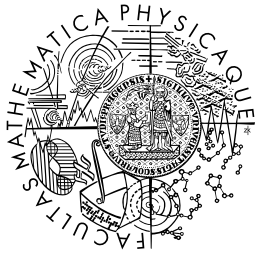

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

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 source code.

Keywords: face tracking gaze tracking image analysis

I could not finish this work without the help of many people. But most of all, Samy is my hero!

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 corresponding brain circuitry develops early in infants (@cite) 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. 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.

Eye trackers can actually be used for communication, namely for human-computer interaction. Wide variety of algorithms have been proposed (@cite) for on-screen keyboard and general desktop control. 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. Possible applications in gaze tracking in cell phones are a vast topic, such as locking the screen when not looked at or displaying a note when somebody looks over your shoulder.

Hereby we offer an open-source solution in the belief that it will help further development within this scientific field. The task itself is by certainly new, and as will be detailed in the following 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.

(*TODO: idea contributions: homography, radial iris locator*)
(*TODO: code contributions: face and eye trackers*)
(*TODO: the survey in Related Work is also noteworthy*)

By the author's preference, the program tends to rely on complex, rigid models. Such an approach is also beneficial because models with more free parameters would require too much training data.

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 reasonable precision, sophisticated mechanical structures were often built 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. An especially precise technique was developed by 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.

A method known as *electrooculography* provides a less invasive alternative. There is a constant voltage gradient across the eye from back to front, in magnitude of about 1 mV. Rotation of this small dipole is measurable, 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 spatial resolution is generally poor.

Details on other purely analog methods such as this can be found in a 1967 book by Yarbus [1]. 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

(*TODO: split into subsections*)

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. 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. Few approaches are able to encompass both of these tasks within a single model.

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(@cite) is based on matching a template image using gradient descent optimization and can be considered the cornerstone of object tracking. The original paper describes tracking a rectangular grayscale template by horizontal and vertical shift. We should note here that such a problem can also be solved on a global scale, i.e. using normalized cross-correlation and the Fast Fourier Transform for high performance. However, there have been numerous generalizations of this concept to a broader class of motion models such as affine(@cite) or perspective(@cite) transformations where global optimization is not feasible. Our program uses many of these generalizations extensively.

An especially sophisticated generalization is the Adaptive Appearance Models(@cite) (AAM). 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

[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 [1].

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 transformations gracefully.

Purely geometric image transformations have 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.(@cite) 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 (*TODO: ...*)

Each template need not be limited to a single image. Instead, each of these images can be extended to a reasonable neighborhood by a per-pixel linear function. The linear coefficients are typically estimated from several nearby templates using Principal Component Analysis, and this approach is widely known as Eigenfaces(@cite). This approach leads to a mathematical concept of a high-dimensional manifold that covers all feasible face images. By a further extension, each point in the high-dimensional space can be assigned six values that correspond to the six degrees of freedom of a rigid body. In this view, a face tracker simply extracts these six parameters from the corresponding point in space. Methods using this approach typically rely on high-level algebraic concepts to improve their performance.

Eye tracking is rather a simple task once the head pose is known. However, there has not yet been a clear consensus on the appropriate method. 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. The generalized Hough transform(@cite) provides a global method of searching for circles with one-pixel precision. In many applications(@cite), this is enough.

It is possible and sometimes more robust to use an appearance-based method, similar to the face tracking approaches outlined above. (*TODO: ...*)

(*TODO: ...*) Methods with partially controlled lighting are especially efficient for eye tracking because the mammalian eye is reflective both on its outer and the inner surface. Human eyes are 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, eyes in the image

(*TODO: finish*)

(*TODO: youtubers*)

(*TODO: @cite wang shi 16 etc. thoroughly*)

1.3 Notation

We mostly follow the notation used in the classical book Multiple View Geometry [**?**]. Boldface letters (e.g., vec$x$) denote real column vectors. Monospace letters (e.g., $\mathbf{C}$) denote real matrices. An upper index next to these (e.g., $\mathbf{C}^i$) identifies the i-th matrix within a list, n-th power of matrix would be always expressed using parentheses, e.g., $(\mathbf{S})^n$. The central dot $\cdot$ can mean either scalar or matrix

multiplication. Bounds in the summation symbol are not explicitly written down where they are obvious from the context (e.g., matrix dimensions).

