribulations of Using an Eye-tracking System", in *Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '00, The Hague, The Netherlands: ACM, 2000, pp. 273–274, ISBN: 1-58113-248-4. DOI: 10.1145/633292.633452. [Online]. Available: http://doi.acm.org/10.1145/633292.633452.

[210] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[211] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis", in *International Conference on Artificial Neural Networks*, Springer, 1997, pp. 583–588.

[212] F. Schulze, "How the Humble Stereomicroscope Found its Way into Modern Surgery: The Zeiss Operating Microscope", *Micscape Magazine*, 2013.

[213] B. Sheliga, V. Brown, and F. Miles, "Voluntary saccadic eye movements in humans studied with a double-cue paradigm", *Vision Research*, vol. 42, no. 15, pp. 1897–1915, 2002.

[214] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000, ISSN: 0162-8828. DOI: 10.1109/34.868688. [Online]. Available: http://dx.doi.org/10.1109/34.868688.

[215] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning", *Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[216] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning From Simulated and Unsupervised Images Through Adversarial Training", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[217] K. Sippel, E. Kasneci, K. Aehling, M. Heister, W. Rosenstiel, U. Schiefer, and E. Papageorgiou, "Binocular Glaucomatous Visual Field Loss and Its Impact on Visual Exploration - A Supermarket Study", *PLoS ONE*, vol. 9, no. 8, e106089, 2014. DOI: 10.1371/journal.pone.0106089. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0106089.

[218] K. Sippel, T. Kübler, W. Fuhl, G. Schievelbein, R. Rosenberg, and W. Rosenstiel, "Eyetrace2014 - Eyetracking Data Analysis Tool", in *Proceedings of the International Conference on Health Informatics*, 2015, pp. 212–219, ISBN: 978-989-758-068-0. DOI: 10.5220/0005352902120219.

[219] *Smarteye*, http://smarteye.se/, Accessed: 2019-01-01.

[220] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator", *Applied and Computational Harmonic Analysis*, vol. 43, no. 2, pp. 233–268, 2017.

[221] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[222] B. Strobel, M. A. Lindner, S. Saß, and O. Köller, "Task-irrelevant data impair processing of graph reading tasks: An eye tracking study", *Learning and Instruction*, vol. 55, pp. 139–147, 2018.

[223] M. Subbarao and T. Choi, "Accurate recovery of three-dimensional shape from image focus", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 266–274, 1995.

[224] Y. Sugano and A. Bulling, "Self-Calibrating Head-Mounted Eye Trackers Using Egocentric Visual Saliency", in *Proceedings of the 28th ACM Symposium on User Interface Software and Technology (UIST)*, Nov. 5, 2015, pp. 363–372. DOI: 10.1145/2807442.2807445.

[225] M. Sugiyama, M. Krauledat, and K.-R. MÃžller, "Covariate shift adaptation by importance weighted cross validation", *Journal of Machine Learning Research*, vol. 8, no. May, pp. 985–1005, 2007.

[226] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye, "A two-stage weighting framework for multi-source domain adaptation", in *Advances in Neural Information Processing Systems*, 2011, pp. 505–513.

[227] Y. Sun, S. Duthaler, and B. J. Nelson, "Autofocusing in computer microscopy: selecting the optimal focus algorithm", *Microscopy Research and Technique*, vol. 65, no. 3, pp. 139–149, 2004.

[228] S. Suwajanakorn, C. Hernandez, and S. M. Seitz, "Depth from focus with your mobile phone", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3497–3506.

[229] M. Suzuki, N. Yamamoto, O. Yamamoto, T. Nakano, and S. Yamamoto, "Measurement of driver's consciousness by image processing-a method for presuming driver's drowsiness by eye-blinks coping with individual differences", in *Systems, Man, and Cybernetics Society*, IEEE, vol. 4, 2006, pp. 2891–2896.

