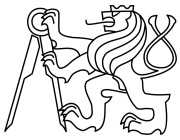
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	Translation of eye movement into mouse cursor movement
Student:	Jan Škařupa
Supervisor:	Ing. Radomír Polách
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	Until the end of winter semester 2018/19

Instructions

Study libraries OpenCV [1] and/or SimpleCV [2] and appropriate algorithms for analysis of the eye movement recorded by camera.
Analyse how to translate eye movement into mouse cursor movement.
Design and implement prototype application for Linux and Windows operating systems in the C++ programming language.
Test implemented applications on several human subjects with regard to reliability and accuracy.

References

[1] OpenCV. Available at: http://opencv.org/
[2] SimpleCV. Available at: http://simplecv.org/

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague March 17, 2017

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Bachelor's thesis

Translation of eye movement into mouse cursor movement

Jan Škařupa

Supervisor: Ing. Radomír Polách

16th May 2017

Acknowledgements

I would like to thank my entire family for supporting me during my studies and during my work on this bachelor thesis.

Also I would like to thank Ing. Radomír Polách for proposing this interesting topic.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 16th May 2017 .....

Citation of this thesis

Škařupa, Jan. *Translation of eye movement into mouse cursor movement.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Tato bakalářská práce se zabývá tématem sledování pohybu očí, konkrétně detekcí bodu pohledu. Zkoumá současné metody a algoritmy. Dále je modifikuje pro použití při překladu pohybu očí na pohyb kurzoru. Práce uvažuje pouze případy, kdy jsou oči snímány obyčejnou web kamerou. Upravené algoritmy testuje a zkoumá jejich použitelnost.

Klíčová slova počítačové vidění, sledování očí, detekce duhovky, detekce zornice, bod pohledu, pohyb kurzoru

Abstract

This thesis reviews current methods for eye tracking, specifically for point of gaze detection. It redesigns them for use in application that translates eye movement into movement of a mouse cursor. It considers only cases when the eye movement is captured with regular non-expensive web camera. It tests precision of implemented approach and discuss its usability.

Keywords computer vision, eye tracking, iris detection, pupil detection, point of gaze, cursor movement

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Eye tracking and point of gaze

Eye tracking is a process of detecting motion of the eye ball (or more precisely of the pupil) relative to the head. The earliest mention is from second half of 18. century, when reading process was studied. Naturally eye movement was observed without any technology. Nowadays various methods for eye tracking exists, as well as many applications.

Today's application

Even today the eye tracking is mainly used for research. However it spread from purely medical research to other areas including commercial one. Here I present short list [9] of most common research areas:

- Medical Research: Eye tracking in combinations with conventional research methods or other biometric sensors can even be helpful for diagnosing diseases such as Attention Deficit Hyperactivity Disorder (ADHD), Autism Spectrum Disorder (ASD), Obsessive Compulsive Disorder (OCD), Schizophrenia, Parkinsons and Alzheimers disease.
- Psychology Research: Useful information about human attention and others psychological factors can be derived from movement of eyes.
- Market Research: Many leading brands use the eye tracking as a tool to evaluate their products, designs, advertising or even the shopping behavior of their customers according to amount of attention given to subject.
- Usability Research: The classic example is website testing. Using eye tracking one can evaluate how hard it is to locate important elements of UI.

- Human factors: By eye monitoring it is possible to estimate condition of the individual. That could be used for example as security element in cars.

Future is promising even more applications. With increasing digitalization, eyes could be used as common interface between human and computer.

Types of eye tracking

Eye tracking techniques can be divided into two main categories based on the fact whether they use specialized hardware (eye tracker) or not. First category of techniques is called "intrusive", the other "non-intrusive". Examples of intrusive methods are:

- Optical tracking: This technique uses head-mounted device for eye capturing. Some sort of light source can be added to help detection. Infrared light is often used.
- Eye-attached tracking: This technique uses hardware attached directly to the eye. (Commonly some sort of contact lens.)
- Electric potential measurement: based on measuring of electric potentials.

All intrusive methods are naturally quite constraining or uncomfortable for the user. We will focus on second category. In non-intrusive eye tracking input is plain image containing observed eyes. We can find location of the eye centre in the image, but obviously for point of gaze detection previous calibration is needed. Algorithms for eye localization also falls into several categories:

- Feature based: Set of features (specific patterns) is predefined. Then the image is probed and every region is tested for presence of every feature. Specific configuration of features may be evaluated as object detection.
- Model based: This approach tries to fit to predefined templates.
- Hybrid models: These are combination previous methods.

In this work I will try to improve non-intrusive point of gaze detection.

1.2 Goals

The overall goal of this thesis is to design an algorithm which can translate eye movements into movement of the cursor. Input images will be captured by regular integrated web camera. We do not consider cases where extra hardware, as head-mounted tracking device, is involved. Algorithm should satisfy following requirements and constrains:

1. Algorithm performs on low quality images.
2. Algorithm is fast enough to work in real time.
3. Algorithm estimates PoG with precision sufficient for controlling cursor.

To meet these requirements, solution for following problems must be designed:

1. Perform face detection in relatively short time.
2. Detect location of the eye centre precisely. Detection must be invariant to lightning conditions and resistant to noise.
3. Define reference point or other framework to transform eye centre location into point of gaze estimation. Such reference should be invariant to face expression and head motion.
4. If necessary design filtering algorithm for latter two points. Filter must not cause noticeable delay.

Chapter 2

Related work

2.1 Face recognition

The initial step for any kind of eye tracking is to recognize faces in the image. Face detection is basically special case of object-class detection. The face detection algorithm must be able to recognize faces despite all their differences. Its performance should be as invariant to changes of light condition as possible. Finally we often want to use such algorithms in real time applications therefore it must be fast.

