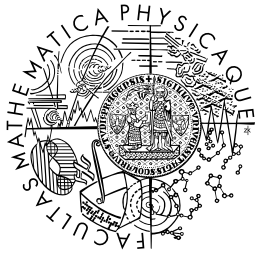



FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

**MASTER THESIS**

Adam Dominec

# Software-based eye tracking

Department of Software and Computer Science Education

| | |
|---|---|
| Supervisor of the master thesis: | RNDr. Barbara Zitová, Ph.D. |
| Study programme: | Computer Science |
| Study branch: | Software Systems |

Prague 2017

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date .....  signature of the author

Title: Software-based eye tracking

Author: Adam Dominec

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Barbara Zitová, Ph.D., Institute of Information Theory and Automation, Czech Academy of Science

Abstract: This thesis presents a software library for eye gaze tracking. The typical use case is a person watching their computer screen. All data is obtained from a single video camera and is processed in real time. The resulting software is freely available including its source code.

Keywords: face tracking gaze tracking image analysis

# Contents

# 1. Introduction

The main idea behind this program is to view human eye as a means of communication. It is rather intuitive for us to recognize other people's gaze just by looking at their face thanks to high contrast coloring of the eye itself. The relevant brain circuitry develops early in infants and is equivalent among individuals. This seems to be an evidence of evolutionary purpose—the human eye is built in a way so as to display the gaze clearly [26]. Therefore it should also be possible for a computer to estimate gaze from the data us humans have available, that is, a color image.

Thanks to the fact that almost every laptop computer or cell phone is now equipped with a video camera, the hardware requirements of this software are easy to satisfy. Future programs built on top of this library will also exhibit minimal hardware requirements that enable them to be used almost immediately upon download and completely for free.

The possible applications are a vast topic. Along the communication concept, eye trackers can be used for human-computer interaction. Wide variety of algorithms have been proposed for on-screen keyboard and general desktop control [17]. In contrast to classical input devices, they make computers accessible to disabled users that have difficulty to move their limbs. They can also improve the user comfort, such as by speeding up mouse motion in accordance to the gaze.

Running a gaze tracker quietly in the background (perhaps in a low quality) can also be helpful. When the screen of a cell phone is not looked at, it can automatically lock to improve security. On the other hand, it can display a message when somebody reads over your shoulder.

The gaze provides valuable information even when we do not concentrate on it. In psychological experiments, it is often used as a synonym of the subject's attention. An industrial application of this method is found in marketing and user experience design, where gaze recordings can be used to evaluate the quality of presentation of a product, or of an interface layout. Knowing the user's center of attention can also serve as a cue for adaptive rendering in video games because the rest of the screen need not be displayed in full detail. Tracking the gaze of a car driver can be used to improve safety [20].

## 1.1 History

Perhaps a more appropriate heading of this section would be *Unrelated work*. Indeed, before we get to the overview of our software competitors, we should shortly review past research of gaze tracking in general. Its history spans half a century before the computer era.

The interest in eye tracking originates in the field of psychology. Gaze designates the focus of attention of the subject which in turn can tell much about ongoing cognitive processes. Furthermore, eye movement itself is the result of a rather complex neural system; nowadays, it is perhaps the best studied part of human cognition.

In order to record data with a reasonable precision, sophisticated mechanical structures used to be constructed around the subject's head. Eye movement was

then measured either directly from an object attached to the eye, or indirectly from small displacements in the eye region. For example, an especially precise technique was developed by Alfred Yarbus in 1954 and requires to stick a mirror to the surface of the eye using a small suction cup.[1] Eye movement can then be recorded directly on a piece of photographic film via the reflection of a point light source shining at the eyes. We should note here that many other attachment mechanisms were less user-friendly than the suction cups.

A method known as *electrooculography* provides a less invasive alternative. Physiologically, there is a constant voltage gradient across the eye from back to front, in magnitude of about 1 mV. Rotation of this small dipole generates a measurable magnetic field, so it is possible to almost directly measure the speed at which the eye moves. The advantage gained is temporal resolution: this method allows to draw a graph of angle against time. However, the actual direction is an integral of the measured value, and therefore tends to deteriorate and the spatial resolution is generally poor.

Details on other purely analog methods such as this can be found in a 1967 book [33]. The various methods suggested for eye tracking since the end of 19th century are a story of human curiosity and can be seen as a proof of the effort directed towards this area.

## 1.2 Related work

Rapid development of computers and digital video cameras has removed most of the hardware constraints mentioned so far. The demand for non-intrusive gaze tracking is high in many branches of science. For example, it is obvious that the results of a psychological experiment can greatly vary with emotional influence of the environment, thus the influence should be kept minimal.

Letting the subject move their head without constraints, invariance against head pose becomes a serious challenge. Head pose estimation and gaze tracking are typically handled as two separate tasks to be performed in series. Only few approaches are able to encompass both of these tasks within a single model.

### 1.2.1 Face tracking

In this thesis, we believe that it is sufficient to examine the image of the face alone. Using the data obtained this way, we intend to cancel the effects of head movement, so that eyes can be tracked independently. However, the problem can also be viewed as a more general task of head pose estimation.

Many substantially different approaches have been suggested for head pose estimation, and there seems to be no consensus so far. The tracker by Kanade, Lucas and Tomasi [16] can be considered the cornerstone of object tracking. It is based on matching a template image using gradient descent optimization; more precisely, the original paper describes tracking a rectangular grayscale template by horizontal and vertical shift. Nowadays, such a simple problem can also be solved on a global scale, e.g., using normalized cross-correlation and the Fast

[1] The article is not cited here because it has been published only in Russian and does not seem generally available. However, Yarbus provides details on the method in [33].

Fourier Transform. However, there have been numerous generalizations of this concept to a broader class of motion models such as affine (e.g., [2]) or perspective transformations, in which cases a global optimization is not feasible. These are often referred to as the *deformable template models*. Our program uses several of such generalizations extensively.

An especially sophisticated generalization are the Active Appearance Models (AAM) [4; 24]. In this method, a planar mesh is overlaid on the object, subdividing the template image into polygon-shaped cells. Each of the cells is responsible of stretching its cut-out image part when its vertices are displaced. All vertices of the mesh can be displaced separately, and they are essentially the model parameters to be optimized. The degrees of freedom of this model can be customized to application needs, so that the method will handle either rigid- or soft-body motion gracefully. The original paper suggests to learn the motion constraints by a Principal Component Analysis of manually annotated training data.

Purely geometric image transformations exhibit poor invariance to changes in lighting, so they are well combined with element-wise image transformations. A simple option is to acquire multiple templates of the object in question, and either just select the best candidate for each input image, or allow their arbitrary linear combinations. If the amount of training data grows large, more efficient and robust schemes are necessary. The template images can be arranged in a search structure to speed up the lookup of the most similar template (i.e., the nearest neighbor).

A template need not be limited to a single image, and may be given implicitly, such as in [8]. It is especially common to extend each template by a per-pixel linear function in a small neighborhood. The linear coefficients are typically estimated from several nearby templates using the Principal Component Analysis, and this approach is widely known as Eigenfaces [18; 23]. This approach leads to a mathematical concept of a high-dimensional manifold that covers all feasible face images. Each point in the high-dimensional space can be assigned a feature vector that correspond to the modeled degrees of freedom [1]. In this view, a face tracker simply extracts these parameters from the corresponding point in space.

Another direction of development is to track several small templates simultaneously, and to impose constraints upon their relative positions. These are called the *Deformable Parts Models*. The image is analysed within each of the parts, and their positions are updated in an iterative manner. In order to exploit all the information available, their displacement in each iteration should be planned for all at once. Reasonable choices of such a decision engine include the Support Vector Machines [28] and Random Forests [22]. Given the typically high complexity of this algorithm, each of the templates is usually limited to a simple displacement, i.e., they remain as axis-aligned rectangles. The facial landmarks tracked by such Deformable Parts Models can be used for geometric reasoning on the head pose.

A noteworthy group of approaches uses some extra hardware, most commonly a depth camera. This data stream may be made optional alongside a video, as in [18]. Thanks to the geometric nature of the depth data, it can significantly improve the overall precision.

For a thorough comparison of all the face tracking methods in terms of performance and requirements upon the input data, we can recommend the excellent

(although slightly outdated) survey [19].

There have also been many software tools published aside from the scientific journals. Some of them are available including the source code such as [5; 15].

### 1.2.2 Eye tracking

Eye tracking seems to be a simple task once the head pose is known—it almost feels that the eyes are made so as to be clearly visible. However, it is rather difficult to develop a method that would generalize well, e.g., across users. It appears that many of the methods that have been proposed in the literature are only appropriate in conditions specific to each of them.

Much research relies on the fact that both human iris and pupil are circles, so their camera projection is always an ellipse. When the gaze direction is reasonably bounded, these projected shapes can safely be considered to remain circular. Under this assumption, the generalized Hough Transformation can be used to search for circles with one-pixel precision.

Unfortunately, the Hough Transformation is only effective around the boundary of the circle. One possibility how to incorporate the whole mass inside the circle is the Mean Shift algorithm. Essentially, that is an application of the Expectation-Maximization scheme: a circle is iteratively repositioned to the mean of pixel weights within it. This concept has been used in many simple programs, and is also included in complex models such as [31].

It is quite common to model some more specific aspects of the eye. A reasonable choice are the eye corners [38] and the eyelids [35].

It is possible and sometimes more robust to use an appearance-based method [25], that is, crop out a small image in the eye region and consider its all pixels a high-dimensional vector. This approach usually requires the eye position to be precisely normalized to a center point, so that eye images with varying gaze directions only differ in the iris and pupil position. Obviously, there are many free parameters (such as iris color, skin tone, shape of the eyelids and amount of eyelashes) that ought to be ignored by the gaze tracker. To this end, it is necessary to provide enough training data that covers all these cases, to prevent overfitting.

In contrast to these methods based solely on an image, many rely on some extra hardware, such as cameras or lights. Methods with partially controlled lighting are especially efficient for eye tracking because the mammalian eye is reflective both on its inner and its outer surfaces. The retina of the human eye is distincly reflective in red and near infrared light, which is the cause of the red-eye effect commonly observed in photography. If an infrared light source is placed next to the camera, the user's pupils will shine brightly, whereas the rest of the eye and the scene brightens only subtly. The pupil shape can be obtained simply by contrasting this image to the one when the infrared light is turned off.

Having a point light source in a known position relatively to the camera also creates predictable reflections on the outer eye surface, called the *Purkinje images* [9]. The shape of the eye is just quite enough complex (as detailed in Section 3.2.2) and almost the same across individuals, so the eye pose can be deduced from such glares by geometric calculations. As shown in [30], this approach can be used for precise tracking with only minimal calibration.

Devices using Purkinje reflections can be identified easily, since they contain several infrared lights that quickly switch on and off. According to the author's experience, the Tobii EyeX tracker [11] belongs to this group. The tracker has been used to obtain some of our testing data; for a more detailed description, see Attachment B. Tobii is one of the pioneer corporations in eye tracking, and many of their products are available on the market.

If even higher precision is required, it may still be necessary to attach a camera to the user's head and point it closely to the eyes. Although such methods are slowly being deprecated by the less intrusive ones presented so far, it is important to note that there has also been much advancement in camera manufacturing. Indeed, it is possible to build very small and lightweight cameras that will make almost no obtrusion to the user's view and comfort [13; 27]. Such a tracking rig may be preferred in many controlled scenarios such as in psychological laboratories.

For a thorough survey of eye trackers including the obscure historical ones, the reader is directed to [9].

Regarding the relation between eye rotation and on-screen gaze position, some sources suggest that only a linear function is necessary [38]. On the other hand, if the head and scene model is precise enough, it may be appropriate to calculate the gaze explicitly as a ray in space [31].

### 1.2.3 Testing databases

Many data sets for gaze tracking are publicly available for download.[2] We have examined many of them in the hope that they could be used for the training and testing of our own program.

Perhaps the most profound benchmark is the BioID database [12]. It consists of about 1500 grayscale images, each of them annotated with the position of each pupil, the eye corners and thirteen more prominent facial landmarks. The images are all limited to a common resolution of $384 \times 286$ pixels, which makes the iris about 10 pixels in diameter. This image resolution could be enough for our purposes. Unfortunately, the eye positions are given in whole pixel units only, and the actual precision is yet somewhat worse. Since we intend to locate the iris center with subpixel precision, this data set is not sufficient.

Another noteworthy piece is the MPIIGaze data set [37], which is accompanied by a neural network-based gaze tracker. It contains more than 200,000 photos in full color and quite a high resolution, acquired using a built-in camera of a laptop computer. Its annotations are another extreme case: the on-screen gaze position is known in each image, but nothing else. In theory, we could use it to evaluate the performance of our program. Unfortunately, even that is not appropriate because almost every photo in the MPIIGaze data set has been taken in different light conditions than others. Furthermore, the lighting is often poor and the photos are blurry.

Very appealing is the approach taken by the authors of the SynthesEyes data set [32]. They used real people only for creation of the 3-dimensional models of their heads. These models are then used to create a virtually infinite number of

[2] A very nice list of data sets for computer vision, including gaze estimation, is maintained at `http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm#face`.

training images. The resulting computer generated images show a small region centered around an eye. All of the randomly generated parameters, such as viewing direction and gaze direction, are recorded in a data file. We should note that the variation of parameters is taken to an extreme—for example, several of the images are entirely black due to contrast variation. Also, because of the wide range of viewing angles, this data set does not fit our needs well.

In the end we decided to collect own testing data. These are described in detail in Attachment B.

## 1.3 Organization

Chapter 2 provides a brief overview of the goals of this thesis. It is followed by Chapter 3, where we analyse all aspects of gaze tracking relevant to our method.

The following two chapters, 4 and 5, explain the general-purpose algorithms used in our program and their specific application to our needs, respectively. An experimental evaluation of the resulting program is provided in Chapter 6.

The accompanying data medium is described in the attachments. Specifically, its directory structure is explained in Attachment A.

## 1.4 Notation

For matrix algebra we follow the notation used in the classical book Multiple View Geometry [10]. In order to reduce ambiguity, we add several more rules:

| | |
|---|---|
| $c$ | italic lowercase letter ..... scalar |
| diag | regular type word ..... function |
| F | uppercase letter ..... function |
| $S$ | italic uppercase letter ..... set |
| $\mathbf{x}$ | boldface letter ..... real column vector |
| $\mathtt{M}$ | uppercase monospace letter ..... real matrix |
| $\mathtt{M}^{-\mathtt{I}}$ | ..... inverse matrix |
| $\mathtt{M}^{i}$ | .....indexed matrix name (e.g., index in a list of matrices) |
| $\cdot$ | central dot symbol .....scalar or matrix multiplication |
| $\lvert\mathbf{x}\rvert$ | vertical bars ..... vector $L_2$ norm |

# 2. Goals

Given the fact that a majority of laptop computers are capable of constrantly recording the users using a video camera, it is attractive to analyse this stream. We set off to pursue the following goals:

- Build an **open-source** solution for interactive gaze tracking. Gaze tracking is by no means a new task, and has been detailed in the previous section, there are many software libraries that solve it. Unfortunately, it is rare that such a library would be released with open source code, and these few are usually incomplete or outdated. We believe that sharing the source code will help further development in this scientific field.