[230] L. Świrski and N. Dodgson, "Rendering Synthetic Ground Truth Images for Eye Tracker Evaluation", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '14, Safety Harbor, Florida: ACM, 2014, pp. 219–222, ISBN: 978-1-4503-2751-0. DOI: 10.1145/2578153.2578188. [Online]. Available: http://doi.acm.org/10.1145/2578153.2578188.

[231] L. Świrski, A. Bulling, and N. Dodgson, "Robust Real-time Pupil Tracking in Highly Off-axis Images", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '12, Santa Barbara, California: ACM, 2012, pp. 173–176, ISBN: 978-1-4503-1221-9. DOI: 10.1145/2168556.2168585. [Online]. Available: http://doi.acm.org/10.1145/2168556.2168585.

[232] L. Świrski and N. A. Dodgson, "A fully-automatic, temporal approach to single camera, glint-free 3D eye model fitting [Abstract]", in *European Conference on Eye Movements*, Lund, Sweden, Aug. 2013. [Online]. Available: http://www.cl.cam.ac.uk/research/rainbow/projects/eyemodelfit/.

[233] E. Tafaj, G. Kasneci, W. Rosenstiel, and M. Bogdan, "Bayesian Online Clustering of Eye Movement Data", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '12, Santa Barbara, California: ACM, 2012, pp. 285–288, ISBN: 978-1-4503-1221-9. DOI: 10.1145/2168556.2168617. [Online]. Available: http://doi.acm.org/10.1145/2168556.2168617.

[234] N. M. Taylor, R. H. Eikelboom, P. P. Van Sarloos, and P. G. Reid, "Determining the accuracy of an eye tracking system for laser refractive surgery", *Journal of Refractive Surgery*, vol. 16, no. 5, S643–S646, 2000.

[235] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[236] F. Timm and E. Barth, "Accurate Eye Centre Localisation by Means of Gradients.", *International Conference on Computer Vision Theory and Applications*, vol. 11, pp. 125–130, 2011.

[237] *Tobii*, https://www.tobii.com/, Accessed: 2019-01-01.

[238] C. Tomasi and T. Kanade, *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[239] M. Tonsen, J. Steil, Y. Sugano, and A. Bulling, "InvisibleEye: Mobile Eye Tracking Using Multiple Low-Resolution Cameras and Learning-Based Gaze Estimation", *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 106, 2017.

[240] M. Tonsen, X. Zhang, Y. Sugano, and A. Bulling, "Labelled Pupils in the Wild: A Dataset for Studying Pupil Detection in Unconstrained Environments", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '16, Charleston, South Carolina: ACM, 2016, pp. 139–142, ISBN: 978-1-4503-4125-7. DOI: 10.1145/2857491.2857520. [Online]. Available: http://doi.acm.org/10.1145/2857491.2857520.

[241] S. Trösterer, A. Meschtscherjakov, D. Wilfinger, and M. Tscheligi, "Eye Tracking in the Car: Challenges in a Dual-Task Scenario on a Test Track", in *Proceedings of the 6th AutomotiveUI*, ACM, 2014.

[242] M. Uhrich, R. Underwood, J. Standeven, N. Soper, and J. Engsberg, "Assessment of fatigue, monitor placement, and surgical experience during simulated laparoscopic surgery", *Surgical Endoscopy*, vol. 16, no. 4, pp. 635–639, 2002.

[243] R. Valenti and T. Gevers, "Accurate Eye Center Location through Invariant Isocentric Patterns", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1785–1798, 2012.

[244] F. Vera-Olmos, E Pardo, H Melero, and N Malpica, "DeepEye: Deep convolutional network for pupil detection in real environments", *Integrated Computer-Aided Engineering*, vol. 26, no. 1, pp. 85–95, 2019.

[245] A. Villanueva, V. Ponz, L. Sesma-Sanchez, M. Ariz, S. Porta, and R. Cabeza, "Hybrid method based on topography for robust detection of iris center and eye corners", *Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 9, no. 4, p. 25, 2013.

[246] P. Viola and M. J. Jones, "Robust real-time face detection", *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[247] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, vol. 1, 2001, pp. I–511.