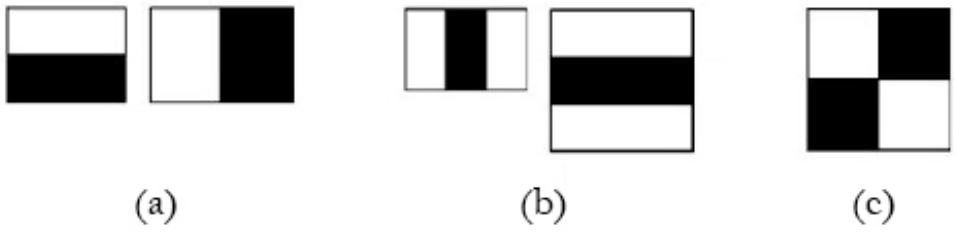
In context of computer vision we are mostly talking about detecting features. A feature is - generally speaking - a pattern which we are looking for in our complex input. It may be point, edge, difference in pixel intensity, etc. For complex structures (such as faces) it is impossible to define set of desired features manually. Therefore some sort of machine learning is used.

2.1.1 Haar Cascade Classifier

One of the algorithm I tested during my work was Haar Cascade Classifier. It was proposed by Paul Viola and Michael Jones in 2001 in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features". [5] It based on detection specific features, so called "Haar-like features". Three example of Haar-like features are shown in figure 2.1. They can be regarded as convolution kernels. They consist of shape divided between two regions. For each region we calculate the sum of pixel's intensities and then the difference between those sums. So basically it calculates brightness difference between adjacent areas.

Shape of Haar-like feature is most often rectangle. For rectangle shaped features we can convert our input into integral image. By doing this we can compute sums of our regions with $\mathcal{O}(1)$ complexity.

Figure 2.1: Haar-like features [5]



Cascade function training

Every cascade classifier must be trained on some dataset of positive images (those who contain face) and negative images (those who do not). All instances of kernels are generated (for every size and location). Then they are applied to every image in training database. Threshold value is decided for every kernel such that it provides best classification results. After that weights are assigned to the images based on number of miss-classifications and these steps are repeated (with regard to weighting) to optimize setting of thresholds. This procedure is called AdaBoost.

After sufficient result is achieved we choose features with lowest error rate. One feature with set threshold value is called "weak classifier" as it provides only slightly better result in face detection then random choosing. They must be used in group to provide useful results. Therefore final classifier is a weighted sum of these weak classifiers. Number of selected features is optional. With increasing size of the set the computational complexity will naturally increase as well. Trained set of OpenCV library contains around 6000 features.

Classification cascade

To detect faces with trained classifier algorithm uses sliding window to explore all image sections. However in real life applications most sections of the image does not contain any face at all. It is computationally wasteful to apply all feature kernel to them. For this reason kernels are split into groups. Groups are applied gradually and their size increases. If any group decides that window does not contain image face process is stopped and window moves to other position. If all the groups agrees then a face is detected.

2.1.2 Classification from Histogram of Oriented Gradients

Another algorithm for face detection that I tested is based on the classic Histogram of Oriented Gradients [6] feature combined with a linear classifier and sliding window detection scheme, which was already mentioned above. Technique based on HOG was very successfully used for detection of pedestrians.

Histogram of Oriented Gradients

Unlike previous method this one works with image gradients. First step is to calculate vector field of gradients. It is usually done by Sobel operator with size 1. (i.e. Filtering image with kernels $S = [-1\ 0\ 1]$ and S^T and calculating magnitude and direction of resulting vectors.) Gradients are afterwards clustered into squares of specific size. This size is based on the expected size of the searched object. It must be fraction of object size.

Histogram with respect to gradient orientation is created afterwards. To achieve results which will be invariant to uniform illumination changes, histogram is normalized to the 2L norm.

Algorithm then search field of histograms looking for specific pattern. Technique of sliding window is used again and histograms in windows are exploit by trained linear classifier.

Figure 2.2: Histogram of Oriented Gradients

(a) Input image. (b) Field of gradients from $n \times n$ region collapsed into N-dimensional vector.

2.2 Facial Landmarks Detection

After detecting faces using one of the approach described above we might want to obtain additional information about the face. Face detectors usually returns only regions. Based on used detector we can assume rough orientation (e.g. Was detector trained for frontal faces? Etc.), but details remain unknown. Faces have common stable characteristic which we can look for. For example: inner and outer corners of eyes, lips, chin, etc. Different implementations specify different number (and location) of these points. Figure 2.3 shows example of points defined by Vahid Kazemi and Josephine Sullivan in their paper "One Millisecond Face Alignment with an Ensemble of Regression Trees".

Naturally these points are mostly located on those parts of human face that creates edges in the image. Therefore similar approach will be used as when detecting presence of a face itself. Nevertheless one single point would create only hardly distinguishable feature. Fortunately we have prior knowledge about human face (e.g. approximate relative locations of the points). Further we have already acquired the face region. Therefore we can constrain location of points and try to fit them to a predefined model.

Implementation is often based on regression trees. Xiangxin Zhu for instance states: "Our model is based on a mixtures of trees with a shared pool of parts; we model every facial landmark as a part and use global mixtures to capture topological changes due to viewpoint."

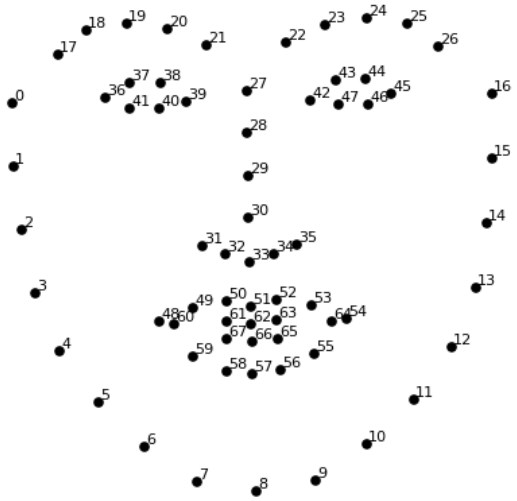


Figure 2.3: Facial Landmarks Detection points

Figure shows 68 points which represents initial model for facial landmarks detection.

2.3 Pupil centre localization

After successful eye region alignment, the next step of PoG detection is to determine the location of pupils centers. A lot of work has been made in this area lately.

At first I decided to rehearse approach proposed by Fabian Timm and Erhardt Barth in 2011 [1] as it achieved good results in their comparative testing on the BioID database. (see fig. 2.3.1) Moreover algorithm is quite straightforward and its implementation is not exceedingly demanding. However several issues arisen when this approach was tested in context of PoG detection. Namely: low precision in general, insufficient robustness against image noise, insufficient robustness against occlusions. Due to the characteristics of testing on BioID database these problems remained hidden. I discuss them thoroughly later.

