Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

Article

Implicit calibration using Probable Fixation Targets

Pawel Kasprowski [1,*], Katarzyna Harężlak [1] and Przemysław Skurowski [1]

[1] Institute of Informatics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland;
* Correspondence: pawel.kasprowski@polsl.pl; Tel.: +48-32-237-1339

Version November 12, 2018 submitted to

Abstract: Proper calibration of eye movement signal registered by an eye tracker seems to be one of the main challenges in popularizing eye trackers as yet another user input device. Classic calibration methods taking time and imposing unnatural behavior of users have to be replaced by intelligent methods that are able to calibrate the signal without conscious cooperation with users. Such an implicit calibration requires some knowledge about the stimulus a person is looking at and takes into account this information to predict probable gaze targets. The paper describes one of the possible methods to perform implicit calibration: it starts with finding probable fixation targets (PFTs), then uses these targets to build a mapping - probable gaze path. Various possible algorithms that may be used for finding PFTs and mapping are presented in the paper and errors are calculated utilizing two datasets registered with two different types of eye trackers. The results show that although for now the implicit calibration provides results worse than the classic one, it may be comparable with it and sufficient for some applications.

Keywords: eye tracking; calibration; eye movement; optimization

1. Introduction

With growing access to cheap eye tracking devices, utilizing simple web cameras, and advancements in image processing and analysis, it has become possible to incorporate eye tracking as yet another human computer interface. However, one of the most important problems when considering eye tracking usage on end users market is the necessity of calibration. Most eye trackers require to be calibrated prior to their each usage. Unfortunately, currently used calibration procedures are cumbersome and inconvenient. Additionally, to achieve reliable results, they often require qualified operators that control the process. As the result, eye tracking is reliable only in laboratory conditions and usage of eye tracking "in the wild" by unqualified users is often impossible.

The goal of the research summarized in this paper is to develop calibration methods that make utilizing eye tracking also possible for unattended and not experienced end users. The idea that the calibration may be done implicitly, during normal user's activities is our main research hypothesis. The possibility of such a calibration has been confirmed in several papers [1–4], however, the main challenge is to achieve generality of the solution and accuracy that is sufficient for end users' eye tracking usage for various eye tracking applications.

1.1. Calibration

Every eye tracker produces some output, which may be multidimensional and depends on the type of the eye tracker used. It may be information about eye center coordinates, light reflections (glints) positions, signal intensity (for IROG or EEG eye trackers) or just the whole eye image utilized by appearance based methods. This output may be treated as a set of attributes $e_1, ...e_n$. The purpose of the calibration is to build a model (calibration function $f_{calibration}$) that converts this output into information about the gaze point – a place where the examined subject is looking at the particular moment. The gaze point is given in scene coordinates and typically is presented as a point in a two dimensional scene space (Equation 1) but may also be defined in 3D space [5].

$$gaze_i(x, y) = f_{calibration}(e_1, e_2, ...e_n) \quad (1)$$

© 2018 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

The classic calibration procedure requires subject to look for some time at the predefined locations on a scene (so called *targets*). Having some amount of eye tracker output data with the corresponding known gaze locations it is possible to build the calibration function that will be able to map a future eye tracker output to gaze points. Quality of the function depends on the amount of training data available, diversity of recorded gaze points' positions and quality of the recordings [6]. While two first aspects may be adjusted by prolonging the calibration procedure, the latter is difficult to inspect.

Such a calibration scenario typically requires focusing at one target and leave eyes stable for a long time to collect necessary data. As human eyes tend to change focus several times a second, gazing at not moving target is not natural. Therefore, such a procedure is cumbersome and inconvenient for users and prone to errors when participants do not look where they are expected because of tiredness or loss of focus.

Another problem is that the accuracy of results derived by the calibration function tends to decrease in time [1,7]. In order to solve this problem the calibration procedure must be repeated from time to time which makes the usage of an eye tracker even more inconvenient.