(*TODO: Something more: vec, diag, ω*)

2. Problem Analysis

This chapter provides a thorough analysis of available data from the camera and 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.

2.1 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. Throughout the calculation, we rely on that the user's face is visible, and that the angle difference of gaze and camera direction is small.

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 completely different means of calibration. We allow the user to move their head but we cannot *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. may change abruptly and by all means. 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 change, it may be necessary to re-calibrate.

2.2 Face

Human face consists of many muscles. Like the pupil, they mainly serve for communication. Of special interest for us are regions that remain mostly fixed to the skull in normal conditions. (*TODO: image*) These include most notably the chin, base of the nose and small outer regions around and slightly below the eyes. The chin is especially prominent and easy to track, but may become completely misleading if the user opens their jaw. (*TODO: ...*)

2.3 Eye

2.3.1 Eye Anatomy

The eyeball consists of several parts, most notably:

- Transparent cornea, which makes a fixed lens and provides some mechanical protection
- Flexible and reflective iris
- Flexible lens stretched by several muscles to control its optical magnitude

- Purely white sclera, which serves as a hard shell of the eyeball
- The vitreous body, consisting mostly of transparent gel to maintain inner pressure
- The light-sensitive retina

Furthermore, each eyeball is embedded in a hole called (*TODO: what—eye chamber, perhaps?*) that provides fixation and actuation. There are (*TODO: how many*) separately controlled muscles stretching between to the eyeball and the eye chamber.

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

Sclera is white and pupil is black. Iris is something in between. The iris diameter remains constant but the pupil diameter can change dramatically.

In a healthy case, the inner and outer boundary of the iris (the pupil and the limbus, respectively) should have a precisely circular shape. Generally, the inner shape of the iris is less reliable of these two. Its radius changes depending on the amount of incoming light and the emotional state of the user. This motion is controlled from (*TODO: which brain region—thalamus?*) and therefore cannot be directly controlled by will. In case of a disease or an injury, it is also possible that some of the muscles stop working properly and the inner iris shape is no longer a circle.(*TODO: image*)

2.3.2 Eye Movement

Human eyes have developed not only as an organ of sight, but also as a means of communication. When attending to an object or another person, people will usually turn their eyes directly towards it.

The main reason of this is that the overall acuity of our visual system increases towards an area near the optical axis. The corresponding spot on the retina is called the fovea, and it is located about (*TODO: how many?*) degrees horizontally towards the nose. The density of photopic (daylight-sensitive) cells is up to (*TODO: how much*) 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 (*TODO: how many*) muscles provide (*TODO: how many*) degrees of freedom 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*. Thanks to it, the effect of eye roll can be implicitly precalculated during calibration.

(*TODO: saccades, fixation, smooth pursuit*)

2.4 Gaze

Under the assumption of small angle divergence, we can model movement of the pupil as a simple translation.

Given the eye–face offset, onscreen gaze center is assumed to be a homography. This saves us from explicitly modeling the viewed scene. The non-linear factor is quite huge, unfortunately.

2.5 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 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 otherwise bad, 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 @cite. The most notable of these issues is 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 geometrical 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 different color filters to each cell of the light sensor. (*TODO: explain some more*) Contrast on subpixel level makes each color channel sum up to a different amount, even if there are no 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.

2.6 Image

The image as acquired from the camera is a rectangular grid of colored points. However, certain parts of the computation require a continuous and smooth image model in order to obtain sub-pixel precision. In order to apply classic optimization methods, we need to geometrically transform (i.e., stretch) the image, and evaluate its partial derivatives at random points.

This problem has three possible solutions:

- Use simple interpolation and ignore the inaccuracy induced. This is the approach applied in this thesis.
- Use simple interpolation on a blurred image so that the inaccuracy disappears. This is the solution implicitly used by many software libraries.
- Interpolate using a sophisticated function. Surprisingly enough, commonly used interpolation functions are not suitable for a precise model. An interpolator should obviously not be biased towards zero, and therefore, functions such as sinc cannot be used directly.

  Bias disappears if we use the function value only as a weight in a weighted average formula. However, that often makes image derivatives prohibitively hard to calculate. This may lead us to only estimating the image derivatives by a simple formula—but by making such a step we effectively classify to the first category above.

