Christoph Leitner, BSc

Eye Tracking In Cataract Surgeries

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur
Master's degree program: Telematik

submitted to
Graz University of Technology

Supervisor
Univ.-Prof. Dipl.-Ing. Dr.techn. Horst Bischof
Dipl.-Ing. Dr.techn. Matthias Rüther

Graz, August 2018

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch

Datum

Unterschrift

Abstract

Unoperated cataract is the main cause for blindness as well as a major cause for moderate to serve vision impairment, but it can be healed with a cataract surgery. If the cataract is combined with an astigmatism, it is necessary for the surgeon to know the center of the eye and the orientation of the astigmatism at any moment of the surgery, to achieve a good vision restoration. At the current state, the surgeon has to use a special marking tool to mark the astigmatism axis with ink directly on the eye. The initial marking accuracy of this process strongly depends on the experience of the surgeon and the marks can get blurred or disappear during the surgery, which will cause a decreased vision restoration quality.

To overcome this problem, we present an eye tracking approach specially designed for cataract surgeries. By tracking the center of the eye as well as the astigmatism axis during the complete surgery, the surgeon can have access to this information at any point of the surgery via a camera stream which is used to perform the surgery. Additionally, this information cannot disappear or can get blurred over time due to blood vessels that burst or applied water flushes. Therefore this approach is intended to deliver more accurate position and orientation information in the artificial lens placement step.

In the first part of this thesis, we present an appearance-based tracking approach for cataract surgeries, that is real-time capable. It is based on template tracking with a robust image feature based homography estimation for doing the template update. Additionally, the robustness is increased by applying an adaptive image blending method when updating the templates. In addition to the template tracker, a tracking by detection approach is presented. As a detector, the aggregated channel features (ACF) detector is used. The detection results are subsequently refined to figure out the center of the pupil of the detected eye. This is done by applying a polar transform to the detected eye and finding the border between iris and lens. Finally, both methods are evaluated with respect to their runtime and accuracy. The template tracker can perform with more than 30 frames per second and has a mean distance error of 1.059% which is below the maximally allowed error of 1.27%. The ACF detector itself can achieve more than 210 frames per seconds and has a mean distance error of 2.405% whereas the refinement can perform with 2 frames per second and a mean distance error of 1.769%.

Keywords: eye tracking, cataract surgery, template tracking, homography estimation, ACF detector, tracking by detection

Kurzfassung

Die häufigste Ursache für Blindheit ist grauer Star. Dieser kann jedoch mit einer grauen Star Operation entgegen gewirkt werden. Soll im Zuge der Operation nicht nur der graue Star, sondern auch ein Astigmatismus geheilt werden, muss der Chirurg während der gesamten Operation den Mittelpunkt des Auges, sowie die Ausrichtung des Astigmatismus kennen. Hierfür wird mit einem speziellen Markierwerkzeug und einer Art Filzstift die Astigmatismus-Achse direkt am Auge markiert. Die Markiergenauigkeit ist dabei sehr stark von der Erfahrung des Chirurgen abhängig. Ein weiteres Problem, welches während der Operation auftreten kann ist, dass die Markierungen verschwimmen oder gänzlich verschwinden können. Dies wiederum kann zu einer Verschlechterung des Operationsergebnisses führen.

Um diese Problematik zu bewältigen, wird in dieser Arbeit ein Eye Tracking Ansatz vorgestellt, der speziell für graue Star Operationen entwickelt wurde. Dabei werden der Mittelpunkt des Auges, sowie die Ausrichtung des Astigmatismus über einen zur Verfügung gestellten Kamerastream über die gesamte Operationsdauer getrackt. Der Vorteil dieses Ansatzes ist es, dass die tracking Information im Laufe der Operation nicht verschwimmen oder ganz verschwinden kann. Zusätzlich hat der Chirurg jederzeit Zugriff auf diese Informationen, wodurch eine genauere Platzierung der eingeführten künstlichen Linse ermöglicht wird.

Im ersten Teil der Arbeit wird ein echtzeitfähiger Eye Tracker für grauer Star Operationen vorgestellt, welcher auf Template Tracking basiert. Das Template Update erfolgt dabei über eine robuste Homographie Schätzung und einen adaptiven Bildverschmelzungsansatz. Im zweiten Teil der Arbeit wird ein Tracking by Detection Ansatz vorgestellt, welcher auf dem Aggregated Channel Features (ACF) Detektor basiert. Anschließend wird das Detektionsergebnis schrittweise verbessert, um den Mittelpunkt des Auges robuster zu detektieren. Hierfür wird das detektierte Auge polartransformiert und anschließend wird der Übergang zwischen Iris und Linse gefunden. Abschließend werden der Tracker und der Detektor einer Laufzeit- und Genauigkeitsanalyse unterzogen. Der Tracker erreicht eine Laufzeit von über 30 Bildern pro Sekunde und eine durchschnittliche Abweichung vom Sollwert von 1,059%, welcher unter dem vorgegebenen Wert von 1,27% liegt. Der ACF Detektor erreicht eine Laufzeit von 210 Bildern pro Sekunden und eine durchschnittliche Abweichung vom Sollwert von 2,405%. Die Detektionsverfeinerung erreicht eine Laufzeit von 2 Bildern pro Sekunde und eine durchschnittliche Abweichung von 1,769%.

Schlagwörter: Eye Tracking, Grauer Star Operation, Template Tracking, Homographie Schätzung, ACF Detektor, Tracking by Detection

Acknowledgments

First of all, I would like to thank my supervisors Prof. Dr. Horst Bischof and Dr. Matthias Rüther who gave me the opportunity to work on this interesting project. Their guidance, professional support and valuable feedback helped me to finish this thesis.
In addition, I would like to thank my lab colleague Felix Kühnel for the helpful advice he gave and the discussions we had.
Furthermore, I would like to thank my dear friend and former roommate Fabian Oblinger for the interesting and fruitful discussions, his advice, his support and the great time we had during our complete Bachelor and Master studies.
Last but not least I would like to thank my family and girlfriend for their love, mental support and encouragements that helped me during my Bachelor and Master studies.

Contents

List of Figures

List of Tables

1 Introduction

Globally, of the 7.33 billion people alive (as of 2015), an estimate of 252.6 million people live with vision impairment. 36 million people of these were blind and 216.6 million had moderate to severe visual impairment [9]. According to the World Health Organization (WHO), unoperated cataract is with 35% the main cause for blindness and with 25% a major cause for moderate to severe vision impairment [78]. This leads to the fact that approximately 66.75 million people live with vision impairment caused by unoperated cataract.

For a better understanding of the following text, Figure 1.1 should give an overview of the main parts of the human eye and where they are located. The clear front surface of the eye is called cornea and is primarily used to focus light. The pupil is controlling the amount of light reaching the back of the eye by automatically adjusting the size. It can be compared with the aperture control of a camera. Directly behind the pupil, the crystalline lens is located. It is used to further focus light so the eye can automatically focus on near and approaching objects. The focused light hits the retina which acts as a light-sensitive sensor and converts optical images into electronic signals. The optic nerve transmits these signals to the part of the brain that controls our sense of sight. This part of the brain is called virtual cortex [61].

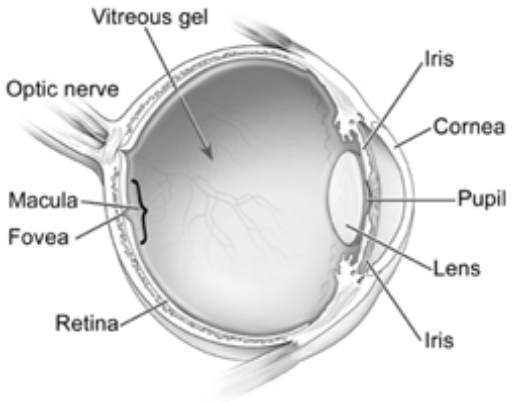
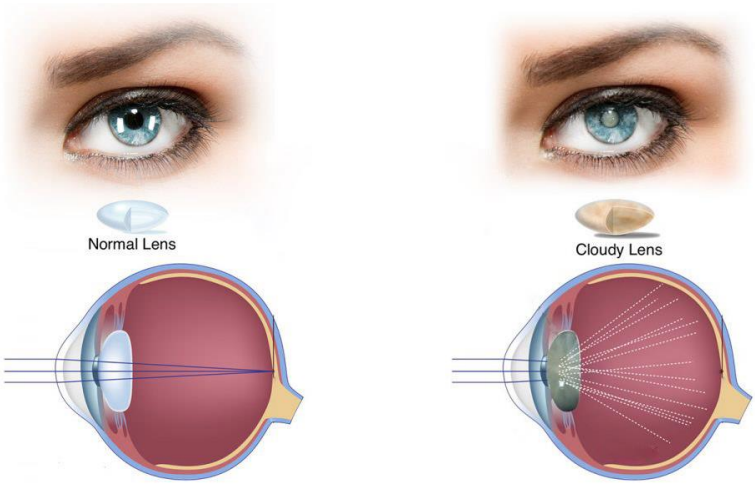


Figure 1.1 Main parts of the human eye [79]

A cataract is a clouding of the lens inside the eye (Figure 1.2) and it causes vision loss, which cannot be corrected with glasses, contact lenses or corneal refractive (refractive errors are for example nearsightedness, farsightedness or astigmatism) surgery. The lens of the eye is primarily made of water and protein. The protein is arranged in such a way, that it keeps the lens clear and let's light pass through. As we age, the protein can

clump together and make the lens cloudy. Generally, a cataract starts out small and has only little impacts on one's vision. Over time, the cataract can grow which makes seeing harder. It may be noticed as a little blur, like looking through a cloudy piece of glass [5, 69, 79]. Figure 1.3 gives a comparison on how a person with regular vision (Figure 1.3 (a)) and a person with cataract (Figure 1.3 (a)) sees the same scene.



(a) (b)

Figure 1.2 Comparison of a normal lens (a) and a cloudy lens (b) caused by a cataract (adapted from [80])

(a) (b)

Figure 1.3 Comparison of two scenes viewed by a person with normal vision (a) and a person with cataract (b) [5, 79]

After a cataract is properly diagnosed, the next step is to restore one's vision with a cataract surgery. Due to [80] cataract surgeries are one of the most commonly performed operations worldwide and are considered as an extremely safe and effective surgical procedure. The procedure itself consists of four main steps and is typically performed on an outpatient basis and no overnight stay is required [69]. In the first step, an approximately 3mm small incision is made at the corneal margin (Figure 1.4 (a)). Afterward, a phacoemulsification probe is inserted which breaks the cataract into microscopic fragments with ultrasound. The fragments can be aspirated with the help of the tooltip (Figure 1.4 (b)). After the natural lens is completely removed, an artificial foldable intraocular lens (IOL) can be inserted through the incision (Figure 1.4 (c)). Once the artificial lens is in the eye, it unfolds and can be placed in the right position by the surgeon (Figure 1.4 (d)). The small incision heals naturally without the need for a suture [81]. The IOL will not only lead to a clear vision, they can furthermore correct nearsightedness, farsightedness and modern IOLs can correct astigmatism as well.

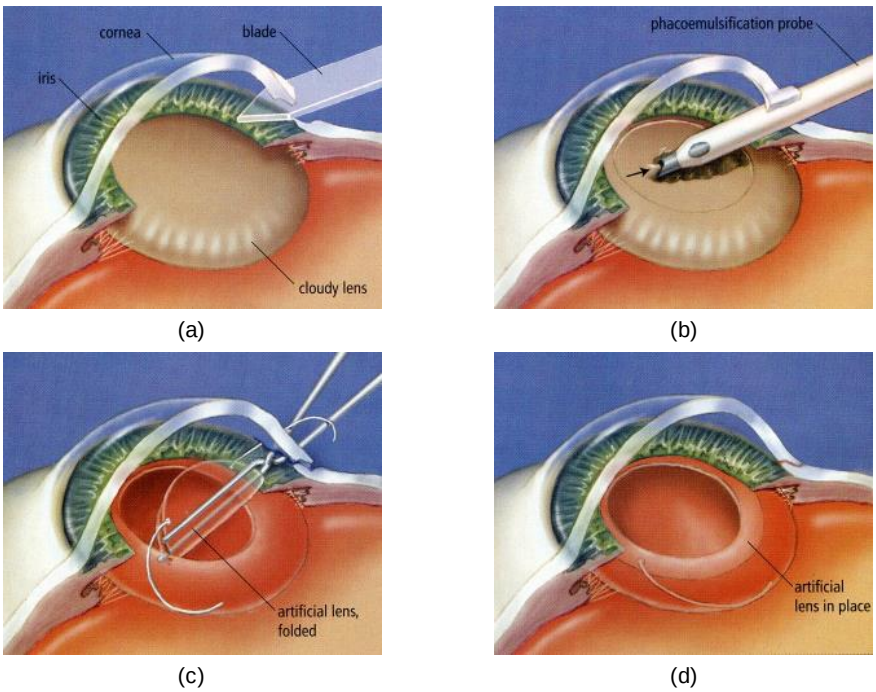


Figure 1.4 Four main steps of a cataract surgery ((a) incision, (b) lens removal, (c) lens implantation, (d) result) (adapted from [81])

Astigmatism is usually caused by an irregular shape of the cornea. Instead of a symmetrically round shape, one meridian is significantly more curved than the meridian perpendicular to it [32]. To correct astigmatism during a cataract surgery a toric IOL is used.

Toric IOLs have different powers in different meridians of the lens to correct the asymmetric power of the eye. Compared to a regular cataract surgery, a cataract surgery with astigmatism correction is essentially the same but still has some important differences. Prior to the surgery, measurements are done to figure out the correct power and orientation of the implant. Toric IOLs have special markers on the lens to help the surgeon to see the orientation of the astigmatism correction on the lens. Once the lens is implanted to the eye, the surgeon can rotate the lens so that the correction axis is properly aligned with the astigmatism axis. The usage of toric IOLs do not increase the risk of a complication, but a misaligned toric IOL can cause blurred vision which cannot easily be corrected with eyeglasses or contact lenses [33]. To overcome this problem and help the surgeon to correctly align the correction axis of the lens with the astigmatism axis a tracking approach is presented, which tracks the position and the axis of the eye during the whole surgery.

1.1 Problem Statement

In cataract surgery, the refractive requirements are steadily increasing. Not only by the patients but also by the surgeons. Besides nearsightedness and farsightedness, also astigmatism should be recovered after the surgery. To get close to this objective a variety of different lenses is available on the market. Before the surgeon can start with the surgery, he needs to figure out the center of the eye and the rotation of the astigmatism. This information is needed subsequently in the last step of the surgery, where the surgeon needs to place the lens in the eye in a very precise way. A rotation or decentration of the implanted lens can lead to deterioration of one's vision.

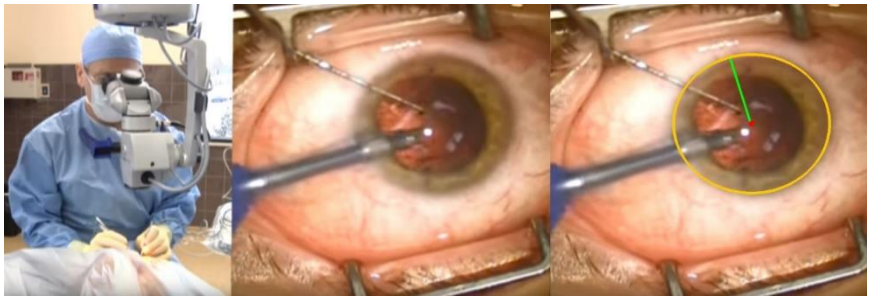
(a) (b)

Figure 1.5 Astigmatism axis marking with a bubble marker ((a) marking with a bubble marker, (b) the red rings show the marks on the eye) [22]

At the current state, the surgeon has to use a special marking tool to mark the astigmatism axis [37]. An example of such a tool is given in Figure 1.5 (a). In this specific case, a bubble marker with a special gentian violet ink is used. The initial marking accuracy of

this process strongly depends on the experience of the surgeon. Additionally, we can see, the marks are quite blurry (Figure 1.5 (b)) and this can get worse during the surgery, as blood vessels can burst or water flushes are applied. Less accurate marking will directly lead to less accurate lens positions, which will decrease the overall quality of the vision restoration.

To overcome this problem, we present an eye tracking approach for cataract surgeries. Again, at the beginning of the surgery, which will be the start of the tracking process, the center of the eye, as well as the astigmatism axis needs to be accurately marked by the surgeon. But this is a much simpler task now, as it can be done on an image and not directly on the real eye. Additionally, this procedure can easily be repeated several times until the surgeon is satisfied with his marking results.



(a) (b) (c)

Figure 1.6 Surgeon performing a cataract surgery ((a) surgeon who performs a cataract surgery with a camera and "head-mounted" display, (b) camera stream observed by the surgeon, (c) camera stream with proposed visualization for eye center and astigmatism axis) (adapted from [56])

During the surgery, the surgeon does not directly look at the patient's eye, instead, he uses a device with a camera (which observes the eye) and a "head-mounted" display (which shows the camera stream) (Figure 1.6 (a)). Figure 1.6 (b) shows the camera stream which is observed by the surgeon via the "head-mounted" display. By tracking the center of the eye as well as the axis of the astigmatism during the complete surgery, the surgeon can have access to this information at any point of the surgery. Nevertheless, it will be most likely need in the last steps of the surgery, where the toric IOL is implanted in the eye. Additionally, this tracking information can be directly visualized to the camera stream which is observed via the "head-mounted" display (Figure 1.6 (c)). This information cannot disappear or can get blurred over time due to blood vessels that burst or applied water flushes, as it can happen when the regular marking technique is used. Therefore this approach is intended to deliver more accurate position and orientations information in the toric IOL placement step.

As one can imagine, the proposed tracker has to deal with very challenging datasets which can include the following problems:

- The camera which is observing the surgery can be moved in x- and y-direction
- The scale can change due to zooming
- Under and overexposure is possible
- Reflections are possible
- The eye can move in 3D space
- The eye will undergo a non-rigid deformation during the surgery
- The lens will undergo an appearance change during the surgery (cloudy lens, broken cataract in the lens, no lens, artificial lens)
- Blood vessels can burst
- The eye can be covered by the surgeon, surgical tools, water flushes or blood

Figure 1.7 gives a visual overview of the challenges the eye tracker has to deal with. It shows the positions change of the capturing device as well as the movement of the eye. Row (a) shows the appearance change of the lens during the surgery. (a.1) shows the cloudy lens, (a.2) the lens after the cataract was broken up and (a.3) the eye after the lens was removed. In (a.4) the new artificial lens is already implanted. Row (b) shows how water and blood can change the appearance of the eye. In addition to that, reflections which are caused by blood and water are demonstrated. Finally, row (c) gives an overview of challenges that are caused by surgical devices.

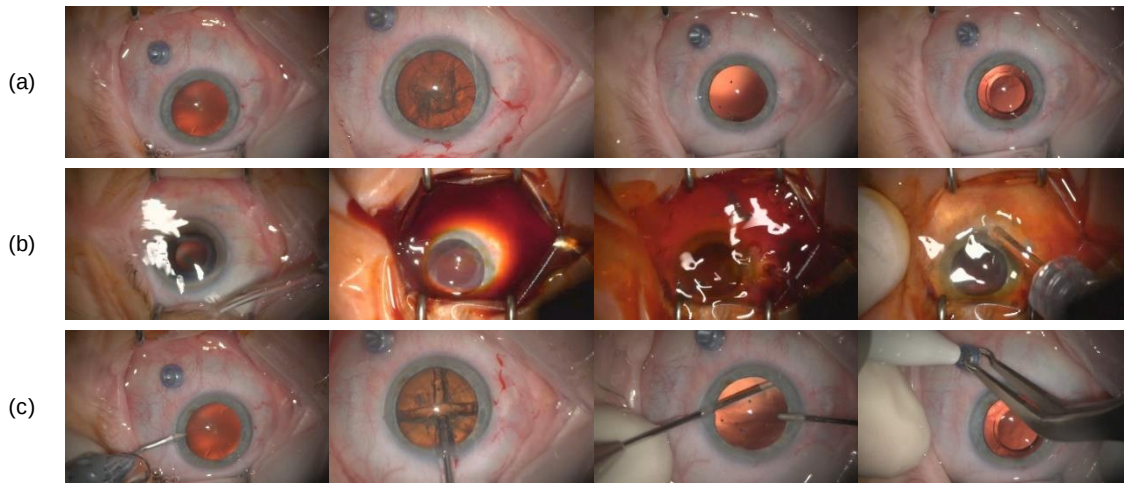


Figure 1.7 challenges of the eye tracker ((general) different position of the camera and the eye, scale change, reflections,(a) appearance change of the lens during the surgery, (b) influence of water flushes and blood, (c) influence of surgery tools)

Based on the findings of [37], for manual marking a mean postoperative toric misalignment error of 3.5° can be reached, and this should be decreased by the presented approach. Additionally, a distance error of less than 0.15mm should be reached. Based on the findings of [30], the mean white-to-white corneal diameter for adults is 11.8mm. Therefore the 0.15mm lead to a distance error of 1.27%. Last but not least, the surgeon needs access to this information at any point of the surgery. Therefore the provided eye tracker needs to be real-time capable.

1.2 Motivation

Object tracking is a very common problem in computer vision. It can be defined as tracing the progress of objects as they move around in the scene. Furthermore, some trackers can provide additional information about orientation, area or shape of an object [2, 47, 52, 53, 74]. In general, object tracking is a very challenging problem as it has to deal with several difficulties like abrupt object motion, changing the appearance of the fore- and the background, non-rigid object structures, occlusion, noise in the image and camera motion. To simplify tracking tasks, constraints to the motion or the appearance of an object can be set. For example, one can assume a uniform velocity or acceleration without any abrupt changes. In addition to that prior knowledge, the number, the size and the shape of the object can be used to simplify the problem [47, 74]. Nevertheless, object tracking has a very broad range of applications such as automated surveillance, traffic monitoring, vehicle navigation, human-computer interaction or assistance in medical surgeries [2, 26, 53, 74].

Furthermore, there have been a lot of investigations into general eye tracking systems as the eyes and their movements play an important role in expressing a person's desires, needs, cognitive processes and emotional states. The importance of eye movement with regards to the perception of and attention to the world is confirmed. It is the method through which the information is gathered, which is necessary to identify the characteristics of the visual world. In addition to that, the eyes can be considered as relatively stable compared to other facial features. Therefore general eye tracking plays an important role in creating human-computer interfaces, attentive user interfaces and understanding human affective states. In addition to that, eye movements are the least affected by disabilities and can, therefore, be used for assistive technologies. The unique geometric, photometric and motion characteristics of an eye can be used for face detection, face recognition and understanding facial expressions. [1, 28].