  For a good overall precision of gaze tracking, a very precise is a crucial component. To this end, we design all the calculations so as to work reliably and properly on subpixel scale.

- Create a **benchmark rig** for evaluation of the tools in terms of performance. By implementing several different methods for each of the subtasks at hand, we intend to compare them in terms of performance. Further evaluation can help choose an appropriate method for a specific application.

  Because our problem setting is already quite specific, we also provide testing data sets that fit the given conditions. In order to properly evaluate the reasons why some methods perform better than others, it is necessary to make extra annotations about each testing sample. Publicly available data sets lack this kind of information.

- We decided to design a **novel algorithm** for face tracking, and another one for eye tracking. Given the amount of algorithms that have been already presented in the literature, this plan may seem pointless. However, it seems that there is still room for improvement, and discovering new methods may bring more insight.

Where possible, we try to employ complex and rigid models that are specific to the tasks at hand. There are two reasons for this. First and foremost, we would need too much training data in order to learn a generic machine learning algorithm. It turns out that none of the publicly available data sets for face and gaze tracking are well suitable for our problem setting. For example, most of them are designed for recognition from a static image, and some even lack color information.

Secondly, there has been much progress in the field of machine learning currently, and we can hardly compete with the manpower and computational capacity of these scientific teams. Configuring a machine learning tool is simply not the aim of this thesis, by the author's personal preference.

# 3. Problem Analysis

This chapter provides a thorough analysis of the data stream provided by the camera, and of the objects displayed therein. We discuss the anatomy of human face and eyes in particular, and all relevant technical details about commonly used web cameras.

## 3.1 Conditions

We expect that the user is working with their personal computer or a similar device such as a tablet. The user and the device can freely move around; the only constraint there is that the camera must be fixed to the computer screen.

Figure 3.1: Example input frame (see Attachment B).

Throughout the computation, we rely on that the user's face and both eyes are visible, and that the angle difference of the gaze and the camera direction is small. We do not impose any further constraints on the face pose relatively to the camera. For example, it is not necessary that the camera be upright, since some users might find it more comfortable to attach a camera to the bottom of their screen upside down. Such a constraint could also pose a difficulty for users working with a vertically rotated screen.

For a good calibration, we require active cooperation from the user: they are asked to watch a dot moving around the screen. If necessary, this requirement can be somewhat lifted by using a calibration file defined earlier or by some completely different means of calibration. We allow the user to move their head but we cannot actually rely on this because many users (e.g. disabled ones) may have difficulties to move.

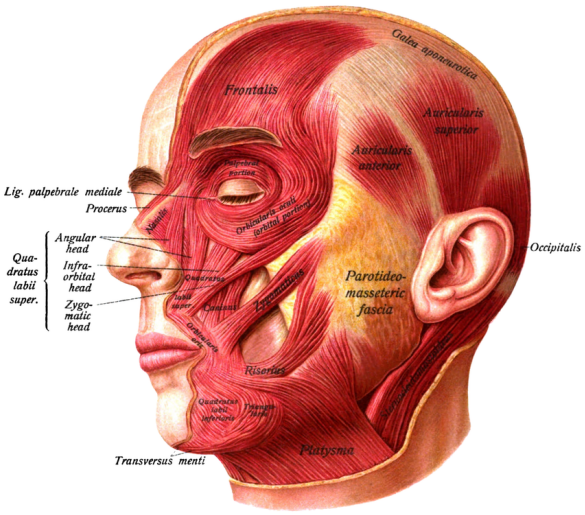
Ambient lighting must be good enough for the camera to deliver a high-contrast and sharp image. The image brightness, white balance etc. are allowed to change abruptly and always. There is, however, a strong requirement on the light being rather diffuse than directional, and making no sharp shadows on the

user's face. If the light conditions substantially start to differ from the ones during calibration, it may be necessary to re-calibrate.

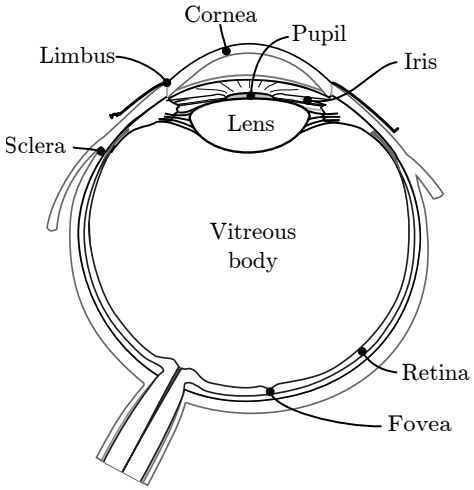
Cluttered environment is not an issue as long as the extraneous objects do not occlude the view and do not oversaturate (dazzle) the camera. An example of what the input image could look like is given in Figure refi:analy-shot.

## 3.2 Human head

We consider it important to introduce some basic terms and facts about human anatomy in order to better understand the problem at hand.




(a) Human face. (J. Sobota, public pomain)




(b) Human eye. (Adopted from `commons.wikimedia.org`, CC0)

Figure 3.2: Illustration of an undesired result of the Poisson reconstruction.

### 3.2.1 Face anatomy

The face is the frontal part of human head. Facial muscles are specific in that they are attached to the skin, and their almost sole purpose is communication. As seen in Figure 3.2a, there is hardly a spot on the face that would not have muscles underneath. Although they can be controlled by will, they often actuate subconsciously. There are kinds of subtle movements, usually induced by basic emotions such as fear, that can not be prevented by will. In a summary, the facial skin is flexible with a virtually infinite number of degrees of freedom, and almost its whole area is controlled by muscles.

Of special interest for us are regions that remain mostly fixed to the skull in normal conditions. These include the nose (especially its upper part), the chin and small outer regions around and slightly below the eyes. The chin is very prominent and easy to track, but may become completely misleading if the user opens their mouth. Given the flexibility discussed above, it is almost impossible to estimate head pose precisely, but we can get a good estimate by choosing some of these less expressive regions.

The face also contains many features that hinder gaze recognition. Typically, people keep their eyelids open just quite enough so as not to occlude the pupil. This poses a serious challenge to all techniques based on the iris: in many people, only about half of the iris area is visible.

On the edges of eyelids there are eyelashes. Depending on the viewing angle, these can also significantly occlude the eye. As if this all was not enough, the user may wear strong makeup and a mascara to emphasize these undesirable factors.

Assistive tools like glasses and contact lenses also have to be considered when designing a gaze tracker, although they are technically not a part of the face. Glasses and lenses (including the natural ones) will cause an unpredictable displacement of the iris and pupil image. A flexible enough gaze estimator may be capable of handling this distortion implicitly. In fact, glasses with solid rims can help head pose estimation as their position relatively to the skull remains reliably constant.

### 3.2.2 Eye anatomy

The eyeball consists of several functional parts. The most notable ones are depicted in Figure 3.2b:

- Transparent *cornea*, a fixed lens also providing some mechanical protection
- Flexible and reflective *iris*, which serves as a diaphragm
- Flexible *lens* stretched by several muscles to control its optical magnitude
- Purely white *sclera*, which serves as a hard shell of the eyeball
- *Vitreous body*, a transparent gel to maintain the inner pressure
- Light-sensitive *retina*

Furthermore, each eyeball is embedded in a hole called the *orbit* that provides fixation and actuation. There are six separately controlled muscles stretching between to the eyeball and the orbit.

Throughout the literature, eye is modeled either as a sphere [36], or with an extra spherical section for the cornea [30]. Some sources, such as [31], also explicitly model the eyelids.

The sclera of human eye is made of white collagen. The color of the iris, in turn, can vary anywhere between bright blue and dark brown, including some less common shades like green and amber. It feels quite intuitive that these distinctive colors have evolved for the purposes of nonverbal communication. This claim is known as the *cooperative eye hypothesis* and has been, in part, proven by experiment [26].

In addition, the aperture inside the iris, known as the *pupil*, displays the retina through the lens. The retina absorbs light in order to acquire as much information as possible, and therefore it is mostly black. A notable exception is the red-eye effect that may appear in directional light. Its usefulness to eye tracking has been discussed in Section 1.2.2, but since we assume diffuse lighting, this effect should not be present.

The inner and outer boundary of the human iris (the pupil and the limbus, respectively) are concentric circles. Generally, the inner shape of the iris is the less reliable of these two. An extreme case is when the pupil is permanently distorted and non-circular due to a damage [3, p.5] or a disease [3, p.146].

In healthy people, the radius of the pupil changes depending on the amount of incoming light and on their emotional state. The constriction is controlled by the parasympathicus, a nervous system generally related to comfortable actions such as eating and sleeping. The dilation is controlled by the sympathicus, which in turn is the main actuator of the "fight or flight" stress response. Both the sympathicus and parasympathicus are parts of the autonomous nervous system and as suggested by the name, they cannot be directly controlled by will.

### 3.2.3 Eye movement

When attending to an object (or another person), people will automatically turn their eyes directly towards it. This process is usually subconscious and some of its aspects are involuntary.

The main reason for eye movement is that the overall acuity of our visual system increases towards a small area near the optical axis. The corresponding spot on the retina is called the *fovea*, and it is located about 5° off-axis horizontally towards the nose [30]. It covers only about 1.5° of the visual field. The density of photopic (daylight-sensitive) cells in the fovea is up to 20 times more than in the peripheral areas of the retina.

Although gaze can be precisely controlled by will, there are many peculiarities about eye motion that depend on old brain circuitry and are common to all humans.

Assuming that each muscle can apply force only by stretching, and not extending itself, the six muscles provide three degrees of freedom (DoF) to the eye motion—but only two DoF are necessary to control the gaze direction. A third DoF, namely rotation around the optical axis (the *roll*) is actively used by the brain. The effect is slight, but because fovea is offset from the optical axis, this could make the gaze direction unpredictable given the optical axis only. Fortunately, the roll is deterministic with respect to the remaining two rotation angles. This fact is known as the *Donder's Law* [9], and thanks to it, the effect of eye roll can be implicitly precalculated during calibration.

The gaze is usually controlled by high-level brain regions automatically so as to observe all necessary details about the environment. This process is mainly based on visual feedback, and can be predicted to an extent by simple image processing, such as in [34]. There are three basic actions that our visual system is capable of: a *saccade*, a *fixation*, and a *smooth pursuit*.

The purpose of saccades is to point the gaze in a new direction. During eye movement, the visual input is distorted by motion blur, so it is desirable that it be as quick as possible. Indeed, the angular acceleration in the beginning and in the end of a saccade reaches the order of $10{,}000°/s^2$ The periods when the eye is static are called fixations.

It is important to note that what we feel like a constant gaze direction is actually a sequence of saccades and fixations. Although a single fixation may suffice to identify an object, the whole visual system relies on its ability to perform

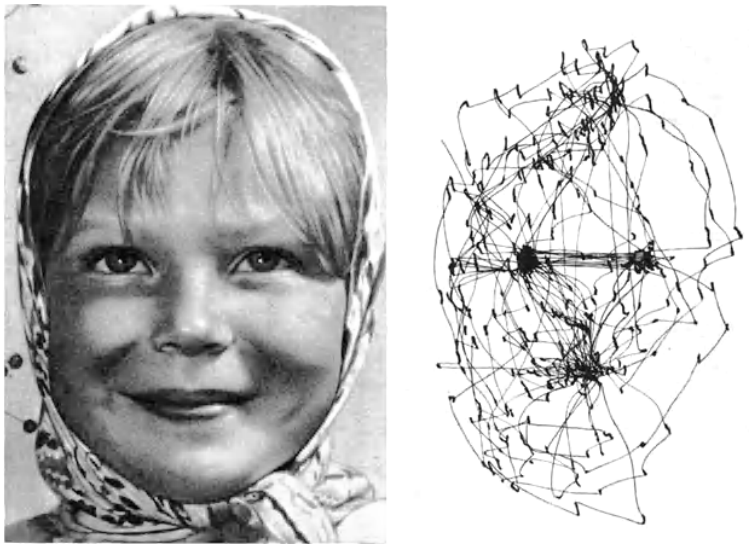


Figure 3.3: Gaze recording. (L. Yarbus [33])

saccades constantly. The scale of these saccades varies greatly, a microsaccade measured in an experiment can be considered small within the fixation threshold in another one. For a rough estimate, we can say that the eye moves less than 1° during a fixation. This corresponds to the 1.5° diameter of the fovea: when attending to an object, it is enough to project it quite anywhere on the fovea, in order to get a good image. Figure 3.3 shows a sequence of saccades and fixations as recorded when viewing a static image for three minutes.

Our tracker will ignore these details about eye motion since they are much below the expected resolution of our program. It is important to remember, though, that the speed of saccades makes the gaze very unpredictable. For example, it would not be wise to impose a maximum gaze offset relatively to the previous known position.

Finally, there is a special kind of eye movement for following objects that actually move around. This is referred to as the smooth pursuit, and it is characterized by a smooth, saccade-less eye movement.

People are usually unable to perform smooth pursuit without a moving object in sight. On the other hand, if a moving target is drawn on the screen with a low frame rate, our perception of it will be blurry because the eye will traverse a smooth path during each frame. Being so predictable and precise, smooth pursuit is perfect for the calibration of a gaze tracker.

## 3.3 Gaze

The gaze is the direction of focus of the user. We assume that it is a ray originating in the eye, and that its direction is unambiguously given by the eye rotation. Furthermore, we expect the ray be pointed to somewhere on the computer screen. The typical setting is drawn out in Figure 3.4.

Gaze is affected by both the head pose and eye rotation. Although it may seem more efficient to only move our eyes when using a computer, people usually move their heads quite a lot. In fact, prolonged static pose of the head can cause pain in the spine.

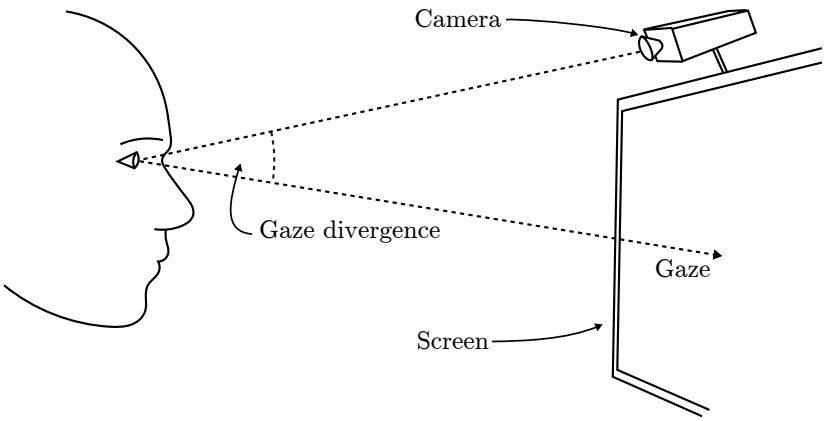



Figure 3.4: Sketch of the scene.

There are no strict constraints regarding the scene configuration. Perhaps the only requirement is our assumption of small angle divergence. The gaze angle when the user looks directly into the camera and when they look to the farthest point on the screen should not differ by more than about 30°. This means that the iris can be anisotropically stretched by a factor of about 0.9, which can still be approximated as a circle.[1]

We model movement of the iris as a simple translation. It should be clear that because the head and the eyes differ by an order of magnitude in scale, gaze estimation is numerically unstable. A small variation in the eye rotation induces an even smaller variation in the iris position, but may correspond to a large change of gaze.