(*TODO: what is our image model, then.*) (*TODO: shifted derivatives*)

3. Algorithms

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

3.1 Linear algebra

3.1.1 Singular Value Decomposition

Input: matrix $\mathbf{A}$
Output: orthonormal matrices $\mathbf{U}, \mathbf{V}$ and a vector $\mathbf{s}$ such that $\mathbf{A} = \mathbf{U} \cdot \mathrm{diag}(\mathbf{s}) \cdot \mathbf{V}^\mathsf{T}$.

(*TODO: Jacobi algorithm!*)

The system matrix in our case is of shape $K \cdot (n-1) \times m$, which leads to an overall time complexity of $O(m^2 K(n-1))$.

3.1.2 Homogeneous Linear Solving

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

Let us consider the SVD(*TODO: abbreviation*) of $\mathbf{A} = \mathbf{U} \cdot \mathrm{diag}(\mathbf{s}) \cdot \mathbf{V}^\mathsf{T}$. The result $\mathbf{x}$ is the i-th column of $\mathbf{V}^\mathsf{T}$, where i is chosen so that $\mathbf{s}_i^2$ is minimal.

Proof. Let us define a vector $\mathbf{v} = \mathbf{V}\mathbf{x}$. This is effectively a change of basis, and because $\mathbf{V}$ is an orthonormal matrix, the norm $|\mathbf{v}| = |\mathbf{x}|$ is preserved. Because also $\mathbf{U}$ is an orthonormal matrix, it can be safely crossed out of the norm $|\mathbf{A}\mathbf{x}| = |\mathrm{diag}(\mathbf{s}) \cdot \mathbf{v}|$. This in turn can be expressed as the square root of $\sum_i (\mathbf{s}_i \mathbf{v}_i)^2 = \sum_i \mathbf{s}_i^2 \mathbf{v}_i^2$.

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 $\mathbf{x} = \mathbf{V}^{-\mathsf{I}}\mathbf{v} = \mathbf{V}^\mathsf{T}\mathbf{v}$. □

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 can be proven by adding a small perturbation to $\mathbf{s}_j$.

3.1.3 Principal Component Analysis

(*TODO: is this used in the program?*)

3.2 Nonlinear optimization

(*TODO: Gradient descent*)
(*TODO: line search*)

3.3 Fourier Analysis

3.3.1 Fast Fourier Transform

Input: vector $\mathbf{v} \in \mathbb{C}^n$.
Output: vector $\mathbf{f} \in \mathbb{C}^n$ such that $\mathbf{x}_i = \sum_j \mathbf{f}_j \omega^{ij}$ for all i, where ω is n-th root of unity.

Let us define the function FFT : $\mathbb{R}^{2m} \to \mathbb{R}^m$.

Proof. @cite CLRS □

3.3.2 Convolution

Input: matrices $\mathbf{M} \in \mathbb{R}^{n \times m}, \mathbf{T} \in \mathbb{R}^{k \times l}$.
Output: matrix $\mathbf{R} \in \mathbb{R}^{|n-k| \times |m-l|}$ as defined below.

3.3.3 Normalized Cross-Correlation

Input: matrices $\mathbf{M} \in \mathbb{R}^{n \times m}, \mathbf{T} \in \mathbb{R}^{k \times l}$.
Output: matrix $\mathbf{R} \in \mathbb{R}^{|n-k| \times |m-l|}$ as defined below.

(*TODO: there are more use cases beyond the obvious one.*)

3.4 Geometry

3.4.1 Direct Linear Transformation

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

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 is generally different from a geometric least-squares solution. The former can be obtained in a much more stable way than the latter, so we start with algebraic least squares fitting and use the result for initialization of a nonlinear optimization method. Secondly, the measured data points are defined only up to scale, so it is not possible to formulate a linear system straight away. A technique called Direct Linear Transformation is widely used to overcome this issue.

We are presented with a set projective correspondences of the form

$$\mathbf{H}\mathbf{x}^k = \alpha_k \mathbf{p}^k, 1 \leq k \leq K,$$

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

Unfortunately, these equations do not form a linear system with respect to the unknown matrix $\mathbf{H}$ because there is an unknown scale factor α_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.

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

$$\mathbf{M}^{i,j}_{i,j} = 1, \mathbf{M}^{i,j}_{j,i} = -1.$$

From this definition it follows that for any $\mathbf{p} \in \mathbb{R}^n$, the product $\mathbf{p}^\mathsf{T}\mathbf{M}^{i,j}\mathbf{p} = \mathbf{p}_i\mathbf{p}_j - \mathbf{p}_j\mathbf{p}_i = 0$. The vectors $\{\mathbf{M}^{i,j}\mathbf{p}\}, 1 \leq i < j \leq n$, are all orthogonal to a given $\mathbf{p} \in \mathbb{R}^n$.