Although eye tracking, in general, is widely researched it is still not completely robust for a wide range of applications. In this thesis, we present an appearance-based tracking approach with a real-time capability. This approach is specially designed for tracking

eyes under harsh environments. It is based on template tracking with a robust image feature based homography estimation for doing the template update. Additionally, the robustness is increased by applying an adaptive image blending method when updating the templates. In addition to the template tracker, a tracking by detection approach is presented. As a detector, the aggregated channel features (ACF) detector from Piotr Dollar's Matlab toolbox is used [18]. The detection results are subsequently refined to figure out the center of the pupil of the detected eye. This is done by applying a polar transform to the detected eye and finding the border between iris and lens.

1.3 Overview

This thesis is structured as follows: In section 2 an overview of general object trackers is given. In addition to that, different tracking approaches for general eye trackers are explained. In chapter 3, the theoretical foundations for this thesis are given. It starts with an insight to template matching and goes on with an explanation of the detector which is used in chapter 5. Additionally, transformations of 2D and 3D space are explained. Namely, the projective geometry of 2D space and the epipolar geometry. Other than that, the random sample consensus (RANSAC) algorithm as well as Laplacian image pyramids as discussed. In chapter 4, the template based tracking approach is presented. The initialization, as well as the different steps which are needed to perform the template tracking, are explained. Finally, the tracker is validated regarding runtime and accuracy performance. Chapter 5 explains the tracking by detection approach with its detection refinement which improves the center detection of the eye. As well as the template based tracker, the detection tracker gets validated regarding runtime and accuracy performance. Finally, in chapter 6 a detailed conclusion of the performance of the two presented tracking approaches as well as an outlook to future work is given.

2 Related Work

The next two sections (2.1 General Object Tracking and 2.2 General Eye Tracking) give an overview of techniques which are used in various tracking applications.

2.1 General Object Tracking

General object tracking can be defined as tracing the progress of objects or object features as they move around in the scene. In other words, object tracking can be described as generating the trajectory of an object over time, by locating its position in every frame of a video. Object trackers may also provide the complete region in the image that is occupied by the tracked object at every time instant. In general, object tracking is a very challenging problem, as it has to deal with difficulties like abrupt object motion, changing the appearance of the fore- and the background, non-rigid object structures, occlusion, noise in the image and camera motion, to name a few. By using some prior knowledge, some of these problems can be simplified. Based on the findings of [74], general object tracking can be divided into three main categories: point tracking, kernel tracking and silhouette tracking (Figure 2.1). In addition to that, representative work for each category is shown in Table 2.1.

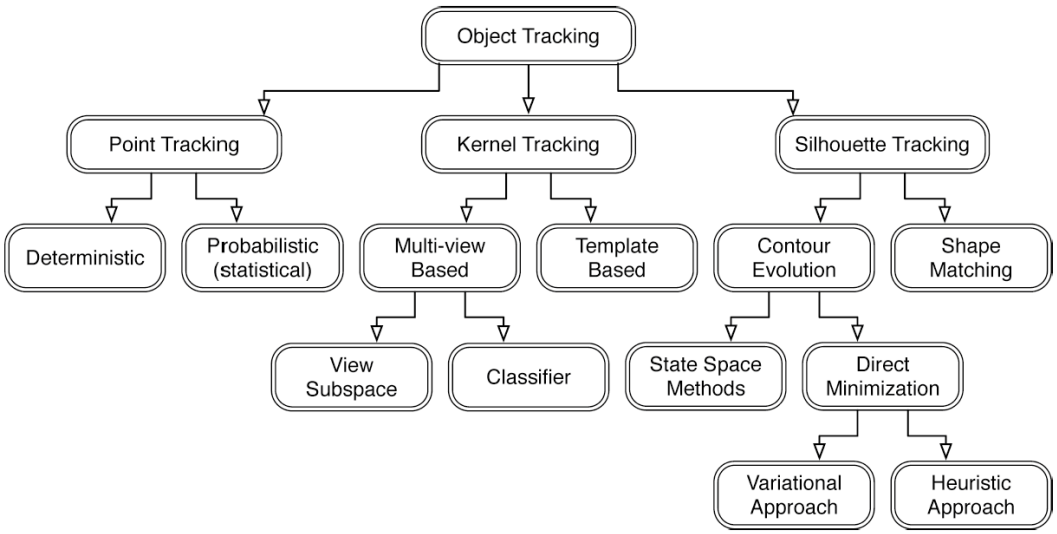


Figure 2.1 Classification of general object tracking methods based on [74]

The selected object representation can limit the type of motion or deformation an object can undergo. If an object is represented as a point, only a translational model can be

applied. When a geometric shape representation, like a rectangle or an ellipse, for the object representation is used, parametric motion models like affine or projective transformations are appropriate. For non-rigid objects, silhouette or contour is the most descriptive representation and both parametric and nonparametric models can be used to specify their motion.

Classification	Demonstrative Work
Point Tracking	
Deterministic Methods	• Feature Point Correspondence in the Presence of Occlusion (MGE Tracker) [58] • Resolving Motion Correspondence for Densely Moving Points (GOA Tracker) [71]
Statistical Methods	• Estimation of Object Motion Parameters from Noisy Images (Kalman Filter) [10] • Tracking and Data Association (JPDAF) • Maximum Likelihood Method for Probabilistic Multihypothesis Tracking (PMHT) [66]
Kernel Tracking	
Template and Density Based Appearance Models	• Kernel-Based Object Tracking (Mean-Shift) [17] • Good Features to Track (KLT) [62] • Object Tracking With Bayesian Estimation of Dynamic Layer Representation (Layering) [68]
Multi-View Appearance Models	• EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation (Eigentracking) [8] • Support Vector Tracking (SVT) [3]
Silhouette Tracking	
Contour Evolution	• Condensation - Conditional Density Propagation for Visual Tracking (State Space Models) [36] • Morphing Active Contours (Variational Methods) [7] • Region-Based Strategies for Active Contour Models (Heuristic Methods) [57]
Matching Shapes	• Tracking Non-Rigid Objects in Complex Scenes (Hausdorff) [35] • Temporal Spatio-Velocity Transform and its Application to Tracking and Interaction (Hough Transform) [60] • Object Reacquisition Using Invariant Appearance Model (Histogram) [39]

Table 2.1 Representative work of general object tracking methods based on [74]

Following a brief introduction to the main tracking categories will be given:

Point Tracking

In the point tracking approach, detected objects are represented by points. The association of points is based on the previous object state which can include object position and motion. For this approach, an external mechanism to detect the object in every frame is necessary. Figure 2.2 (a) shows an example of object correspondences [74].

Kernel Tracking

Kernel tracking refers to object shape and appearance, whereby the kernel, which represents the object, is, in general, a primitive object region, like a rectangle or an ellipse. Objects are tracked by computing the motion of the kernel in consecutive frames (Figure 2.2 (b)). This motion is usually a parametric transformation such as translation, rotation or affine transformation [74].

Silhouette Tracking

Some interesting objects for tracking (e.g. hands, heads, shoulders, …) cannot be well described by simple geometric shapes. Silhouette tracking methods use the information encoded inside the object region and tracking is performed by estimating the object region in each frame. Therefore they provide an accurate shape description for more complex objects. Given the objects models, silhouettes are tracked by either shape matching or contour evolution (Figure 2.2 (c)) [74].

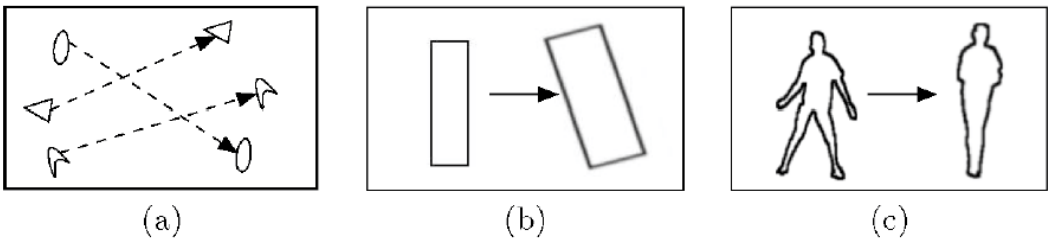


Figure 2.2 Different tracking approaches ((a) multipoint correspondences, (b) parametric transformation of a rectangular patch, (c) contour evolution) (adapted from [74])

As point tracking is not advanced enough and the eye can easily be described by a simple primitive object, kernel tracking is the most promising approach for our eye tracking approach. Therefore this approach will be discussed further in the following section.

2.1.1 Kernel Tracking

In kernel tracking approaches, the motion of the object is typically computed in the form of a parametric motion like translation, rotation or affine transformation. The object is represented by primitive object regions like a rectangle or an ellipse. Kernel tracking algorithms differ in terms of the used appearance representation, the number of tracked objects and the method for estimating the object motion. Based on the used appearance representation two subcategories can be defined, namely, template and density-based appearance models and multi-view appearance models.

Template and Density-Based Appearance Models

The most common approach in this category is template matching. Template matching is a brute force method, wherein an image I the object template O is searched. The object template is defined via the previous frame. The position of the template in the current frame is computed by a similarity measure (e.g. cross correlation). The template usually is formed by image intensity, color features or image gradients. Due to the brute force search, template matching can be quite cost intensive. To reduce the computational costs the search window for the object can be reduced to the surrounding of its previous position [52, 74].

In [17] they use a weighted histogram computed from a circular region to represent the object. Instead of a brute force method for locating the object, a mean shift procedure is used (further details on the mean shift algorithm can be found in [14, 16]). The appearance similarity is iteratively maximized by the mean shift tracker, by comparing the histograms of the object Q and the hypothesized object location P. As a similarity measure, the Bhattacharyya distance is used. At each iteration, the similarity is increased and this process is repeated, until convergence is reached. In practice, this takes about 4 iterations.

An online template update was first introduced by [38]. The generative model for the appearance is modeled as a mixture of three components. The stable appearance component which is learned with a relative long time-course, a two-frame transient component and a noise process. The stable component identifies the most reliable appearance for motion, that is, the region of the object whose appearance does not quickly change over time. Quickly changing pixels are identified by the transient component and outliers that arise due to noise are handled by the noise process.

Multi-View Appearance Models

In the previous tracking method, the model representation is gathered online from the most recent observation. One issue is, that the object may appear different from different views. This means, if the object view changes during tracking, the appearance model may not be valid anymore. To overcome this problem, the object can be offline learned from different views and afterward used for tracking [74].
In [8] they present a view based approach for tracking rigid and articulated objects which rely on eigenspace techniques. For computing the affine transformation from the current image of the object to the image reconstructed, eigenvectors are used. By using a robust formulation of subspace matching they showed, that they can track objects over a long time, in which the object can undergo affine image motion and changes of view.
In [3] they propose a support vector tracker (SVT), which fuses an optic-flow-based tracker with a support vector machine (SVM) classifier. SVM is a general classification scheme, which finds the best separation between two classes. For SVT, the positive class consists of images of the object to be tracked, while the negative class consists of all things that should not be tracked. To estimate the position of the object, the SVM classification score is optimized.

2.2 General Eye Tracking

As the eyes and their movements play an important role in expressing a person's desires, needs, cognitive processes and emotional states a lot of investigations have been done to create human-computer interfaces, attentive user interfaces and understanding human affective states. As described in [1, 28] general eye tracking can be further divided into shape based, feature-based and appearance based tracking approaches. In addition to that, there are tracking approaches with active infrared illumination.

2.2.1 Shape-Based Tracking

One very common approach in eye tracking is to find the location of the iris or pupil based on their circular shape. Also, the exterior shape of the eye (e.g. the eyelids) can be used to improve the tracking results [1]. As said before, many eye trackers only detect and track either the iris or pupil. Depending on the viewing angle, iris and pupil appear as an ellipse and therefore simple elliptical shape models can be used for eye tracking. Such simple methods are not capable to deal with the variations of eye features like eyelids, eye corners and eyebrows. Therefore more complex shape models can be used which allow a more detailed modeling of the eye shape [28].

In [41] they present a longest line detection (LLD) algorithm to obtain the center of the pupil. The algorithm assumes, that the pupil is of arbitrary circular shape. Then the longest vertical and horizontal line of the pupil is found and the center of the longest vertical or horizontal line represents the center of the pupil. The accuracy of the algorithm is influenced by the pupil's shape.
A straightforward way for eye detection which relies on a circular Hough transform is proposed in [40]. The facial image is cropped to the required face region and afterward, a threshold is applied to the gradient magnitude of the cropped face image. As the iris is nearly circular, the Hough transform is used to detect the iris based on the gradient magnitudes. As many assumptions have been made for this work, it is not suitable for most real-world applications.
An approach which relies on more complex shapes is proposed by [75]. The deformable eye template consists of a circle for the iris and two parabolas for the eyelids. The eye template is fitted to the image by minimizing an energy function which is given as a combination of terms due to valley, edge, peak, image and internal potentials. For this algorithm good initialization is necessary and it has difficulties when the iris is partially hidden by the boundaries of the eye.

2.2.2 Feature-Based Tracking

In feature-based eye tracking methods, the characteristics of the human eyes are explored to figure out distinctive features. Limbus, pupil and corneal reflections are common features which are used for eye detection [28].
In [65] they first identify the face in the image with a skin color model. After the face is detected, they search for the pupils by looking for two dark regions which satisfy certain anthropometric constraints and lie within a certain area of the face. For a given situation, the dark regions can be located by applying a fixed threshold to the grayscale image. Depending on the person itself and different lighting conditions, this threshold can vary. To overcome this problem they developed an iterative thresholding algorithm. After the pupils are found, they can be tracked in the following frames by finding the darkest pixels in a small search window around the current location.
Corneal reflections are virtual images of light sources that illuminate the eye and are created by the front surface of the cornea, which acts as a convex mirror. The pupil center and the corneal reflections can be used to estimate the gaze. In [27] a general mathematical model is presented which allows estimating the gaze based on the center of the pupil and one or more corneal reflections.
In [73] an approach is presented, which eliminates the influences caused by glasses and other accessories. In the proposed scheme, the gray difference between the face, pupils

and corneal reflections are used to detect the eyes. The intensity of the pupil is usually much lower than the intensity of the reflections and therefore the eye region can then be detected as the intersection of the low-gray region and the high-gray region. At the same time reflections which are caused by glasses of accessories are removed according to their size, geometric structure and other relevant features.

2.2.3 Appearance-Based Tracking

Besides the shape, also the appearance is an important descriptor for the eye. These methods are also known as image template or holistic methods. Appearance-based methods detect and track eyes directly on the photometric appearance. These methods are independent of the actual object and are in general capable of tracking other objects than eyes too. Appearance-based methods are carried out in a spatial or transformed domain and try to overcome issues due to illumination changes. In practice, however, such techniques are only tolerant to some moderate illumination changes [28].
In [54] they generalize the eigenface approach of [70] to view-based and modular eigenspaces for detection and recognition. This view-based formulation allows recognition under varying head orientations and the modular description not only allows eye detection, furthermore other important facial features like the nose and the mouth can be detected. The eigenspaces for the different facial features are called eigeneyes, eigennoses and eigenmouths.
The eigeneye approach of [54] is extended in [34]. The initial localization of the eye is done in the eigeneye space. This initial position is refined with a Hough transform based on the edge information. Afterward, a homogeneity measure is used to eliminate invalid hypothesis. Finally, a robust method is used to select the best circle among all possible circles.

2.2.4 Tracking With Active Infrared Illumination

Methods which rely on an active light source for detecting the eye are called active light approaches, otherwise, they are called passive light approaches. Active light methods are very common for indoor eye tracking systems and most of them use near-infrared light sources to illuminate the eye. If the light source is located close to the optical axis of the camera the pupil appears very bright as most of the light is reflected by the retina. This effect is similar to the red eye effect which can appear in photography when using a flash [1, 28].

In [45] they did investigations on how the magnitude of the bright pupil response change with different individuals. There they figured out, that the brightness can change dramatically although they exposed the image to the same amount of ambient light and the viewing angle stayed the same. Based on that findings, thresholds can be set manually and will work for a lot of scenarios. But to get more robust tracking results, the threshold for bright pupil detection needs to be adaptive.

2.3 Conclusion

The literature research has shown, that template tracking looks very promising for the eye tracking approach. Templates are formed by using simple geometric shapes and it carries both, spatial and appearance information. As we assume challenging lighting conditions as well as significant object appearance changes, it is necessary to dynamically adapt the tracking template. The general eye tracking research showed, that the center of the eye can be detected by the circular shape of the iris. Using additional shapes like the eyelid is not possible, as they are not visible during the surgery. Using an active IR light source could help to robustly detect the center of the pupil, but with the current surgical device, this is not possible.

3 Background

This chapter provides an overview of the theoretical background and methodologies which are used in this thesis. Chapter 3.1 gives an insight into template matching and chapter 3.2 explains the aggregated channel features (ACF) detector which is used in the tracking by detection approach. In chapter 3.3 the projective geometry of 2D space and the epipolar geometry is explained. The random sample consensus (RANSAC) algorithm is explained in chapter 3.4. Finally, in chapter 3.5 the Laplacian image pyramid is explained.

3.1 Template Matching

Template matching is a low-level technique in computer vision and is commonly used for pattern recognition tasks. It allows identifying parts of an image that matches the given image pattern [55]. In other words, template matching is an approach which allows finding areas of an image which are similar (match) to a template image. To perform template matching, three primary components are necessary. The template, the search image and a similarity measure. Related to [11] anything fashioned, shaped, or designed to serve as a model from which something is to be made can be seen as a template. Comparing in respect of similarity is the technical understanding for matching.
The simplest approach for template matching is, to use a sliding window to find the given template in the image. Figure 6.1 illustrates this simple approach, where the template is assumed with similar scale and pose in the search image.

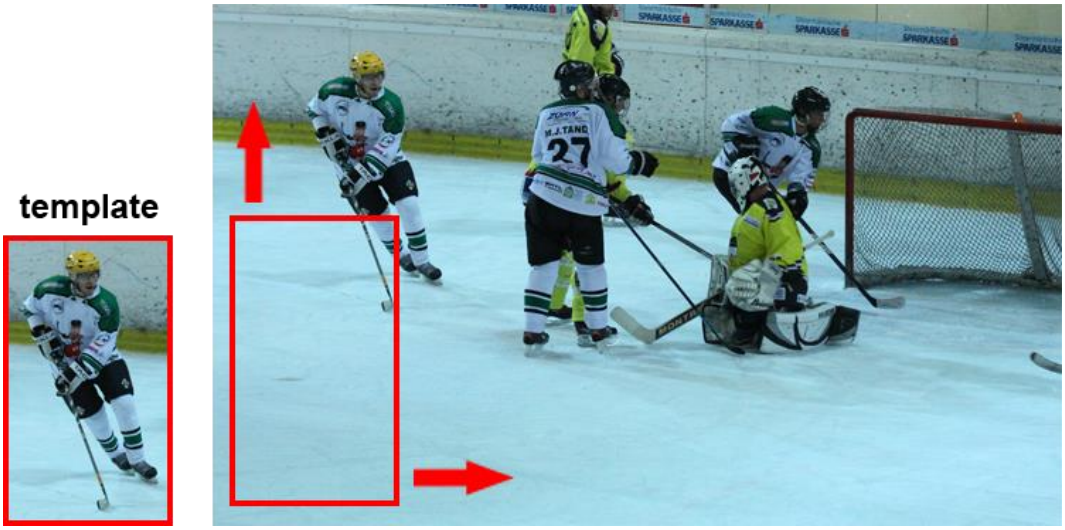


Figure 3.1 Sliding-window template matching approach

This simple template matching approach can be made more robust, by using a full search algorithm, which additionally considers 360° rotation and different scales by using a pyramid approach. Unfortunately, this is a very time-consuming process with large computational costs. Therefore, there is a need for fast and robust template matching approaches which do not rely on such an exhaustive search but deliver results of similar accuracy [31].

A template may additionally show some extrinsic, as well as intrinsic variability. The simplest extrinsic variability is a corruption by noise. Additionally, it can vary due to different illumination or different viewpoints from where the object is observed. A non-rigid deformation of the template can be seen as intrinsic variability. This non-rigid deformation can be for example intrinsic variability through physical objects (e.g. different writing styles of the letter 'a') or temporal evolution of an object (e.g. variation of the mouth while talking). Figure 3.2 shows extrinsic as well as intrinsic variability of three different template classes, 'letter', 'face' and 'mouth'. Corruption by noise, variability due to different viewpoints, object intrinsic variability as well as temporal variability is visualized there. To allow good template matching, robust similarity measures are necessary.