These issues result in the search for alternative ways of the calibration. There were many other scenarios tested, including: a click based calibration [8], smooth pursuit calibration [9], vestibulo-ocular reflex (VOR) based calibration [10] and others. However, all those methods share the same burden: they take time, are not very convenient for users and their results lose accuracy in time. For this reason a growing interest in methods able to perform the calibration implicitly, during normal user's activities may be observed.

1.2. Implicit calibration

The main assumption of the presented research is that when information about an observed scene is available, it is possible to predict – with some probability – where a subject will look at. Having this information and data from an eye tracking device, it is possible to construct a model that is able to estimate the gaze point, similarly to the classic calibration. Such a calibration does not require any explicit and active cooperation with users – in fact the users may even not be aware that they are being calibrated. Therefore, such a calibration is called *implicit calibration* and because it may be performed on-line, during the whole eye tracking session it may also be treated as the continuous calibration.

Implicit calibration always requires information about the scene, the user is looking at. The probability that user looks at specific points may be calculated using various criteria. The simplest algorithm may utilize center location paradigm that states that most time user is looking at the center of a scene [11]. More sophisticated algorithms may use saliency map models or any other predictions [12]. It is worth noting that the implicit calibration may replace a classic calibration but may also be used only to improve or maintain an already performed calibration [13,14].

1.3. The state of the art

The problem of implicit calibration has been already analyzed in several publications, however, to the best of our knowledge, there are still no out-of-the-box solutions, which may be directly used by potential users.

One of the first papers utilizing solutions similar to implicit calibration was [1]. The authors aimed to correct systematic calibration drift using clues about gaze position obtained during participant's work. The clues were locations of mouse clicks (with the assumption that people look where they click) and were used to check the calibration quality and invoke recalibration if necessary. The similar assumption was made in [14], where the initial calibration was completely removed and calibration was done only based on mouse click positions.

When the number of clicks is insufficient, it is possible to use other information about the scene, e.g. identify objects that should be fixated to accomplish tasks, such as labels, icons to find etc. This idea was used to correct drift errors in the calibrated signal in [7,13].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

When a scene does not consist of easily distinguishable objects, as in the case of user interfaces, a more general solution must be used that tries to predict fixation targets taking into account only information about the scene a user is looking at. Such a solution involves creation of a saliency map - a function that estimates for each scene point a probability that user will look at this point [15].

Probably the first paper in which saliency maps were used as an input to the calibration model was [3], extended later in [16]. The authors used graph-based visual saliency (GBVS) model [17] and face detector to calculate saliency for each frame of a movie. The saliency map was later aggregated to improve optimization. The same saliency model was used in [11]. Another example includes [4] where the map was built using a self developed RCNN based algorithm.

Instead of using algorithmic saliency model it is also possible to use genuine human gazes recorded earlier for the same image as it was done in [18]. However, such a solution is possible only when the same stimulus is presented many times and is not feasible in general case.

Some methods use not only a pure saliency maps but utilize also natural constraints like center bias that assumes that gaze locations are biased towards the center of the scene during free viewing [19,20].

When saliency for every frame is known, it may be used to estimate parameters of a calibration model that, for the given data, maximizes the likelihood that the computed gaze falls in the region of high saliency. There are different optimization methods used: Gaussian process regression [16], Bayesian network [11], K-closest points and mixture model methods [18] or minimization of the KL-divergence between distributions [4].

Most experiments published so far recorded a set of gazes while a person was watching static images [4,11,18,21]. Only in [16] the authors used a set of frames from a movie and computed a saliency map for each frame. Results presented in one of the recent papers [22] showed that implicit calibration may be used also in an unconstrained environment.

1.4. Paper's contribution

The paper describes a novel technique to perform an implicit calibration. This technique is based on the assumption that in most cases there are only few points (targets) for a scene being observed where users may fixate their gazes. After finding this targets (called Probable Fixation Targets - PFTs) it will be possible to build a calibration model (Figure 1).

Figure 1. The image with two probable fixation targets (PFTs). It may be assumed that while looking at the image, most of the time people will fixate on one of the regions inside the yellow squares.