We avoid explicit modeling of the viewed scene by assuming that the gaze is given by a homography, as it will be detailed in Section 5.4.

## 3.4 Camera

A suitable yet very simple mathematical model of a video camera is the pinhole camera. Light rays passing through a certain point in space (figuratively, the pinhole) are projected onto an image plane. The axis of symmetry of this system is called the optical axis, and the intersection of the optical axis with the image plane is assumed to be the origin point in image coordinates.

Real-world cameras can suffer from many kinds of degeneracies off this model:

- Imprecise manufacturing. The image origin point may be offset from the optical axis. The sensor may be stretched and skewed, so that orthonormal vectors in image plane are not always sensed as orthonormal.

- Blur. Refractive lens are usually inserted into the optical path so that light need not pass through an infinitely small pinhole, but rather though a small disc. This approach results in that only light rays originating from a certain surface in the scene, known as the focal plane, are properly projected onto the image plane. Points from a plane parallel with the focal plane will be

[1] Assuming that the camera lies in the screen plane, this factor equals $\cos(30°) \approx 0.87$.

displayed as if convolved with a disc kernel; the radius increases with the parallel distance from the focal plane, and the shape is mostly a projection of the camera iris.

The lens may be dirty or scratched, which casues a slight and uniform foggy blur. In very small cameras with relatively high resolution, the wave nature of light may cause additional blur by diffraction at edges of the camera iris.

- Lens aberration. The lens itself is a thick solid object and therefore can never fulfill its physical model perfectly. The nonlinear effect usually manifests itself by stretching sensed points in or out relatively to the image origin point. If carefully measured, this geometric transformation within the image plane can be very well cancelled.

  Because the refractive index of materials varies with wavelength (this fact is known as dispersion), the nonlinearities are also wavelength dependent. There are software tools to reduce percieved color aberration, but this problem is under-determined and can never be solved exactly. A proper solution would require to densely sample the spectrum at each image point but we only have three values roughly corresponding to red, green and blue.

  These effects are often well compensated in high-end cameras by sophistication of the optical system. On the other hand, they also decrease with the lens size, and cameras with a narrow or almost closed iris can be fairly well aproximated as pinhole cameras with no lens.

- Moiré. In most consumer cameras, image colors are obtained by attaching a color filter in front of the sensor, so that each light-sensitive element receives either red, green or blue light only. These colors are interleaved in a fine pattern. Color pixel can be obtained from each element by looking at two neighbors neighbors that have different filters, in the hope these neighbors are lit by roughly the same color. Alas, contrast on subpixel level makes each of these color sensors receive a different amount of light, even if there are no actual colored objects in the scene. When imaging fine black-and-white colored structures, spurious and highly saturated colors may appear.

  This effect can hardly occur when displaying human faces. Quite to the contrary, it can be used for manual focusing of the camera, if necessary. Moiré is an optical effect between the imaged object and the light sensor, and usually will not be affected by image processing such as denoising algorithms. We can put a black and white grid nearby the user's head and tune the camera focus until color moiré appears.

## 3.5 Image

The image acquired from the camera is a rectangular grid of colored points, expressed as a matrix $\mathtt{M} \in \mathbb{R}^{n \times m}$. However, certain parts of the computation require a continuous image model in order to obtain a sub-pixel precision. In such cases, the image function is defined by bilinear interpolation:

$$\begin{aligned} \mathrm{I}(\mathbf{p}, \mathtt{M}) = (1-t)\cdot &\left((1-s)\mathtt{M}_{i,j} + s \cdot \mathtt{M}_{i,j+1}\right) \\ +t &\left((1-s)\mathtt{M}_{i+1,j} + s \cdot \mathtt{M}_{i+1,j+1}\right), \end{aligned} \tag{3.1}$$

where $\mathbf{p} = (i + t, s + j)^\mathsf{T}$, $s, t \in [0, 1)$ and $i, j$ are whole numbers.

The partial derivatives of an image are approximated as

$$\mathrm{I}_x(\mathbf{p}, \mathtt{M}) = \mathrm{I}\left((\mathbf{p}_1 - \frac{1}{2}, \mathbf{p}_2)^\mathsf{T}, \mathtt{D}\right), \tag{3.2}$$

where $\mathtt{D}_{i,j} = \mathtt{M}_{i,j+1} - \mathtt{M}_{i,j}$.

Note that each derivative is sampled at a slightly shifted position. While this may seem needlessly picky, precise coordinates become very important when working with the image pyramid in Section 5.2. A half-pixel shift in a downscaled image can represent a very large distance in the original pixel grid, and ignoring this displacement would actually make our algorithms fail.

Nevertheless that the bilinear interpolation is an attractive choice, it does not produce a smooth function. We shall need to interpolate not only the image but also its derivatives up to the second order. In general, this problem has three possible solutions.

Firstly, we can interpolate both the image and its discretely evaluated derivatives using a simple formula, and ignore the inaccuracy induced. This is the approach applied in this thesis.

The second option is to use such simple interpolation on a blurred image in the hope that the inaccuracy disappears. This solution is implicitly used by many software libraries.

Finally, it is possible to interpolate using a sophisticated filter such as the Lanczos function $\mathrm{sinc}(x) \cdot \mathrm{sinc}(x/2), x \in (-2, 2)$. Both its first and second derivatives are continuous and bounded within $(-1, 1)$. The problem is that they are prohibitively hard to calculate. In the end, this may lead us to only estimating these derivatives by a simple formula—but by making such a step, we effectively classify to the first category above.

# 4. Algorithms

This chapter provides mathematical tools and computational methods for the problems to be presented in the next section.

## 4.1 Numeric Tools

Firstly, we shall describe several classical algorithms that we use to solve generic mathematical problems.

**Definition 1.** *The diagonal matrix constructor* $\mathrm{diag}^{n,m} : \mathbb{R}^k \to \mathbb{R}^{n\times m}$*, creates a matrix with a given vector along the main diagonal, with all remaining entries set to zero:*

$$\mathrm{diag}(\mathbf{d})_{i,i} = \mathbf{d}_i, \mathrm{diag}(\mathbf{d})_{i,j} = 0 \text{ for all } i \neq j.$$

*The input dimension $k$ must equal to the lesser value of $n, m$.*

**Definition 2.** *The Singular Value Decomposition (SVD) of a given matrix* $\mathtt{A} \in \mathbb{R}^{n\times m}$ *are orthonormal matrices* $\mathtt{U} \in \mathbb{R}^{n\times n}, \mathtt{V} \in \mathbb{R}^{m\times m}$ *and a vector* $\mathbf{s}$ *such that* $\mathtt{A} = \mathtt{U} \cdot \mathrm{diag}^{n,m}(\mathbf{s}) \cdot \mathtt{V}^\mathsf{T}$*,* $\mathbf{s}_i \geq 0$ *and* $\mathbf{s}_i \geq \mathbf{s}_j$ *for all* $i \leq j$*. The dimension of* $\mathbf{s}$ *equals to the lesser value of* $n, m$*.*

**Claim 3.** *The Singular Value Decomposition exists for any real matrix* $\mathtt{A}$*. In the specific case where* $\mathtt{A}$ *is symmetric, then* $\mathtt{U} = \mathtt{V}$*.*

### 4.1.1 Linear solving

*Input:* matrix $\mathtt{A} \in \mathbb{R}^{n\times m}$ of rank $m$, vector $\mathbf{b} \in \mathbb{R}^n$.
*Output:* vector $\mathbf{x}$ such that $|\mathtt{A}\mathbf{x} - \mathbf{b}|$ is minimal.

Let us consider the SVD of $\mathtt{A} = \mathtt{U}\cdot\mathrm{diag}(\mathbf{s})\cdot\mathtt{V}^\mathsf{T}$. The result is $\mathbf{x} = \mathtt{V}\cdot\mathrm{diag}(\mathbf{t})\cdot\mathtt{U}^\mathsf{T}\mathbf{b}$, where $\mathbf{t}$ is given by $\mathbf{t}_i = \mathbf{s}_i^{-1}$.

*Proof for the case $m = n$.* In the case of a square matrix $\mathtt{A}$, clearly $\mathtt{A}\mathbf{x} = \mathtt{U} \cdot \mathrm{diag}(\mathbf{s}) \cdot \mathtt{V}^\mathsf{T}\mathtt{V} \cdot \mathrm{diag}(\mathbf{t}) \cdot \mathtt{U}^\mathsf{T}\mathbf{b} = \mathbf{b}$, because $\mathrm{diag}(\mathbf{s}) = \mathrm{diag}(\mathbf{t})^{-\mathsf{I}}$. □

### 4.1.2 Homogeneous linear solving

*Input:* matrix $\mathtt{A}$
*Output:* vector $\mathbf{x}$ such that $|\mathbf{x}| = 1$ and $|\mathtt{A}\mathbf{x}|$ is minimal.

Let us consider the SVD of $\mathtt{A} = \mathtt{U}\cdot\mathrm{diag}(\mathbf{s})\cdot\mathtt{V}^\mathsf{T}$. The result $\mathbf{x}$ is the last column of $\mathtt{V}$.

*Proof.* Let us define a vector $\mathbf{v} = \mathtt{V}^\mathsf{T}\mathbf{x}$. This is effectively a change of basis, and because $\mathtt{V}$ is an orthonormal matrix, the norm $|\mathbf{v}| = |\mathbf{x}|$ is preserved. Because also $\mathtt{U}$ is an orthonormal matrix, it can be safely crossed out of the norm $|\mathtt{A}\mathbf{x}| =$

$|\mathrm{diag}(\mathbf{s}) \cdot \mathbf{v}|$. This in turn can be expressed as the square root of $\sum_i (\mathbf{s}_i \mathbf{v}_i)^2 = \sum_i \mathbf{s}_i^2 \mathbf{v}_i^2$, and the root can be omitted in optimization.[1]

Finally, this constrained minimization problem can be solved by the method of Lagrange multipliers. Equating $\nabla(\sum_i \mathbf{s}_i^2 \mathbf{v}_i^2) = \lambda \nabla(\sum_i \mathbf{v}_i^2)$ leads to the system of equations $\mathbf{s}_i^2 \mathbf{v}_i = \lambda \mathbf{v}_i$ for all $i$. These necessary constraints are satisfied only by choosing $\lambda = \mathbf{s}_i^2$ for some $i$. Deliberately assuming that all $\mathbf{s}_j^2 \neq \mathbf{s}_i^2$ for all $j \neq i$, we must also set the remaining elements $\mathbf{v}_j = 0$, and therefore $\mathbf{v}_i = 1$. In order to find a global minimum, we pick $i$ such that $\mathbf{s}_i^2 \mathbf{v}_i^2 = \mathbf{s}_i^2$ is minimal. The result is $\mathbf{x} = \mathtt{V}^{-\mathsf{T}} \mathbf{v} = \mathtt{V}\mathbf{v}$. □

If $\mathbf{s}_i^2 = \mathbf{s}_j^2$ for some $j$, the global minimum is not unique but this method finds a correct solution anyway. This claim is left without a proof.

### 4.1.3 Quadratic polynomial fitting

*Input:* set of samples $P = \{\mathbf{p}^i \in \mathbb{R}^2\}$ of function $\mathrm{score} : \mathbb{R}^2 \to \mathbb{R}$.
*Output:* quadratic polynomial $\mathrm{Q} \in \mathbb{R}^2 \to \mathbb{R}$ minimizing $\sum_i |\mathrm{score}(\mathbf{p}^i) - \mathrm{Q}(\mathbf{p}^i)|^2$ and vector $\mathbf{x}$ maximizing $\mathrm{Q}(\mathbf{x})$.

Given several samples of a real function nearby its maximum, we would like to get a precise estimate of the maximum location.

Let us express the least-squares fit as a linear system:

$$(x^2, xy, x, y^2, y, 1)^\mathsf{T} \mathbf{q} = \mathrm{score}(\mathbf{p}^i) \text{ for each } i, \text{ where } \mathbf{p}^i = (x, y)^\mathsf{T}. \tag{4.1}$$

The polynomial Q can be expressed in terms of the solution $\mathbf{q}$ as the symmetrical matrix

$$\mathtt{Q} = \begin{pmatrix} 2\mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \\ \mathbf{q}_2 & 2\mathbf{q}_4 & \mathbf{q}_5 \\ \mathbf{q}_3 & \mathbf{q}_5 & 2\mathbf{q}_6 \end{pmatrix}.$$

Let us split this matrix into the following blocks:

$$\mathtt{A} = \begin{pmatrix} 2\mathbf{q}_1 & \mathbf{q}_2 \\ \mathbf{q}_2 & 2\mathbf{q}_4 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{q}_3 \\ \mathbf{q}_5 \end{pmatrix}. \tag{4.2}$$

The global extremum $\mathbf{x}$ of this polynomial is obtained by solving $\mathtt{A}\mathbf{x} = -\mathbf{b}$.

*Proof.* Firstly, the polynomial can be evaluated using the matrix $\mathtt{Q}$ as $\mathrm{Q}(\mathbf{x}) = \mathbf{h}^\mathsf{T} \mathtt{Q} \mathbf{h}$, where $\mathbf{h} = (\mathbf{x}_1, \mathbf{x}_2, 1)^\mathsf{T}$. (See Definition 7.) The vector $\mathbf{q}$ finds a least squares fit of this equation. Then, we set the gradient $\nabla \mathrm{Q}(\mathbf{x}) = 2\mathtt{Q}\mathbf{h}$ to zero. □

### 4.1.4 Principal Component Analysis

*Input:* set of samples $P = \{\mathbf{p}^i \in \mathbb{R}^n\}$.
*Output:* orthonormal basis $\{\mathbf{q}^i\}$ such that the orthogonal projection of $P$ onto each $\mathbf{q}^i$ has maximal variance with all $\mathbf{q}^j, j < i$ being fixed.

[1] We also omit the summation bounds for $i$. We hope this improves readability since the bounds are unambiguously implied by the vector dimensions.

Let us consider the covariance matrix $\mathtt{C} = \sum_{i,j}(\mathbf{p}^i - \mathbf{m})\,(\mathbf{p}^j - \mathbf{m})^\mathsf{T}$ (up to scale), where $\mathbf{m} = \frac{1}{|P|}\sum_i \mathbf{p}^i$ is the center of mass. Let us further consider the Singular Value Decomposition of this symmetrical matrix $\mathtt{C} = \mathtt{U}\cdot\mathrm{diag}(\mathbf{s})\cdot\mathtt{U}^\mathsf{T}$. The sought basis vectors are the columns of $\mathtt{U}^\mathsf{T}$, in their order.