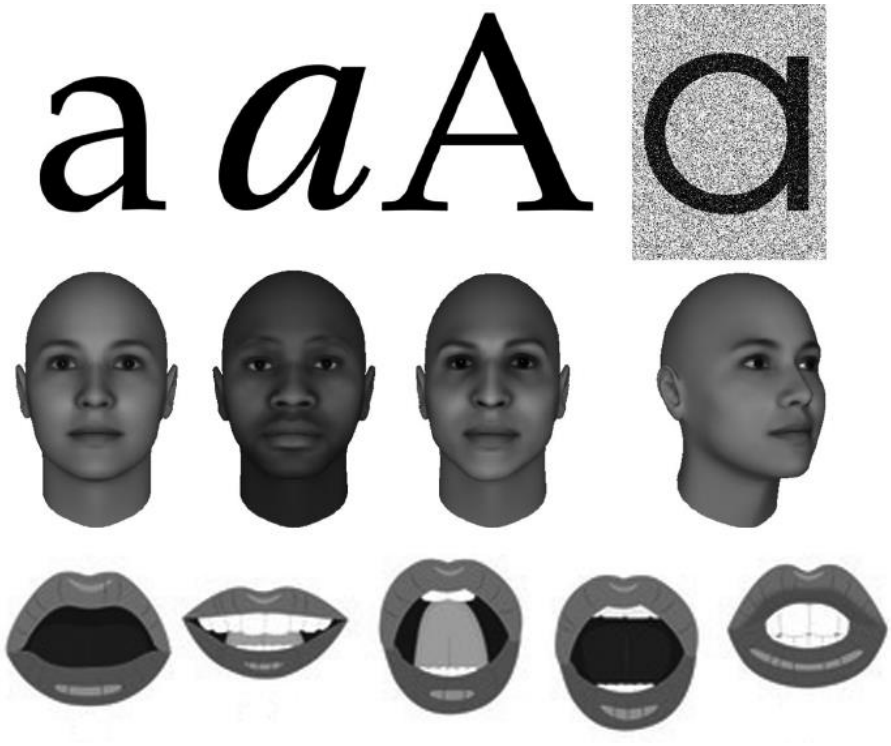


Figure 3.2 Extrinsic and intrinsic template variability for 'letter', 'face' and 'mouth' (adapted from [11, 46])

3.1.1 Similarity Measures

As already discussed before, templates can show some variability. To allow a robust template matching, robust similarity measures are necessary. A very simple way to compute the similarity between a template and an image region is to describe both as a vector $(\boldsymbol{x}, \boldsymbol{y})$ of pixel values and use a distance metric as similarity measure [11, 31]. Quite typical is the use of the sum of squared distance defined by equation 3.1 as well as the Euclidean distance or so called $L^2 - norm$ which is defined by equation 3.2.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} (x_i - y_i)^2 \tag{3.1}$$

$$d_{L2}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{3.2}$$

As a measure of match for a given distance function $d(\boldsymbol{x}, \boldsymbol{y})$ the similarity measure $s(\boldsymbol{x}, \boldsymbol{y})$ (equation 3.3) can be used. A perfect match results into $d(\boldsymbol{x}, \boldsymbol{y}) = 1$ and goes to 0 for increasing mismatches.

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{1 + d(\boldsymbol{x}, \boldsymbol{y})} \tag{3.3}$$

In addition to that, the normalized cross-correlation (NCC) is often adapted for similarity measures due to its better robustness to illumination changes and is therefore by far the most widely used similarity measure [59, 64]. In general, the normalized cross-correlation for two images $\boldsymbol{I}$ and $\boldsymbol{T}$ is defined by equation 3.4 where $\bar{I}$ and $\bar{T}$ are the mean intensity values of the corresponding images $\boldsymbol{I}$ and $\boldsymbol{T}$. Correlation values are obtained in the range of $[-1,1]$, where positive values indicate a higher similarity [31, 43].

$$NCC(\boldsymbol{I}, \boldsymbol{T}) = \frac{\sum_{x,y} (\boldsymbol{I}(x,y) - \bar{I})\,(\boldsymbol{T}(x,y) - \bar{T})}{\sqrt{\sum_{x,y} (\boldsymbol{I}(x,y) - \bar{I})^2 \cdot \sum_{x,y} (\boldsymbol{T}(x,y) - \bar{T})^2}} \tag{3.4}$$

3.2 ACF Detector

ACF stands for aggregated channel features and its detection framework is mostly described in [19]. Additional information can be found through [20, 21, 44]. The ACF detector is the successor of the integral channel features (ICF) detector which was introduced by [21]. The ICF detector computes multiple registered images channels by using linear and non-linear transformations of the input image. Afterward, features are extracted from each channel using sums over rectangular regions. Such features are referred as integral channel features. Boosting is used to train and combine decision trees over these features to distinguish between objects and background. The boosting algorithm of their choice is AdaBoost [25]. Finally, a multiscale sliding window approach with non-maximal suppression (NMS) is used to detect the object. They did a performance validation of various channels (histogram of oriented gradients, color (greyscale, RGB, HSV and LUV) and gradient magnitude) alone and in conjunction and showed that a combination of histogram of oriented gradients, LUV and gradient magnitude gives the best results. Additionally, they tested further boosting algorithms (RealBoost and LogiBoost) and they figured out, that the choice of the boosting algorithm plays almost no role in the performance.

The ACF detector is quite similar to the ICF detector. Both detectors use the same channel features as well as boosted classifiers. The key difference between the two frameworks is the feature generation. ICF uses sums over rectangular channel regions, while ACF uses pixel lookups in aggregated channels as features. Based on [19] the framework of the ACF detector is conceptually straightforward and an overview of the framework is given in Figure 3.3.

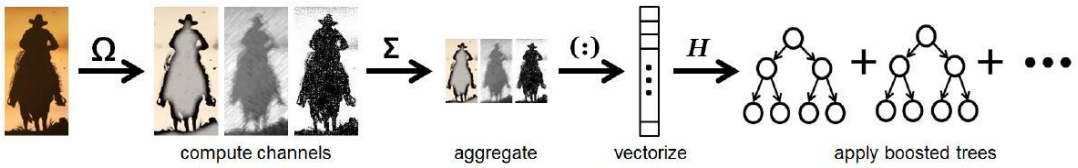


Figure 3.3 Overview of the aggregated channel feature detection framework [19]

From a given input image I several channels $C = \Omega(I)$ are computed. The channels are namely: normalized gradient magnitude (1 channel), histogram of oriented gradients (6 channels) and LUV color channels (3 channels). Prior the 10 channels are computed I is smoothed with a Gaussian filter. The channels C are divided into 4×4 blocks and every block of pixels is summed up. Finally, the resulting lower resolution channels are

again smoothed with a Gaussian filter. Features are single pixel lookups in the aggregated channels. Given a $h \times w$ detection window, there are $h/4 \cdot w/4 \cdot 10$ candidate features. Boosting with multiple rounds of bootstrapping is used to train and combine 2048 depth-2 decision trees over these features to distinguish objects from the background. As a boosting algorithm, again, AdaBoost is used. Finally, an efficient multiscale sliding-window approach is applied to do the object detection.
A Matlab toolbox [18] is provided which is easy to use. To train the detector we only need to provide a directory which contains the object images, plus an additional directory which contains the object annotations (a text file which holds the bounding box information of the object in the corresponding image).

3.3 Geometry in 2D and 3D space

The next chapters will give an overview of geometries of 2D and 3D space. Namely projective geometries in 2D space and the epipolar geometry which is a projective geometry between two views. Before these geometries are further discussed, a brief introduction to geometric primitives is given. Most of the formulations are based on [29] and also additional information to the discussed topics of this chapter can be found there.

3.3.1 Geometric Primitives

A point in the plane can be represented by a pair of coordinates (x, y) in $\mathbb{R}^2$. Considering $\mathbb{R}^2$ as a vector space, the coordinate pair (x, y) is vector. A line in the plane is given by the equation $ax + by + c = 0$, where different values for a, b and c will give different lines. Thus, a line can now be represented by the vector $(a, b, c)^T$. The vectors $(a, b, c)^T$ and $k(a, b, c)^T$ represent the same line, for any non-zero k. An equivalence class of vectors under this equivalence relationship is known as homogenous vector. A point $\boldsymbol{x} = (x, y)^T$ lies on the line $\boldsymbol{l} = (a, b, c)^T$ if and only if $ax + by + c = 0$. This can be written as an inner product of vectors where the point $\boldsymbol{x}$ is represented as a 3-vector with an additional 1 as the third coordinate. The set of vectors $(kx, ky, k)^T$ for varying values of k, all represent the same point $(x, y)^T$. Thus, just as with lines, points are represented as homogenous vectors. Therefore the point in form $\boldsymbol{x} = (x_1, x_2, x_3)^T$ represents the point $(x_1/x_3, x_2/x_3)^T$ in $\mathbb{R}^2$. As already mentioned before, the point $\boldsymbol{x}$ lies on the line $\boldsymbol{l}$ if and only if $\boldsymbol{x}^T\boldsymbol{l} = 0$. Given two lines $\boldsymbol{l}$ and $\boldsymbol{l}'$ the point of intersection is given by $\boldsymbol{x} = \boldsymbol{l} \times \boldsymbol{l}'$ where $\times$ defines the vector or cross product. Finally, the line through two given points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is defined by $\boldsymbol{l} = \boldsymbol{x} \times \boldsymbol{x}'$.
In 3-space, a point is represented as a 4-vector in homogeneous coordinates. The homogenous vector $\boldsymbol{X} = (X_1, X_2, X_3, X_4)^T$ with $X_4 \neq 0$ represents the point $(X_1/X_4, X_2/X_4, X_3/$

$X_4)^T = (X, Y, Z)^T$ of $\mathbb{R}^3$ with inhomogeneous coordinates. A plane in 3-space can be written as $\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$ or as $\pi_1 X_1 + \pi_2 X_2 + \pi_3 X_3 + \pi_4 X_4 = 0$ in a homogenous way. More concise, a plane in 3-space can be written as $\pi^T \boldsymbol{X} = 0$, which expresses, that the point $\boldsymbol{X}$ is on the plane π. A plane is uniquely defined by the join of three points (equation 3.5), or the join of a line and a point, in general position (points and lines are not collinear). Three distinct planes intersect in a unique point (equation 3.6) and two distinct planes interest in a unique line.

$$\begin{bmatrix} \boldsymbol{X}_1^T \\ \boldsymbol{X}_2^T \\ \boldsymbol{X}_3^T \end{bmatrix} \pi = 0 \qquad 3.5$$

$$\begin{bmatrix} \pi_1^T \\ \pi_2^T \\ \pi_3^T \end{bmatrix} \boldsymbol{X} = 0 \qquad 3.6$$

3.3.2 Projective Geometry in 2D

Projective geometry is the study of properties that are invariant under a group of transformations [29]. Therefore 2D projective geometry is the study of properties of the projective plane P^2 which are invariant under a group of transformation called projectivities. Projectivities are further known as collineation, projective transformation or homography. A projectivity is an invertible mapping $\boldsymbol{h}$ from points in P^2 to points in P^2 that maps lines to lines. Therefore three points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ lie on the same line only if $\boldsymbol{h}(\boldsymbol{x}_1)$, $\boldsymbol{h}(\boldsymbol{x}_2)$ and $\boldsymbol{h}(\boldsymbol{x}_3)$ do. $\mathrm{P}^2 \rightarrow \mathrm{P}^2$ is a projectivity if and only if there exists a non-singular 3×3 matrix $\boldsymbol{H}$ such that for any point in P^2 represented by a vector $\boldsymbol{x}$ equation 3.7 is true.

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{x}' = \boldsymbol{H}\boldsymbol{x} \qquad 3.7$$

The matrix $\boldsymbol{H}$ consist of nine elements with eight independent ratios. Therefore the projective transformation has eight degrees of freedom. As already mentioned projective geometry consists of a group of transformations which are defined through a hierarchy. We will introduce these transformations hierarchy starting from the most specialized one until the projective transformation is reached.

Isometry Transformations

Isometries are the simplest form of projective geometry and preserve Euclidian distance. An isometry is represented as represented in equation 3.8 where $\varepsilon = \pm 1$.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \varepsilon\, cos(\theta) & -sin(\theta) & t_x \\ \varepsilon\, sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad 3.8$$

If $\varepsilon = 1$ the isometry is orientation preserving and is therefore a Euclidian transformation. If $\varepsilon = -1$ the isometry reverse the orientation. A planar Euclidian transformation can be written in block form (equation 3.9) where $\boldsymbol{R}$ is a 2×2 rotation matrix and $\boldsymbol{t}$ is a translation 2-vector.

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0^T & 1 \end{bmatrix} \boldsymbol{x} \quad 3.9$$

Special cases of the isometry are pure rotation ($\boldsymbol{t} = 0$) and pure translation ($\boldsymbol{R} = \boldsymbol{I}$). A planer Euclidian transformation has three degrees of freedom, where one is used for the rotation and two for the translation. The transformation can be computed from two point correspondences. The invariants of the isometry transformation are length (distance between two points), angle (angle between two lines) and area.

Similarity Transformations

A similarity transformation is an isometry transformation extended by an isotropic scaling s. In the case of a Euclidian transformation extended with a scaling s, the matrix representation of the similarity transformation is shown by equation 3.10. The more concise block form is presented in equation 3.11.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\, cos(\theta) & -s\, sin(\theta) & t_x \\ s\, sin(\theta) & s\, cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad 3.10$$

$$\boldsymbol{x}' = \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ 0^T & 1 \end{bmatrix} \boldsymbol{x} \quad 3.11$$

The similarity transform is further known as equi-form transformation as it preserves ‘shape’. The planar similarity form has four degrees of freedom. The three known from

the isometry transform (length, angle, area) plus one for the scale. Two point correspondences are enough to compute a similarity transform. The invariants are the angle between lines, the ratio of length as well as the ratio of areas.

Affine Transformations

An affine transformation, also known as affinity, is a non-singular linear transform which is followed by a translation. Its matrix representation is shown in equation 3.12 and its block form in equation 3.13 where $\boldsymbol{A}$ represents a 2×2 non-singular matrix.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.12}$$

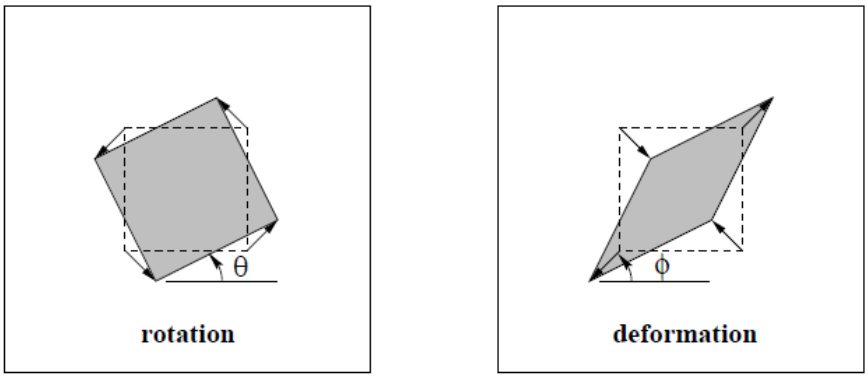
$$\boldsymbol{x'} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ 0^T & 1 \end{bmatrix} \boldsymbol{x} \tag{3.13}$$

The planar affine transformation has six degrees of freedom according to its six matrix elements. Three point correspondences are necessary to compute an affine transformation. For a better understanding of the geometric effect of the matrix $\boldsymbol{A}$, it can be split into a rotation and a non-isotropic scaling. A single value decomposition (SVD) can be used to decompose $\boldsymbol{A}$ (equation 3.14), where $R(\theta)$ and $R(\phi)$ are rotations by θ and ϕ and $\boldsymbol{D}$ is a diagonal matrix (equation 3.15) consisting of the scaling parameters λ_1 and λ_2. A schematic example of the arising distortions caused by the affine transformation is shown in Figure 3.4.

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = (\boldsymbol{U}\boldsymbol{V}^T)(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T) = R(\theta)\big(R(-\phi)\boldsymbol{D}R(\phi)\big) \tag{3.14}$$

$$\boldsymbol{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{3.15}$$

The invariants of an affinity transform are parallel lines, the ratio of lengths of parallel lines and the ratio of areas. Depending on whether $det(\boldsymbol{A})$ is positive or negative the affinity transformation is orientation preserving or orientation reversing. Since $det(\boldsymbol{A}) = \lambda_1\lambda_2$ this property only depends on the sign of the scaling factors.



(a) (b)

Figure 3.4 Distortion arising from affine transformation ((a) rotation by $R(\theta)$, (b) deformation $R(-\phi)\boldsymbol{D}R(\phi)$) [29]

Projective Transformations

A projective transformation is a general non-singular linear transformation of homogeneous coordinates. It can be seen as a generalization of an affine transformation. Its matrix form is shown in equation 3.16 and its block form in equation 3.17 with vector $\boldsymbol{v} = (v_1, v_2)^T$.

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{3.16}$$

$$\boldsymbol{x}' = \boldsymbol{H_p x} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{v}^T & v \end{bmatrix} \boldsymbol{x} \tag{3.17}$$

The matrix has eight independent ratios amongst the nine elements. Therefore the projective transformation has eight degrees of freedom, with two degrees for scaling, two for rotation, two for translation and two for the line at infinity. To compute a projective transformation four point correspondences are necessary. The only remaining invariant is the cross ratio of four collinear points. Compared to affine transformations, it is not possible to distinguish between orientation preserving and orientation reversing projectivities.

Summary of Projective Transformations in 2D

Figure 3.5 and Table 3.1 summarize the projective transformations in 2D space. Figure 3.5 visualizes the appearing distortions and Table 3.1 gives a summary of matrix representation, the degrees of freedom, the invariants and how many point correspondences are necessary to compute the desired transformation.

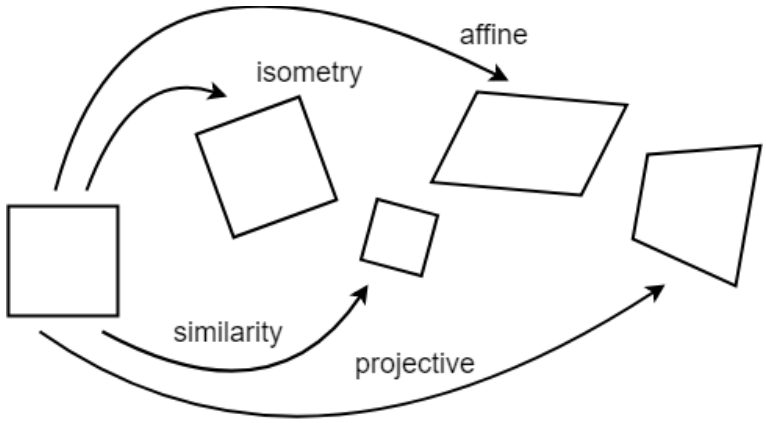


Figure 3.5 Projective transformations in 2D space (adapted from [29, 67])

Group	Matrix	DOF	Points	Invariants
isometry	$\begin{bmatrix} \varepsilon\cos(\theta) & -\sin(\theta) & t_x \\ \varepsilon\sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$	3	2	length, angle and area
Similarity	$\begin{bmatrix} s\cos(\theta) & -s\sin(\theta) & t_x \\ s\sin(\theta) & s\cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$	4	2	angle, ratio of lengths and areas
Affine	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	6	3	parallel lines, ratio of lengths of parallel lines and ratio of areas
projective	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$	8	4	cross ratio of four collinear points

Table 3.1 Projective transformations in 2D space (adapted from [29])

3.3.3 Epipolar Geometry

The epipolar geometry is the geometry of a two camera system and describes the relationship between two image planes and a 3D world point $\boldsymbol{X}$ [4, 24, 29]. It only depends on internal camera parameters and the relative pose of the cameras. A camera model is used to describe the relation between 3D world coordinates and coordinates on the 2D image plane. By introducing the world point $\boldsymbol{X}$ as a homogeneous 4-vector $\boldsymbol{X} = (X, Y, Z, 1)^T$, the image point $\boldsymbol{x}$ as a homogenous 3-vector $\boldsymbol{x} = (x, y, 1)^T$ and the camera projection matrix P as a homogenous 3×4 matrix the mapping from world to image coordinates can be written as $\boldsymbol{x} = P\boldsymbol{X}$.

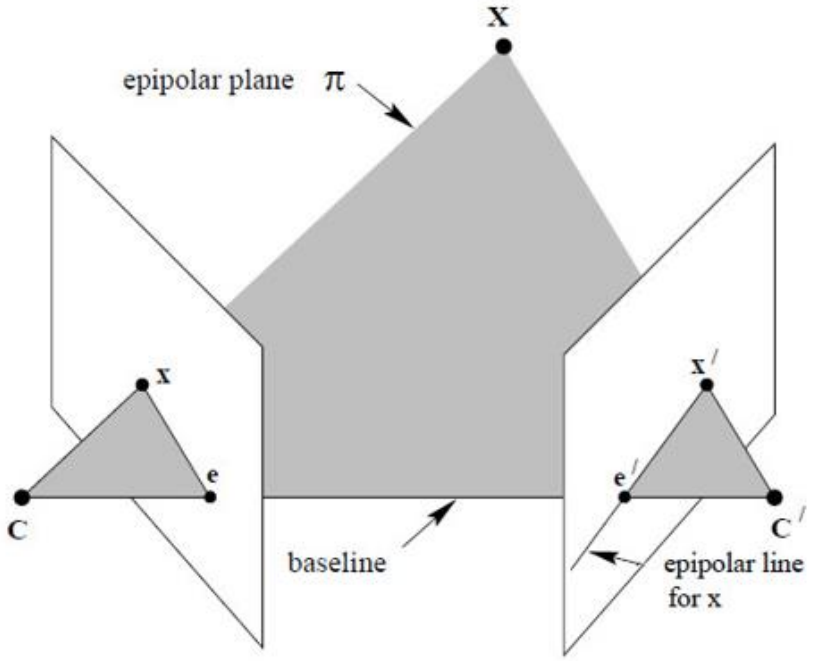


Figure 3.6 Epipolar geometry – point correspondence geometry (adapted from [29])

The 3D world point $\boldsymbol{X}$ is observed by two cameras with their optical centers $\boldsymbol{C}$ and $\boldsymbol{C}'$. $\boldsymbol{x}$ is the image of $\boldsymbol{X}$ in the first camera and $\boldsymbol{x}'$ is the image of $\boldsymbol{X}$ in the second camera. As we can see in Figure 3.6 the image points $\boldsymbol{x}$ and $\boldsymbol{x}'$, the world point $\boldsymbol{X}$, as well as the optical centers $\boldsymbol{C}$ and $\boldsymbol{C}'$ are coplanar. This means, all points lie on the same plane, which is called the epipolar plane π. This property is most significant for the search of corresponding points. The points $\boldsymbol{e}$ and $\boldsymbol{e}'$ are the intersections of the base line (line connecting the camera centres $\boldsymbol{C}$ and $\boldsymbol{C}'$) and the image planes and are called epipoles of the two cameras. In other words, the epipole is the image of the camera center of one view

in the other view. The epipolar line is the intersection of one epipolar plane π with the image plane and all epipolar lines intersect in the epipole. Therefore a point $\boldsymbol{x}$ in image one, back projected to a ray in space is an epipolar line $\boldsymbol{l}'$ in image two. The 3D point $\boldsymbol{X}$ must lie on this ray and therefore the image of $\boldsymbol{X}$ must lie on the epipolar line $\boldsymbol{l}'$. In terms of correspondence finding, the corresponding point to $\boldsymbol{x}$, namely $\boldsymbol{x}'$, has to lie on the line $\boldsymbol{l}'$.