Later on I decided to change base of pupil centre localization to George Anjith's approach [2]. It uses convolution operators and Starburst algorithm [3] to detect iris borders. Pupil centre is afterwards derived from iris location. This technique proved itself to be more convenient for needs of this work. It is described in section 2.3.2.

2.3.1 Timm's approach: localization by means of gradient

In the following section I will provide description of Timm's and Barth's approach, including mathematic definition. Discussion on precision and other problems follows.

Algorithm

Center Localization by Means of Gradient technique works on previously selected eye region. It is based on the assumption, that pupil is the darkest area in the region. Also that it is surrounded with brighter iris and even brighter sclera. Therefore in vector field of gradient (calculated from pixels intensities), vectors will be pointing towards pupil centre.

For every possible pupil centre (which corresponds to every pixel in region) the algorithm exploits vector field to determine how much gradients fits assumption stated above. Timm and Barth define this exactly:

> *Let c be a possible centre and g_i the gradient vector at position g_i. Then, the normalized displacement vector d_i should have the same orientation (except for the sign) as the gradient g_i. If we use the vector field of (image) gradients, we can exploit this vector field by computing the dot products between the normalized displacement vectors (related to a fixed centre) and the gradient vectors g_i.*

The optimal centre $c*$ of a circular object in an image with pixel positions $x_i, i \in \{1, \ldots, N\}$ is then given by

$$c^* = \arg\max_{\mathrm{c}} \left\{ \frac{1}{N} \sum_{i=1}^{N} (d_i^T c_i)^2 \right\}, \tag{2.1}$$

$$d_i = \frac{x_i - c}{||x_i - c||_2}, \qquad \forall i : ||g_i||_2 = 1. \tag{2.2}$$

To ensure that the maximum of the objective function will lay in dark area and thus more likely in pupil region, they add weight w_c to each potential centre. Therefore we get:

$$\mathrm{c}^* = \arg\max_{\mathrm{c}} \left\{ \frac{1}{N} w_c \sum_{i=1}^{N} (d_i^T c_i)^2 \right\}, \tag{2.3}$$

where $w_c = I*(cx; cy)$ is the gray value at $(cx; cy)$ of the inverted and smoothed image. Smoothing is done by applying Gauss filter.

Post processing

Furthermore they discuss false detection caused by other dark elements (e.g.: eyebrows, glasses, etc.) in eye region. Such elements can have gradient similar to the eye around themselves. Solution consists of post processing results of the objective function before finding its maximum. Threshold based on the maximum value is applied and all regions connected to borders of the eye region are removed. This solution is very effective but relies hugely on correct size of the region. During my testing two problematic scenarios occurred.

1. Region too large: If the region is too large, it will contain entire eyebrow without it touching borders. In that case it is not removed and can cause false detections.

2. Region too small: If the region is too small, eyelid contours (usually quite dark) can be touching borders. And because upper part of the eye is often overshadowed by superciliary ridge these two regions can merge into one after thresholding. Entire iris will be removed consequently.

Especially second case causes serious problems, therefore if used, eye regions must be chosen carefully.

Issues in PoG application

Entire stated algorithm base its logic on the fact, that pupil centre is the darkest point in eye region. However this is only truth to the certain point of precision. (Note that in context of translation PoG to cursor movement we require high precision as even delicate movement of the eye might result in consignable movement of the cursor.) Timm's algorithm is robust and even on images with low resolution and quality performs very well in terms of finding darkest point in pupil region. Here I list conditions under which such point is usually not the pupil centre.

1. Occlusions: Strong direct illumination causes reflects on cornea. Especially in night time the computer screen is reflected right in the pupil region as user usually faces screen directly. This results in pupil centre being bright.

2. Strong illumination from one specific angle: Even if the source of light is not as direct and strong as computer screen mentioned above, it often comes from one direction (from window, etc.). Consequently pupil might be darkest at the opposite side.

3. Noise: Finally, even if the illumination is uniform, we are facing problem of camera noise. This causes "trembling" of detected point. Naturally we can apply some sort of filter either to image itself or directly to the point location, however every filtering causes delay which might be inconvenient.

These problems led me to the conclusion, that detecting pupil centre directly is insufficient and detection of iris itself is needed.

Testing on BioID dataset

Timm and Barth tested their algorithm on BioID dataset. I present most interesting part of their comparative table.

Method	$\epsilon \leq 0.05$	$\epsilon \leq 0.1$	$\epsilon \leq 0.15$	$\epsilon \leq 0.2$	$\epsilon \leq 0.25$
Asadifard and Shanbezadeh, 2010	47.0%	86.0%	89.0%	93.0%	96.0%
Valenti and Gevers, 2008	**84.1%**	90.9%	(93.8%)	(97.0%)	98.5%
Trkan et al., 2007	(18.6%)	73.7%	(94.2%)	**(98.7%)**	**99.6%**
Cristinacce et al., 2004	(57.0%)	**96.0%**	**(96.5%)**	(97.0%)	(97.1%)
.....	...	...	...	...	...
Timm & Barth	82.5%	93.4%	95.2%	96.4%	98.0%

Table 2.1: Comparison of Timm's detection algorithm on the BioID database.

Brackets indicate values that have been accurately measured from authors graphs.

2.3.2 Anjith's approach: Convolution operators & Starburst

In the following section I will provide description of G. Anjith's approach. [2] As already mentioned, it detects iris borders, from which iris respectively pupil centre is derived.

Many algorithms - as well as Anjith's - for iris detection use two-stage approach. In first stage rough eye centre is detected and the exact position is then refined in second stage. This separation allows use of different methods. In first stage we need robust and computationally inexpensive algorithm which can reliably performs on larger areas. For second stage, the precision is the priority.

Rough eye centre localization by Circle Hough Transform

For rough eye centre estimation algorithm assume the shape of the iris is circle. The radius boundaries of the searched circle are calculated from size of the head. The ratio between head size and iris size is hard-coded. (It is empirical value.) Also, as well as in Timm's approach 2.3.1, it is assumed that surrounding of the iris is brighter therefore vectors gradient field point outwards.