Given several random samples, we want to find the most prominent aspects of the data. The basis vectors are supposed to capture (and extract) all covariance, so that when expressed in the basis $\{\mathbf{q}^i\}$, the data will be decorrelated. Before we explain the formulas presented above, let us start with a lemma:

**Lemma 4.** *If a normally-distributed random vector* $\mathbf{x}$ *with covariance matrix* $\mathtt{C}$ *is transformed to* $\mathtt{A}\mathbf{x}$*, the resulting random vector has covariance matrix* $\mathtt{ACA}^\mathsf{T}$*.*

*Proof.* Using the fact that covariance is a linear quantity, we derive:

$$\begin{aligned}\mathrm{cov}\Big((\mathtt{A}\mathbf{x})_i, (\mathtt{A}\mathbf{x})_j\Big) = \mathrm{cov}\Big(\sum_k(\mathtt{A}_{ik}\mathbf{x}_k), \sum_l(\mathtt{A}_{jl}\mathbf{x}_l)\Big) = \sum_k\Big(\mathtt{A}_{ik}\sum_l \mathrm{cov}(\mathbf{x}_k,\mathbf{x}_l)\mathtt{A}_{jl}\Big)\\ = \sum_k\Big(\mathtt{A}_{ik}\sum_l \mathtt{C}_{kl}\mathtt{A}^\mathsf{T}_{lj}\Big) = \sum_k \mathtt{A}_{ik}(\mathtt{CA}^\mathsf{T})_{kj} = (\mathtt{ACA}^\mathsf{T})_{ij}. \quad (4.3)\end{aligned}$$

□

*Proof of the algorithm.* Setting $\mathtt{A} = \mathtt{U}^\mathsf{T}$, we obtain $\mathtt{ACA}^\mathsf{T} = \mathtt{U}^\mathsf{T}\mathtt{U}\cdot\mathrm{diag}(\mathbf{s})\cdot\mathtt{U}^\mathsf{T}\mathtt{U} = \mathrm{diag}(\mathbf{s})$. This means that if the data are expressed in the orthonormal basis $\mathtt{U}$, their vector components are no longer correlated.

Picking the first basis vector $\mathbf{q}^1$ (such that the variance of the orthogonal projection of $P$ onto $\mathbf{q}^1$ is maximal) then amounts to selecting the basis vector of $\mathtt{U}$ with the largest variance (i.e., the first column of $\mathtt{U}^\mathsf{T}$) and so forth.

□

### 4.1.5 Gradient descent

*Input:* function $\mathrm{F} \in \mathbb{R}^n \to \mathbb{R}$, its gradient $\nabla\mathrm{F}$ and initial location $\mathbf{p}_0 \in \mathbb{R}^n$.
*Output:* local minimum of F.

This algorithm is an iterative scheme. The next step is given by the formula

$$\mathbf{p}^{i+1} = \mathbf{p}^i - s\cdot\nabla\mathrm{F}(\mathbf{p}^i), \quad (4.4)$$

where $s$ defines the step length. In our implementation, even the step length is calculated by an iteration:

$$s_{j+1} = s_j - \frac{1}{2}\frac{\mathrm{G}'_j(0)}{\mathrm{G}_j(1) - \mathrm{G}_j(0) - \mathrm{G}'_j(0)}, \quad (4.5)$$

where $\mathrm{G}_j(t) = \mathrm{F}\Big(\mathbf{p}^i - s_j\cdot t\cdot\nabla\mathrm{F}(\mathbf{p}^i)\Big)$. The initial step length $s_0$, and the number of iterations both in the inner and the outer loops, are heuristically chosen constant values.

The iterative scheme for $s_j$ is inspired by the fact that we would evaluate $\mathrm{F}(\mathbf{p}^{i+1})$ anyway, to ensure that it is an improvement over $\mathrm{F}(\mathbf{p}^i)$. Having computed these values, we can already approximate the function G as a quadratic

polynomial, and find its minimum. If the improvement is small enough, we accept the so-obtained step length and proceed with the outer iteration.

It is possible to calculate the optimal step length $s$ using second derivatives. However, such an approach has shown rather unstable in our experiments.

Functions based on image pixel data may change strongly from a pixel to its neighbor. In order for this algorithm not to skip such important details, the initial step length $s_0$ is set to one pixel.

**Claim 5.** *When supplied with a function of the form* $\mathrm{F}(\mathbf{x}) = a|\mathbf{x} - \mathbf{x}_0|^2 + b$ *for* $\mathbf{x}_0 \in \mathbb{R}^n$ *and* $a, b \in \mathbb{R}$*, this algorithm finds the global optimum* $\mathbf{x}_0$ *on the first iteration.*

This algorithm is provided without a proof of convergence. We intend to use it on functions that are very noisy and a radially symmetric quadratic polynomial can hardly even serve as an approximation. Ensuring that some necessary conditions are satisfied by the function F can be much more difficult than the proof itself. We consider this minimization scheme an intuitive heuristic, and we admit that it may fail to converge in some circumstances.

This algorithm is perhaps the simplest optimization scheme applicable on our problems, and its performance can be hindered by many issues that are quite common. For example, it is important that all the parameters of F are expressed in units of a similar scale. In our implementation, we take care of this explicitly either by expressing all the parameters in the same unit (e.g., pixels) or by scaling them by a heuristic factor.

### 4.1.6 Random Sample Consensus

*Input:* set of point pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$ and distance limit $d \in \mathbb{R}$.
*Output:* matrix $\mathtt{H}$ maximizing the count of $k$ such that $|\mathrm{cart}(\mathtt{H}\mathbf{x}^k) - \mathrm{cart}(\mathbf{p}^k)| < d$.

The Random Sample Consensus (Ransac) is a probabilistic and iterative scheme.

Each trial begins by randomly selecting several point correspondencies, called the minimal support set $S_0$. The number of pairs is chosen so that the homography $\mathtt{H}^0$ that relates them exactly, that is $(mn-1)/(n-1)$ for a $\mathtt{H} \in \mathbb{R}^{n\times m}$. The support set $S_{i+1}$ is defined as all point pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$ that satisfy $|\mathrm{cart}(\mathtt{H}^i\mathbf{x}^k) - \mathrm{cart}(\mathbf{p}^k)| < d$.

The homography $H^{i+1}$ is initialized by applying the DLT algorithm on $S_i$. This result is further refined by gradient descent minimization of the geometric error on the subset $S_i$. The iteration stops as soon as $|S_{i+1}| \leq |S_i|$.

In real scenarios we assume that some of the input point pairs can be related by a homography within the distance limit, while others are useless outliers. In order for the iteration to converge to the correct solution, it is necessary that no outlier is selected in the minimal support set.

We repeat the whole iteration schema many times in the hope that at least one of the minimal support sets will be free of outliers.

The probability that after we pick a minimal support set containing no outlier

at least once in $n$ iterations is

$$p = 1 - \left( \binom{t}{s} \binom{t}{u}^{-1} \right)^n, \tag{4.6}$$

given that $t$ is the total number of input pairs, $u$ is the number of outliers and $s$ is the size of the support set. This equation can be solved for $n$ and thanks to the fact that binomial coefficients are well approximated by the normal distribution, it can be incorporated into a program. For more details, see [10, p.117].

## 4.2 Image processing

**Definition 6.** *The cross-correlation of images* $\mathrm{I}, \mathrm{J} \in \mathbb{R}^2 \to \mathbb{R}^n$ *is the function* $\mathrm{I} \star \mathrm{J} \in \mathbb{R}^2 \to \mathbb{R}$ *defined as:*

$$(\mathrm{I} \star \mathrm{J})(\mathbf{p}) = \int_{\mathbf{x} \in \mathbb{R}^2} \mathrm{I}(\mathbf{p} + \mathbf{x})^\mathsf{T} \mathrm{J}(\mathbf{x}) \mathrm{d}\mathbf{x}.$$

*Both of the images are considired zero outside of their bounds.*

### 4.2.1 Normalized cross-correlation

*Input:* images $\mathrm{I}, \mathrm{J} \in \mathbb{R}^2 \to \mathbb{R}^n$.
*Output:* function $\mathrm{C} \in \mathbb{R}^2 \to \mathbb{R}$ that is invariant to local changes in brightness and contrast of I, J but otherwise proportional to $\mathrm{I} \star \mathrm{J}$.

The sought function C is obtained from $\mathrm{I} \star \mathrm{J}$ by an explicit normalization against the changes to brightness and then to contrast. Consider the intersection $R$ of image bounds of I and J. Let us define the box filter $\mathrm{B}(\mathbf{x})$ to be $1/|R|$ when $\mathbf{x} \in R$, and zero otherwise. Here, $|R|$ is used to denote the area of $R$ (assuming that it is a rectangle).

For an image I, let us define the *brightness-normalized image* $\mathrm{I}_b$ as:

$$\mathrm{I}_b = \mathrm{I} - (\mathrm{I} \star \mathrm{B}).$$

Then, let us define the contrast-normalized image $\mathrm{I}_c$ as:

$$\mathrm{I}_c(\mathbf{x}) = \frac{\mathrm{I}(\mathbf{x})}{\sqrt{(\mathrm{I}^2 \star \mathrm{B})(\mathbf{x})}}, \tag{4.7}$$

where $\mathrm{I}^2(\mathbf{x}) = (\mathrm{I}(\mathbf{x}))^2$ is calculated element-wise.

In practice, all the cross-correlations involved are calculated using the Convolution Theorem, by applying the Fast Fourier Transform and a tiling scheme alongside with element-wise operations. Much of them can also be precalculated in advance. the overall computation time is $O(mn \cdot \log(pq))$, where $m \times n$ is the resolution of the larger image and $p \times q$ is the resolution of the smaller one.

### 4.2.2 Circle Hough Transformation

*Input:* radius $r > 0$ and discretized greyscale image $\mathtt{M} \in \mathbb{R}^{n \times m}$ containing a dark circle of radius $r$.
*Output:* center $\mathbf{c}$ of the circle.

Let us have a vote accumulator $\mathtt{A} \in \mathbb{R}^{n \times m}$, initially set to zero. Each image pixel $\mathbf{p}$ casts a vote to the estimated position

$$(i, j)^\mathsf{T} = -r \frac{\nabla \mathrm{I}(\mathbf{p})}{|\nabla \mathrm{I}(\mathbf{p})|} \tag{4.8}$$

with the increment:

$$\mathtt{A}_{j,i} \leftarrow \mathtt{A}_{j,i} + |\nabla \mathrm{I}(\mathbf{p})|. \tag{4.9}$$

The result $\mathbf{c}$ is obtained by fitting a quadratic polynomial around the maximum element of $\mathtt{A}$.

Each sample of the image gradient assumes for a moment that the limbus is passing through it, and makes a guess where the center of the circle would be. Thanks to the fact that the radius $r$ is known a priori, we only need to determine the direction.

This is essentially a simplified version of the general Circle Hough Transformation. The general case does not require the radius to be known a priori. Instead, the radius is estimated for each vote from the *isophote curvature* as in [14; 29]. In our experiments, that method turned out to be very unreliable. The isophote curvature depends upon the second-order derivatives, and these tend to be noisy.

## 4.3 Geometry

### 4.3.1 Derivatives of a three-point affinity

*Input:* set of point pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k, 1 \le k \le 3$ and vector $\mathbf{v}$, where $\mathbf{x}^k, \mathbf{p}^k, \mathbf{v} \in \mathbb{R}^2$.
*Output:* Jacobian matrices $\partial \mathtt{A}\mathbf{v} / \partial \mathbf{p}^k$, where $\mathtt{A} \cdot \hom(\mathbf{x}^k) = \mathbf{p}^k$.

**Definition 7.** *The homogeneous conversion function* $\hom : \mathbb{R}^n \to \mathbb{R}^{n+1}$ *maps a Cartesian vector to its homogeneous counterpart:*

$$\hom(\mathbf{x}) = (\mathbf{x}_1, \ldots, \mathbf{x}_n, 1)^\mathsf{T}.$$

**Definition 8.** *The Cartesian convertor* $\mathrm{cart} : \mathbb{R}^n \to \mathbb{R}^{n-1}$ *maps a homogeneous vector to its Cartesian counterpart:*

$$\mathrm{cart}(\mathbf{x}) = (\mathbf{x}_1 / \mathbf{x}_n, \ldots, \mathbf{x}_{n-1} / \mathbf{x}_n)^\mathsf{T}.$$

Let us define the matrix $\mathtt{C}$ as

$$\mathtt{C} = \left( \hom(\mathbf{x}^1), \hom(\mathbf{x}^2), \hom(\mathbf{x}^3) \right), \tag{4.10}$$

The sought partial derivatives are given by

$$\frac{\partial \mathtt{A}\mathbf{v}}{\partial (\mathbf{p}^k)_i} = \frac{\partial (\mathbf{p}^1, \mathbf{p}^2, \mathbf{p}^3)^\mathsf{T}}{(\partial \mathbf{p}^k)_i} \cdot \mathtt{C}^{-\mathsf{I}} \cdot \mathbf{v}. \tag{4.11}$$

This formula follows directly from the fact that $\mathtt{C}$ is constant wrt. all $\mathbf{p}^k$, and from the following lemma:

**Lemma 9.** *The transformation matrix* $\mathtt{A}$ *is (uniquely) given by the formula*

$$\mathtt{A} = (\mathbf{p}^1, \mathbf{p}^2, \mathbf{p}^3)^\mathsf{T} \cdot \mathtt{C}^{-\mathsf{I}}.$$

*Proof.* Splitting $\mathtt{A} \cdot \hom(\mathbf{v})$ into two matrix multiplications this way has an intuitive motivation. Firstly, the vector $\mathbf{v}$ is expressed as an affine combination of points $\mathbf{p}^k$. This is accomplished by the multiplication by $\mathtt{C}^{-\mathsf{I}}$. We can verify that this combination $\sum_i \alpha_i \mathbf{v}_i$ is affine because the weights sum up to $(\mathtt{C} \cdot (\alpha_1, \alpha_2, \alpha_3))_3 = \hom(\mathbf{v})_3 = 1$.

Then, the same weights can be used to express $\mathtt{A} \cdot \hom(\mathbf{v})$ as an affine combination of $\mathbf{x}^k$. The weights $\alpha_1, \alpha_2, \alpha_3$ are called the *barycentric coordinates.* □

### 4.3.2 Direct Linear Transformation

*Input:* set of point pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$ related by an unknown homography.
*Output:* estimated homography matrix $\mathtt{H}$ .

**Definition 10.** *The vectorization operator* $\mathrm{vec} : \mathbb{R}^{m \times n} \to \mathbb{R}^{mn}$ *is defined as*

$$\mathrm{vec}(\mathtt{M})_{mi+j} = \mathtt{M}_{i,j} \text{ for all } i, j.$$

*This essentially means reading out all the matrix elements in row-major order.*

We are presented with a set projective correspondences of the form

$$\mathtt{H}\mathbf{x}^k = \alpha_k \mathbf{p}^k, 1 \leq k \leq m, \tag{4.12}$$

where $\alpha_k$ are unknown scalars and $\mathtt{H}$ is an unknown matrix that we want to calculate.

Let us define a set of antisymmetric matrices $\{\mathtt{M}^{i,j} \in \mathbb{R}^{n \times n}\}$, $1 \leq i < j \leq n$, each having only two nonzero elements:

$$\mathtt{M}^{i,j}_{i,j} = 1, \mathtt{M}^{i,j}_{j,i} = -1. \tag{4.13}$$

Then, $\mathtt{H}$ is obtained by solving the homogeneous linear system

$$\mathrm{vec}\left((\mathbf{p}^k)^\mathsf{T} \mathtt{M}^{i,j} (\mathbf{x}^k)^\mathsf{T}\right)^\mathsf{T} \cdot \mathrm{vec}(\mathtt{H}) = 0, \tag{4.14}$$

where all $i, j, k$ generate the system rows, under the constraints that $i < j$ and either $\mathbf{p}^k_i$ or $\mathbf{p}^k_j$ is the maximal element of $\mathbf{p}^k$.