The mathematical representation of the epipolar geometry is the fundamental matrix F. It maps a point $\boldsymbol{x}$ in one image to a line $\boldsymbol{l}'$ in the other image (equation 3.18)

$$\boldsymbol{x} \mapsto \boldsymbol{l}' \tag{3.18}$$

With a camera matrix P and a point $\boldsymbol{x}$ in the image, it is possible to determine a set of points in space which map to this point. In other words, we can do a back projection of an image point to a ray in 3D space. This ray can be formed by two know points and a line which connects them (equation 3.19). These two points are the camera center $\boldsymbol{C}$ and the point $P^+\boldsymbol{x}$. $P^+ = P^T(PP^T)^{-1}$ is the pseudo inverse of P for which $PP^+ = I$ holds.

$$\boldsymbol{X}(\lambda) = P^+\boldsymbol{x} + \lambda\boldsymbol{C} \tag{3.19}$$

The camera center $\boldsymbol{C}$ and the point $P^+\boldsymbol{x}$ are imaged by the second camera P' at $P'\boldsymbol{C}$ and $P'P^+\boldsymbol{x}$. The epipolar line $\boldsymbol{l}'$ in the second image is the line connecting these two points (equation 3.20). The point $P'\boldsymbol{C}$ is the epipole $\boldsymbol{e}'$ of the second camera. The fundamental matrix F can be computed as shown in equation 3.21.

$$\boldsymbol{l}' = (P'\boldsymbol{C}) \times (P'P^+\boldsymbol{x}) = \boldsymbol{e}' \times (P'P^+)\boldsymbol{x} = F\boldsymbol{x} \tag{3.20}$$

$$F = \boldsymbol{e}' \times (P'P^+) \tag{3.21}$$

For any two corresponding points in the two images, the fundamental matrix F holds the condition shown in equation 3.22. To compute the fundamental matrix, at least seven point pairs are necessary, but the easier way is to use eight point pairs. The algorithms for computing the fundamental matrix F are discussed in [29].

$$\boldsymbol{x}'^T F\boldsymbol{x} = 0 \tag{3.22}$$

Equation 3.23 shows the computations for the epipolar lines as well as the computation for the epipoles.

$$\boldsymbol{l}' = F\boldsymbol{x} \qquad \boldsymbol{l} = F^T\boldsymbol{x}' \qquad F\boldsymbol{e} = 0 \qquad F^T\boldsymbol{e}' = 0 \tag{3.23}$$

3.4 RANSAC

RANSAC stands for random sample consensus and was introduced by [23]. It is an iterative method to estimate the parameters of a mathematical model from an observed dataset including inliers and outliers. It is a non-deterministic algorithm that produces a robust estimation with a certain probability. This probability can be increased by allowing more iterations. Given a dataset that includes inliers and outliers, RANSAC uses a voting scheme to find the optimal result. The assumption hereby is, that noisy samples will not vote consistently for any single model and there are enough samples that agree to a good model. The algorithm basically consists of two steps, which are iteratively repeated.

1. The algorithm first selects a random sample set from the input data that is minimally required to estimate the desired model. With this minimal subsample dataset the model is estimated.
2. Secondly, the algorithm determines the number of inlier samples when the model from step 1 is applied. Any sample that lies within a defined error threshold is defined as inlier, any other sample is defined as an outlier.

This random selection is repeated until a termination condition is reached. The complete RANSAC algorithm is visualized in Figure 3.7.

Objective

Robust fit of a model to a data set S which contains outliers.

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.
(ii) Determine the set of data points S_i which are within a distance threshold t of the model. The set S_i is the consensus set of the sample and defines the inliers of S.
(iii) If the size of S_i (the number of inliers) is greater than some threshold T, re-estimate the model using all the points in S_i and terminate.
(iv) If the size of S_i is less than T, select a new subset and repeat the above.
(v) After N trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i.

Figure 3.7 The RANSAC robust estimation algorithm [29]

3.5 Laplacian Image Pyramid

An image pyramid is a multi-scale image representation technique which is commonly used in multiresolution image analysis and image compositing. For example, we want to search for an object in a scene, but do not know the actual size of the object. In this case, we will need to create a set of images with different resolution and search for the object in all images. As these images, when kept in a stack, look like a pyramid, they are called image pyramid [48].

Laplacian Image pyramids were introduced by [12] and are based on Gaussian pyramids. In a Gaussian pyramid, the original image I_0 is filtered with an Gaussian filter and subsampled to obtain the image I_1. I_1 is the so called "reduced" version of I_0. In a similar way, we form I_2 as a reduced version of I_1, and so on. Figure 3.8 illustrates the creation of a Gaussian pyramid with five levels.

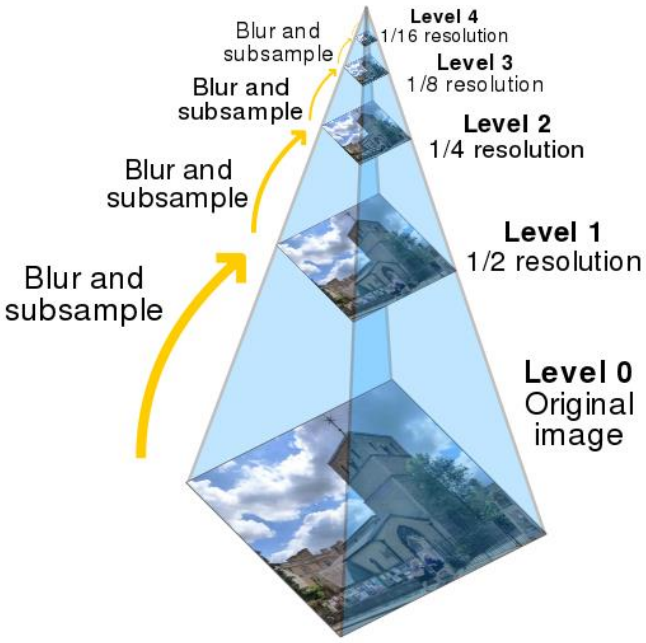


Figure 3.8 Gaussian Image pyramid creation [15]

The Laplacian pyramid now is very similar to the Gaussian pyramid. The difference is, that the Laplacian pyramid is a sequence of error images $L_0, L_1, \dots, L_N$, where each image is the difference between two levels of the Gaussian pyramid. For $0 \leq l < N$, each image for the pyramid is achieved by equation 3.24. As there is no image for I_{N+1} to create L_N,

we define $L_N = I_N$. $EXPAND$ applied to an image I_l of the Gaussian pyramid will yield to an image $I_{l,1}$ which is of the same size as I_{l-1}.

$$L_l = I_l - EXPAND(I_{l+1}) = I_l - I_{l+1,1} \tag{3.24}$$

Figure 3.9 gives an overview of how the Laplacian pyramid is generated. The Gaussian images of the middle row are obtained by expanding the image of the Gaussian pyramid. Each level of the Laplacian pyramid is the difference between the corresponding and the next higher level of the Gaussian pyramid.

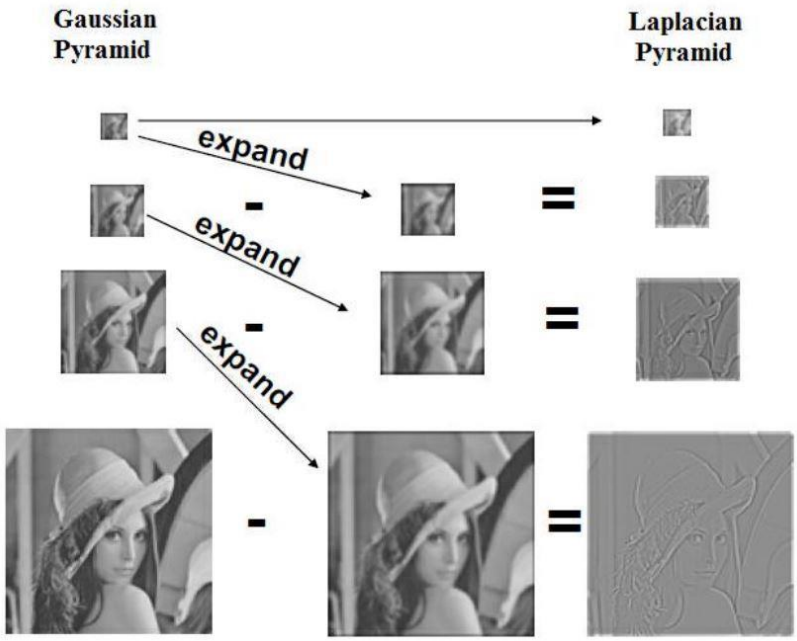


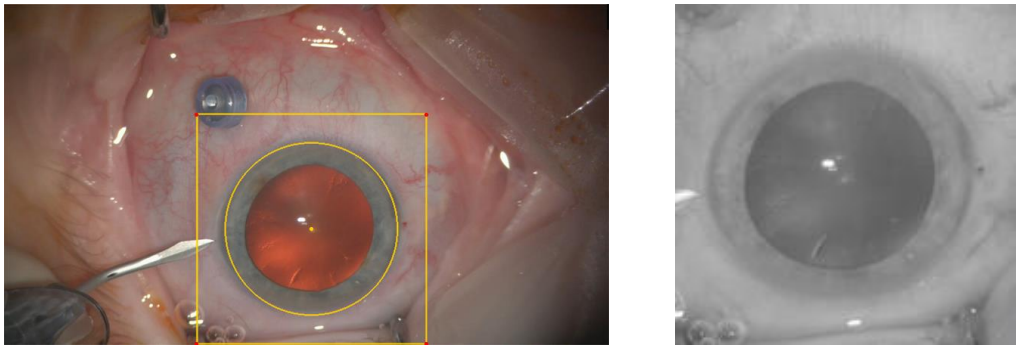
Figure 3.9 Creation of the Laplacian pyramid from expanded Gaussian pyramid images [42]

4 Template Based Tracking

In this chapter, we present an appearance-based tracking approach that is real-time capable. This approach is specially designed for tracking eyes under harsh environments. Thereby we assume x and y movements of the capture device, scale changes due to zooming, illumination changes, 3D motion of the eye, reflections caused by water or blood, appearance change of the lens caused by the lens removal and the artificial lens implantation, coverage of the eye by the surgeon or surgical instruments and non-rigid deformation of the eye caused by the surgical procedure itself. As mentioned in [31] the tracking update becomes more difficult for dynamic scenes, therefore the tracking template gets dynamically adapted to these changes. The fundamental concepts for this tracker are based on [31] and are namely robust homography estimation and adaptive image blending for the template update.

4.1 Tracker Initialization

At the beginning of the surgery (first frame of the surgery sequence), the tracking template needs to be manually initialized. As the tracker was only tested with recorded data with no information about the astigmatism axis, this is done with one click into the center of the eye and another on the border of the iris. When the proposed tracker is used during surgeries with the real surgical device, this initialization routine needs a slightly different interface which needs to be optimized for the surgeon needs. Figure 4.1 (a) shows how the template is marked in the original image and Figure 4.1 (b) shows the extracted tracking template which is a rectangle with a fixed size in which the eye is centered.



(a) (b)

Figure 4.1 Template tracker initialization. ((a) template marked in the original image, (b) extracted tracking template with fixed size and centered eye)

The tracking template is obtained by transforming the marked image region from the image coordinate frame I (ImageFrame) to the predefined and rectified template coordinate frame T (TemplateFrame) by applying a projective transformation H_A. As already discussed in the chapter Projective Transformations, four point correspondences are necessary to compute a projective transformation. Therefore, we compute the projective transformation H_A with the help of the four corner points of the ImageFrame I ($\boldsymbol{x}_i^I$) and the four corner points of the TemplateFrame T ($\boldsymbol{x}_i^T$) (equation 4.1). The computation itself is done with the help of the open source computer vision library OpenCV [49] and its function $getPerspectiveTransform$ [50].

$$\boldsymbol{x}^T = \boldsymbol{H}_p \boldsymbol{x}^I \tag{4.1}$$

Afterward, the received projective transformation is applied to the ImageFrame to receive the TemplateFrame ($warpPerspective$ [50] from OpenCV is used). When the perspective transformation is done, a bilinear interpolation is applied. Bilinear interpolation delivers smoother results as nearest neighbor interpolations, but is still faster as bicubic interpolation. The template extraction is illustrated in Figure 4.2. For a better understanding, a stick figure is used instead of the round eye.

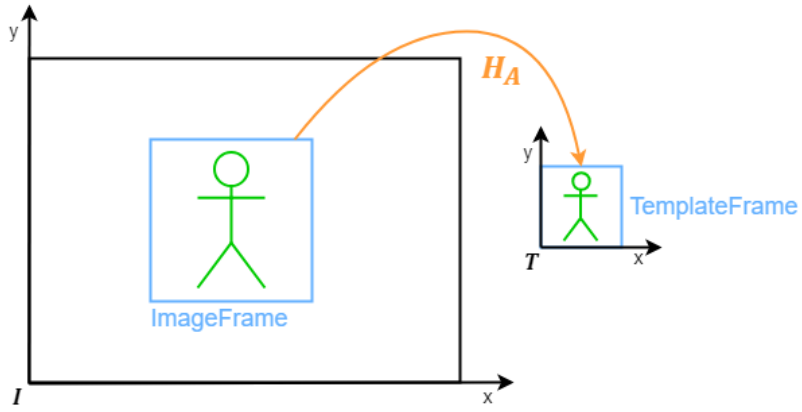


Figure 4.2 Tracking template extraction. Transformation of the image region to the template frame with the projective transformation H_A

4.2 Template Tracking

As already stated, our requirement is to robustly estimate the motion between two consecutive frames. This is done by a robust image feature based homography estimation

in the template coordinate frame T. The computed transformation then allows a computation of the recent mapping from the image to the template frame. Not only the tracking object can move from one frame to another, but also the camera system itself can move in x and y direction and the magnification can change as well. For better tracking results this movement has to be compensated. The surgery device can provide information about its current position, magnification and the pixels per millimeter ratio for $magnification = 1$. This information can be used to compute the relative movement between the current frame and the start frame. The pixels per millimeter ratio is used to converter die relative movement from mm to pixels (equation 4.2).

$$\begin{aligned} x &= pixels\_per\_mm * (x\_current_{mm} - x\_start_{mm}) \\ y &= pixels\_per\_mm * (y\_current_{mm} - y\_start_{mm}) \\ m &= magnification\_current / magnification\_start \end{aligned} \tag{4.2}$$

The relative movement can now be used to compute the transformation between the current and the start frame. The computation of the transformation matrix T is shown in equation 4.4. In this equation, w stands for the image width and h for the image height in pixesls.

$$T = \begin{bmatrix} m & 0 & \frac{w * (1-m)}{2} + m * (-x) \\ 0 & m & \frac{h * (1-m)}{2} + m * y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

To get the new image frame with the compensated camera motion, first the inverse of the last transformation T_{last}^{-1} and subsequent the current transformation T_{now} has to be applied to the old image frame (equation 4.4)

$$ImageFrame = T_{now} * T_{last}^{-1} * oldImageFrame \tag{4.4}$$

Again, the four corner points of the image frame with the compensated camera motion, as well as the four corner points of the template frame are used to compute a temporal projective transformation H_{A_temp} ($getPerspectiveTransform$ [50] from OpenCV). This temporal projective transformation H_{A_temp} can now be applied to the ImageFrame to get the temporal TemplateFrame ($warpPerspective$ [50] from OpenCV) (Figure 4.3).

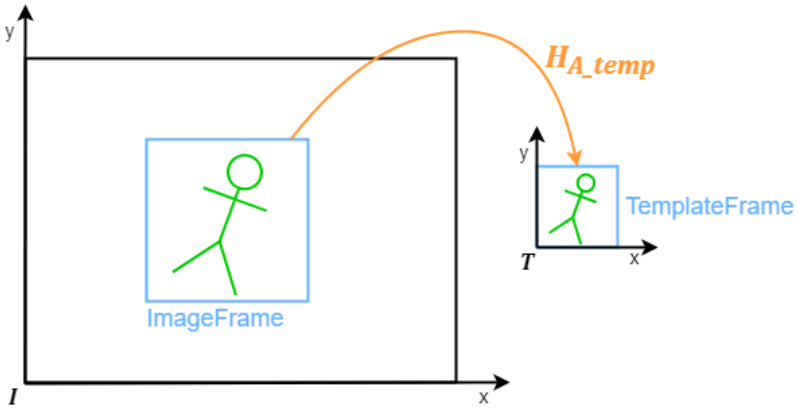


Figure 4.3 Template tracking. Temporal projective transformation H_{A_temp} between the image frame and the template frame

4.2.1 Feature Extraction

We now extract features from the previous template frame. As our aim is a robust and real-time capable eye tracker we decided to use image patches as features. We extract small and overlapping image patches along the iris, as this is the part of the eye which will stay stable over the complete surgery. The pupil will change over time as the cataract gets removed and an artificial lens is implanted and the sclera (white of the eye) can change due to bursting blood vessels. Figure 4.4 gives an overview of the feature extraction for an exemplary template. The violet squares are used to illustrate exemplarily the overlapping image patches and the green dots visualize the center of all image patches.

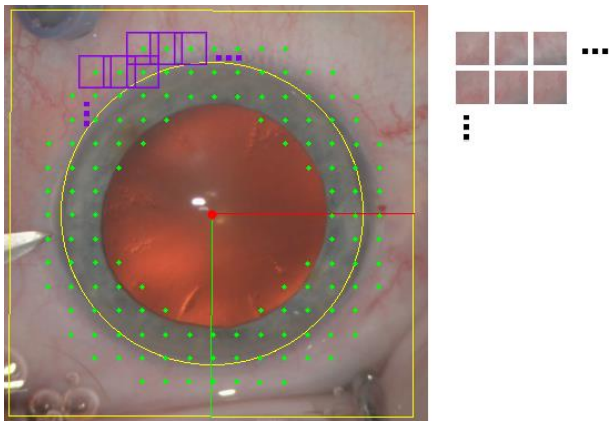


Figure 4.4 Template frame feature extraction. The patches (with the green dot as center) are circularly arranged along the iris, as the iris will stay the same for the complete surgery.

4.2.2 Feature Matching

After the features are extracted from the previous template frame, they need to be found in the current temporal template frame. For matching each feature with the temporal template frame we use the normalized cross-correlation (NCC) as a similarity measure. We rely on the normalized cross-correlation as it is not only a simple but also an effective method for measuring the similarity between two image regions. In addition to that, it is insensitive to linear brightness and contrast variations [76]. As the computation of the normalized cross-correlation can get very time consuming when applied to the whole image for every single image patch, we define a region of interest around the old feature center. This can be done due to our assumption that patches can only move in a small surrounding of their old location. Every match which exceeds a specific correlation threshold is considered as correct and saved as inlier.

4.2.3 Homography Estimation

With the feature matches which were found with the help of the normalized cross correlation, a homography can be estimated. To improve the quality and robustness of the homography we use a two-step approach. In the first step, we try to refine the feature matches by removing outliers. This is done with the help of the epipolar geometry and their mathematical representation, the fundamental matrix F. In other words, the fundamental matrix F is used as a method for pre-filtering outliers. As discussed in chapter 3.3.3, the epipolar geometry describes the relationship between two image planes and a 3D world point. We can use this concept here as our input stream is a sequence of images of a 3D object with not too much motion in between. For computing the fundamental matrix F at least eight point pairs $(x_i \leftrightarrow x'_i)$ are necessary. Any point pair which cannot fulfill equation 4.5 is considered as an outlier. Point pairs which can fulfill this equation are considered as inliers. For the computation of the fundamental matrix OpenCV with its function $findFundamentalMat$ [51] is used. As a computation method for the fundamental matrix, the RANSAC algorithm is used. Additionally, this function directly delivers the inlier point set for the found fundamental matrix.

$$x_i'^T F x_i = 0 \tag{4.5}$$

With the refined inlier set which was found with the help of the fundamental matrix, the affine transformation H_B between the previous template frame and the temporal template frame can be computed. For the computation of the affine transformation, at least three point pairs are necessary. As three point pairs are quite less to robustly estimate the

correct affine transformation, at least 40 point pairs have to be found to continue. The estimation of the affine transformation H_B is done in a robust way and relies on RANSAC. To get the new image frame from the template frame equation 4.6 can be used. The transformation pipeline from the previous template frame to the temporal template frame to the new image frame is illustrated in Figure 4.5.

$$ImageFrame = H_{a\_temp}^{-1} * H_B * previousTemplateFrame \quad 4.6$$

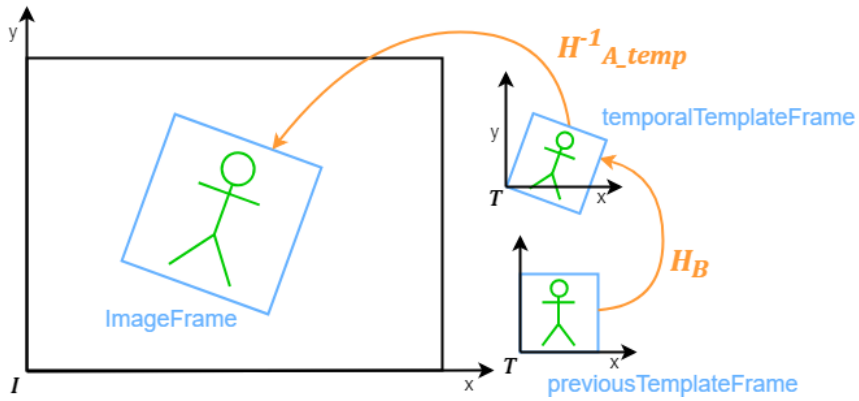


Figure 4.5 Template Tracking. Transformation pipeline from the previous template frame to the temporal template frame to the image frame

Finally the new projective transformation H_A between the image frame and the predefined and rectified template coordinate frame T can be computed (Figure 4.6) and a new template frame for the next tracking step can be extracted. Again this is done by using the four corner points of the image frame and four corner points of fixed template frame and the help of OpenCV ($getPerspectiveTransform$ and $warpPerspective$ [50])