With increasing dimension, this set of vectors becomes heavily redundant. Clearly, a minimal solution would form a basis of the subspace $\mathbb{R}^n/\mathbf{p}$ orthogonal to $\mathbf{p}$ and thus would consist of $n-1$ vectors, whereas this approach generates $\frac{1}{2}n\cdot(n-1)$ vectors. Some sources[2] suggest to pick an arbitrary subset of size $n-1$ from the matrices defined above and use these for the whole data set. 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 $\mathbf{M}^{i,j}$ 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 $K\cdot(n-1)$ equations, where K is the number of correspondence pairs $\mathbf{x}^k \leftrightarrow \mathbf{p}^k$. Its unknowns will be precisely the elements of the matrix $\mathbf{H}$. Each equation of the system represents a constraint of the form $\mathbf{p}^\mathsf{T}\mathbf{M}\mathbf{H}\mathbf{x} = 0$, which is the orthogonality constraint as proposed above. This constraint can be reformulated in terms of the Frobenius inner product $\langle \mathbf{A}, \mathbf{B}\rangle_F = \sum_{i,j} \mathbf{A}_{i,j} \cdot \mathbf{B}_{i,j}$ and the vector outer product as follows:

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

Viewing the matrices as vectors of their elements, this finally leads to an equation in the suitable form:

$$\mathrm{vec}(\mathbf{p}^\mathsf{T}\mathbf{M}\mathbf{x}^\mathsf{T})^\mathsf{T} \cdot \mathrm{vec}(\mathbf{H}) = 0,$$

where $\mathrm{vec} : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ stacks all matrix elements to a vector, in row-major order. Each correspondence pair and each selected antisymmetric matrix $\mathbf{M}^{i,j}$ for that pair provide one such equation, and the row vectors $\mathrm{vec}(\mathbf{p}^{k\mathsf{T}}\mathbf{M}^{i,j}\mathbf{x}^{k\mathsf{T}})^\mathsf{T}$ can be stacked to form the system matrix.

Homogeneous linear systems such as this one can be solved using the Singular Value Decomposition (SVD).

(*TODO:* ...) 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 $\mathbf{H}\mathbf{x}^k = \alpha_k \mathbf{p}^k$. 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 loop through the resultant singular vectors and choose the one with the minimal reprojection error.

(*TODO: normalization: subtract mean, enforce unity variance*)

3.4.2 Four-point homography

A minimal case of a 2-dimensional homography estimation is a correspondence of 4 point pairs. The general formula as generated by DLT(*TODO: abbreviation?*) would be needlessly bloated for this setting.

Let us write the point sets as columns of a 3×4 matrix, and let us denote the equivalence of two point sets up to scale by the $\sim$ (tilde) character:

$$\mathbf{A} \sim \mathbf{B} \Leftrightarrow \exists \alpha_1 \dots \alpha_4 : \mathbf{A} \cdot \begin{pmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_4 \end{pmatrix} = \mathbf{B}.$$

Also, let us declare the following set as the *canonical configuration*:

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

We will now present the formulas to convert an arbitrary 4-point set to and from the canonical configuration.

For a given point set $\mathbf{A} = (\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$, the homography $\mathrm{can}(\mathbf{A})$ is defined as:

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

where $\alpha_1 \dots \alpha_3$ are chosen so that $\mathrm{can}(\mathbf{A}) \cdot \mathbf{d} = (1, 1, 1)^\mathsf{T}$. Note that this requires no linear solving, just an element-wise division. If the points $\mathbf{A}$ are affine independent, it clearly follows that $\mathrm{can}(\mathbf{A}) \cdot \mathbf{A} \sim \mathbf{C}$.

The homography $\mathrm{dec}(\mathbf{A})$ is, in turn, defined as:

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

where $\alpha_1 \dots \alpha_3$ are chosen so that $\mathrm{dec}(\mathbf{A}) \cdot (1, 1, 1)^\mathsf{T} = \mathbf{d}$. This requires us to solve the 3×3 linear system $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \cdot (\alpha_1, \alpha_2, \alpha_3)^\mathsf{T} = \mathbf{d}$. As a result it follows that $\mathrm{dec}(\mathbf{A}) \cdot \mathbf{C} \sim \mathbf{A}$. Note that the vector $\mathbf{d}$ is exactly the right-hand side of this system. A neat side-effect is that the inverse matrix $\mathrm{dec}(\mathbf{A})^{-\mathsf{I}} = \mathrm{can}(\mathbf{A})$ has correct scale. More importantly, it means that the mapping dec is linear with respect to all elements of $\mathbf{d}$.