The paper describes all consecutive steps that are necessary to perform the implicit calibration using probable fixation targets (PFTs). The steps include:

- saliency maps prediction,
- conversion of saliency maps to lists of probable fixation targets (PFTs),

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

- choosing a mapping (a set of PFTs) using miscellaneous criteria and heuristic algorithms,
- using the chosen PFT mapping to build a calibration model.

The experimental part of the paper checks the usability of the proposed solutions using two datasets containing uncalibrated eye movement data recorded by different eye tracking devices and various stimulations. The datasets are publicly available and may serve as a convenient testbed for future calibration algorithms.

The paper summarizes the idea introduced in two ETRA conference publications [23,24], however, it significantly extends it by:

- Analyses of algorithms for PFT (targets) creation using saliency maps.
- Introduction of new mapping comparison functions.
- More extensive analyses of results and possible algorithms' parameters.
- Introduction of the new dataset.
- Implementation of all presented algorithms in publicly available new version of ETCAL library [25].

2. Methods and Materials

The original idea utilized during the research presented in this paper is introducing an additional step of processing during which saliency map is used to build a short list of probable fixation targets (PFTs). Having a limited number of such targets for every fixation, it is possible to build the model assuming that most of the time a user would look at one of these targets (Figure 2).

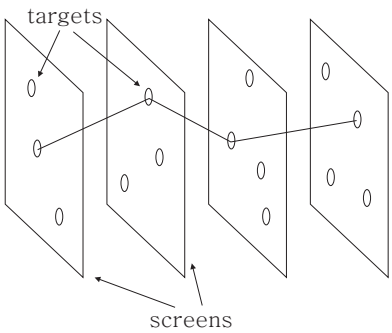

Figure 2. Fixations with targets. The figure shows subsequent scenes/screens with several targets (PFTs) on each. Horizontal axis represents time. The path visualizes one possible mapping.

The first step of the processing algorithm is the extraction of targets from a saliency information. The saliency information may be derived using:

- Expert knowledge.
- Ground truth saliency recorded earlier for other observers of the same scene.
- Saliency map created by any of saliency models.

The result of that first step is the list of probable fixation targets (PFTs) for every fixation.

The next step of the processing formally may be defined as follows: There are M fixations and for each fixation F_i there is a list of m_i PFTs: $\{T_i^{(1)}, T_i^{(2)}, ...T_i^{(m_i)}\}$ where T is a (x, y) pair. Additionally, an eye tracker output is defined for each fixation. The eye tracker output is a set of attributes $(e_1, e_2, ...e_n)$ and it will be denoted as $E_i(e)$ in the subsequent text. The e may be any set of attributes depending on the type of eye tracker: eye center coordinates in eye camera, eye-glint vector, EOG signal etc.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

$$\begin{aligned} F_1 &: E_1(e), \{T_1^{(1)}, T_1^{(2)}, ...T_1^{(m_1)}\}, \\ F_2 &: E_2(e), \{T_2^{(1)}, T_2^{(2)}, ...T_2^{(m_2)}\}, \\ &... \\ F_M &: E_M(e), \{T_M^{(1)}, T_M^{(2)}, ...T_M^{(m_M)}\}. \end{aligned} \tag{2}$$

Subsequently, to calculate parameters of the calibration function, one of the PFTs must be chosen for every fixation. So, for every fixation an index of one of its PFTs is chosen denoted as c_i where $c_i \in \langle 1, m_i \rangle$. The list of chosen indexes $c_1 ... c_M$ is called a *mapping*.

$$mapping = \{c_1, c_2, ..., c_M\} \tag{3}$$

The main research issue for the algorithm of implicit calibration presented in this paper is providing the best mapping. Two problems have to be considered:

- How to compare mappings in order to estimate, which one is better when no ground truth information is known (it is called mapping comparison function (*MCF*) in the subsequent text)
- How to find the best possible mapping using the *MCF* for comparison. Due to a very large number of possible mappings a heuristic algorithm must be used here.

When specific mapping is found, using some *MCF* and optimization function, it is possible to create a pair: $(E_i(e), T_i^{(c_i)})$ for each F_i. A list of such pairs may be used to calculate parameters of a calibration function in the same way as during a classic point of regard calibration [25].