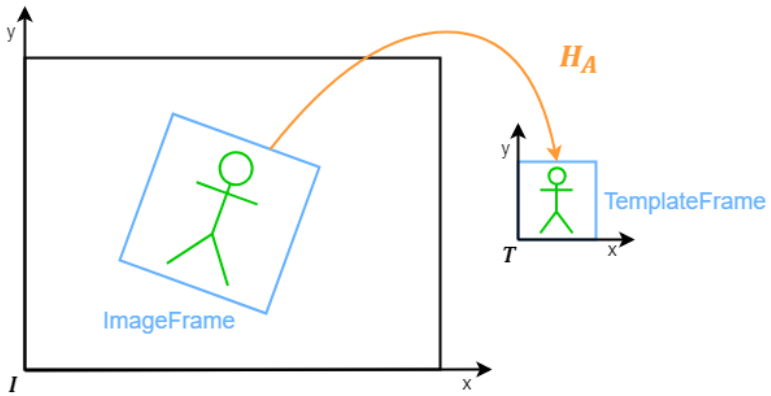


Figure 4.6 Template Tracking. Final projective transformation H_A to get the new template frame for next tracking step

4.2.4 Template Blending

As mentioned before, in our approach we use an adaptive image blending method for updating our templates. From an initial template, we adapt variations of each pixel incrementally over time. This blending approach allows handling illumination changes, object appearance variation or object pose changes while retaining the object structures [31, 38]. We use a weighting function that consists of a difference weight w_d, a quality weight w_q and a constant temporal weight w_t. The resulting pixel wise weighting function W is given in equation 4.7.

$$W(x,y) = w_d(x,y) * w_q(x,y) * w_t(x,y) \tag{4.7}$$

The difference weight w_d is defined by the smoothed and normalized absolute difference between the previous and the current template frame. By doing that, we get high weights for regions where the object undergoes changes and small weights for regions that stay the same. This will smooth out structures that do not belong to the tracked object but will also blur regions where surgical devices are used or reflections appear. To overcome this issue the quality weight w_q is introduced. This weight takes care, that only good matches get updated. In the first test, we only used inliers from the template matching to create this weighting mask. By doing that, surgical devices get not updated into the tracking template, but it can also happen that parts of the iris stop updating as no good match was found in the previously performed template matching process. Therefore, we defined higher weights for the inlier matches and smaller weights for the outlier matches. Doing so will allow surgical devices to adapt to the template frame, but also helps in recovering to the regular eye after the device was removed from the eye again. The constant temporal weight is given by a scalar t for the existing template frame and $(1-t)$ for the new observation. New observations can incorporated to the tracking template very fast by using a small value for t or can be slightly considered by using a larger value. In our testing, we figured out that for our approach best results are achieved, when the temporal weight t is defined quite high (0.8). This means, in our tracking approach, new observations are incorporated quite slow to the tracking template. The flexible image blending approach presented in [77] is extended to a multi resolution flexible blending approach and is used as a blending mechanism in this thesis. As proposed in [13] a Laplacian pyramid is used as a multi resolution method for image blending, where L_{t_i} stands for the i^{th} Laplacian pyramid level of the template T_t. The final multi resolution template blending is given in equation 4.8. Figure 4.7 shows the appearance change of the tracking template over a surgery based on the used blending approach.

$$T_t = \sum_{i=1}^{N} \frac{W_{t_i}(x,y) * L_{t_i}(x,y) + W_{t-1_i}(x,y) * L_{t-1_i}(x,y)}{W_{t_i}(x,y) + W_{t-1_i}(x,y)} \quad 4.8$$

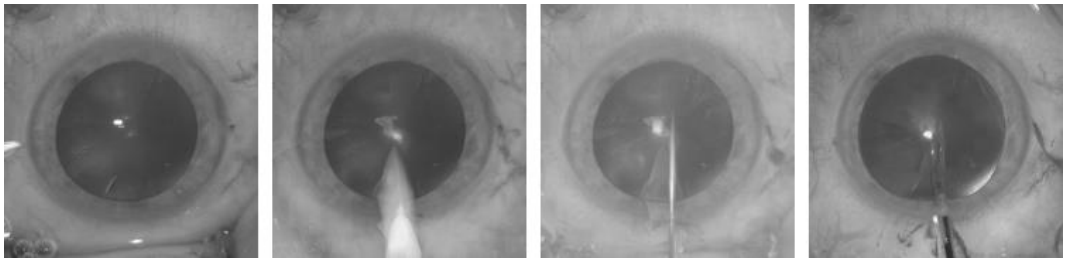


Figure 4.7 Template blending evolution. Appearance change of the tracking template through the surgery

4.2.5 Fallback

When occlusions or big appearance changes caused by the surgeon, surgical devices, water or blood occur during the surgery, it can happen that not enough good feature matches can be found to compute a correct homography. In this case, a so-called fallback is initiated. Here we try to find the complete template in the input image in a defined search window based on the previous position of the last found template. As a similarity measure, again the normalized cross-correlation is used. If no good match is found, the search window is increased with the next fallback. This procedure is repeated until the search window reaches a predefined maximum size. When the maximum size of the search window is reached, the template gets still searched with the next fallback, but the window size is not increased anymore. If the correlation value exceeds a specific threshold a good match is found and the successive fallback number, as well as the search window size, are reset to its initial values. Additionally, a correction transformation (equation 4.9) can be computed. This correction transformation defines the offset between the previous and the new position of the template. To compute the corrected image frame equation 4.10 can be used.

$$T_{corr} = \begin{pmatrix} 1 & 0 & x_{cor} \\ 0 & 1 & y_{corr} \\ 0 & 0 & 1 \end{pmatrix} \quad 4.9$$

$$ImageFrame = H_A^{-1} * T_{corr} * TemplateFrame \quad 4.10$$

In addition to figuring out the new position of the template in the image, we do a stepwise recovery of the initial template with our multi-resolution blending approach. This is done to get rid of accidentally inserted surgical tools or reflections.
In the experiment section we show, that it can happen, that the proposed fallback solution cannot find the template in the search window. To improve the fallback procedure, it is necessary to figure out if the eye is still in the search window and to provide an alternative method to reinitialize the tracker. Therefore we present an eye detection approach in chapter 5.

4.2.6 Flowchart of the Template Tracking Approach

Figure 4.7 summarizes the complete tracking approach in a flowchart.

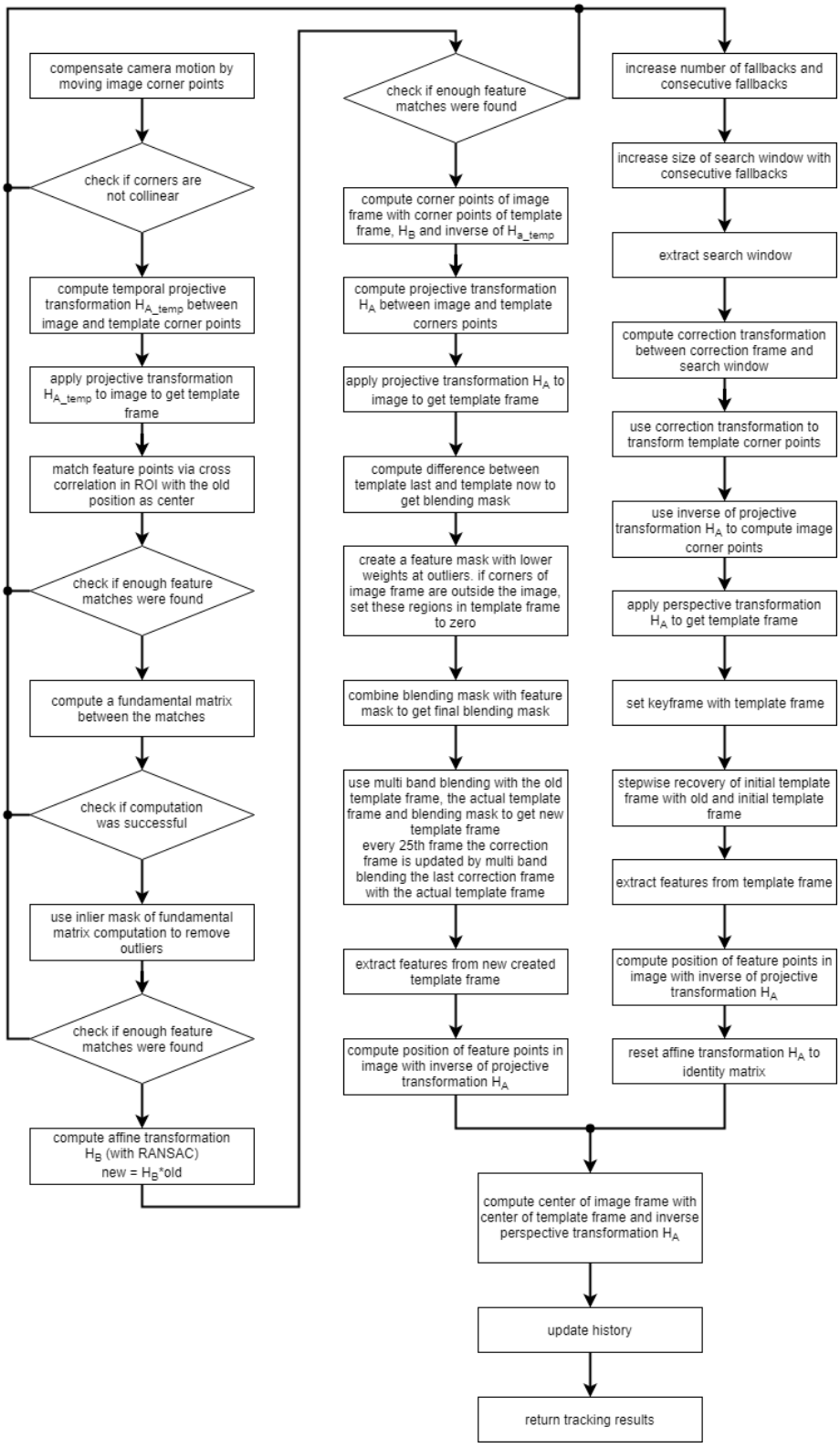


Figure 4.8 Flowchart of the complete template tracking approach

4.3 Experiments & Results

To evaluate the performance of the proposed template tracking approach different experiments were performed. In the first experiment, the runtime of the tracker and in the second experiment the position accuracy was validated.
The tracker was tested on images with an input resolution of 1920x1080 pixels and the extracted and rectified templates have a resolution of 180x180 pixels. The extracted features from the template have a size of 30x30 pixels and an overlap of 65% which leads to approximately 136 extracted features per template. To define feature matches as inliers, a correlation threshold of 0.75 has to be exceeded and at least 40 inliers have to be found to perform the homography estimation. Otherwise, the fallback routine is activated.

4.3.1 Runtime Performance Validation

The runtime performance validation was done on a benchmark dataset consisting of 3000 frames. The first input frame is used to initialize the tracker and therefore the tracking itself was only executed on 2999 frames. The runtime for the complete computation was measured and with this value, the runtime per frame and the frame rate was derived. The evaluation was done with three different modes of the tracker and on three different workstations. The hardware of the three workstations is listed in Table 4.1.
First and second mode of the tracker are thought as debug modes. In the first mode, all tracking results including text files for the pose and images for templates and visual tracking results are stored to the hard drive. The tracking visualization is active and the tracking results are printed to the console. Figure 4.9 shows the active tracking visualization. It shows the current input frame, with the tracked eye. Additional the tracking template with additional statistical information (detected features, number of inlier features, correlation value and number of fallbacks) is visualized.
Mode two only prints the tracking results to the console and shows the visualization.
Mode three is the production mode where only the necessary tracking results are written to the console. Our main goal for the tracker was to achieve a real-time capability. As we get 30 frames per second (FPS) as input frame rate and we can process more than 30 FPS with each workstation in the production mode (Table 4.2) this goal was accomplished.

Workstation	Operating System	CPU	RAM
WS 1	Windows 7	Intel Core i7-2600K @3.4Ghz	8 GB
WS 2	Windows 8	Intel Core i7-4770 @ 3.4GHz	16 GB
WS 3	Windows 10	Intel Core i5-4690 @ 3.5GHz	16 GB

Table 4.1 Hardware used for the runtime performance experiment of the template tracker

	Mode 1 [FPS]	Mode 2 [FPS]	Mode 3 [FPS]
WS 1	3.187	23.242	32.395
WS 2	5.261	23.187	34.254
WS 3	6.207	29.188	44.471

Table 4.2 Runtime performance validation for different modes of the template tracker (mode 1 and mode 2 are thought as debug modes, mode 3 is the production mode)

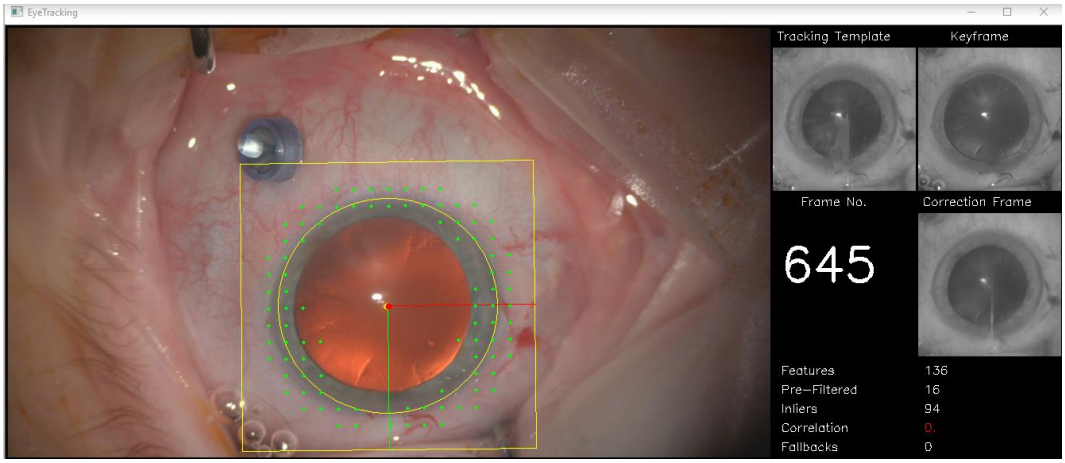


Figure 4.9 Eye tracker visualization (active track, tracking template, statistical information)

4.3.2 Accuracy Validation

To validate the accuracy of the tracker six different surgery sequences were used consisting of more than 85000 frames. All datasets include a camera and eye movement, appearance change of the lens due to the cataract removal and lens implantation, reflections caused by water and blood and occlusions caused by the surgeon or surgical

devices. Basically, all the challenges which were already discussed in chapter 1.1 are included in the tested datasets. As there was no ground truth data available the center of the eye was manually labeled in 322 frames by fitting an ellipse to the eye. As we have no medical background and there were no markers on the eye, it is really hard to mark the rotation of the eye within the surgery. Therefore the eye axis could not be labeled and furthermore, no qualitative validation of the eye axis tracking accuracy could be done. But we carefully watched all the sequences and checked that there were no unwanted or wired rotation changes. Figure 4.10 visualizes the orientation correctness for an example dataset over 2075 frames while a surgical device is moved inside the eye. Figure 4.10 (a) shows the moment when the surgical device enters the eye. Figure 4.10 (b) and (c) show different zoom levels while the device was moved around in the eye. Finally, Figure 4.10 (d) shows a bigger movement of the tool. From a visual inspection point of view, the visualized rotation vectors (red and green vector) stay correct over the shown frameset.

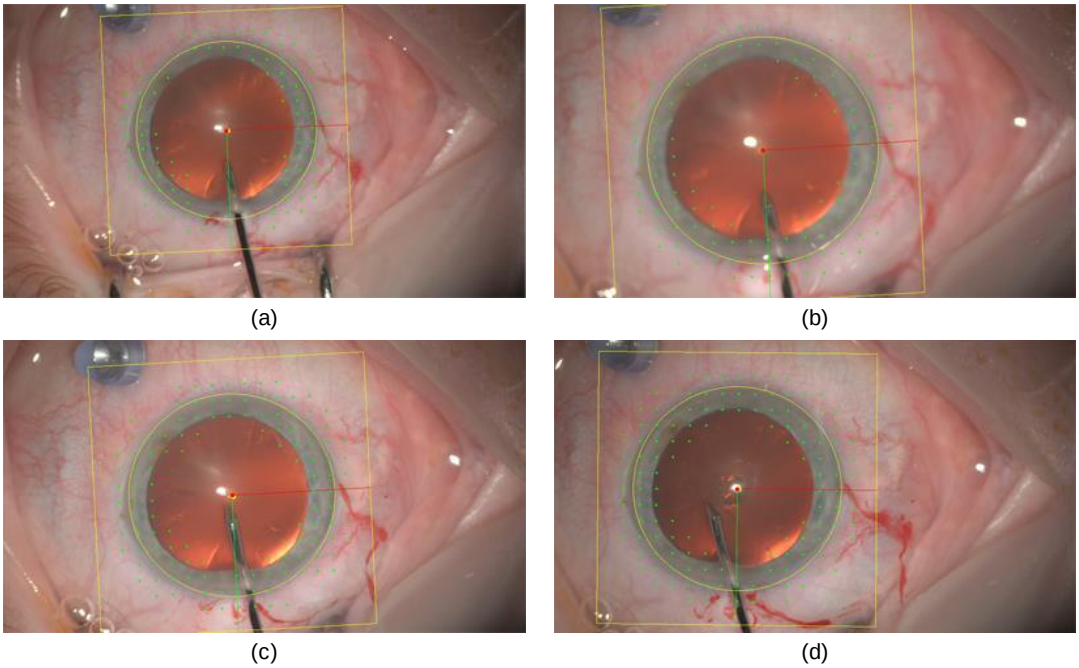


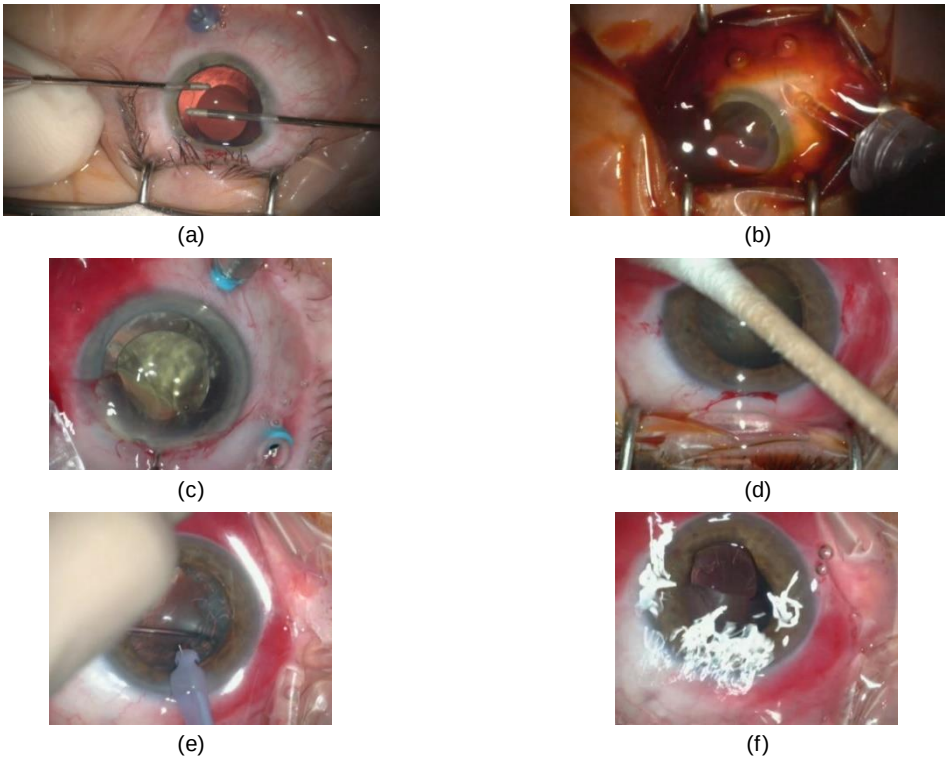
Figure 4.10 Orientation validation of the template tracker ((a) entrance of tool at frame 842, (b) different zoom level at frame 1137, (c) different zoom level at frame 1787, (d) bigger tool movement at frame 2917)

Table 4.3 gives an overview of how many images each dataset has, how many ground truth frames were extracted and how good the tracker performed on each dataset with respect to entered fallbacks. As we can see the tracker performs very well on the dataset ‘Test 1’ but performs quite bad on the dataset ‘Test 5’. This variance comes from the

quality and difficulties in each of the datasets. The test sequence ‘Test 2’ lose the track after two tools are inserted and the artificial lens is moved (Figure 4.11 (a)). Test sequence ‘Test 4’ (Figure 4.11 (b)) as well as test sequence ‘Test 5’ (Figure 4.11 (c)) lose the track after a water flush is applied. Test sequence ‘Test 6’ has a lot of fallbacks caused by different reasons. For example, a cotton swab is used to remove some blood (Figure 4.11 (d)), the surgeon, as well as some surgical tools, occlude the eye (Figure 4.11 (e)) or a water flush is applied (Figure 4.11 (f)). Occlusions, as well as big appearance changes, will decrease the quality of the feature matches and as already mentioned before, at least 40 good feature matches are necessary to perform the homography estimation. This homography is necessary for the motion estimation between two frames. If this number of good features matches can’t be achieved, no good tracking update can be guaranteed and therefore the fallback procedure is called. Depending on when the track, for how long the track and how often the track gets lost the number of fallbacks can be higher or smaller. Some fallbacks are caused by big occlusions and by definition no track can be found and other fallbacks appear because the track was lost due to bad feature matches. Nonetheless, over the entire dataset, we achieve a good track rate 58.16% and a fallback track rate of 41.84%. This means that for nearly half of the images no good track can be found. Therefore this part has to be improved by a more robust reinitialization of the tracker after the track got lost.

Test Sequence	Frames	Ground Truth Frames	Fallbacks	Good Track [%]	Fallback Track [%]
Test 1	22187	69	1373	93.812	6.188
Test 2	12431	50	6538	47.406	52.594
Test 3	3240	31	458	85.864	14.136
Test 4	767	12	422	44.980	55.020
Test 5	23264	77	15671	32.638	67.362
Test 6	23264	83	11166	52.003	47.997
overall	85153	322	35628	58.160	41.840

Table 4.3 Percentage of good and bad tracks of the template tracker



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

Figure 4.11 Reasons for fallbacks in template tracking ((a) Test 2, (b) Test 4, (c) Test 5, (d)-(f) Test 6)

Table 4.4 gives an overview of the accuracy performance of the tracker. Thereby the distance between the center of the pupil and the center of the ground truth annotated images were computed. As distance measure, the Euclidian distance and the signed distance in x- and y-direction is used. Additionally, the mean distance error, based on an approximate eye size of 679 pixels in % is given. In the best case, we achieve a mean Euclidian distance of 6.819 pixels (1.004%) and in the worst case a mean Euclidian distance of 37.395 pixels (5.507%). As we have seen in the fallback validation (Table 4.3) for some datasets a lot of frames have no good track, which means we have a high number of fallbacks. In other words, this will influence our distance validation negatively.