There are, loosely speaking, two difficulties in homography fitting as compared to affine transformations. Firstly, it has a nonlinear component so an algebraic least-squares solution (as defined in [10, p.93]) is generally different from a geometric least-squares solution (i.e., the sum of errors). The Direct Linear Transformation computes only former of these two. The relation between these two approaches is addressed in the next section.

Secondly, it is not possible to formulate a linear system straight away. The equations (4.12) do not form a linear system with respect to the unknown matrix $\mathtt{H}$ because the measured data points are defined only up to scale—there is an unknown scale factor $\alpha_k$ involved in each equation. Differences among the scale

factors make up the perspective part of the homography, so although they can make the problem numerically unstable, we cannot impose any limits on these values. The core idea of Direct Linear Transformation is to find a set of vectors that must be orthogonal to the solution, and then solve the resulting homogeneous system.

**Lemma 11.** *The vectors* $\{\mathtt{M}^{i,j}\mathbf{p}\}, 1 \leq i < j \leq n$*, where* $\mathtt{M}^{i,j}$ *is defined by* (4.13)*, are all orthogonal to a given* $\mathbf{p} \in \mathbb{R}^n$*.*

*Proof.* This follows from the definition of $\mathtt{M}^{i,j}$: for any $\mathbf{p} \in \mathbb{R}^n$, the product $\mathbf{p}^\mathsf{T}\mathtt{M}^{i,j}\mathbf{p} = \mathbf{p}_i\mathbf{p}_j - \mathbf{p}_j\mathbf{p}_i = 0$. □

We intend to construct a basis of the subspace $\mathbb{R}^n/\mathbf{p}$ orthogonal to $\mathbf{p}$. With increasing dimension, the set of vectors from Lemma 11 becomes heavily redundant. Although $\{\mathtt{M}^{i,j}\mathbf{p}\}$ generate the subspace in question, there are $\frac{1}{2}n \cdot (n-1)$ of such vectors, whereas only $n-1$ vectors are necessary to form a basis. Some sources suggest to pick an arbitrary subset of size $n-1$ from the matrices $\{\mathtt{M}^{i,j}\}$ and use these for the whole data set.[10] A more proper solution is to select these matrices specifically for each correspondence pair $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$ so as to avoid possible degeneracies. In particular, we find the largest vector element $\mathbf{p}_l$ (in absolute value) and then select all matrices $\mathtt{M}^{i,j}$ such that $i = l$ or $j = l$. Each of the vectors generated this way contains the value of $\mathbf{p}_l$ at a different position (i.e., a different vector element), therefore the vectors are linear independent and they form a basis of $\mathbb{R}^n/\mathbf{p}$.

Now we can formulate the linear system. It will consist of $m \cdot (n-1)$ equations, where $m$ is the number of correspondence pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$. Its unknowns will be precisely the elements of the matrix $\mathtt{H}$. Each equation of the system represents a constraint of the form $\mathbf{p}^\mathsf{T}\mathtt{M}\mathtt{H}\mathbf{x} = 0$, which is the orthogonality constraint as proposed above (from now on, the row-dependent indices $i, j, k$ are omitted for readability). This constraint can be reformulated in terms of the Frobenius inner product $\langle \mathtt{A}, \mathtt{B} \rangle_F = \sum_{i,j} \mathtt{A}_{i,j} \cdot \mathtt{B}_{i,j}$ and the vector outer product as follows:

$$\mathbf{p}^\mathsf{T}\mathtt{M}\mathtt{H}\mathbf{x} = \sum_{i,j} (\mathbf{p}^\mathsf{T}\mathtt{M})_i \mathtt{H}_{i,j} \mathbf{x}_j = \langle \mathbf{p}^\mathsf{T}\mathtt{M}\mathbf{x}^\mathsf{T}, \mathtt{H} \rangle_F = 0.$$

Viewing the matrices as vectors of their elements, this finally leads to an equation in the suitable form (4.14). Each correspondence pair and each selected antisymmetric matrix $\mathtt{M}$ for that pair provide one such equation, and the row vectors $\mathrm{vec}(\mathbf{p}^\mathsf{T}\mathtt{M}\mathbf{x})^\mathsf{T}$ can be stacked to form the system matrix.

**Definition 12.** *Given point pairs* $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$ *and a homography matrix* $\mathtt{H}$*, the geometric error is defined as*

$$\sum_k |\mathrm{cart}(\mathtt{H}\mathbf{x}^k) - \mathrm{cart}(\mathbf{p}^k)|^2.$$

*The cartesian convetor* cart *has been formulated Definition 8.*

In real scenarios, the input point sets are disrupted by noise. As already noted, solving this overdetermined system in least squares sense provides the algebraic least-squares solution. However, if the input vectors are degenerate and only span a subspace of $\mathbb{R}^m$, there are multiple solutions and not all of them are

proper solutions of the original equation $\mathtt{H}\mathbf{x}^k = \alpha_k \mathbf{p}^k$. In particular, it is possible that a solution will produce points at infinity, meaning that $\mathbf{p}_n \approx 0$ and that the Cartesian counterpart of $\mathbf{p}$ is undefined. Instead of avoiding such degeneracies explicitly, we step into the algorithm for homogeneous solving. We loop through the possible choices of $\mathbf{s}_i$ and choose the one that minimizes the geometric error.

As suggested in [10, p.107], it is important to normalize the input points if this estimated matrix $\mathtt{H}$ should resemble the optimal solution. Let us define the sets of *normalized points* $\{\hat{\mathbf{x}}^k\}$ and $\{\hat{\mathbf{p}}^k\}$ using affine transformations $\hat{\mathbf{x}}^k = \mathtt{A}\mathbf{x}^k$ and $\hat{\mathbf{p}}^k = \mathtt{B}\mathbf{p}^k$, where $\mathtt{A}, \mathtt{B}$ are chosen so that the respective point set has zero mean and unity variance. If the DLT algorithm applied to $\hat{\mathbf{x}} \leftrightarrow \hat{\mathbf{p}}$ returns the homography $\hat{\mathtt{H}}$, then the overall solution is $\mathtt{B}^{-1}\hat{\mathtt{H}}\mathtt{A}$.

### 4.3.3 Four-point homography

*Input:* set of point pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$, where $1 \leq k \leq 4$.
*Output:* matrix $\mathtt{H}$ such that $\mathtt{H}\mathbf{x}^k \sim \mathbf{p}^k$.

**Definition 13.** *The similarity operator $\sim$ denotes the equality of homogeneous vectors up to scale. When applied to matrices $\mathtt{A}, \mathtt{B} \in \mathbb{R}^{n\times k}$, the equality is posed between their corresponding columns:*

$$\mathtt{A} \sim \mathtt{B} \Leftrightarrow \exists \mathbf{c} \in \mathbb{R}^k : \mathtt{A} \cdot \mathrm{diag}(\mathbf{c}) = \mathtt{B}.$$

**Definition 14.** *The canonizer function* $\mathrm{can} : \mathbb{R}^{3\times 4} \to \mathbb{R}^{3\times 3}$ *is defined as*

$$\mathrm{can}(\mathtt{A}) = \begin{pmatrix} \alpha_1(\mathbf{b}\times\mathbf{c})^\mathsf{T} \\ \alpha_2(\mathbf{c}\times\mathbf{a})^\mathsf{T} \\ \alpha_3(\mathbf{a}\times\mathbf{b})^\mathsf{T} \end{pmatrix},$$

*where $\alpha_1 \ldots \alpha_3$ are chosen so that* $\mathrm{can}(\mathtt{A}) \cdot \mathbf{d} = (1,1,1)^\mathsf{T}$.

**Definition 15.** *The decanonizer function* $\mathrm{dec} : \mathbb{R}^{3\times 4} \to \mathbb{R}^{3\times 3}$ *is defined as*

$$\mathrm{dec}(\mathtt{A}) = (\alpha_1\mathbf{a}, \alpha_2\mathbf{b}, \alpha_3\mathbf{c}),$$

*where $\alpha_1 \ldots \alpha_3$ are chosen so that* $\mathrm{dec}(\mathtt{A}) \cdot (1,1,1)^\mathsf{T} = \mathbf{d}$.

The sought four-point homography is given by

$$\mathtt{H} = \mathrm{dec}(\mathbf{p}_1, \ldots, \mathbf{p}_4) \cdot \mathrm{can}(\mathbf{x}_1, \ldots, \mathbf{x}_4). \tag{4.15}$$

A minimal case of a 2-dimensional homography estimation is a correspondence of four point pairs. If the input points are in a general configuration (i.e., none three are collinear), then there is an exact solution—no least squares fitting necessary. The general formula as generated by Direct Linear Transformation would be needlessly bloated for this setting.

Let us declare the following set as the *canonical configuration* $\mathtt{C}$:

$$\mathtt{C} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

The formulas given above can be used to convert an arbitrary four-point set to and from the canonical configuration.

**Lemma 16.** *For an affine independent four-point set* $\mathtt{A}$*,*

$$\mathrm{can}(\mathtt{A}) \cdot \mathtt{A} \sim \mathtt{C} \text{ and } \mathrm{dec}(\mathtt{A}) \cdot \mathtt{C} \sim \mathtt{A}$$

*Proof.* The proof follows directly from the definition of can($\mathtt{A}$) and dec($\mathtt{A}$). □

This already proves that $\mathtt{H}\mathbf{x}^k \sim \mathbf{p}^k$ as required.

Note that evaluating the function can($\mathtt{A}$) requires no linear solving, just an element-wise division. In order to evaluate dec($\mathtt{A}$), we have to solve the $3\times3$ linear system $(\mathbf{a}, \mathbf{b}, \mathbf{c})\cdot(\alpha_1, \alpha_2, \alpha_3)^\mathsf{T} = \mathbf{d}$. Note that the vector $\mathbf{d}$ is exactly the right-hand side of this system. A neat side-effect is that the inverse matrix $\mathrm{dec}(\mathtt{A})^{-\mathsf{I}} = \mathrm{can}(\mathtt{A})$ has correct scale. More importantly, it means that the mapping dec is linear with respect to all elements of $\mathbf{d}$.[2]

### 4.3.4 Derivatives of a homography

*Input:* set of point pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$, where $1 \le k \le 4$ and vector $\mathbf{v} \in \mathbb{R}^2$.
*Output:* Jacobian matrices $\partial\mathtt{H}\mathbf{v}/\partial\mathbf{p}^k$, where $\mathtt{H}\mathbf{x}^k \sim \mathbf{p}^k$.

The partial derivative wrt. the $i$-th coordinate of the last point is given by

$$\partial\mathtt{H}\mathbf{v}/\partial\mathbf{p}_1^4 = \mathtt{H} \cdot \mathrm{dec}(\mathbf{p}^1, \mathbf{p}^2, \mathbf{p}^3, \mathbf{e}^i) \cdot \mathbf{v}, \tag{4.16}$$

where $\mathbf{e}^i$ is the natural basis vector for the $i$-th coordinate and $\mathtt{H}$ is exactly as in (4.15). The remaining three matrices are obtained by a permutation of the input points.

*Proof.* This follows from the linearity of $\mathtt{H}$ wrt. $\mathbf{p}^4$, by applying formulas for derivative of the matrix product. □

It seems that we could use the fact that any permutation of the canonical points can be expressed as a homography,[3] and incorporate it into (4.16). Unfortunately, the unknown scale factors make such an approach inefficient. The preferable method is therefore to explicitly permute the points so that the point in question becomes the last one, and to repeat the original scheme.

In our implementation, we need to evaluate the Jacobian matrices in each image pixel. Note that all the matrices can be precalculated as long as the homography $\mathtt{H}$ is constant. What remains then are four matrix multiplications per pixel—and some extra cost due to the conversion to Cartesian coordinates.

[2] The mapping is not linear with respect to $\mathbf{a}, \mathbf{b}$ nor $\mathbf{c}$. However surprising may this asymmetry seem, it is caused by the unknown scale factors.

[3] The homography $\mathtt{R} = \begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{pmatrix}$ has the effect $\mathtt{R} \cdot \mathtt{C} \sim \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$.

# 5. Implementation

Let us now use the theory and knowledge from previous two chapters to design a gaze tracker. After sketching out the basic traits of the program, we continue by listing out various approaches to each of the subproblems. There are indeed many options which make difference in computational difficulty, precision and robustness to possible difficulties in the input data. They are presented here, and will be evaluated in the next section under varying conditions.

## 5.1 Overview

On a first run, the program needs a calibration procedure to learn specific parameters related to the user. Some calibration is also necessary on each run and during longer periods of execution because of changing light conditions.

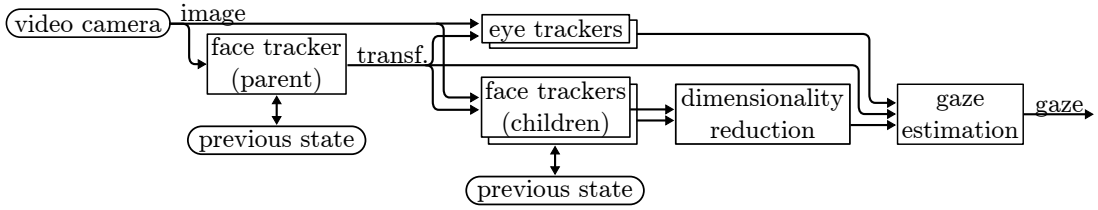



Figure 5.1: Overall scheme of gaze tracking.

The pipeline of computation is presented in Drawing 5.1. Firstly, the face is locally tracked from the previous frame by the so-called parent tracker. Only if this step fails, the program executes a global face localization procedure. This first step results in a geometric image transformation.

Next, the precise position of both eyes is established within a reasonable region in the face. This step requires the face to be localized properly in the image but since the eyes can move very quickly, no information from previous frames is taken into account.

Finally, the 6 degrees of freedom of the head and the 2 degrees of freedom of each eye are fed into a gaze estimator that outputs coordinates in screen reference frame.

## 5.2 Face tracking

*Input:* reference image R and current image I.
*Output:* geometric transformation $\mathrm{T} : \mathbb{R}^2 \to \mathbb{R}^2$ such that $\int_{\mathbf{p}\in\mathbb{R}^2} |\mathrm{R}(\mathbf{p}) - \mathrm{I}(\mathrm{T}(\mathbf{p}))|^2$ is minimal.[1]

The tracking is split into two parts: a *parent tracker* that covers the whole face, and several *child trackers* that capture more subtle degrees of freedom. All of the trackers are deformable templates along the lines of [2].

[1] Both of the images are considered zero outside their bounds.

The reference image $R$ is a single frame that is used during the whole course of operation. Although this approach does not tolerate long term changes (such as when the user takes on their glasses), it makes our method robust against slow drift of the trackers.

Each of the trackers minimizes the energy function

$$\int_{\mathbf{p}} |\mathrm{R}(\mathbf{p}) - \mathrm{I}(\mathrm{T}(\mathbf{p}))|^2,$$

where $T : \mathbb{R}^2 \to \mathbb{R}^2$ is a differentiable image transformation. Note that we do use the color information—this seems to be quite rare in the literature.