The iris is searched by performing convolution of the gradient of the image and class of convolution kernels. Using convolution is similar to using circle Hough Transform (described in 2.3.3.1). However operating on the gradient dispose the need to use an edge detector. This is very beneficial since it problematic to set right parameters for the edge detection without prior knowledge of the input.

Anjith provides exact mathematic definition for convolution operator:

> *The convolution operator is designed as a complex operator with magnitudes as unity. [. . .]*

$$O_{COA}(m,n) = \begin{cases} \frac{1}{\sqrt{m^2+n^2}}(\cos\theta_{mn} + i\sin\theta_{mn}) & iffR^2_{min} < m^2 + n^2 < R^2_{max} \\ 0 & otherwise \end{cases} \tag{2.4}$$

where,

$$\theta_{mn} = \arctan(\frac{n}{m}) \tag{2.5}$$

> *Parameters m and n denote the coordinates of the kernel matrix with respect to the origin. The operator is scaled for equal contributions of circles in the radius range.*

[...] An additional weighing factor (β) is included to increase the contribution of horizontal gradients. Equation for convolution kernel can be made a real-valued kernel as

$$C_{RCC} = \beta Re(O_{COA}) \otimes S_x + \frac{1}{\beta} Im(O_{COA}) \otimes S_y \tag{2.6}$$

where $\otimes$ denotes the convolution operator; Sx and Sy denote the 3×3 Schaar kernels in x and y directions respectively.

Here Anjith as well as Timm and Barth (2.3.1) wants to benefit dark areas of the image. However instead of taking just inversed intensity value of the evaluated point, he considers also point's adjacent area. Weight of adjacent pixels lowers as distance from point increases. This is achieved by weighting kernel W_A.

This is definitely improvement as camera noise cause a lot of intensity distortions. The average intensity of each point in image is calculated as

$$W = (255 - I) \otimes W_A \tag{2.7}$$

Where I and denote the image and W_A is defined as

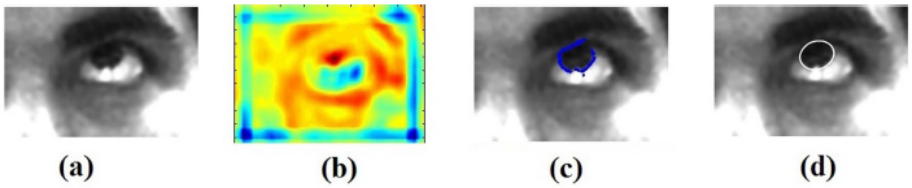
$$W_A(m,n) = \begin{cases} \frac{1}{\sqrt{m^2+n^2}} & if f m^2 + n^2 < R_{max}^2 \\ \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

To obtain final correlation output (CO) Anjith sums the result of convolution with averaged intensities of pixels. Sum is weighted with scalar $\lambda \in [0; 1]$. Here we can see another difference from Timm's approach as Timm used element-wise multiplication for weighting. Anjith's approach is somewhat finer but the actual effect is questionable. If lambda is large it may lead to false detections in dark areas (e.g. eyebrow). On contrary a small lambda may not create enough difference between dark and bright point. Since we are only looking for approximation of centre point I would prefer multiplication.

$$CO = \lambda(I \otimes C_{RCC}) + (1 - \lambda)W \tag{2.9}$$

We can search in result of convolution for maximal value to detect rough eye centre. However the convolution operator itself can produce more (local) maxim and hence false detection. To increase robustness of the algorithm Anjith calculates peak to side lobe ratio. Candidate with highest ratio is selected as approximate eye centre.

Figure 2.4: Stages of Anjith's eye centre detection



(a) Cropped eye region, (b) Correlation surface from the proposed operator, (c) Selected candidate boundary points, (d) Fitted ellipse.

Edge refining and ellipse fitting

After rough eye centre estimation Anjith employs Starburst algorithm (see 2.3.3.2) to detect iris borders. Algorithm searches in radial direction finding strongest edges with agreeing gradient. Note that edges can be chosen with sub-pixel precision. The step does not have to be integral, values of partial derivatives can be obtained as weighted average.

Based on image quality result will contain less or more outliers. However even in prefect image the outliers will be present as iris is never surrounded by sclera entirely. Except for cases unnaturally opened eye, iris is always partially covered with eye lids. Also top part of the eye can be overshadowed. Rays of Starburst algorithm will likely pass trough these sector and create outliers afterwards. It is convenient to filter results by their distance from centre. But since ray origin point (rough centre) can lie anywhere in iris region (or even outside) average distance will change accordingly to the ray angle. Anjith therefore creates angle versus distance plot, which can be filtered with median filter.

After initial filtering RANSAC algorithm (see 2.3.3.3) is used for ellipse fitting. Anjith does not describe procedure exactly, yet he states, that fitting in one iteration of RANSAC is done by Fitzgibbon's Direct Least Squares Fitting. [4] We may therefore assume, that more than 5 points (smallest sufficient points to determine rotated ellipse exactly) are selected into one sample. From fitted ellipse exact iris centre is derived. You can see examples of each fitting stage in figure 2.4.

Testing on BioID dataset

Table 2.2: Comparison of Anjith's detection algorithm on the BioID database.

Method	$\epsilon \leq 0.05$	$\epsilon \leq 0.1$	$\epsilon \leq 0.15$	$\epsilon \leq 0.2$
MIC (cite!)	**86.0%**	91.6%	94.5%	96.9%
Timm & Barth	82.5%	93.4%	95.2%	96.4%
Anjith	85.0%	**94.3%**	**96.6%**	**98.1%**

2.3.3 Used concepts, techniques and algorithms

In this section I will present brief overview of techniques and algorithms frequently used in eye tracking applications.

2.3.3.1 Circle Hough Transform

Circle Hough Transform (CHT) is specialization of Hough Transform, which is feature extraction technique commonly used in computer vision and image analysis. It was proposed by Richard Duda and Peter Hart in 1972. It is used for finding imperfect instances of given object. Such imperfections can be result of noise, result of missing data, presence of outliers and others.