Test Sequence	Min Dist. [px]	Max Dist. [px]	Mean Dist. [px]	Mean Dist. [%]	Mean Dist. x [px]	Mean Dist. y [px]
Test 1	0.883	31.991	6.819	1.004	-1.395	-0.922
Test 2	0.449	79.912	17.444	2.569	-5.339	1.335
Test 3	2.947	179.993	17.956	2.644	-9.250	-10.249
Test 4	4.938	24.701	16.167	2.381	-10.132	-7.314
Test 5	2.309	179.865	37.395	5.507	-10.563	-4.573
Test 6	1.116	106.475	25.317	3.729	-1.314	-15.751
overall	0.449	179.993	22.052	3.248	-5.257	-6.400

Table 4.4 Distance from the center of the pupil of the tracker and the ground truth images

To get a better impression on the real performance of the tracker the same validation was repeated for images only, where a good track was found (Table 4.5). The numerical results are additionally visualized in Figure 4.12. By removing the fallback images from the validation we see a dramatical improvement in the results. The best mean Euclidean distance has dropped from 6.819 pixels (1.004%) to 6.294 pixels (0.927%) and the worst mean Euclidean distance from 37.395 pixels (5.507%) to 8.187 pixels (1.206%). Overall we achieve a mean Euclidean distance of 7.192 pixels. We have an average eye size of approximately 679 pixels, so the Euclidian distance of 7.192 pixels leads to a very good distance error of 1.059%. This distance error is below 1.27% which was specified in the chapter Problem Statement as the maximally allowed error. By looking at the results of mean distance in x- and y-direction we see a slightly offset into quadrant three. This may be caused by a combination of a slightly wrong initialization of the tracker (compared to how the ground truth images ware labeled) and a slightly drift during the track. Furthermore, the ground truth labels itself can hold an error.

Tes Se-quence	Min Dist. [px]	Max Dist. [px]	Mean Dist. [px]	Mean Dist. [%]	Mean Dist. x [px]	Mean Dist. y [px]
Test 1	0.883	31.991	6.294	0.927	-1.096	-0.739
Test 2	0.449	22.631	6.654	0.980	1.060	2.399
Test 3	2.947	17.826	8.105	1.194	-2.147	-3.010
Test 4	4.938	9.677	7.187	1.059	-1.609	5.750
Test 5	4.296	11.962	7.261	1.069	-6.396	-0.480
Test 6	3.357	15.285	8.187	1.206	-6.543	-1.499
overall	0.449	31.991	7.192	1.059	-3.090	-0.643

Table 4.5 Distance between the center of the pupil of the tracker and the ground truth images where a good track was found

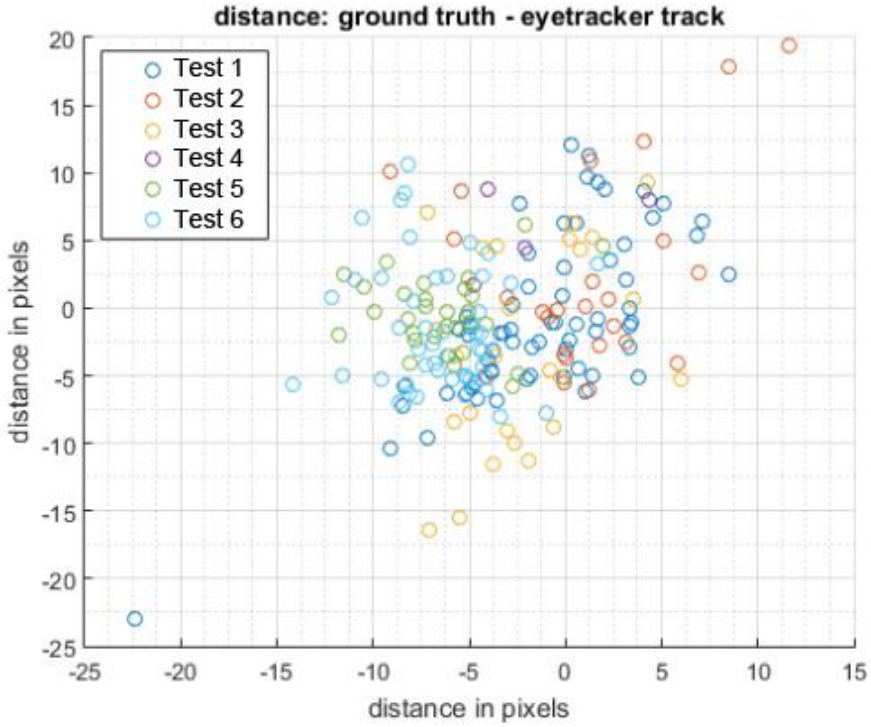


Figure 4.12 Distance between the center of the pupil of the tracker and the ground truth images where a good track was found

4.4 Conclusion

In this chapter, we introduced a template tracking approach which relies on image blending to update the tracking templates. By doing this it becomes robust against noise, appearance changes of the eye and small occlusion.

The tracker was only tested with recorded datasets where the initialization was done with one click into the center of the eye and another to the border of the iris. The astigmatism axis was always defined horizontal (0°). For future work, it is necessary to optimize this initialization step to the surgeon needs where he not only defines the center and radius of the eye but also the astigmatism axis. In the current approach, this axis was always defined in the same way, as it has no influence on the performance of the tracker.

With the runtime performance test, we approved that we can achieve more than 30 frames per seconds on different workstations in the production mode, which means, we are real-time capable. We only need to provide the tracking results to the surgical device and the device takes care of the visualization.

The rotation accuracy of the template eye tracker could not be qualitatively validated as no ground truth data was available. Additionally, the medical background was missing to label the rotation axis of the eye by hand. But carefully done manual inspection showed good rotation correctness results. For future work, it is necessary to get datasets where the eye axis is already marked with some ink or the recorded data is labeled by a qualified person like a surgeon or an ophthalmologist.

When computing the distance between the centers of the eye of good tracks and ground truth images we get a mean Euclidian distance of 7.192 pixels. With an approximate eye size of 679 pixels, this means we have a distance error of 1.059% and this error is below the required 1.27%.

When too less good feature matches are found to perform a good homography estimation the fallback procedure is activated. With the fallback validation, we figured out, that the tracker is really robust in most scenarios but cannot find its way back when too big changes in the appearance or too long occlusions appear. This means the current fallback solution is not robust enough to lead the tracker back to the correct position and continue tracking. For checking if the tracker is still tracking the eye and to make the fallback solution more robust in chapter 5 a tracking by detection approach is presented.

5 Tracking by Detection

As we have seen with the template tracking approach, it is possible that the tracking can get lost due to occlusions or too big appearance changes of the eye during the surgery. With the fallback experiment, we showed, that the basic fallback solution which tries to find the template in a search window with the normalized cross-correlation is not robust enough to lead the track back to its correct position. For a robust tracking result, it is necessary to know when the track is too far away from its expected position and to have a fallback solution to reinitialize the tracker to the correct position. To fulfill this requirement we present a tracking by detection approach.

5.1 Approach / Implementation

As a detector, the aggregated channel feature detector (ACF) from Piotr Dollar's Matlab toolbox is used [18]. The detector itself was described in [19–21, 44] and the basic background of the detector is described in chapter 3.2. The ACF detector is a fast and effective sliding-window detector which is best suited for quasi-rigid objects. To see if the detector can handle the non-rigid eye deformations which occurs during the surgery a quick test was performed. This test already showed very promising results and also confirmed that the detector is very fast and easy to use. Therefore we decided to go forward with this approach. The detector delivers a bounding box in which the eye is present. The center of this bounding box is not necessarily the center of the eye. Therefore we present a method to find the center of the eye based on the bounding box image we get from the detector.

5.1.1 Refinement of the Detector Results

As mentioned before, the ACF detector from Piotr Dollar's Matlab toolbox will deliver a bounding box which holds the eye. As shown in Figure 5.1 the center of this bounding box is not necessarily equal to the center of the eye. For this reason, the general eye detection needs to be refined to figure out the center of the eye. As discussed in previous chapters the lens will change over time. First the cataract is broken, afterward, it gets removed and finally, an artificial lens is implanted. The sclera can change its appearance due to bursting blood vessels and the border between sclera and iris sometimes has a smooth transition. The only part of the eye which stays quite constant during the surgery is the iris itself and the border between iris and lens. It seems quite natural to use the border between iris and lens to figure out the center of the eye. Therefore this edge needs to be detected. To reduce the edge detection problem to 1D space we apply a

polar coordinate transformation to the detected and extracted bounding box with the center of the bounding box as the coordinate origin. Figure 5.2 shows the polar transformed detection result.

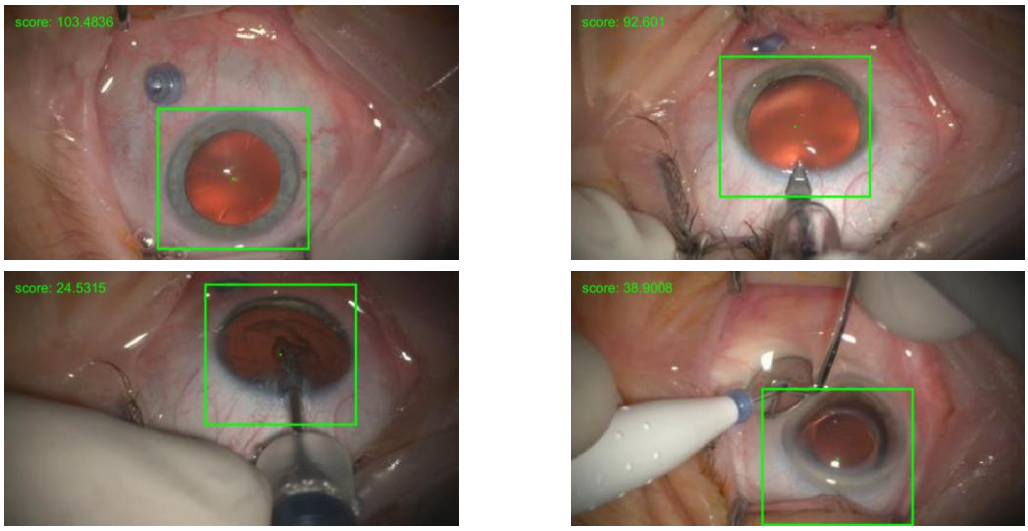


Figure 5.1 Detection results from the ACF detector. Center of bounding box is not necessarily the center of the eye

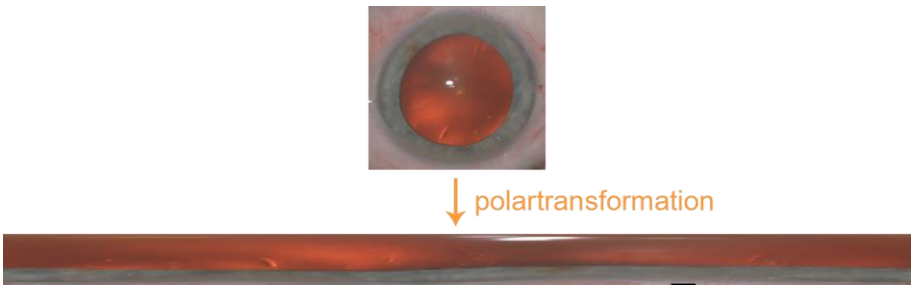


Figure 5.2 Polar transformation of the detected eye

As we already have discussed and directly can see in Figure 5.3 there are different causes which can lead to reflections on the eye. In this specific case, a water flush is applied during the surgery. These reflections will lead to unwanted edges in the edge detection and therefore they need to be detected separately and removed from the iris-lens edge detection result. This reflections can be seen as very bright spots in the gray image and can therefore very easily be detected via thresholding. Not only these bright spots, also very dark spots can cause not relevant edges and need to be removed beforehand. To make sure we get rid of most of these bright and dark spots we create a

so-called highlight mask. The initial mask which is received via thresholding is further progressed with some morphological operations. The resulting highlight mask which helps to get rid of the undesired edges for the iris-lens border detection is shown in Figure 5.4.

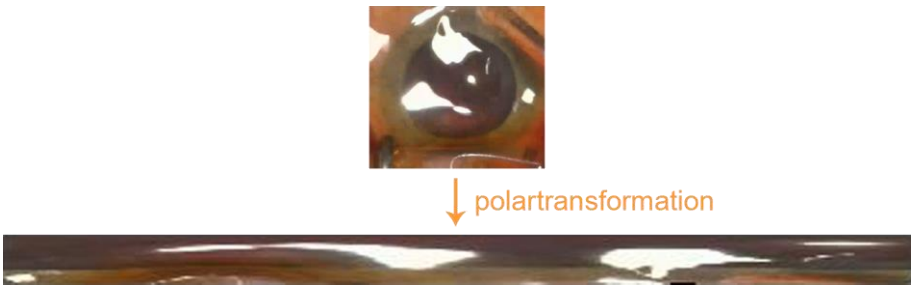


Figure 5.3 Polar transformation of the detected eye with reflections from a water flush

Figure 5.4 Highlight mask which holds bright and dark spots to improve iris-lens border detection

To figure out the border between iris and lens in a robust way for different eye appearances, the edge detection is performed in different color spaces and the individual edge detection results are combined afterward. Two different eye detections and their polar transformations in RGB, gray, H, S and V color space are shown in Figure 5.5. As we can see there, the color spaces gray, S and V hold good iris-lens boundary information and are therefore further processed. In a first step, a horizontal Sobel operator is applied to the gray, S and V image and only large gradients are extracted. In Figure 5.6 we can see the gradient images for each color channel in the first row and the extracted large gradients back-projected into the original color channel image in the second row. This already gives some good edges on the border between lens and iris. In addition, we can see, that the border seems to have the highest gradient within a small region. Therefore we apply a second step in which we extract the highest gradients within small stripes. The results back-projected into the original color channel images can be seen in the third row of Figure 5.6. The detected edges of each color channel are combined separately and some additional morphological operations are applied to get rid of some wrong detected edges. Afterward, the edges of the three different color channels are combined (Figure 5.7 (a)) and again morphological operations are applied to improve the detection and get the final edge detection result (Figure 5.7 (b)).

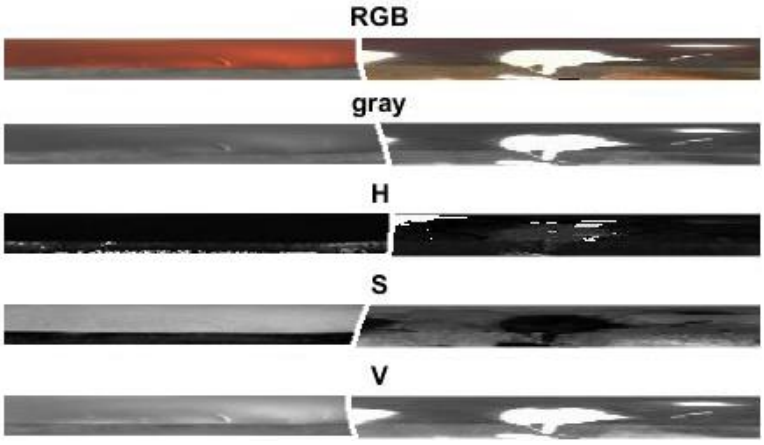


Figure 5.5 Polar transformation of two different eye templates in RGB, gray, H, S and V color space

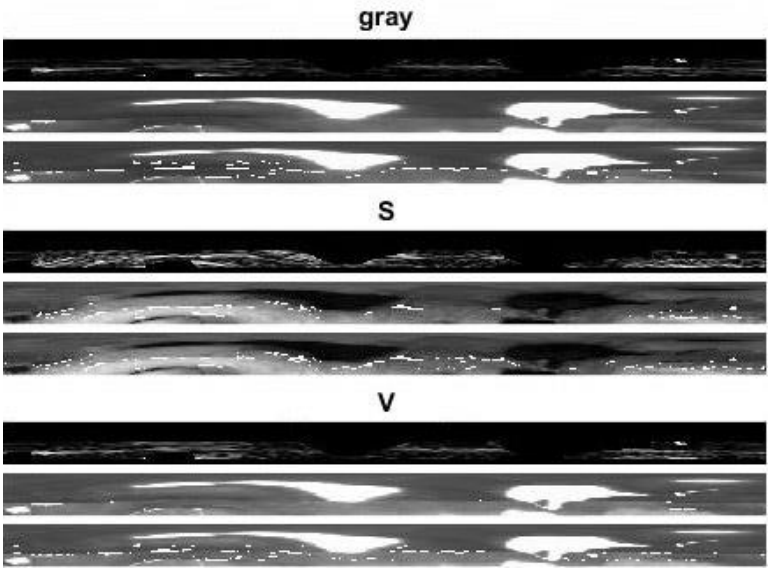
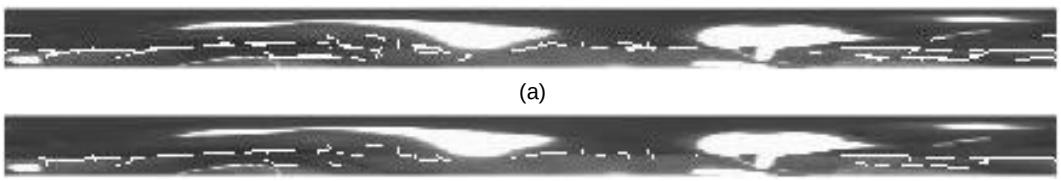


Figure 5.6 Gradient images after applying horizontal Sobel filter (per color channel: first row: gradient image, second row: large gradients, third row: large gradients within a small stripe)

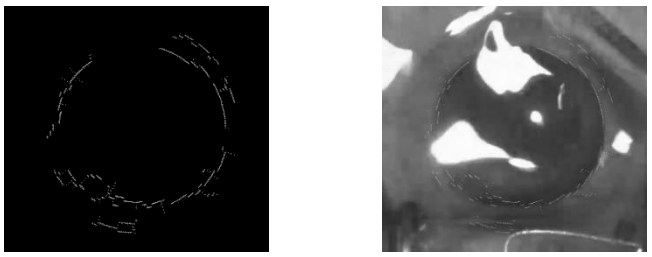


(a)

(b)

Figure 5.7 Combined gradient images (a) and final gradient image after applying addition morphological operations to the combined gradient image (b)

Now the detected edges of the polar transformed image can be back-transformed into the Cartesian coordinate image. For a better understanding of the results, the back-transformed image (Figure 5.8 (a)) is overlaid with the previously detected eye (Figure 5.8 (b)).



(a) (b)

Figure 5.8 Back transformed edges of the polar transformed image to the Cartesian image ((a) plain edges, (b) edges overlaid to the gray image)

The final step now is, to fit an ellipse to the detected points. This is done with the Matlab implementation of [63] which is based on [6, 72]. In this approach, an ellipse is fitted to a set of points by examining all major axis and getting the minor axis using a Hough transformation. By restricting the minimal and maximal length of the major axis as well as the ratio of the major and the minor axis the algorithm complexity can be reduced. For the center of the pupil, the mean center of the best three ellipses is computed. The final pupil center detection results can be seen in Figure 5.9. The best three fitting ellipses are visualized in red (1st), yellow (2nd) and pink (3rd). The mean center of the ellipses is colored blue, the ground truth center of the eye is green and the center of the bounding box is cyan.

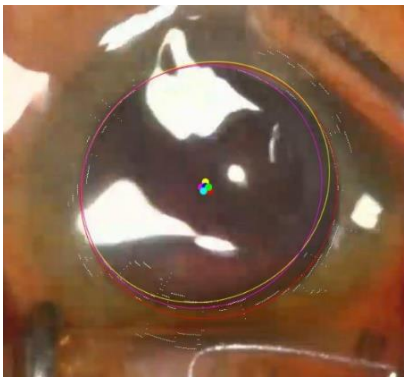


Figure 5.9 Final center detection of the pupil of the eye detector (red: best fitting ellipse, yellow: second best fitting ellipse, pink: third best fitting ellipse, blue: mean center of best three ellipses, green: ground truth center of the eye, cyan: center of the bounding box)

5.1.2 Flowchart of the Detection Approach

In Figure 5.10 a complete flowchart of the eye detection with eye refinement approach is visualized.