These two mappings can be composed to provide any four-point homography as necessary:

$$\mathrm{dec}(\mathbf{B}) \cdot \mathrm{can}(\mathbf{A}) \cdot \mathbf{A} \sim \mathbf{B}.$$

The resulting mapping is linear with respect to the last column of $\mathbf{B}$.

(*TODO: what are the overall computational requirements?*)

3.4.3 Derivatives of a homography

We wish to calculate the derivative of a four-point homography with respect to the coordinates of its four control points. The linearity of $\mathrm{dec}(\mathbf{A})$ implies that the derivative $\frac{\partial}{\partial \mathbf{d}_i}\mathrm{dec}\,(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = \mathrm{dec}\,(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{e}_i)$, where $\mathbf{e}_i$ is the unit vector for the coordinate axis i.

Furthermore, we need to calculate derivatives with respect to $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$. Theoretically, we could use the fact that any permutation of the canonical points can be expressed as a homography,[1] and incorporate these homographies into the formulas above. Unfortunately, there are several unknown scale factors involved that make such an approach inefficient. The preferable method is to explicitly permute the points so that the point in question becomes the last one, and to repeat the original scheme.

(*TODO: what are the overall computational requirements, per pixel?*)

3.5 Machine learning

(*TODO: Ransac*) (*TODO: Boosted random forests*)

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

4. 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.

4.1 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.

The pipeline of computation is presented in Drawing @ref. Firstly, the face is locally tracked from the previous frame. Only if this step fails, the program starts a global face localization procedure. This first step results in a 3-dimensional translation and rotation mapping from the initial pose.

Next, the precise position of both eyes is established within a reasonable region in the face. This steps requires the face to be localized properly in the image but since the eyes can move very quickly, this uses no prior from the previous frame.

Finally, the 6 degrees of freedom of the head and the 4 degrees of freedom of the eyes are fed into a gaze estimator that outputs coordinates in screen reference frame.

4.2 Face Tracking

Our program supports two methods of face tracking, which represent two major groups of tracking techniques in general. First is feature-based, second is appearance-based. A surface within the face is tracked pixel per pixel. Depending on the transformation allowed, we get different methods:

- Similarity transformation. 4 DoF. Aligning in grid makes no sense at all.
- Affinity. 6 DoF. Aligning in grid makes little sense.
- Barycentric transformation. 6 DoF. Possible to align in grid.
- Homography. 8 DoF. Possible to align in grid. Appropriate from a theoretical standpoint.

(*TODO:* *blabla*). There is a *master tracker* that covers the whole face. It 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 robustess against changes in lighting, the master trackers is color-normalized.

4.2.1 Markers

In addition to the master tracker, several small trackers are arbitrarily placed within the face area that are supposed to track specific small features. Technically, it is possible to track any skin region with a distinctive texture, so an automated feature detection performs well enough.

Each of the markers is allowed to freely move around. They are tracked from their previous known position relatively to the master tracker. If the tracking score drops below a certain limit or a marker escapes the face area, it is reset to its original position and tracking continues from there.

4.2.2 Grid

This method follows the approach of Active Appearance Models. The master 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 barycentric cells (i.e., a triangular mesh), where the derivatives wrt. vertex coordinates are quite easy to derive. The extension of this method to homographic cells (i.e., a quadrangular mesh) relies on the theory developed in Section 3.4.3.

4.3 Eye Tracking

There are a plenty of eye trackers available 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.

These approaches have been combined in many different ways, resulting in the following methods available to the host application:

4.3.1 Limbus gradient

This method relies on the edge between the iris and the sclera. Assuming that the iris is much darker than the sclera, their shared boundary (the limbus) should be sensed as a circle of high contrast. We assume the eyelids to be of uniform color, so they have zero value in the gradient domain and can be neglected.

4.3.2 Limbus polynomial gradient

The eye region of the face is actually not uniform as it contains many wrinkles and directional light may cause strong shadow. These facts cause a low-magnitude gradient that might modify the global optimum of the method described above. In order to gain some robustness, we can apply a heuristic filter to the gradient values. If designed with care, the method can still operate at nearly the same speed.