Suppose we are looking for presence of objects described by N parameters. In case of CHT this object would be obviously circle. Since circle in R^2 is defined as $(x - x_c)^2 + (y - y_c)^2 = r^2$ parameter space would be three dimensional (x_c, y_c, r). Without loss of generality we can assume case, when we are looking for circle with fixed radius. As r becomes constant, 3D space is collapsed into 2D. Here every point from our dataset votes for all centers (i.e (x_c, y_c)) which equation of circle for voting point. Afterwards objects are chosen from parameter space based on number of votes and other constrains (minimal distance between objects etc).

Voting process can be implemented as gradual addition of the binary object masks to the accumulator matrix which implements parameter space. Choosing candidates is done as finding local maxim.

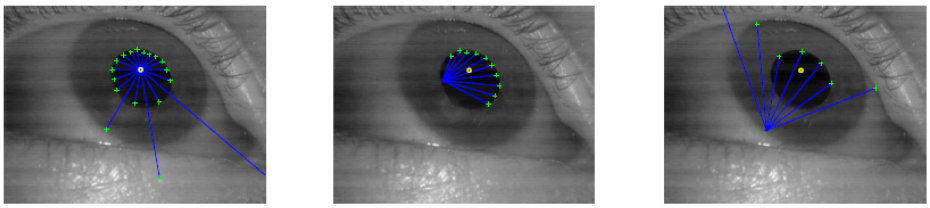
2.3.3.2 Starburst

Suppose we have want detect pupil borders (circle) in bitmap image. One option is to use circle Hough Transform however this algorithm operates on set of discrete points and on bitmap would not produce any useful result. To overcome this problem various edge detectors has been used. Yet if we already know approximate pupil centre location using edge detectors such as canny is computationally wasteful. Another - probably more serious - problem is the right configuration of edge detector. Especially when nature of images changes dramatically. D. Li et al. in their work regarding PoG detection using head-mounted eye tracker proposed algorithm called 'Starburst'. [3]

Edges are detected with set of rays, which are extended from approximate centre in radial direction. (As shown in 2.5 a.) As ray proceed in its direction it looks at gradient of the image. If the gradient vector (respectively its part parallel to the ray) at certain point exceeds given threshold it is marked as border point. Usually rays have also specified maximal distance they can travel.

After that second iteration of rays is send from every found border point. Their directions are limited to 50 degrees around the ray that originally generated the border point. It is expected that if the border point is inlier its rays will mostly generate inliers again. If it is outlier the rays are less likely to generate border point or such point will not be consistent with other points. Therefore ratio between inliers and outliers is increased.

Figure 2.5: Starburst algorithm example



Example a) shows first iteration of rays with two false detections. One ray reached border and did not generate border point. Other two figures shows second iteration from inlier (b) and from outlier (c). [3]

2.3.3.3 Random Sample Consensus

Random Sample Consensus or RANSAC is non-deterministic iterative method to estimate parameters of a mathematical model from set of points. It is designed to prevent outliers to affect estimated parameters. It can be used both as a fitting algorithm or as a outlier detection method.

It is based on assumption that there is more inliers that fit real (searched) model than outliers which randomly fits model with arbitrary parameters.

RANSAC randomly choose N points from the set, where N is smallest number sufficient to determine model parameters. Algorithm then counts how many points fits this particular setting for some maximal error. This is iteratively repeated. Parameters of model with most inliers are result of the algorithm. It is consensus of random sample.

In presence of outliers it will most likely produce better results than method which minimize the error between model and all points in dataset (e.g. Least Square method).

Chapter 3

Design

In the following section I will describe algorithms I modified or designed for my application. I will try to keep this description as general as possible. In some cases is design influenced by used libraries (OpenCV, Dlib). To avoid confusion and uncertainties I will include mathematical definitions.

Algorithms are mostly based on work of G. Anjith [[2]] and D. Li et al. [3] and they are modified to suit conditions of translation of point of gaze into cursor movement.

3.1 Pupil centre detection

Here I will describe all stages of eye centre detection. As I already mentioned when reviewing Timm's approach 2.3.1, direct centre localizations often end up being biased from real centre due to lighting condition. Also the detection is affected by noise. (Noise causes "shaking" of the centre point.) Filtering on the other hand causes bothersome delays. Therefore in my application I am detecting iris location.

Context of use is quite beneficial for the algorithm. When considering person controlling the cursor with its eye, we can expect following conditions:

1. Face will be dominant part of the captured image.
2. Face will be mostly in direct position relative to the screen.
3. Eyes will be usually regularly open.

Furthermore we can set lower and upper bound on size of the iris by defining its approximate ratio to the head size. This ratio of course vary among different people but for setting boundaries it is stable enough.

Detection of the eye centre will be executed in following stages:

1. Rough centre detection by convolution operators.
2. Iris border points detection by Starburst algorithm.
3. Border refining by conic fitting.
4. Extraction of the eye centre as centre of the conic.

3.1.1 Rough centre detection

I am using similar approach as described here subsection 2.3.2.

Regarding third condition mentioned above I expect sclera to be visible (and brighter than iris) and thus create gradient of high magnitude on borders with iris. Vectors of gradient will be pointing outwards from the eye. I will search through matrices of partial derivatives looking for pattern corresponding to the iris borders. (I am dealing with x and y directions separately to simplify equations as much as possible. Anjith's use of the complex form seems redundant to me.)

Let me define the search process formally. First of all I designed an eye pattern as

$$\Psi_x(x,y) = \begin{cases} \frac{x}{|x|+|y|} & \text{iff } R_{min}^2 < x^2 + y^2 < R_{max}^2 \\ \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $\Psi_y(x,y)$ is defined in the same way only having y in the numerator. Accumulators (in sense of Hough Transform filters) are calculated as

$$ACC_x = \frac{\partial I}{\partial x} \otimes \Psi_x \quad \text{where} \quad \otimes \text{ denotes convolution,} \tag{3.2}$$

and ACC_y respectively for other partial derivative. Also we want to benefit dark areas over bright ones as others did. We will use inversion of image I blurred with Gaussian filter.

$$I_{INV} = (255 - I) \otimes G_B, \quad \text{where} \quad G_B = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \tag{3.3}$$

Finally we can obtain rough centre as

$$c^* = \max\{(ACC_X + ACC_Y) \circ I_{INV}\} \tag{3.4}$$

3.1.2 Iris border points detection

From the rough centre Starburst algorithm is launched (as described in 2.3.3.2). I deploy 40 rays in radial direction. The important modification I made here from Li's implementation is that ray do not stop after overcoming specified threshold T.