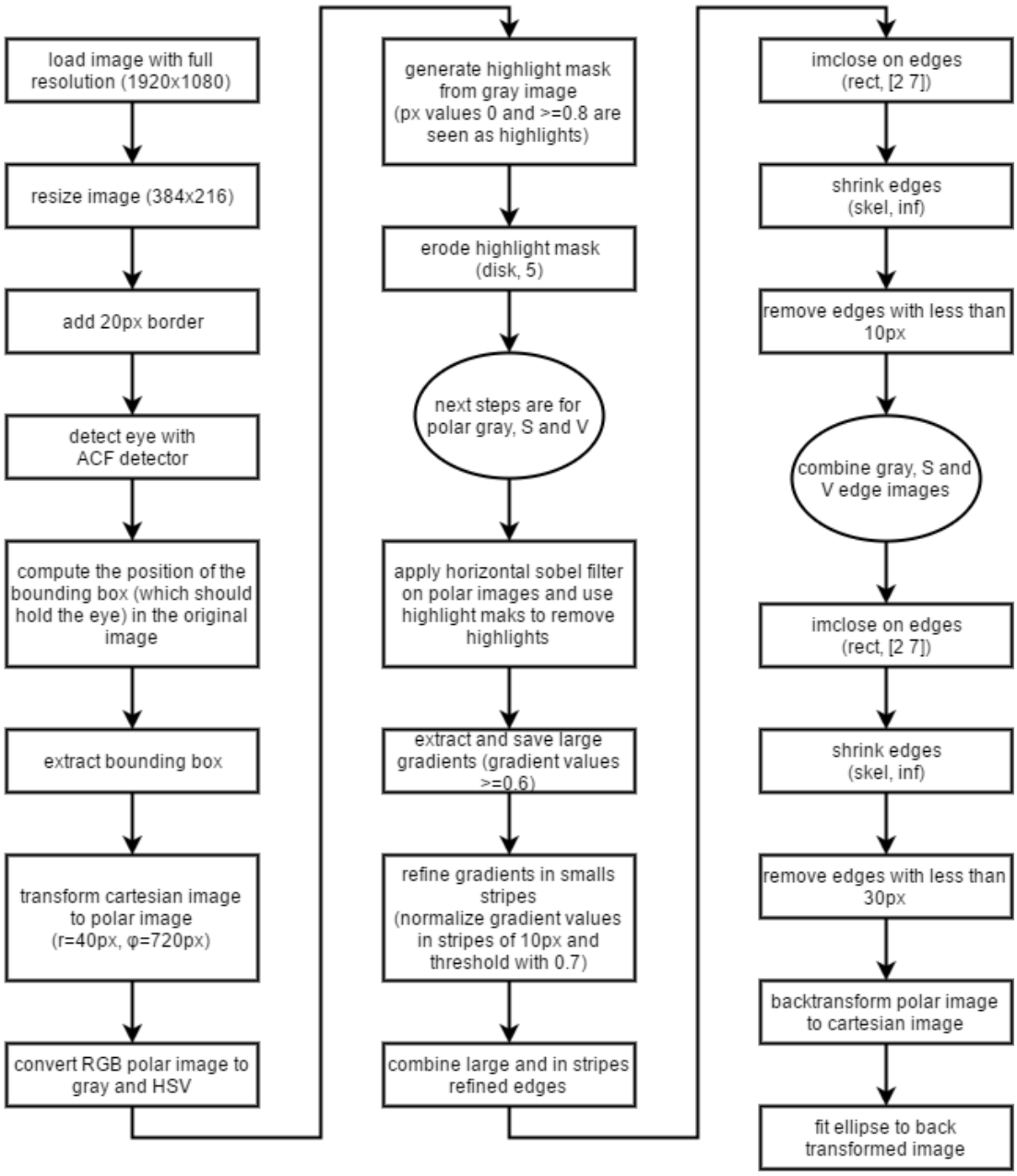


Figure 5.10 Flowchart of the complete eye detection and refinement approach

5.2 Experiments & Results

To evaluate the performance of the proposed detection and refinement approach different experiments were performed. In the first experiment general performance of the detector, in the second experiment the runtime, in the third experiment the false positive detection rate and in the fourth experiment the accuracy of the refinement is validated.

5.2.1 General Performance Validation

For a first general performance validation, 720 frames with ground truth bounding boxes were generated whereby 187 frames were used to train the detector and the remaining 533 frames were used to validate the detector. We made the decision to not use the ground truth set from the template based tracker, but to generate a new dataset to learn the tracker. This gives us the opportunity that at some later point, all newly generated ground truth frames can be used to learn the detector and compare the detection results to the same ground truth data set that was used for the template based tracker accuracy validation.

This first general performance test was performed on the full resolution images (1920x1080 pixels) as well on the 1/5 resolution images (384x216 pixels). As validation metrics, the distance of the centers of the ground truth and detected bounding box, as well as the overlap of the two bounding boxes were computed. The overlap computation is shown in equation 5.1 whereby the area of the ground truth bounding box is noted as A_{GT}, the area of the detected bounding box as A_{DT} and the area of intersection as A_I. Additionally, this is visualized in Figure 5.11.

$$overlap = \frac{A_I}{A_{GT} + A_{DT} - A_I} \tag{5.1}$$

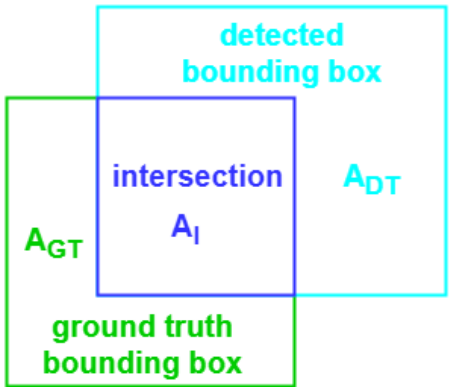


Figure 5.11 Overlap of the ground truth and the detected bounding box as validation metrics

In addition to the position of the eye, the detector additionally delivers a confidence score. The relations between distance and score (a, c) and overlap and score (b, d) for the full resolution as well as for the 1/5 resolution are shown in Figure 5.12. The associated statistical results are shown in Table 5.1. The mean distance for 1/5 resolution is with respect to the full resolution to make the results comparable. As we can see from this table, full resolution and 1/5 resolution perform nearly equally when it comes to detection rate and overlap. 1/5 resolution achieves slightly worse results regarding ground truth distance. When limiting our validation to detections with a score greater or equal to 75 we get rid of a few outliers and achieve a mean Euclidian distance of 17.5 pixels with the full resolution detector. This correlates to a distance error of 2.578% with a mean eye size of 679 pixels. With 1/5 resolution we achieve a mean Euclidian distance of 21.181 pixels which correlates to a distance error of 3.119%.

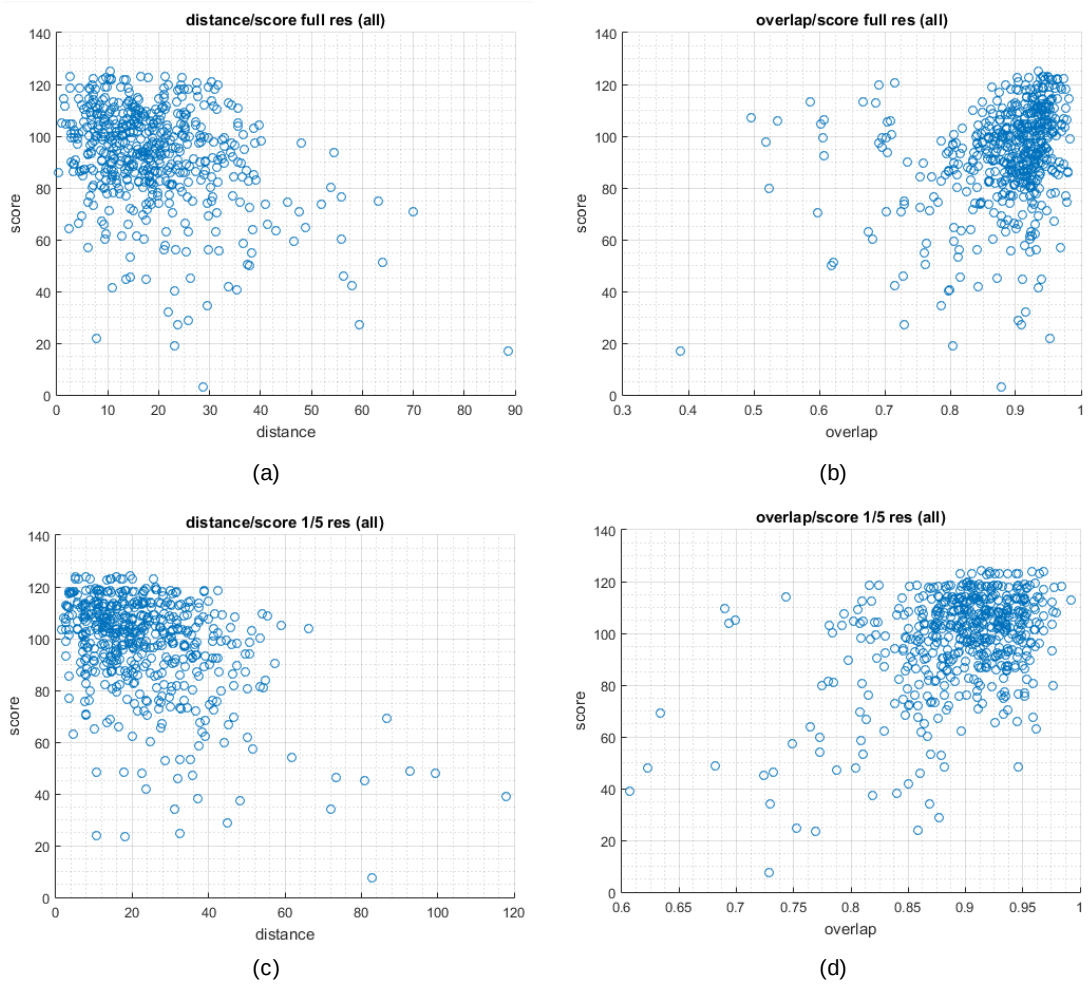


Figure 5.12 Relation between distance/score and overlap/score of the detection results for full resolution (a,b) and 1/5 resolution (c,d)

	All Scores				Score >=75			
	Mean Dist. [px]	**Mean Dist. [%]**	**Mean Over-lap [%]**	**Detec-tion Rate [%]**	**Mean Dist. [px]**	**Mean Dist. [%]**	**Mean Over-lap [%]**	**Detec-tion Rate [%]**
Full Reso-lution	18.984	2.796	88.640	99.812	17.500	2.578	89.422	84.991
1/5 Reso-lution	23.004	3.388	89.590	99.812	21.181	3.119	90.348	88.931

Table 5.1 Statistical results regarding the general performance of the eye detector

5.2.2 Runtime Performance Validation

As with the template based tracking approach the runtime performance validation was done with the same benchmark dataset consisting of 3000 frames. As there is no initialization necessary the runtime can be evaluated over the entire set of 3000 frames. The evaluation was done with two different resolutions of the input images and on two different workstations which are listed in Table 5.2. In the first experiment, the runtime performance of the detector was done with the full resolution images (1920x1080 pixels) and in the second experiment, it was done with 1/5 of the full resolution (384x216 pixels). The achieved frames per second are shown in Table 5.3. By decreasing the image resolution by 1/5 the detector can perform with more than 210 frames per second, which means, it can perform more than 25 times faster than with the full resolution which is quite obvious as the images are 25 times smaller. Also, the real-time capability is surpassed with this results. By decreasing the image resolution to 1/5, we saw that the distance error is increased by 0.541% but simultaneously we increase the detection rate by more than 25. Therefore this detection accuracy decrease is outperformed by the runtime improvement and that is why the detector for the refinement step will be run on 1/5 resolution images. When the refinement is applied the runtime goes down to approximately 2.5 frames per second. This is mostly related to a quite slow and not runtime optimized Matlab implementation, which could be improved when the program is translated to C++. Additionally, the Ellipse fitting is quite slow in its current version.

Workstation	Operating System	CPU	RAM
WS 1	Windows 7	Intel Core i7-2600K @3.4Ghz	8 GB
WS 3	Windows 10	Intel Core i5-4690 @ 3.5GHz	16 GB

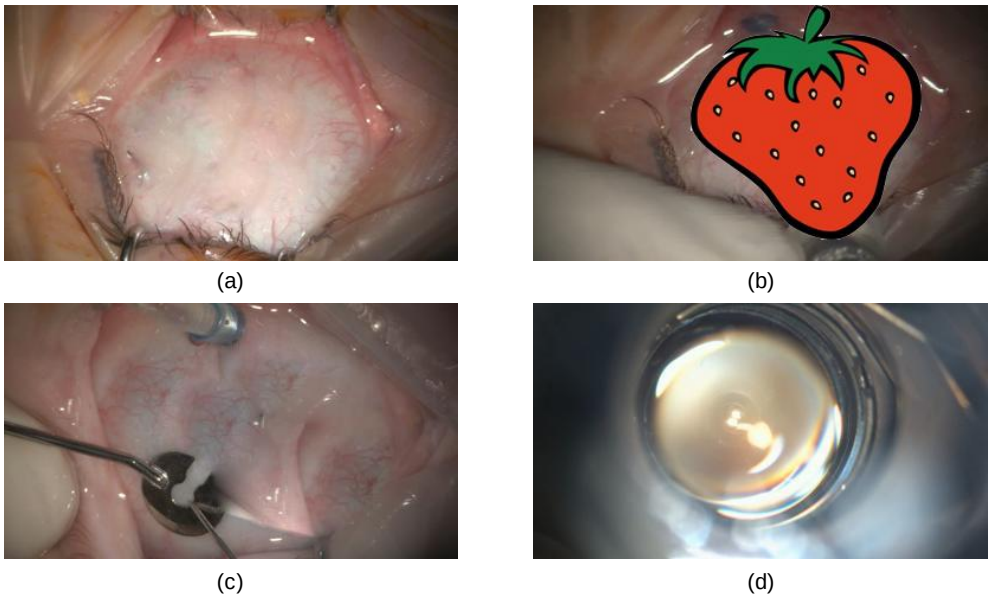
Table 5.2 Hardware used for the runtime performance experiment of the detection approach

	Full Resolution [FPS]	1/5 Resolution [FPS]	Refinement [FPS]
WS 1	7.159	210.150	2.428
WS 3	8.543	218.131	2.799

Table 5.3 Runtime performance validation of the detection approach

5.2.3 False Positive Detection Validation

For this experiment, 110 frames were created where the eye was removed or some other object was placed over the eye. Figure 5.13 shows four examples. The purpose of this experiment is to check if the detector detects an eye in situations where no eye is visible.



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

Figure 5.13 Examples of removed/modified eyes to test false positive detection rate

The false positive rate is computed as demonstrated in equation 5.2. FP stands for the number of false positives, which means an eye was detected were no actual eye was. TN stands for the number of true negatives which means no eye was detected when there was no eye. The objective is, that the false positive rate is very low. When we set the confidence threshold to 75 as we did before, we achieve a false positive rate of 2.727% with only three false positives (Table 5.4). The three detected false positives are similar to Figure 5.13 (d) which are considered as very hard test cases. There is no eye available but the structure is very similar to an eye.

$$false\ positive\ rate\ [\%] = \frac{FP}{FP + TN} * 100 \qquad 5.2$$

	All Scores	**Score >=75**
False Positive (FP)	35	3
True Negative (TN)	75	107
False Positive Rate	31.818%	2.727%

Table 5.4 Eye detector false positive detection rate

In addition to that test, an additional false positive test was performed were those removed/modified eye images were placed next to images with an eye (Figure 5.14). For all test cases, the actual eye was detected correctly.

(a)

(b)

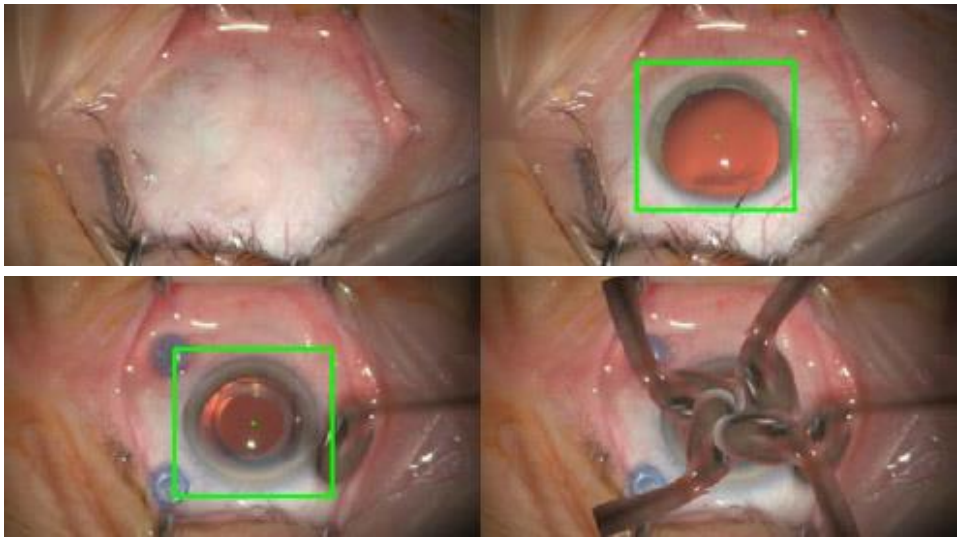


Figure 5.14 Examples of removed/modified eyes next to real eye images to test false positive detection rate

5.2.4 Refinement Accuracy Validation

In the final experiment, the accuracy performance of the refinement was validated. Therefore the detector was retrained with all 720 ground truth frames. The validation was done with the 322 ground truth frames which were already used in the template tracker validation. By doing that, the accuracy validation results of the template tracker and the detector can be compared. Table 5.5 gives an overview of the detection rates of the eye detector. When no threshold is applied we achieve a detection rate of 99.689%. When we limit the valid detections, like before, to detections with a confidence score greater or equal to 75 we still get a very high detection rate of 96.273%. The accuracy results for the detector itself are shown in Table 5.6 and for the refinement approach in Table 5.7. With the applied threshold the detector itself achieves a mean Euclidian distance of 16.329 pixels which correlates to a distance error of 2.405% for a mean eye size of 679 pixels. The eye detection refinement achieves a mean Euclidian distance of 12.011 pixels when the threshold on the confidence score is applied. This Euclidian distance leads to a distance error of 1.769%. This means the detection result is improved by 0.636%. The distance distribution of the detector results is visualized in Figure 5.15. The achieved distance error is slightly above the request distance error of 1.27% for the tracker but this is okay, as the main purpose of the detector is to check if the tracker is still on track. Additionally, this result is good enough to provide additional information to the tracker to help to improve the reinitialization process when the track got lost.

Test Sequence	Detection Rate All Scores [%]	Detection Rate Score >=75 [%]
Test 1	100.000	98.551
Test 2	100.000	100.000
Test 3	100.000	90.323
Test 4	91.667	66.667
Test 5	100.000	98.701
Test 6	100.000	96.386
overall	99.689	96.273

Table 5.5 Detection rates of the eye detector

	All Scores				Score >=75			
Test Sequ.	**Mean Dist. [px]**	**Mean Dist. [%]**	**Mean Dist. x [px]**	**Mean Dist. y [px]**	**Mean Dist. [px]**	**Mean Dist. [%]**	**Mean Dist. x [px]**	**Mean Dist. y [px]**
Test 1	19.720	2.904	-4.658	-10.528	19.214	2.830	-4.388	-9.962
Test 2	18.819	2.772	-9.830	-6.237	18.819	2.772	-9.830	-6.237
Test 3	23.598	3.475	-0.711	17.069	22.218	3.272	-1.440	16.458
Test 4	31.342	4.616	0.047	-3.686	13.652	2.011	-8.871	3.800
Test 5	11.618	1.711	-0.596	-0.545	11.217	1.652	-1.157	-0.515
Test 6	15.557	2.291	-6.314	-5.758	15.383	2.266	-6.654	-5.538
overall	17.332	2.552	-4.375	-3.332	16.329	2.405	-4.908	-3.162

Table 5.6 Distance between the center of the detected bounding box to the ground truth center of the pupil

	All Scores				Score >=75			
Test Sequ.	**Mean Dist. [px]**	**Mean Dist. [%]**	**Mean Dist. x [px]**	**Mean Dist. y [px]**	**Mean Dist. [px]**	**Mean Dist. [%]**	**Mean Dist. x [px]**	**Mean Dist. y [px]**
Test 1	9.099	1.340	0.474	-1.883	8.775	1.292	1.006	-1.668
Test 2	17.660	2.601	-7.758	1.683	17.660	2.601	-7.758	1.683
Test 3	21.867	3.220	1.518	7.113	20.107	2.961	1.885	6.314
Test 4	28.002	4.124	14.223	-13.541	9.557	1.408	1.639	-4.136
Test 5	13.498	1.988	-3.097	0.557	13.055	1.923	-3.760	0.736
Test 6	7.914	1.166	-1.239	-0.222	7.652	1.127	-1.473	0.007
overall	13.062	1.924	-1.536	0.157	12.011	1.769	-2.120	0.551

Table 5.7 Distance between the refinement result to the ground truth center of the pupil

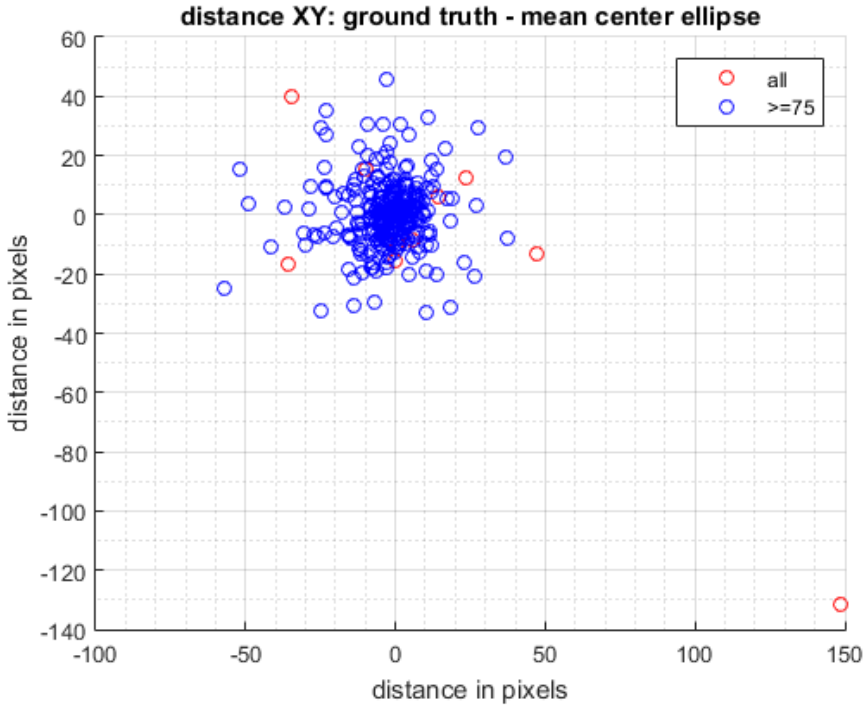


Figure 5.15 Distance distribution between the refinement result to the ground truth center of the pupil

5.3 Conclusion

In this chapter, we showed that the aggregated channel feature detector (ACF) from Piotr Dollar's Matlab toolbox [18] can be used as an eye detector for cataract surgeries. It is easy to learn, fast and accurate as well as robust against noise, appearance changes and small occlusions. Additionally, we showed a detection refinement approach where a polar transformation is applied to the detected eye to detect the border between iris and lens. The detected edge is back-transformed into the Cartesian image and ellipses are fitted to the detected border to figure out the center of the eye. One drawback of the detection approach is, that the rotation of the eye cannot be figured out.

With the runtime performance validation, we showed that by reducing the input resolution from 1920x1080 pixels to 384x216 pixels (1/5 resolution) we can achieve a detection rate of more than 210 frames per seconds for the detector itself. This is more than 25 times faster than using the full resolution images which is quite obvious as the image is 25 times smaller.