Depending on the motion model expressed by the transformation T, we get different kinds of trackers that also differ by their degrees of freedom (DoF):

- Similarity transformation: 4 DoF. The transformation is parametrized by a displacement vector, scale and rotation angle.
- Affinity over a quadrangle: 6 DoF. The transformation is parametrized by its $2 \times 3$ affinity matrix.
- Affinity over a triangle: 6 DoF. This motion is advantageous over the previous one because its parameters exactly correspond to the DoF number of the transformation. The parameters are exactly the vertices of a triangle that defines the tracker area. This is important, as explained below.
- Homography. 8 DoF. Again, the parameters of this motion model are exactly the four vertices that define the tracker area. Homography encompasses all possible views of a planar object.

For performance reasons, a motion model must be chosen at compilation time, and it is shared by all the trackers.

We discretize the formula as a pixel-by-pixel sum and optimize it by gradient descent. The derivatives of the transformation are computed analytically. In case of the homography motion model, these derivatives are not quite obvious to formulate, and as such, they have been described in Section 4.3.4. The formulas for calculating a triangle affinity are comparatively simple, but nevertheless, they have been summarized in Section 4.3.1.

In order to allow large-scale movement, all of the trackers operate upon an *image pyramid*. The pyramid is defined by downsampling the input image to half its resolution in a recursive manner. The top level of the pyramid is an image too small to be downsampled. The tracking starts by a quick estimate on the top level, and this information is successively refined until the bottom level with the original input resolution.

Assuming that the top level of the pyramid is correct, it is enough to refine the trackers up to the current pixel scale. Any greater displacement should have been handled in the levels above. Thanks to this assumption, only few steps of gradient descent (possibly just one) are necessary on each level of the pyramid. That in turn allows us to track large displacements in real time.

The parent tracker provides a reference frame for all other features (e.g., eye trackers) and it supplies the most important parameters to gaze estimation. In

order to gain some robustness against changes in lighting, the parent tracker is color-normalized.

Our program supports two methods how the child trackers are managed. These represent two major groups of tracking techniques in general: the feature-based and the appearance-based method.

### 5.2.1 Markers

This method relies on small-scale features that are usually present in the face and on skin—such as the eyebrows, nostrils, beard and skin defects. Several small trackers are arbitrarily placed within the face area, each of them large just enough so as to track a single feature of the face. Technically, it is possible to track any distinctive texture, so an automated feature detection such as the Harris corner detector (provided by a third-party library) performs well enough.

Each of the markers is allowed to freely move around. (In contrast to the classical Deformable Parts Model such as in [28], we do not calculate any penalty from their relative configuration.) They are tracked from their previous known position relatively to the parent tracker. If the tracking score of a marker drops below a certain limit or a marker escapes the face area, it is reset to its original position and tracking continues from there.

### 5.2.2 Grid

This method follows the approach of Active Appearance Models [4]. The parent tracker is subdivided into a planar mesh, and each cell is responsible of tracking the corresponding part of the image.

In order for the cells to form a contiguous mesh, certain parameters of neighboring cells are bound together and need to be optimized concurrently. Our implementation of the corresponding tracker methods does this explicitly since their parameters are the planar coordinates of their corner vertices. During optimization, the derivatives are summed up in each grid vertex (nevertheless the number of cells that share it) and all grid vertices are updated at once.

This process is commonly done with a triangular mesh (i.e., barycentric cells), where the derivatives wrt. vertex coordinates are quite easy to derive. Our extension of this method to a quadrangular mesh (i.e., homographic cells) relies on the concepts developed in sections 4.3.3 and 4.3.4.

## 5.3 Eye tracking

*Input:* iris radius $r \in \mathbb{R}$ and image I centered at an expected eye location.
*Output:* iris center $\mathbf{c} \in \mathbb{R}^2$.

Once the parent tracker for the face has been precisely localized, the eyes can be easily located using their reference position. The iris radius can also be queried from the parent tracker, given that a radius has been marked in the reference image.

In order for the eye tracking to succeed, it is necessary that these estimates obtained by the face tracker are close to their actual value. Furthermore, it is

necessary that a large enough portion of the iris is visible. This condition depends on the particular algorithm being used but in general, we require something like a half of the iris to be visible. There is no hard limit but the recognition performance clearly decreases with a strong occlusion of the iris.

There are many more important factors that can hinder the performance of eye tracking. Some of these are mentioned in the specific algorithms that follow, and a global overview of the challenges is given in Section 6.2.1. By experience, we believe that these degradations are actually quite common in the real world, so we should not strictly forbid them. However, the problem of eye tracking becomes a difficult one in these adverse conditions.

A plenty of eye trackers have been implemented and tested in our program. From a broader perspective, we took four different approaches:

- Tracking the limbus. There should be a circle with a strong gradient directed outwards.
- Tracking the iris. It should be radially symmetric with a strong gradient on the edges.
- Segmentation. Skin, sclera, iris and pupil should each be uniform, but mutually different in color.
- Machine learning. Instead of modeling, we can feed the image into a regression engine.

Most of the face trackers induce non-uniform scaling, so it might seem appropriate that the eye shape will be an ellipse. However, we assume that eyes are always directed towards the camera, so the limbus should retain a circular shape even if the face is stretched. Tracking an ellipse would an undesirable flexibility, making the calculation prone to failure.

Acknowledging that the scale (e.g., the radius $r$) can be only a few pixels, we aim for a subpixel precision. That is a meaningful pursuit thanks to the continuous image model developed in Section 3.5.

However, many of the algorithms to be presented are based on pixel by pixel evaluation of a scoring function. In such cases, it is not efficient to evaluate at fractional coordinates. Instead, we find the pixel with the maximum score by an exhaustive search. Then we fit a quadratic polynomial to several score values around the maximal pixel, and output the maximum of this polynomial as the global maximum.

The following methods are available to the host application:

### 5.3.1 Limbus gradient

Limbus is the edge between the iris and the sclera. Assuming that the iris is much darker than the sclera, the limbus should be sensed as a circle of high contrast. Curve fitting to detected edge pixels can provide very precise results [13]. Given that we expect to work with images of low quality, the usual approach based on a Canny filter and ellipse fitting is not viable. The Canny filter requires careful tuning of parameters with respect to the amount of blur present in the image.

Instead, we propose an iterative optimization scheme that will directly fit a circle to the image gradient. The energy function is given as

$$\mathrm{E}(\mathbf{c}, r) = \int_{\mathbf{x}\in S} (\mathbf{x} - \mathbf{c})^{\mathsf{T}} \nabla \mathrm{I}(\mathbf{x}) \mathrm{d}\mathbf{x}, \tag{5.1}$$

where $S = \{\mathbf{x} : |\mathbf{x} - \mathbf{c}| = r\}$. This energy is minimized by gradient descent.

As simple as it seems, the formula is motivated by the fact that the gradient on circle boundary is oriented precisely outwards from the iris center. The dot product with the radial vector $\mathbf{x} - \mathbf{c}$ is maximal if our estimate of $\mathbf{c}$ is correct.

We have experimented with applying a transition function on the dot product, to obtain a more robust tracker. For example, it is reasonable to completely ignore pixels where the dot product is negative. Our program can be easily modified to incorporate such a generalization but our experimets show that the improvement is questionable.

It is important to note that we optimization of a function based on image gradient requires image derivatives of second order to be calculated. We have previously stated that these are better avoided, so this new decision deserves an explanation.

This method is mainly designed for refining an estimated eye position on the subpixel level. We assume that the center is already initialized only a few pixel from the optimal solution, so that the strong gradient around the limbus will overrule the noise present in the derivatives. If this algorithm is executed alone, it usually fails to locate the global optimum.

### 5.3.2 Hough Transformation

We can use a voting scheme to find the limbus center, as described in Section 4.2.2. This scheme is used quite often in the literature, e.g., [14; 36].

A free parameter remains to be defined: the resolution of the voting grid. Setting the resolution too high can result in the votes being distributed quite randomly over a wide region around the true center. Especially in the case of blurry or noisy images, it may be reasonable to choose a coarser grid than the actual image resolution.

In fact, we can extend this method to the image pyramid, as defined in Section 5.2. Results from downsampled versions of the input image are added up to the detailed ones in order to agglomerate nearby votes. However, it is necessary to express how the votes are upsampled and summed up to the final accumulator. That means even more free parameters.

### 5.3.3 Dark iris correlation

Assuming that the iris is a dark disc and the rest of the image is much brighter, we can use basic template matching techniques to locate it. This technique is very popular [7; 38]. We obtain some robustness at the expense of computational speed by using the normalized cross-correlation.

Our iris template $\mathrm{T} : \mathbb{R}^2 \to R$ is a black circle on a white background, blurred with a box filter:

$$\mathrm{T}(\mathbf{x}) = \frac{1}{2} + a(|\mathbf{x}| - r), \tag{5.2}$$

where $a$ is an arbitrary constant, and the result is clamped to the range $[0, 1]$.

This method appears to work especially well when tested in geographical regions where the majority of people are brown-eyed and with white skin. It can easily get confused by dark spots around the eyes, such as glass rims or strong makeup. Finally, this method requires most of the iris to be visible, which need not be true because of the viewing angle and the user's personal habit.

### 5.3.4 Personalized iris correlation

The assumption of dark iris is wrong: the iris is often much brighter than the surrounding skin, especially in the case of blue-eyed people. In fact, the iris image varies so greatly among individuals that it is often advocated as a means of identification or authentification [3]. We can, however, rely on the whole iris region (including the pupil) to be a constant, radially symmetric image, and locate it using a personalized template.

In this method, a user-defined template is tracked using the normalized cross-correlation. In contrast to the previous method, it is possible to precisely define the radial color gradient of the iris and to adjust how much sclera should be taken into account. For this flexibility, this tracker has been included in the application—although experiments so far show its performance as poor.

The template is an image from the hard disk, its acquisition must be done offline. The resolution of this image should not be excessively high as it only hurts the computation speed. However, the image should be taken with care and with the eyes wide open, so that the whole limbus is visible.

### 5.3.5 Iris radial symmetry

It may be considerably difficult to obtain a reliable iris image, and generally speaking it is a calibration step that we wish to avoid. Instead, we can use the sole fact that the iris is a radially symmetric object on a white background. An example of a filter based on radial symmetry can be found in [14]. Hereby we design a novel robust algorithm that exploit color information.

The iris colors are recalculated for each frame and each eye position, so only few free parameters remain. This algorithm has built-in skin masking, so that whole segments of the iris are ignored where the limbus is not visible.

Let us define the following functions:

- Angular limbus score $\text{limbus}(\alpha)$ is the gradient magnitude $|\nabla \text{I}(\mathbf{p})|$ for vector $\mathbf{p}$ defined by $\arctan(\mathbf{p}) = \alpha$ and $|\mathbf{p}| = r$.
- Radial color $\text{mean}(t)$ is a mean value of the set $\{\text{I}(\mathbf{p})$ for all $|\mathbf{p}| = t\}$.
- Angular iris score $\text{iris}(\alpha)$ is the weighted arithmetic mean of the set $\{\text{I}(\mathbf{p})$ for all $\arctan(\mathbf{p}) = \alpha\}$. The weight is $\text{w}(\mathbf{p}) = t \cdot (c - |\text{image}(\mathbf{p}) - \text{mean}(t)|)$, clamped to zero, with $t = |\mathbf{p}|$ and $c$ being a value appropriately chosen.

The total score is defined as $\int_\alpha \text{limbus}(\alpha) \cdot \text{iris}(\alpha)$.

Note that the formula for $\text{mean}(t)$ has not been specified yet. Using the arithmetic mean may have a bad impact on the overall success rate. It is appropriate here to use a more robust formula such as the median.

There are essentialy two ways how to generaze the median to color pixels. Firstly, we can pick the median value as sorted by a scalar value (e.g., brightness). This approach becomes unpredictable if many different colors have the same brightness, and specifically in the case of skin tones and iris color, this may be an issue.

The second option is to define the median of set $S$ as the value $\mathbf{m} \in S$ such that the sum of distances $\sum_{\mathbf{x} \in S} |\mathbf{m} - \mathbf{x}|$ is minimized. This formula is consistent with the one-dimensional case and is valid in any metric space. Unfortunately, efficient algorithms to compute this value are too sophisticated, so we loop over $S$ explicitly.

This function is computationally expensive and, depending on the method for mean color calculation, difficult to optimize locally. Our program does not even contain the code to calculate the derivatives. This tracker uses exhaustive search to find the global minimum within a crop-out image.

### 5.3.6 Skin masking

Skin, including the eyelids, can be detected quite easily by an analysis of its color. This approach is perhaps too naive to be advocated in the literature, but it can be found in software libraries such as [21].

We allow several of the eye trackers presented so far to be coupled with a color detector tuned for skin tones. In each pixel, the hue, saturation and value (HSV) is estimated. Pixels whose HSV vector lies within a user-defined cube are excluded from eye fitting.

The method is based on the following heuristic mapping. Given a color vector $\mathbf{c} = (r, g, b)$ consisting of the red, green and blue channel, we define $\mathrm{hsv}(\mathbf{c}) = (h, s, v), h \in [0, 6)$. In order to present simple formulas for $s$ and $h$, we begin by stating two symmetries in this mapping:

$$\begin{aligned} \mathrm{hsv}(b, r, g) &= (h + 2, s, v), \text{ and} \\ \mathrm{hsv}(b, g, r) &= (6 - h, s, v). \end{aligned} \tag{5.3}$$

Then, assuming without loss of generality that $r \geq g \geq b$,

$$\begin{aligned} v &= \frac{1}{3}(r + g + b), \\ s &= \frac{1}{v}(r - b), \text{ and} \\ h &= \frac{g - b}{r - b}. \end{aligned} \tag{5.4}$$

We should explain the heuristic aspects of this approach. In fact, many color spaces have been proposed that calculate the hue, saturation and value in a similar manner. There are also several color spaces based on human color perception that can be used for the purpose of color filtering.

We do not pay much attention to the exact choice of the HSV mapping because the whole idea of skin masking does not stand on solid ground. Due to possible changes of lighting and variation of skin tone among users, it is impossible to classify skin based on its color alone. This method is provided only as a small helper.

Since there are usually many distractive features around the eyes, such as the eyelashes and makeup, the mask obtained by this HSV thresholding is processed by a morphology filter so as to have a smooth boundary and to cover nearby black splotches.

### 5.3.7 Combined estimator

Having implemented several algorithms for the same task, it is possible to adaptively combine their results. An example of two different algorithms summing their results can be found in [14]. A more common approach is to run several trackers in sequence, such as in [7; 31; 38]. Our program supports can combine both of these options in a tree-like manner arbitrarily.

Choosing several trackers that are sensitive to different deteriorations of the image, we can trade off some computational time for much robustness and precision. Each of the trackers chosen should obviously fail in different conditions so that a majority of them is correct on every image. For an empiric comparison on reasonable selection of trackers for a combined estimator, see Section 6.2.1.

After letting each of the selected trackers cast a vote, it is possible to combine them robustly (as compared to the arithmetic average). It is very much possible that one or more of the trackers fail. We have very little prior information about the eye position, but we can exploit the distances among the votes.