Our condition is distinct from Li's. First of all they designed Starburst for use with head-mounted infrared tracking device. Such capturing method naturally provides high quality images thus edges are better defined. Also they were searching for pupil not the iris. These two factor made setting of the threshold easier as they could expect much higher values of derivations on pupil borders.

In my implementation I want to avoid using any kind of thresholds as decision-making elements. It is complicated to set threshold correctly without knowing illumination conditions. I do not abandon concept of thresholds completely, but I use them just as lower (upper) bound constrains.

Therefore rays always travel maximal distance (double of the maximal iris radius) and select point with greatest gradient magnitude as border point candidate. (If any of probed points overcome some lesser threshold.) However this solution creates problem of pupil stopping the rays. Sometimes the contrast is higher between the pupil and iris than between iris and sclera. I apply two mechanism to deal with this issue:

1. Start the ray probing in certain distance from the rough centre. (In the implementation it is $\frac{1}{3}$ of iris maximal radius.)

2. Until higher distance from the rough centre ($\frac{2}{3}$ of iris maximal radius) is reached gradient magnitude is lowered by constant coefficient.

3.1.3 Conic fitting

After acquiring set of border point candidates we need to fit a conic to them to obtain simplified information about iris position. There are many ways how to fit conic and the chosen method affects result a lot.

Lets consider the characteristic of data we acquired first. We know that it may contain outliers. That is a problem we can reasonably deal with. But more importantly there will be a lot of missing points - based on how much was the eye hooded and how close to an eye corner the iris was. We may obtain only fraction of border points. I will return to this point in following sections.

Ellipses versus circles

First choice we have to make is whether we want to try fit an ellipse or a circle. Due to the movement of an eyeball circular iris may appear elliptical in the image. Especially in bigger angles. An ellipse therefore promises more accurate fitting results.

However dealing with ellipses is much more challenging. When using Least Square method one is often searching for general conic. That means solving non-linear LS problem with inequality constrain. The constrain has to be imposed otherwise fit my result in degenerate conic or be a trivial solution. Methods of iterative refining were used in the past. Today we can use Direct ellipse-specific fitting proposed by A. Fitzgibbon. [4]

Instead of dealing with inequality Fitzgibbon "incorporates the scaling into the constraint and impose the equality constraint $4ac - b^2 = 1$." and then solves equation directly using generalized eigenvectors.

The important point here is that such system of equations cannot be easily extended by another constrain. Therefore we cannot easily set maximal value for major and minor axes or their ratio. Consequently if we obtain only fraction of border points it is quite possible that residuals will be better minimized by some extremely outstretched ellipse. Example of this case is shown in figure 3.1.3

This problem occurred regularly during my experiments. The ellipse glitched to some extreme size and made detection very unstable.

Decomposed RANSAC

To simplify the problem I abandoned fitting of an ellipse and focused on circles only. Since extreme angles are not expected the error should not increase significantly. Now I will describe algorithm that provided best results.

I decided to merge the problem of excluding outliers with the problem of bounding iris radius. In classic RANSAC search for circles 2.3.3.3 we select 3 points to obtain 3 equations which define circle exactly. However I define radius as constant and run set of RANSAC iterations to find best centre. I repeat this for range of radii defined by iris maximal and minimal size.

This will obviously ensure that bounding condition will be satisfied but moreover it will increase ability to detect outliers. With such constrained model search will likely succeed even if number of outliers exceeds number of inliers, since it is quite unlikely that outliers will from another circle with given radius.

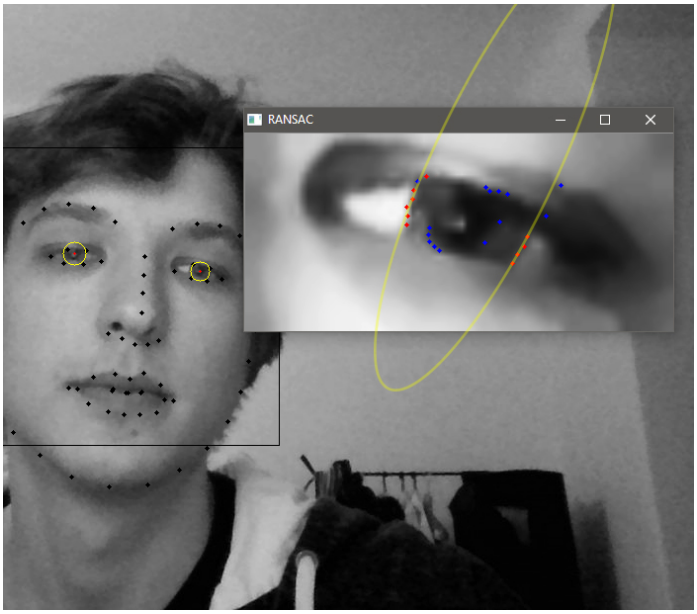


Figure 3.1: Outstretched ellipse as result of Fitzgibbon's fitting.

3.2 Reference point

Before point of gaze detection some sort of calibration must be made to define relationship between eye location in the captured image and real environment. Some complex approaches containing 3D model of head can be applied but since my solution did not achieved sufficient precision (mainly due to low resolution of the webcamera and noise - see chapter 5) I implemented just simple comparing of the iris centre with eye corners.

Eye corners are obtained from facial landmark detection described in previous chapter. Translation function is linear.

Chapter 4

Implementation details

4.1 Libraries

All algorithms were implemented in C++ with use of following libraries.

OpenCV

For my implementation I used mainly OpenCV [7] (Open Source Computer Vision) library. This is commonly used library for computer vision. Opencv.org states that their community has more than 47 thousand members. It can be regarded as standard in term of CV.

Dlib

Dlib [8] is a modern C++ toolkit containing machine learning algorithms implemented in C++ and other auxiliary algorithms. It provides solutions for face detection and feature landmark detection. The latter was used in actual implementation. For face detection I used OpenCV in the end. (I discuss this decision in following section.)