With the general performance validation, we showed that this resolution reduction has not much influence on the accuracy of the detector. When applying a threshold on the

detection confidence score of 75 we achieve a mean Euclidian distance of 17.5 pixels between the center of the detected bounding box and the center of the ground truth bounding box. This Euclidian distance correlates to a distance error of 2.578% for a mean eye size of 679 pixels. When using only 1/5 resolution we get a mean Euclidian distance of 21.181 pixels which is equal to a distance error of 3.119%. This means by reducing the resolution by 1/5 we increase the distance error by 0.54% but simultaneously increase the detection rate by 25 times.

By doing the false positive detection rate validation we showed that the detector achieves a false positive detection rate of 2.727%. This means it is highly unlikely that the detector detects an eye where no eye is. The three false positive detection we had, were very hard cases that looked very similar to an eye.

With the final refinement accuracy validation, we showed that the proposed approach can improve the detection results. We achieve a mean Euclidian distance of 12.011 pixels between the detected center of the eye and the ground truth center. This Euclidian distance correlates to a distance error of 1.769% which is an improvement of the regular detection results by 0.636%. The big drawback of the refinement is that the current Matlab implementation is quite slow. In the current state, we can only execute 2.428 frames per second. The regular detector, with a mean distance error of 2.405%, delivers already good results and can be used to check if the tracker is still on track. The better, but slower, refinement results can be used to give the tracker additional information when the track got lost to improve the reinitialization process.

6 Conclusion and Future Work

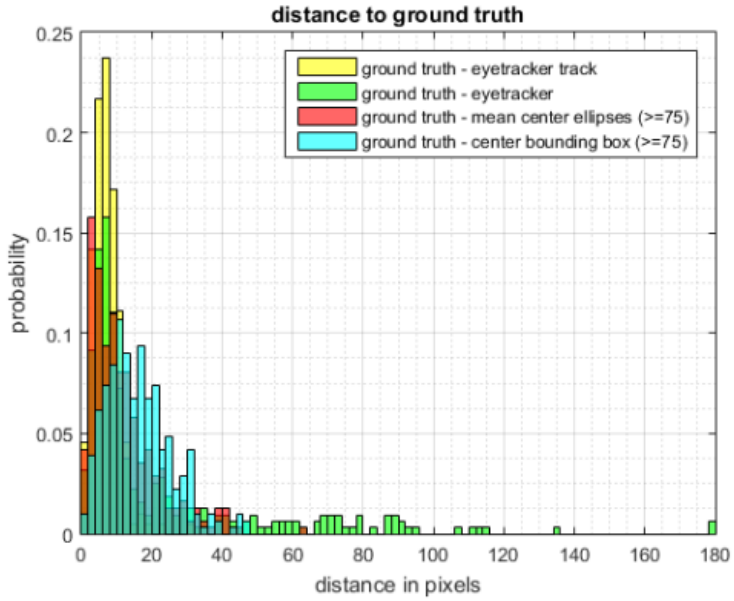
In this thesis, a very promising approach for real-time eye tracking during cataract surgeries is presented. In addition to the eye tracking, also an eye detection approach with a refined eye center detection is presented.

The eye tracker which is presented in chapter 4 is specially designed for tracking eyes under harsh environments. It is capable to deal with camera and eye motion, illumination and appearance changes as well as non-rigid deformations of the eye. The fundamentals of the tracker are template tracking and image blending to update the tracking template. With our runtime validation, we showed that the tracker can perform with more than 30 frames per second which means it is real-time capable. In the accuracy performance test, we showed that we can achieve a very good mean distance error of 1.059% (mean Euclidian distance of 7.192 pixels on an average eye size of 679 pixels) as long as we have a valid track. This distance error is below the requested 1.27%, but with the additional fallback validation, we saw, that for only 58.16% of our processed frames we can achieve a good track. This means the current fallback solution is not robust enough for all situations to lead the tracker back to the eye when it got lost because of occlusions or too big appearance changes. Therefore an additional mechanism is necessary which additionally checks if the track is still valid and helps to get the tracker back to its correct tracking position. With the current datasets, it was not possible to perform a qualitative validation of the rotation accuracy, as no marks on the eye or ground truth labels where available. Although carefully performed visual inspection showed good rotation correctness, for future work it is necessary to get correct labeled ground truth data from a qualified person, like a surgeon or an ophthalmologist, to statistically verify the correct performance of the tracker.

In chapter 5 a detection approach which is based on the aggregated channel feature detector (ACF) from Piotr Dollar's Matlab toolbox [18] is presented. The detector is easy to learn and very fast. In addition to the detector, also a refinement to detect the center of the eye is proposed. During the surgery, the appearance of the eye can change. The only part which stays quite constant is the border between the iris and the lens. This border is detected with a horizontal edge detector in the polar transformed image. The detected edges are back-transformed into the Cartesian coordinate space and an ellipse detection is applied. The runtime performance validation showed, that the detector can perform with more than 210 frames per second when the incoming image resolution is reduced by 1/5 (from 1920x1080 to 384x216). This resolution reduction can be done without any major impact on the accuracy. The detector with the applied refinement can only perform with a bit more than 2 frames per second which is caused by a not runtime

optimized Matlab implementation. When applying a confidence threshold of 75 and executing the detector on the same validation dataset which was used for validating the template tracker, we achieve a detection rate of 96.273% and a mean distance error of 1.769% (mean Euclidian distance error of 12.011 pixels on an average eye size of 679 pixels). This improvement is 0.636% better compared to the results of the general detector.

(a)



(b)

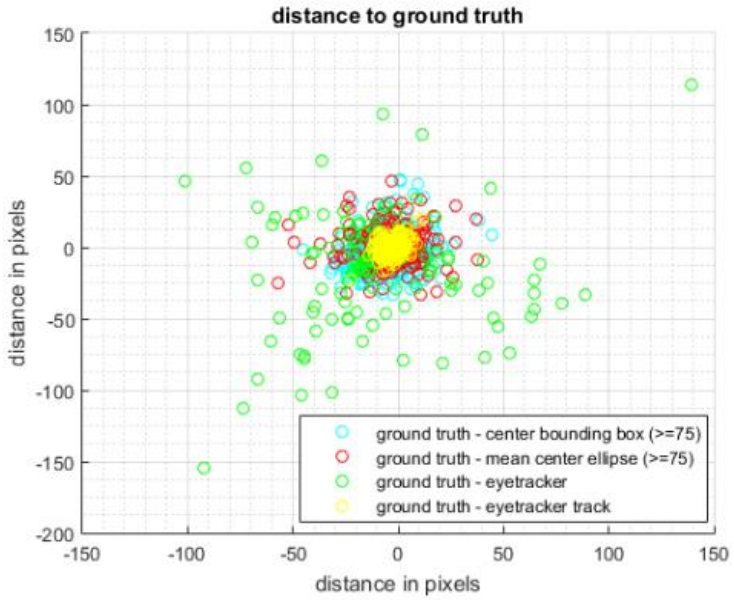


Figure 6.1 Accuracy comparison of the eye tracker and the eye detector ((a) distribution of distance to ground truth, (b) xy-distance to ground truth)

In Figure 6.1 (a) there is the distribution of the distance to the ground truth visualized for the different versions of the tracker and the detector. Figure 6.1 (b) shows the distance distribution in x- and y-direction. What we can see in these two graphs is, that the eye tracker outperforms all other versions when a good track is found (yellow). Second, best performance is achieved with the eye detector and the applied detection refinement (red). Third best results are achieved with eye detector only (turquoise). Worst results are achieved with the eye tracker only (green) which is not surprising as only 58.16% of the processed frames had a good track.

The experiments showed, that the proposed template eye tracker is real-time capable and performs very accurate as long as a good track is found. In future works, it is necessary to improve the overall performance by increasing the good detection rate and the robustness of the fallback solution. One solution could be to use the general eye detector to verify the tracker as it is very fast (≥ 210 FPS) and accurate. The detector with its refinement approach could be used to improve the reinitialization process of the tracker after the track got lost. In addition to that, it is necessary to increase the test dataset base to make sure the tracker, as well as the detector, are not biased. As already mentioned, when increasing the database it is also necessary to get ground truth data for the orientation of the eye, so that the orientation accuracy can by qualitative verified. Finally, it is necessary to get feedback from the surgeons so the tracker and the tracker initialization can be tuned to their specific needs. These are the person who will work with this tracker on a daily basis, and therefore it is absolutely necessary, that the performance is tuned to their specific needs.

Bibliography

[1] A. Al-Rahayfeh and M. Faezipour, “Eye Tracking and Head Movement Detection: A State-of-Art Survey,” *IEEE J. Transl. Eng. Heal. Med.*, vol. 1, 2013.

[2] J. J. Athanesious and P. Suresh, “Systematic Survey on Object Tracking Methods in Video,” *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 1, no. 8, pp. 242–247, 2012.

[3] S. Avidan, “Support Vector Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, 2004.

[4] P. Azad, T. Gockel, and R. Dillmann, *Computer Vision - Das Praxisbuch*, 1st ed. Aachen: Elektor-Verlag GmbH, 2007.

[5] G. Bailey and V. Thompson, “Cataracts,” *All About Vision*, 2018. [Online]. Available: http://www.allaboutvision.com/conditions/cataracts.htm. [Accessed: 26-Apr-2018].

[6] C. A. Basca, M. Talos, and R. Brad, “Randomized Hough Transform for Ellipse Detection with Result Clustering,” *EUROCON 2005 - Int. Conf. Comput. as a Tool*, vol. 2, pp. 1397–1400, 2005.

[7] M. Bertalmio, G. Sapiro, and G. Randall, “Morphing Active Contours,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 7, pp. 733–737, 2000.

[8] M. J. Black and A. D. Jepson, “EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation,” *Int. J. Comput. Vis.*, vol. 26, no. 1, pp. 63–84, 1998.

[9] R. R. A. Bourne *et al.*, “Magnitude, Temporal Trends, and Projections of the Global Prevalence of Blindness and Distance and Near Vision Impairment: A Systematic Review and Meta-Analysis,” *Lancet Glob. Heal.*, vol. 5, no. 9, pp. e888–e897, 2017.

[10] T. J. Broida and R. Chellappa, “Estimation of Object Motion Parameters from Noisy Images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 1, pp. 90–99, 1986.

[11] R. Brunelli, *Template Matching Techniques in Computer Vision - Theory and Practice*. Chichester: John Wiley & Sons Ltd, 2009.

[12] P. J. Burt and E. H. Adelson, “The Laplacian Pyramid as a Compact Image Code,” *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532–540, 1983.

[13] P. J. Burt and E. H. Adelson, “A Multiresolution Spline With Application to Image Mosaics,” *ACM Trans. Graph.*, vol. 2, no. 4, pp. 217–236, 1983.

[14] Y. Cheng, “Mean Shift, Mode Seeking, and Clustering,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, 1995.

[15] Cmglee, “Illustration of an Image Pyramid With 5 Levels,” *Wikipedia*. [Online]. Available: https://commons.wikimedia.org/wiki/File:Image_pyramid.svg. [Accessed: 04-Aug-2018].

[16] D. Comaniciu and P. Meer, “Mean shift: A Robust Approach Toward Feature Space Analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[17] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-Based Object Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, 2003.

[18] P. Dollár, “Piotr’s Computer Vision Matlab Toolbox.” [Online]. Available: https://pdollar.github.io/toolbox/. [Accessed: 11-May-2018].

[19] P. Dollár, R. Appel, S. Belongie, and P. Perona, “Fast Feature Pyramids for Object

Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2014.
[20] P. Dollár, S. Belongie, and P. Perona, "The Fastest Pedestrian Detector in the West," *Br. Mach. Vis. Conf.*, 2010.
[21] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral Channel Features," *Br. Mach. Vis. Conf.*, 2009.
[22] J. H. Farooqui, A. Koul, R. Dutta, and N. M. Shroff, "Management of moderate and severe corneal astigmatism with AcrySof toric intraocular lens implantation – Our experience," *Saudi J. Ophthalmol.*, vol. 29, no. 4, pp. 264–269, 2015.
[23] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[24] D. A. Forsyth and J. Ponce, *Computer Vision - A Modern Approach*. Upper Saddle River: Pearson Education, 2003.
[25] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, 1997.
[26] M. Godec, P. M. Roth, and H. Bischof, "Hough-based Tracking of Non-Rigid Objects," *IEEE Int. Conf. Comput. Vis.*, 2011.
[27] E. D. Guestrin and M. Eizenman, "General Theory of Remote Gaze Estimation Using the Pupil Center and Corneal Reflections," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 6, pp. 1124–1133, 2006.
[28] D. W. Hansen and Q. Ji, "In the Eye of the Beholder: A Survey of Models for Eyes and Gaze," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 478–500, 2010.
[29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge: Cambridge University Press, 2004.
[30] H. Hashemi, M. Khabazkhoob, M. H. Emamian, M. Shariati, A. Yekta, and A. Fotouhi, "White-to-white Corneal Diameter Distribution in an Adult Population," *J. Curr. Ophthalmol.*, vol. 27, no. 1–2, pp. 21–24, 2015.
[31] M. Heber, "Tracking and Visual Quality Inspection in Harsh Environments," Graz University of Technology, 2013.
[32] G. Heiting, "Astigmatism," *All About Vision*, 2018. [Online]. Available: http://www.allaboutvision.com/conditions/astigmatism.htm. [Accessed: 27-Apr-2018].
[33] G. Heiting, "Astigmatism And Cataract? A Toric IOL Can Fix Both," *All About Vision*, 2016. [Online]. Available: http://www.allaboutvision.com/conditions/toric-iols.htm. [Accessed: 27-Apr-2018].
[34] W. Huang and R. Mariani, "Face Detection and Precise Eyes Location," *15th Int. Conf. Pattern Recognit.*, pp. 722–727, 2000.
[35] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge, "Tracking Non-Rigid Objects in Complex Scenes," *IEEE Int. Conf. Comput. Vis.*, pp. 93–101, 1993.
[36] M. Isard and A. Blake, "Condensation - Conditional Density Propagation for Visual Tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, 1998.
[37] R. Jain, S. Aggarwal, and A. Dokania, "A Clinical Study to Compare the Accuracy of Digital and Manual Marking for Toric IOL Alignment," *Int. J. Contemp. Med. Res.*, vol. 4, no. 1, pp. 25–27, 2017.
[38] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust Online Appearance Models for Visual Tracking," *Proc. 2001 IEEE Comput. Soc. Conf. Comput. Vis.*

Pattern Recognit., vol. I, no. 10, p. I-415-I-422, 2001.
[39] J. Kang, I. Cohen, and G. Medioni, "Object Reacquisition Using Invariant Appearance Model," *Proc. 17th Int. Conf. Pattern Recognit.*, vol. 4, pp. 759–762, 2004.
[40] W. M. W. M. K. Khairosfaizal and A. J. Nor'aini, "Eyes Detection in Facial Images Using Circular Hough Transform," *5th Int. Colloq. Signal Process. Its Appl.*, pp. 238–242, 2009.
[41] T. Kocejko, A. Bujnowski, and J. Wtorek, "Eye Mouse for Disabled," *Conf. Hum. Syst. Interact.*, pp. 199–202, 2008.
[42] E. J. Leavline and S. Sutha, "Design of FIR Filters for Fast Multiscale Directional Filter Banks," *Int. J. Sci. Technol.*, vol. 7, no. 5, pp. 221–234, 2014.
[43] J. P. Lewis, "Fast Normalized Cross-Correlation," *Vis. Interface*, 1995.
[44] W. Nam, P. Dollár, and J. H. Han, "Local Decorrelation for Improved Pedestrian Detection," *Neural Inf. Process. Syst.*, 2014.
[45] K. Nguyen, C. Wagner, D. Koons, and M. Flickner, "Differences in the Infrared Bright Pupil Response of Human Eyes," *Proc. Symp. Eye Track. Res. Appl.*, pp. 133–138, 2002.
[46] C. O'Brien, "Speech Sounds Clip Art Set," *Teachers Pay Teachers*, 2017. [Online]. Available: https://ecdn.teacherspayteachers.com/thumbitem/Speech-Sounds-Mouth-Clip-Art-Set-1914282-1525373871/original-1914282-1.jpg. [Accessed: 01-Jun-2018].
[47] S. Ojha and S. Sakhare, "Image Processing Techniques for Object Tracking in Video Surveillance - A Survey," *Int. Conf. Pervasive Comput.*, 2015.
[48] OpenCV, "Image Pyramids," 2015. [Online]. Available: https://docs.opencv.org/3.1.0/dc/dff/tutorial_py_pyramids.html. [Accessed: 04-Aug-2018].
[49] OpenCV, "OpenCV," 2018. [Online]. Available: https://opencv.org/. [Accessed: 06-Aug-2018].
[50] OpenCV, "Geometric Image Transformations," 2018. [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html.
[51] OpenCV, "Camera Calibration and 3D Reconstruction," 2018. [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. [Accessed: 06-Aug-2018].
[52] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A Survey on Object Detection and Tracking Methods," *Int. J. Innov. Res. Comput. Vis. Commun. Eng.*, vol. 2, no. 2, pp. 2970–2978, 2014.
[53] M. J. Patel and B. Bhatt, "A Comparative Study of Object Tracking Techniques," *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 4, no. 3, pp. 1361–1364, 2015.
[54] A. Pentland, B. Moghaddam, and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 84–91, 1994.
[55] N. Perveen, D. Kumar, and I. Bhardwaj, "An Overview On Template Matching Methodologies and Its Applications," *Int. J. Res. Comput. Commun. Technol.*, vol. 2, no. 10, pp. 988–995, 2013.
[56] M. Richie, "Cataract Surgery with Dr. Michael Richie," *YouTube*, 2013. [Online]. Available: https://www.youtube.com/watch?v=0faextOYin4&has_verified=1. [Accessed: 07-Aug-2018].

[57] R. Ronfard, “Region-Based Strategies for Active Contour Models,” *Int. J. Comput. Vis.*, vol. 13, no. 2, pp. 229–251, 1994.
[58] V. Salari and I. K. Sethi, “Feature Point Correspondence in the Presence of Occlusion,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 87–91, 1990.
[59] J. N. Sarvaiya, S. Patnaik, and S. Bombaywala, “Image Registration by Template Matching Using Normalized Cross-Correlation,” *Int. Conf. Adv. Comput. Control. Telecommun. Technol.*, pp. 819–822, 2009.
[60] K. Sato and J. K. Aggarwal, “Temporal Spatio-Velocity Transform and its Application to Tracking and Interaction,” *Comput. Vis. Image Underst.*, vol. 96, no. 2, pp. 100–128, 2004.
[61] L. Segre and S. Bagi, “Eye Anatomy: Parts Of The Eye,” *All About Vision*, 2017. [Online]. Available: http://www.allaboutvision.com/resources/anatomy.htm. [Accessed: 27-Apr-2018].
[62] J. Shi and C. Tomasi, “Good Features to Track,” *IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 593–600, 1994.
[63] M. Simonovsky, “Ellipse Detection Using 1D Hough Transform.” [Online]. Available: https://de.mathworks.com/matlabcentral/fileexchange/33970-ellipse-detection-using-1d-hough-transform. [Accessed: 20-May-2018].
[64] L. Di Stefano, S. Mattoccia, and M. Mola, “An efficient algorithm for exhaustive template matching based on normalized cross correlation,” *Proc. 12th Int. Conf. Image Anal. Process.*, pp. 322–327, 2003.
[65] R. Stiefelhagen, J. Yang, and A. Waibel, “Tracking Eyes and Monitoring Eye Gaze,” *Proc. Work. Percept. User Interfaces*, pp. 98–100, 1997.
[66] R. L. Streit and T. E. Luginbuhl, “Maximum Likelihood Method for Probabilistic Multihypothesis Tracking,” *Proc. SPIE - Int. Soc. Opt. Eng.*, vol. 2235, pp. 394–405, 1994.
[67] R. Szeliski, *Computer Vision : Algorithms and Applications*, Draft. Springer, 2010.
[68] H. Tao, H. S. Sawhney, and R. Kumar, “Object Tracking With Bayesian Estimation of Dynamic Layer Representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 75–89, 2002.
[69] V. Thompson, “Cataract Surgery,” *All About Vision*, 2018. [Online]. Available: http://www.allaboutvision.com/conditions/cataract-surgery.htm. [Accessed: 26-Apr-2018].
[70] M. Turk and A. Pentland, “Eigenfaces for Recognition.pdf,” *Journal of Cognitive Neuroscience*, vol. 3, no. 1. pp. 71–86, 1991.
[71] C. J. Veenman, M. J. T. Reinders, and E. Backer, “Resolving Motion Correspondence for Densely Moving Points,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 54–72, 2001.
[72] Y. Xie and Q. Ji, “A New Efficient Ellipse Detection Method,” *Proc. 16th Int. Conf. Pattern Recognit.*, vol. 2, no. c, pp. 957–960, 2002.
[73] C. Yang, J. Sun, J. Liu, X. Yang, D. Wang, and W. Liu, “A Gray Difference-Based Pre-Processing for Gaze Tracking,” *IEEE 10th Int. Conf. Signal Process.*, pp. 1293–1296, 2010.
[74] A. Yilmaz, O. Javed, and M. Shah, “Object Tracking: A Survey,” *ACM Comput. Surv.*, vol. 38, no. 4, 2006.
[75] A. L. Yuille, D. S. Cohen, and P. W. Hallinan, “Feature extraction from faces using deformable templates,” *Proc. CVPR ’89 IEEE Comput. Soc. Conf. Comput. Vis.*

Pattern Recognit., pp. 104–109, 1989.
[76] F. Zhao, Q. Huang, and W. Gao, “Image Matching by Normalized Cross-Correlation,” *Int. Conf. Acoust. Speech Signal Process.*, pp. 729–732, 2006.
[77] W. Zhao, “Flexible Image Blending for Image Mosaicing with Reduced Artifacts,” *Int. J. Pattern Recognit. Artif. Intell.*, 2006.
[78] “Blindness and visual impairment,” *World Health Organization*, 2017. [Online]. Available: http://www.who.int/en/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Accessed: 26-Apr-2018].
[79] “Facts About Cataract,” *The National Eye Institute*, 2015. [Online]. Available: https://nei.nih.gov/health/cataract/cataract_facts. [Accessed: 26-Apr-2018].
[80] “Cataract Surgery,” *Fort Worth Eye Associates*. [Online]. Available: http://www.ranelle.com/cataract-surgery/. [Accessed: 26-Apr-2018].
[81] “Cataract Surgery Steps: Cartoon,” *Super Eye Care Resources*. [Online]. Available: http://www.supereyecare.com/images/Phaco.jpg. [Accessed: 27-Apr-2018].