In the case of three trackers running in parallel, we can heuristically throw out the vote that is significantly far away from the remaining two. For a higher count of votes (assuming still that the number is reasonably small), this can be generalized as the subset with minimal expected variance. In order to find it, we explicitly loop over all the subsets. The average of this subset is the result of the combined estimator.

Note that in the case of two votes, their arithmetic mean is the only option. In the three-vote case, either a mean of two or the mean of all three can be selected depending on their relative distances.

**Claim 17.** *Given a set $X = \{\mathbf{x}^i\}$ of samples from a random distribution, the expected value of the variance* $\operatorname{var} X$ *is*

$$E \operatorname{var}(\mathbf{x}) = \frac{1}{|X| - 1} \sum_i (\mathbf{x}^i - \hat{\mathbf{x}})(\mathbf{x}^i - \hat{\mathbf{x}})^\mathsf{T},$$

*where $\hat{\mathbf{x}}$ is the arithmetic mean of $X$.*

Another option is to run several algorithms in sequence. In this setting, the first algorithm provides a rough estimate that is subsequently refined by the remaining ones. Since most of our eye trackers use exhaustive pixel-by-pixel search, there are only few meaningful configurations of this kind.

## 5.4 Gaze estimation

*Input:* vectors $\mathbf{e} \in \mathbb{R}^n$ and $\mathbf{f} \in \mathbb{R}^m$ and *gaze parameters* as defined below.
*Output:* vector $\mathbf{p} \in \mathbb{R}^2$.

**Definition 18.** *The gaze parameters consist of an orthonormal basis* $\mathtt{P} \in \mathbb{R}^{2\times n}$ *and a homography* $\mathtt{H} \in \mathbb{R}^{3\times(m+3)}$.

Given the vector **f** obtained from face tracking and vector **e** obtained from eye tracking, and using the scene model described in Section 3.3, we wish to estimate the on-screen position **p** (in pixel units). The methods commonly used for this purpose vary greatly from simple approximations [38] more flexible ones [13; 34] and thorough geometric models [30; 31].

Computer screen is a plane in space, with pixels aligned in a regular square grid. If the face and the eyes were also planar objects, then the gaze in screen coordinates would be related to the face and eyes position by a projectivity. This is obviously not the case, and the image of the face exhibits many more degrees of freedom than a planar rigid body would.

In order to properly model the mapping from our face and eye parameter space to 2-dimensional gaze, we would have to actually approximate the three-dimensional model of the face. Many programs have been published (e.g., [6]) that follow this path. In general, it requires some prior information about the shape of the face, and also much tuning to calibrate a 3d model precisely enough. To avoid these issues, we decided for a simpler approach.

We decided to actually assume the gaze is given by a projective transformation of the face and eye parameters. This approach is limited and even in theory, it does never perfectly fit the data. On the other hand, homographies have a solid mathematical background, and they can be estimated quickly and robustly from data. All invertible homographies form a group that contains the group of invertible affine transformations, and they can also implicitly model inverse proportionalities among their parameters.

It is meaningful to extract four basic parameters about the parent face tracker: its position within the image, its on-screen rotation and scale. The head (as a rigid body) has two more parameters to be covered, but there is a virtually unlimited number of extra parameters that can be obtained by a detailed analysis of the face using the child trackers. Face pose is unknown in each frame, and possibly constant, so there is a high danger of overfitting. We cut down the dimensionality by Principal Component Analysis: out of the all the child tracker parameters, only the two largest principal components are preserved.

It is possible that the user's gaze twitches for a moment, or that the gaze tracking fails in several frames. In order to ignore these faulty measurements, we employ a Ransac fitting.

## 5.5 Calibration

*Input:* interactive session with the user.
*Output:* gaze parameters (see Definition 18).

The purpose of calibration is to learn all necessary parameters about the user's face and the geometry of the screen relatively to the camera. Depending on the tracking quality, the time spent to measure all necessary information may vary.

In the beginning of the calibration session, a single frame is acquired from the camera. Depending on the application, either the program locates the user's

face and eyes, or asks the user to do so manually. For the automatic face and eye localization, a boosted random forest classifier is used within a sliding window.

The calibration session proceeds by presenting the user with a dot moving around the screen. The user is asked to watch the dot carefully until it disappears. In order to make the movement more predictable, the dot moves along a smooth curve with piecewise constant curvature, and a patch of this curve is being drawn ahead of time. The calculated parameters from face tracking are recorded along with the current on-screen position of the dot.

Every several samples obtained this way, we try and apply the Ransac-based homography estimation as described in the previous section. As soon as a fit collects a large enough support of measurements, the calibration procedure stops and outputs the homography it acquired.

We should also note that our program supports a non-interactive method of calibration, for testing purposes on recorded videos. In this case, an extra data file must be provided that lists out the gaze coordinates (i.e., the ground truth) in each frame of the video.

# 6. Results

Having designed and implemented the gaze tracker, we shall now evaluate its performance. There are three aspects to consider when talking of performance: robustness, precision and speed.

## 6.1 Face tracking

In Figure 6.1, we demonstrate the face motion models by fitting a transformation between a pair of images.

The performance of the face tracking methods has not been evaluated in any exact manner because it is difficult to define the ground truth. We could, for example, evaluate their performance on artificial images not related to faces. However, that does not evaluate the algorithms—it would be hardly more than a test of the implementation.

Note that in the case of the triangle affinity tracker, the area of interest is drawn as a quadrangle. This is only a flaw in the user interface: in fact, the bottom two vertices are collapsed together to form the third vertex of a triangle.

## 6.2 Eye tracking

For a start, we compared the overall performance on all data available.

There are two combined trackers in addition to all the simple ones. The one called *serial* is a Hough tracker and a Limbus gradient tracker running in series, in this order. In the tracker called *serioparallel*, we replaced the simple Hough tracker with a parallel combination of Correlation, Hough, and Radial trackers. Again, the result is refined by the Limbus gradient tracker afterwards.

| **Algorithm** | **Mean [px]** | **Q1 [px]** | **Q2 [px]** | **Q3 [px]** |
|---|---|---|---|---|
| serial | 7.30 | 0.59 | 0.90 | 1.43 |
| hough | 8.26 | 0.98 | 1.36 | 2.02 |
| serioparallel | 11.97 | 0.75 | 1.45 | 5.67 |
| radial | 12.11 | 0.89 | 1.73 | 4.44 |
| correlation | 16.48 | 0.97 | 1.99 | 6.83 |
| bitmap | 28.55 | 3.76 | 6.54 | 13.62 |
| limbus | 35.23 | 5.77 | 10.71 | 16.67 |

Table 6.1: Algorithm mean error and quartiles.

Looking at the quartiles Q1 and Q2, it seems that there are many nice cases where the algorithms perform well. Indeed, Figure 6.2 shows the results of each of the trackers are sorted by distance, so that the error distribution becomes apparent.

We evaluated the algorithms separately on each of our data sets separately. Example images from each of these can be seen in Figure 6.3, and they are described in Attachment B.

Apparently, the simple tracker based on Circle Hough Transformation provides the best result. Its performance can be improved on subpixel scale by adding an additional step of the Limbus gradient optimization.

It is quite disappointing that a voting scheme based on three of our trackers appears to perform worse than the best of them.

We should note that our Radial tracker is considerably slower than all the remaining ones. On overly large-resolution images, its execution time can actually harm the overall framerate of gaze recognition.

### 6.2.1 Failure factors

Apparently, the results of eye tracking depend heavily on the qualitative aspects of each image. For this reason, we annotated each of our testing samples in the several regards, as follows. Each of them is evaluated on a subjective scale:

- **Iris** brightness: 1 (not distinguishable from the pupil) to 3 (bright blue)
- Image **contrast**: 1 (image is hardly visible) to 3 (perfect)
- **Glare** or reflection: 1 (not visible), 2 (only a few pixels), 3 (large)

Apart from this, each eye sample has already been labeled with the iris radius.

As expected, there is a strong correlation between many of these quantities:

| **Algorithm** | **Radius** | **Iris** | **Contrast** | **Glare** |
|---|---|---|---|---|
| serial | 21.44 | 0.17 | −0.14 | −0.49 |
| hough | 16.68 | −0.11 | −0.09 | −0.49 |
| serioparallel | 33.03 | 1.80 | 0.19 | 0.73 |
| radial | 37.92 | 0.61 | 0.02 | 0.76 |
| correlation | 34.09 | 3.30 | −0.06 | 1.29 |
| bitmap | 72.23 | 0.74 | 0.54 | 1.43 |
| limbus | 46.57 | 0.54 | 0.37 | 0.41 |

Table 6.2: Covariance of mean error wrt. image properties.

For completeness, we repeated this test with each of our data sets. These detailed results are provided in tables 6.4, 6.5 and 6.6. The overall eye tracking performance on each of the data sets is listed in Table 6.3.

The Eyes data set is apparently much more challenging than the remaining two. The relative advantage of the Correlation tracker on the Instagram data set can be explained by the fact that most of the eyes in that data set are dark, whereas the Models data set contains a prevalence of blue eyes.

Clearly, the performance of the Correlation tracker is harmed by bright iris color. This is confirmed consistently in all of the data sets. The serioparallel tracker has been designed with this fact in mind, so that if correlation tracking fails, the remaining two votes should still provide a good estimate. It seems that the combination technique is inefficient because it improves only few of the results.

| Algorithm | Instagram | Models | Eyes |
|---|---|---|---|
| serial | 6.43 | 6.72 | 23.98 |
| hough | 7.52 | 7.74 | 22.80 |
| serioparallel | 10.64 | 12.44 | 22.40 |
| radial | 11.37 | 11.44 | 28.36 |
| correlation | 14.18 | 18.42 | 22.02 |
| bitmap | 31.04 | 24.39 | 45.47 |
| limbus | 35.53 | 33.50 | 51.08 |

Table 6.3: Algorithm mean error on each dataset.

| Algorithm | Radius | Iris | Contrast | Glare |
|---|---|---|---|---|
| serial | 4.08 | 0.50 | −0.13 | −0.59 |
| hough | 4.12 | 0.23 | −0.08 | −0.74 |
| serioparallel | 14.00 | 2.06 | 0.05 | 0.80 |
| radial | 11.80 | 0.81 | −0.11 | 0.77 |
| correlation | 17.48 | 3.27 | −0.24 | 1.28 |
| bitmap | 41.83 | 1.54 | 0.03 | 2.17 |
| limbus | 19.12 | 1.06 | −0.10 | 0.89 |

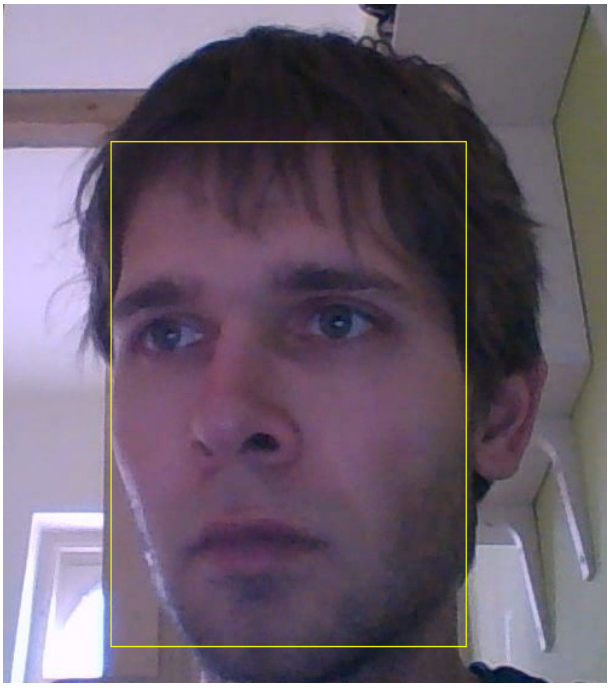
Table 6.4: Error covariance on the Instagram data set.

| Algorithm | Radius | Iris | Contrast | Glare |
|---|---|---|---|---|
| serial | 21.77 | −0.23 | −0.09 | 0.13 |
| hough | 16.66 | −0.47 | −0.06 | 0.11 |
| serioparallel | 34.50 | 0.95 | 0.25 | 0.76 |
| radial | 31.32 | 0.29 | 0.04 | 0.85 |
| correlation | 47.50 | 2.24 | 0.23 | 1.32 |
| limbus | 19.90 | 0.13 | 0.29 | 0.23 |
| bitmap | 55.13 | 0.78 | 0.53 | 1.32 |

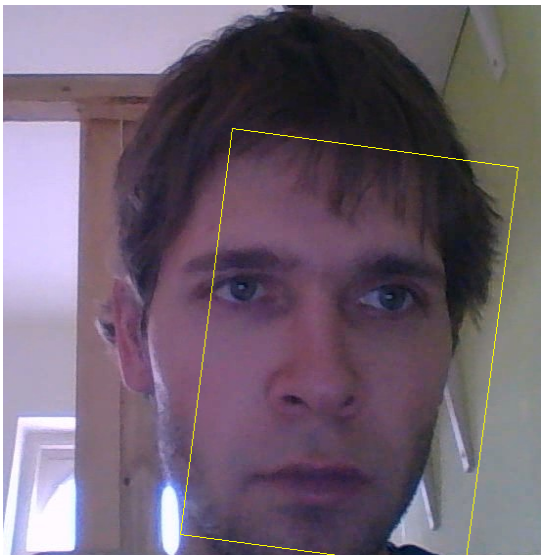
Table 6.5: Error covariance on the Models data set.

| Algorithm | Radius | Iris | Contrast | Glare |
|---|---|---|---|---|
| serial | 99.61 | −1.39 | −2.67 | 1.51 |
| hough | 56.33 | −1.97 | −2.16 | 2.54 |
| serioparallel | 194.48 | 3.59 | 0.44 | 4.07 |
| radial | 301.92 | 0.13 | −0.37 | 7.31 |
| correlation | 109.67 | 5.91 | −0.99 | 2.78 |
| limbus | 524.95 | 2.16 | 4.79 | 4.64 |
| bitmap | 392.63 | 4.26 | 3.33 | 3.20 |

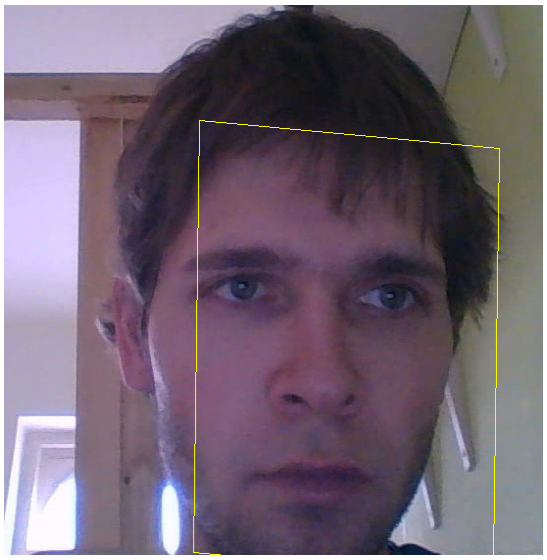
Table 6.6: Error covariance on the Eyes data set.