4.1.1 Face detection with OpenCV and Dlib

Both OpenCV and Dlib libraries provide algorithm for face detection. The overview of techniques these libraries are using is in section 2.1. Firstly I decided to use Dlib since I perform facial landmarks detection on detected face as well. (OpenCV does not provide any algorithm for that.) However Dlib's algorithm was excessively slow.

After compiling Dlib with compiler flag -DUSE_AVX_INSTRUCTIONS=ON (to enable vector instructions) performance increased significantly. However the frame rate only with performing face detection was only about 15 frames per second.

I tried to speed up algorithm by minimizing number of detections. I approximated face region based on result of previous face landmark detections. However this approach proved to be inapplicable. Landmark model is based on the size of the face region. Face region is somewhat bigger then bounding box of landmarks points. Scaling the bounding box inaccurately causes detection fail. Box then gradually shrinks or expand.

Finally I decided to switch to OpenCV's Haar Cascade Classifier. It was improvement. I performed testing on BioID database. However combining OpenCV for face detection and Dlib for facial landmarks detection caused drop in accuracy of the latter algorithm. The Dlib's detector is trained for specific cutout of the face image. When the head is rotated OpenCV returns larger region and causes Dlib to fail detection on the retrograde side of the face.

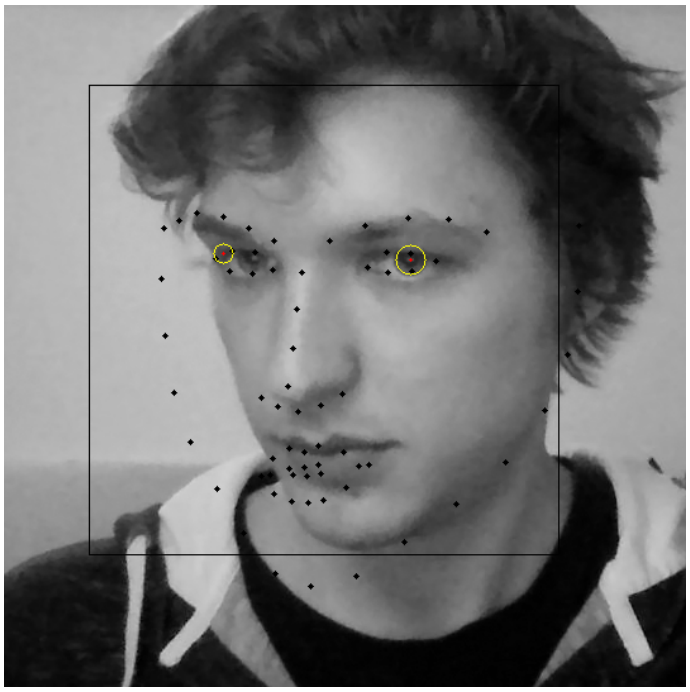


Figure 4.1: Landmark detection error caused by over-sized face region.

Chapter 5

Experiments

In this chapter I describe methods of testing of designed algorithms. I mainly focus on precision of iris centre detection since I did not achieved precision sufficient for mouse cursor control.

5.1 Precision of IC detection

I performed my own experiment with several subjects. The environment for the testing was quite standard. Test was conducted indoors, during daytime. Subjects were seated in front of the screen, mostly facing screen directly. They were asked to minimize spinning of the head as much as possible.

Distance between camera and subjects was approximately 80 cm. Size of the screen was 17 inches. Resolution of the screen was 1366 x 768. Camera captured images in similar resolution. I would consider these conditions as ideal for algorithm performance.

Eleven dots were rendered on the screen - evenly, side by side in one line - as target points. Their y position was in $\frac{2}{3}$ of the screen height. The gap in x-direction was 127 pixels. Subjects were asked to lock their sight on the selected point and when ready, press the dedicated button. After that sequence of 5 images was captured. This took approximately 2 second. Recorded images were stored and the iris detection was performed later.

Captured faces can be found in the "test" folder on attached media.

Figure 5.1: Comparison of gaze at left/right screen border.

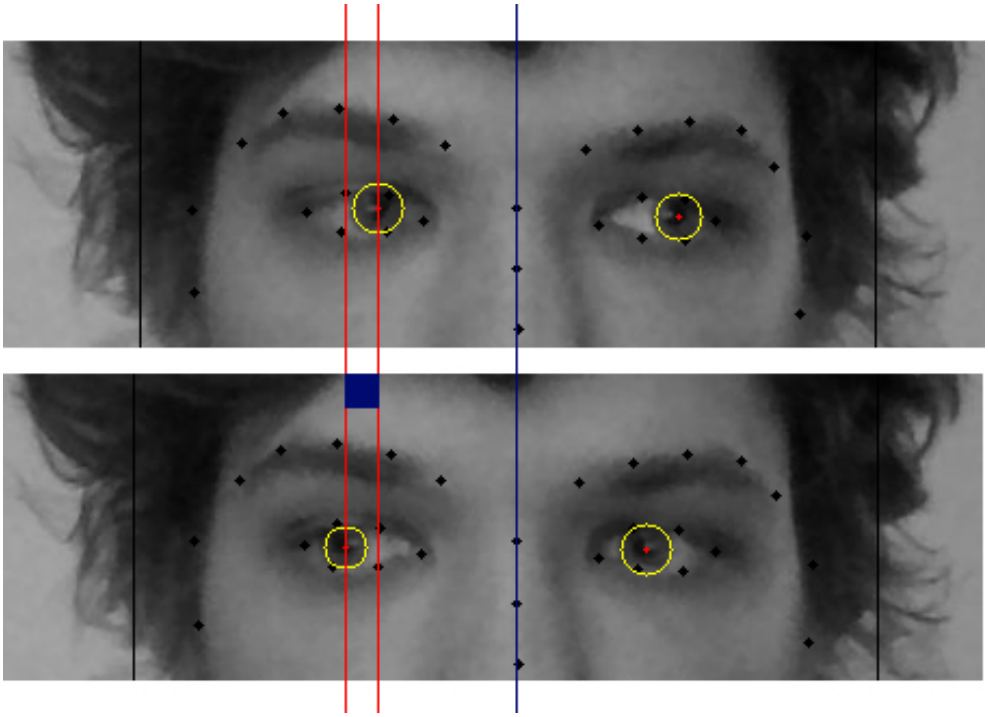


Figure compares the difference in eye positions when looking at left/right screen borders. Blue square denotes size of 16 pixels.