4.3.3 Hough transformation

We can use a voting scheme to find the limbus center. This allows us to apply a sophisticated heuristic filter at almost no cost. However, a free parameter is necessary to define: the resolution of the voting grid. Using each pixel as a voting bin is usually the best choice, but it may fail on blurry images of high resolution because the votes will be distributed quite randomly over a wide region around the true center.

4.3.4 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. An appropriately fast and reliable one is the normalized correlation, using a slightly blurred black circle as the template.

This method appears to work especially well when tested in regions where the majority of people are brown-eyed and with white skin. It can also 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.

4.3.5 Personalized iris correlation

This is yet to be coded and tested. 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. 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. This template requires the user to open wide their eyes, so that the whole limbus is visible.

For tracking a colored object, it is again reasonable to use the normalized correlation.

4.3.6 Iris radial symmetry

We can define a model with few free parameters, relying only on the radial symmetry around the iris. Being highly nonlinear, it is also difficult to optimize and may be trapped in a local minimum. (*TODO: explain in detail!*)

4.3.7 Skin segmentation

This is yet to be coded and tested. It should be a switch that just disables processing of skin regions.

4.3.8 Random forest regression

This is yet to be coded and tested, but it is not going to work anyway.

4.3.9 Convolutional neural networks

This is yet to be coded and tested.

4.3.10 Combined estimator

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. Reasonable selection of trackers for a combined estimator is discussed in the next chapter.

4.4 Gaze Estimation

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 face transformation has 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 two-dimensional gaze, we would have to actually estimate the three-dimensional model of the face. Many programs have been published that follow this path (@cite). In general, it requires some prior information about the shape of the face, and 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 implicitly model inverse proportionalities among their parameters.

(*TODO: combine with the following section in a meaningful manner*)

4.5 Calibration

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.

During the calibration, roughly speaking, the user is asked to watch specific points on screen and their face and eyes are tracked. For each screen position measured this way, the face tracker provides many parameters. Some of these have an implicit meaning, others can be processed to extract useful information and yet others are just noise. We fit a homography mapping some of the face parameters and the eye parameters to each known gaze position.

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.

It is meaningful to extract four basic parameters about a 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 can be obtained from detailed tracking of the face. 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 extra 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 repetitively. As soon as a fit collects a large enough support of measurements, the calibration procedure stops and outputs the homography it acquired.

5. 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. We compare the various methods against each other and against the state of the art.

5.1 Images!

(*TODO:* ...)

5.2 Tables!

(*TODO:* ...)

5.3 Graphs!

(*TODO:* ...)

The program does not perform all that well when compared to the state of the art, but it is certainly much more than a proof of concept.

The results on flawless video streams are already usable, and challenging situations could perhaps be solved with more coding manpower. (*TODO:* ...)

Conclusion

We have designed and implemented a gaze tracker that can serve many real-world purposes. The software is presented as a proof of concept and as a benchmark rig, so it is highly adjustable and modular. Evaluation on real-world data shows that the algorithm used here works well when supplied with flawless input data, and that the results deteriorate steadily as the conditions become more challenging.

Bibliography

[1] Alfred L Yarbus. *Eye movements during perception of complex objects*. Springer, 1967.

[2] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Second Edition. Cambridge university press, 2003.

Attachments

A Directory structure of the attached data

- The directory `bin/` contains ...
- `doc/install.html` and `doc/usage.html` is the user documentation of the program. The file `doc/code.html` contains developer documentation and overall notes about the structure of the code.
- `extern/` contains the less common external libraries that are needed to compile and run the program.
- `src/` contains the whole source code of the program. Apart from it, a Python script `src/io_export_tracks.py` is provided that serves for exporting the camera calibration conveniently from Blender as input to our program. Further files for testing of the source code are provided in `src/test/`.
- The directory `data/` contains data for testing of the program.
- Finally, the file `thesis.pdf` is the electronic version of this document.

B Library interface

The software can be readily used as a gaze tracking library. There are a few decisions that the host application must make in order to initialize the tracking algorithms. (*TODO:* ...)

The file `src/example_main.cpp` is a simple gaze tracking application and an example of the application interface. There are more similar files prefixed with `example_` that showcase smaller bits of the code, and often use methods that should be hidden from the host application.