[248] D. G. Viswanathan, *Features from Accelerated Segment Test (FAST)*, 2009.

[249] G. Wald *et al.*, "Human vision and the spectrum", *Science*, vol. 101, no. 2635, pp. 653–658, 1945.

[250] J. Wan, Q. Ruan, W. Li, and S. Deng, "One-shot learning gesture recognition from RGB-D data using bag of features", *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2549–2582, 2013.

[251] M. Wang, J. Konrad, P. Ishwar, K. Jing, and H. Rowley, "Image saliency: From intrinsic to extrinsic context", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2011, pp. 417–424.

[252] C.-Y. Wee and R. Paramesran, "Measure of image sharpness using eigenvalues", *Information Sciences*, vol. 177, no. 12, pp. 2533–2552, 2007.

[253] R. P. Wildes, "Iris recognition: an emerging biometric technology", *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1348–1363, 1997, ISSN: 0018-9219. DOI: 10.1109/5.628669.

[254] C. E. Willert and M. Gharib, "Digital particle image velocimetry", *Experiments in Fluids*, vol. 10, no. 4, pp. 181–193, 1991.

[255] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis", *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[256] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Learning an Appearance-based Gaze Estimator from One Million Synthesised Images", in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '16, Charleston, South Carolina: ACM, 2016, pp. 131–138, ISBN: 978-1-4503-4125-7. DOI: 10.1145/2857491.2857492. [Online]. Available: http://doi.acm.org/10.1145/2857491.2857492.

[257] D. S. Wooding, "Eye movements of large populations: II. Deriving regions of interest, coverage, and similarity using fixation maps", *Behavior Research Methods, Instruments, and Computers*, vol. 34, no. 4, pp. 518–528, 2002, ISSN: 1532-5970. DOI: 10.3758/BF03195481. [Online]. Available: http://dx.doi.org/10.3758/BF03195481.

[258] D. Wu, F. Zhu, and L. Shao, "One shot learning gesture recognition from rgbd images", in *Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, 2012, pp. 7–12.

[259] Y. Wu, I. Kozintsev, J.-Y. Bouguet, and C. Dulong, "Sampling strategies for active learning in personal photo retrieval", in *International Conference on Multimedia and Expo*, IEEE, 2006, pp. 529–532.

[260] H. Xie, W. Rong, and L. Sun, "Wavelet-based focus measure and 3-d surface reconstruction method for microscopy images", in *International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 229–234.

[261] J. Xu, S. Ramos, D. Vázquez, A. M. López, and D Ponsa, "Incremental Domain Adaptation of Deformable Part-based Models.", in *British Machine Vision Conference*, 2014.

[262] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution", in *Advances in Neural Information Processing Systems*, 2014, pp. 1790–1798.

[263] F. Yang, X. Yu, J. Huang, P. Yang, and D. Metaxas, "Robust eyelid tracking for fatigue detection", in *International Conference on Image Processing*, 2012, pp. 1829–1832.

[264] G. Yang and B. J. Nelson, "Wavelet-based autofocusing and unsupervised segmentation of microscopic images", in *International Conference on Intelligent Robots and Systems*, IEEE, vol. 3, 2003, pp. 2143–2148.

[265] P. T. Yap and P Raveendran, “Image focus measure based on Chebyshev moments”, *Proceedings-Vision, Image and Signal Processing*, vol. 151, no. 2, pp. 128–136, 2004.

[266] A. L. Yarbus, “Eye movements during perception of complex objects”, in *Eye Movements and Vision*, Springer, 1967, pp. 171–211.

[267] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “It’s Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation”, *CoRR*, vol. abs/1611.08860, 2016. [Online]. Available: http://arxiv.org/abs/1611.08860.

[268] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “It’s written all over your face: Full-face appearance-based gaze estimation”, in *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, 2017, pp. 2299–2308.

[269] D. Zhu, S. T. Moore, and T. Raphan, “Robust pupil center detection using a curvature algorithm”, *Computer Methods and Programs in Biomedicine*, vol. 59, no. 3, pp. 145–157, 1999.