As you can see in figure 5.3, even when looking at stable point in ideal conditions iris centre experience shaking. The deviation is about 1 to 2 pixels. In figure 5.1 you can see comparison of two images, captured shortly after each other. Subject in the images is looking at right (respectively left) screen border. The face position is aligned. This comparison revealed that difference in iris movement when looking at borders of 17" screen at approximate distance 80 cm is only 16 pixels.

With resolution 1366 this gives us ratio of 85 screen pixel for 1 pixel of image of captured eyes. With sub-pixel precision (lets say $\frac{1}{2}$ of pixel) controlling the mouse cursor in the x direction would possible. However with regard to the deviation in the steady state error can scale in "worst case" over 150 pixel, which make this technique unusable.

Testing of calculation of PoG roughly confirmed previous calculation.

Figure 5.2: IC detection test - absolute locations

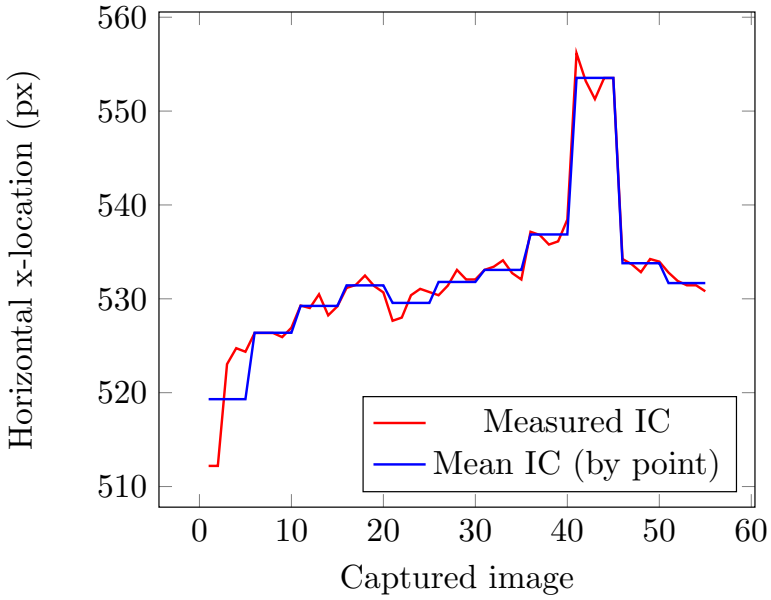
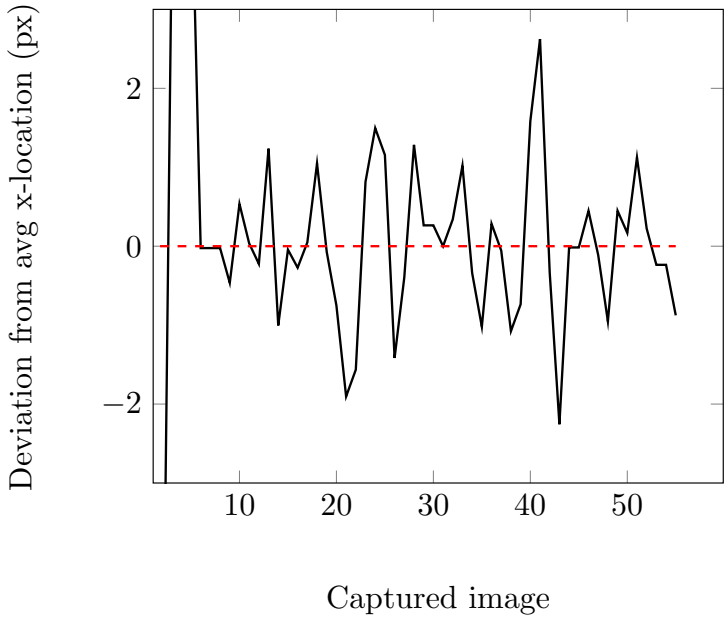


Figure 5.3: IC detection test - deviations from mean location



Conclusion

This work reviewed some algorithms for eye tracking and pupil centre detection. It pointed out some shortcomings and hopefully proposed some useful ideas about PC detection.

However I was not able to achieve sub-pixel accuracy. I showed that this high accuracy is indeed required for translation of the eye movement into mouse cursor movement. To solve this problem, further works must aim for this benchmark. However it is quite possible that only better video input will make controlling of the cursor by eyes possible.

Bibliography

[1] Timm, F.; Barth, E. *Accurate eye centre localisation by means of gradients.* Proceedings of the Int. Conference on Computer Theory and Applications (VISAPP), vol. 1, INSTICC, Algarve, Portugal, pp. 125-130, 2011

[2] Anjith, G.; Aurobinda, R. *Fast and Accurate Algorithm for Eye Localisation for Gaze Tracking in Low-resolution Images.* IET Computer Vision 10.7 (2016): 660-69.

[3] Li, D.; Winfield, D.; Parkhurst, D. J. *Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches.* Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on. IEEE, 2005, pp. 7979.

[4] Fitzgibbon, A.; Pilu, M.; Fisher, R. *Direct least squares fitting of ellipses..* Proceedings of 13th International Conference on Pattern Recognition.

[5] Viola, P.; Jones, M. *Rapid object detection using a boosted cascade of simple features..* Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.

[6] Learn OpenCV. *http://www.learnopencv.com/histogram-of-oriented-gradients/.* [Online; accessed 17-March-2017].

[7] OpenCV. *http://opencv.org/.* [Online; accessed 13-March-2017].

[8] Dlib. *http://dlib.net/.* [Online; accessed 13-March-2017].

[9] Imotions. *https://imotions.com/blog/top-8-applications-eye-tracking-research/.* [Online; accessed 7-March-2017].

Appendix A

Acronyms

IC Iris location

PC Pupil location

PoG Point of gaze

RANSAC Random Sample Consensus

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format
- `test` ..... the directory containing test images