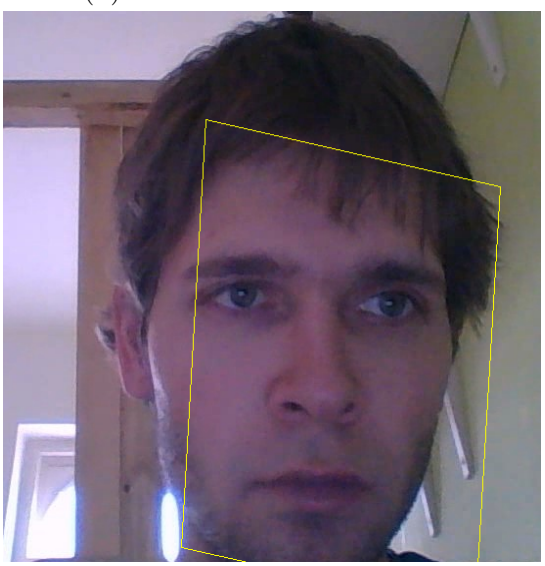
(a) The reference image.



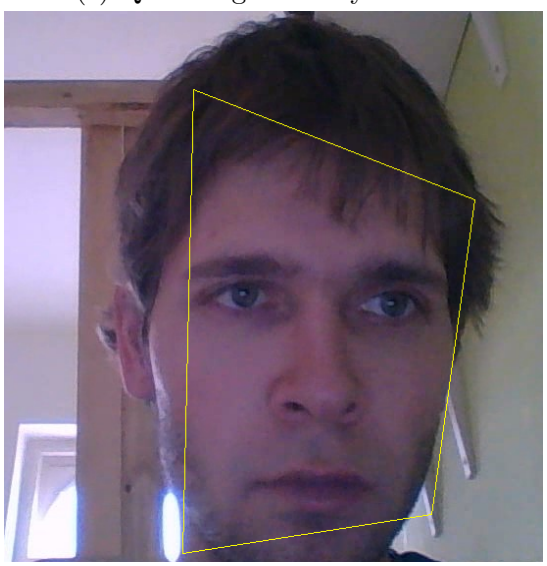
(b) Location-rotation tracker.



(c) Quadrangle affinity tracker.



(d) Triangle affinity tracker.



(e) Homography tracker.



Figure 6.1: Examples of the master tracker performance on two images.

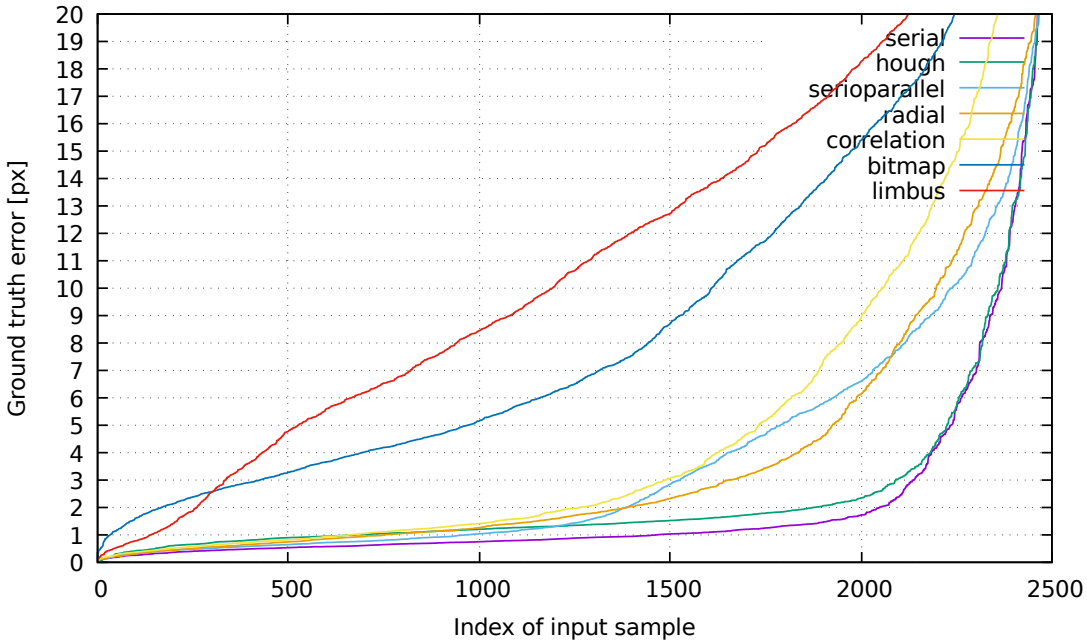
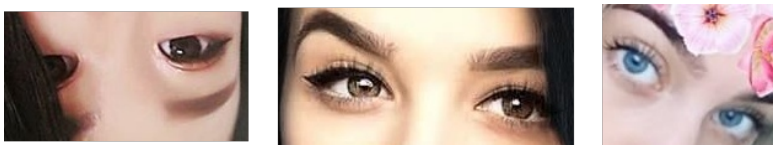
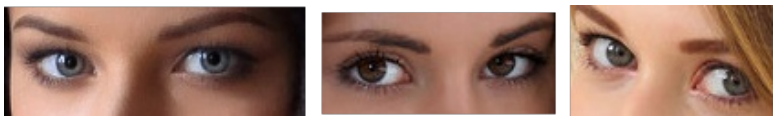


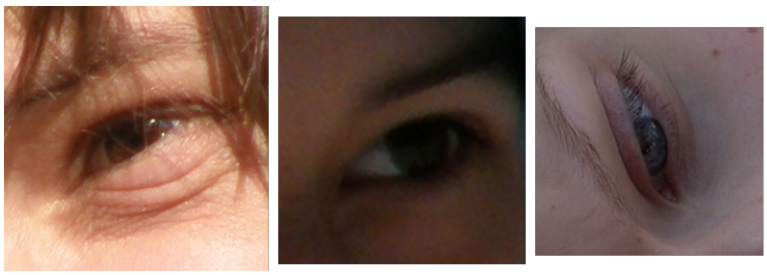

Figure 6.2: Comparison of the eye trackers, each data row sorted by distance.



(a) “Instagram” data set.



(b) “Models” data set.



(c) “Eyes” data set.

Figure 6.3: Example files from the data sets used.

# Conclusion

We have designed and implemented a gaze tracker that can serve many real-world purposes. The program has been only tested on for Linux but it does not depend on any specific features of the system and thus should be highly portable.

On average, we failed to attain a sub-pixel precision in eye tracking, as was the original goal. Evaluation on real-world data shows that the program is indeed capable of such a precision when supplied with flawless data, and it quite often succeeds even in more challenging cases. It is mainly the outliers that spoil the average performance of eye tracking.

In order to deal with these outliers, we have designed a robust calibration method.

In terms of computational speed, the tracker provides an interactive performance of several frames per second on a typical computer. The software is modular and highly adjustable, i.e., it can be configured for high precision or for decreased computational requirements at the expense of tracking quality.

The performance deteriorates steadily as the conditions become more challenging. We have evaluated many test cases in detail and based on these, we provide an analysis of the actual necessary conditions for a smooth operation.

## Future work

A great challenge awaits our algorithm in that they should be compared to Artificial Neural Networks in terms of performance. To this end, an existing software library such as [21] shall be used.

So far, we did not manage to implement the resetting scheme for face trackers as it was described in Section 5.2.1. This method is important for practical use of the program.

Following a similar idea, we should combine both of the eye trackers in a robust manner based on their tracking score. The extreme case is to ignore one of the eyes completely if the tracking fails there.

Finally, the user comfort will be greatly improved by distributing some basic calibration data alongside the code, so that the program can be immediately tested upon download.

The interested reader is advised to download the most recent version of this program, instead of relying on the accompanying medium. The performance and robustness of all the methods implemented will be gradually improved. All the source code, including some extra tools, is published online at `https://github.com/addam/lokeye`.

# Bibliography


[1] Balasubramanian, Nallure, Ye, and Panchanathan. Biased manifold embedding: A framework for person-independent head pose estimation. *CVPR*, 2007.

[2] J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker. *Intel Corporation*, 5(1-10):4, 2001.

[3] K. W. Bowyer and M. J. Burge. *Handbook of Iris Recognition*. Second Edition. Springer, 2016. ISBN 978-1-4471-6784-6.

[4] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE PAMI*, 23(6):681–685, 2001.

[5] Ł. Drumiński. Eyecam program. `https://youtu.be/dZXz3egT7MI`, retrieved 2017-07. source code available.

[6] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. *CVPR*, pages 617–624, 2011.

[7] A. George and A. Routray. Fast and accurate algorithm for eye localization for gaze tracking in low resolution images. *IET Computer Vision*, 10(7): 660–669, 2016.

[8] N. Gourier, J. Maisonnasse, D. Hall, and J. L. Crowley. Head pose estimation on low resolution images. *Multimodal Technologies for Perception of Humans.*, pages 270–280, 2006.

[9] D. W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *PAMI*, 32(3):478–500, 2010.

[10] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Second Edition. Cambridge university press, 2003. ISBN 0521-54051-8.

[11] T. T. Inc. Tobii eyex. `https://www.tobii.com`, retrieved 2017-07.

[12] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz. Robust face detection using the hausdorff distance. In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pages 90–95. Springer, 2001. BioID data set: `https://www.bioid.com/About/BioID-Face-Database`.

[13] M. Kassner, W. Patera, and A. Bulling. Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. 2014.

[14] M. Leo, D. Cazzato, T. De Marco, and C. Distante. Unsupervised eye pupil localization through differential geometry and local self-similarity matching. *PloS one*, 9(8):e102829, 2014.

[15] C. M. Loba. Eviacam program. `http://eviacam.sourceforge.net`, retrieved 2017-07. source code available.

[16] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, 1981.

[17] P. Majaranta and K.-J. Räihä. Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 15–22. ACM, 2002.

[18] L.-P. Morency, P. Sundberg, and T. Darrell. Pose estimation using 3D view-based eigenspaces. *AMFG*, pages 45–52, 2003.

[19] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation in computer vision: A survey. *PAMI*, pages 607–626, 2009.

[20] E. Murphy-Chutorian, A. Doshi, and M. M. Trivedi. Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. *ITSC*, 2007.

[21] M. Patacchiola. Deepgaze library. `https://github.com/mpatacchiola/deepgaze`, retrieved 2017-07. source code available.

[22] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. *CVPR*, pages 1685–1692, 2014.

[23] S. Srinivasan and K. L. Boyer. Head pose estimation using view based eigenspaces. *ICPR*, 4:302–305, 2002.

[24] M. B. Stegmann. Active appearance models. Master's thesis, Technical University of Denmark, 2000.

[25] K.-H. Tan, D. J. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. *WACV*, pages 191–195, 2002.

[26] M. Tomasello, B. Hare, H. Lehmann, and J. Call. Reliance on head versus eyes in the gaze following of great apes and human infants: the cooperative eye hypothesis. *Journal of Human Evolution*, 52(3):314–320, 2007.

[27] P. L. UG. Pupil toolset. `https://pupil-labs.com`, retrieved 2017-07.

[28] M. Uřičář, V. Franc, and V. Hlaváč. Detector of facial landmarks learned by the structured output svm. *VIsAPP*, 12:547–556, 2012.

[29] R. Valenti and T. Gevers. Accurate eye center location and tracking using isophote curvature. *CVPR*, pages 1–8, 2008.

[30] A. Villanueva and R. Cabeza. A novel gaze estimation system with one calibration point. *Transactions on Systems, Man, and Cybernetics, Part B*, 38(4):1123–1138, 2008.

[31] C. Wang, F. Shi, S. Xia, and J. Chai. Realtime 3D eye gaze animation using a single RGB camera. *TOG*, 35(4):118, 2016.

[32] E. Wood, T. Baltrusaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling. Rendering of eyes for eye-shape registration and gaze estimation. In *ICCV*, pages 3756–3764, 2015. SynthesEyes data set: http://www.cl.cam.ac.uk/~eww23/data/syntheseyes_data.zip.

[33] A. L. Yarbus. Eye movements during perception of complex objects. In *Eye movements and vision*, pages 171–211. Springer, 1967.

[34] Z. Yücel and A. A. Salah. Resolution of focus of attention using gaze direction estimation and saliency computation. *ACII*, pages 1–6, 2009.

[35] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *IJCV*, 8(2):99–111, 1992.

[36] W.-z. Zhang, Z.-c. Wang, J.-k. Xu, and X.-y. Cong. A method of gaze direction estimation considering head posture. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(2):103–112, 2013.

[37] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Appearance-based gaze estimation in the wild. In *CVPR*, pages 4511–4520, 2015. MPIIGaze data set: `http://www.mpi-inf.mpg.de/MPIIGazeDataset`.

[38] J. Zhu and J. Yang. Subpixel eye gaze tracking. In *Automatic Face and Gesture Recognition*, pages 131–136. IEEE, 2002.

# Attachments

## A Directory structure

The attached data is organized as follows:

- The directory `bin/` contains a pre-built binary for Linux x86_64 systems with glibc 2.4.
- `readme.html` are brief instructions for installation and usage.
- The directory `src/` contains all of the the source code of the program, in version from 2017-07.
- The directory `screencapture/` is a MSVC++ project helper tool for acquisition of ground truth data from the Tobii EyeX tracker [11]. It has been developed as a part of this thesis, and we believe that it may be helpful to other users of the EyeX tracker. Note that this program can be only ran on Windows since the Tobii API is also limited to that platform.
- The directory `opencv/` contains a distribution of the OpenCV 3.1 library, which is necessary for building our program. Instead of building this library from source, the user is recommended to download a release specific to their system from .
- The directory `data/` contains the testing data sets, as described in the following attachment.
- Finally, the file `thesis.pdf` is the electronic version of this document.

## B Testing Data

There are three data sets provided in the testing data directory. All of them have been manually annotated with the position of iris center, and with a few extra values that describe the qualitative aspects of each image.

- `data/instagram/` are pairs of eyes acquired from random photos tagged as "selfie" on the website `https://www.instagram.com/`. The users who took the original photos come from all around the globe. By the nature of this data source, there are much more female subjects than males, and subjectively it seems that the ethnicities are not distributed accordingly to their global distribution. However, the images have been indeed selected by a random search in a fair manner.
- `data/models/` pairs of eyes cropped out from photos of photo models, taken by professionals. The visual quality of most of the images is excellent, being perfectly in focus and providing enough contrast. All of the subjects are females, and most of them wear makeup. These images have been obtained from many different, publicly available sources.

- `data/eyes/` are single eyes cropped out from the personal archive of the author. The single exception is the image `eyes/lena.jpg`, which is cropped out from the profound Lena image. The situations in which each of the photos were acquired vary greatly, and so does their visual quality and resolution.

The annotations are supplied in the file `data/annotations.js` in JSON format, and have been evaluated subjectively by the author of this thesis. For precise localization of the iris center, a specific tool was developed that is also provided in the source code.

There are also three videos with ground truth data that can be used for an offline evaluation of the program (without using a webcamera). The iris centers ($x, y$ coordinates in pixel units) are listed in CSV format, in the order of the corresponding video frames. Each of these `.avi` videos is accompanied by a `.csv` annotation. The people in these videos have been asked to do some basic tasks on the computer, in order for this data to be realistic.