Every step listed in this section is described in detail in the following subsections. Specifically:

- Extraction of targets (PFTs) from saliency information.
- Choosing the mappings comparison function (MCF).
- Developing heuristic algorithms for searching the mapping space (genetic and incremental)
- Building a calibration model using the ETCAL library ([25]).

2.1. Extraction of targets (PFTs) from saliency information

The main requirement for the implicit calibration is knowledge of the observed scene. Based on its analysis it is possible to build a function calculating, for each point of the scene, the probability that a subject will focus on it.

The simplest algorithm may utilize center location paradigm that states that most time user is looking at the center of a scene [11]. More sophisticated algorithms may use various predictions [12], however the most popular way to calculate the probability function is the application of a saliency model that produces a saliency map [26].

Saliency maps are computational models of human visual attention, the concept was conceived by Koch and Ullman in [27]. It is topographical map of a scene, that describes scene fragments as weights – how likely human gaze will be fixed at certain location. However, despite apparent similarity, saliency maps should not be interpreted or confused with probability density function of the subconscious process. Contrary to the random processes, selection of fixations is not a stochastic process of independent events – it is chaotic [28] – therefore describing it as PDF would be irrelevant.

The first practical, biologically inspired, implementation of saliency model was proposed by Itti, Koch and Niebur [15]. It started intense research on further models based on various fundamental premises and assumptions [29]. It may be the bottom-up approach that uses only image parameters (like contrast) or top-down that uses knowledge about human behavior (like face detector). Recently, methods that combine both approaches gained popularity – e.g. model by Le Meur [30] modelling saliency map with low-level visual feature based saliency map and modelling the fixation selection

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

based on context dependant statistics of saccade length and angle, inhibition of return, and forgetting of previous fixations.

There were 9 different saliency models used in this research. The following list enumerates and briefly describes each of the employed saliency maps models:

- BMS – Boolean maps saliency [31] – a graphical model based on series of thresholds and sophisticated mathematical morphology,
- CovSal [32] – based on the innovation introduced in a certain area when compared to its neighborhood in image, using covariance as a measure,
- FES [33]– Bayesian bi-variate modelling of saliency on the basis of the center-surround model at multiple scales.
- GBVS – graph based visual saliency model by Harel et al. [17], with location vectors dissimilarity used as graph edge values,
- IttiKoch – biologically plausible model, based on modelling of the underlying processes [15],
- LDS – Learning Discriminative Subspaces [34] – based on the image segments projection onto pre-trained dictionary patches of PCA reduced visual features.
- RARE2012 [35] – based on the multiscale innovation (rarity) in location, identified with Gabor filters and information theoretic measure self-information.
- SR-PQFT – is a spectral residual (SR) model (as proposed by Hou and Zhang in [36]), however, in a variant that was developed by Schauerte and Stiefelhagen [37] for color images, where color values are quaternion encoded in opponent color space and quaternion formula of Fourier transform is employed (PQFT),
- SWD [38] – spatially weighted dissimilarity - saliency model also based on the difference from the surrounding, where outlying segments are identified in PCA reduced feature space.

Another option is to create saliency maps using eye tracking information gathered during genuine observations of several participants. We have also done such experiment and used the data to build ground truth ('GT') saliency maps.

Every saliency map had to be transformed to a set of probable fixation targets (PFT). The list should contain only a limited number of points with high saliency. For this purpose we used the watershed algorithm [39] to extract the maxima on the saliency map. The number of PFTs varied for images and maps and ranged from 1 to 16.

Additionally, we also tested if PFTs may be set manually, by the 'fixation expert'. One of the co-authors of the paper manually defined PFTs for each image based on his experience (before seeing any saliency map). This set of PFTs was called 'expert' ('EXP') in the Results section.

Examples of maps and calculated PFTs were presented in Figure 3.

2.2. Choosing the best mappings comparison function (MCF)

In order to calculate parameters of a calibration function it is necessary to choose one target with index c_i for every fixation F_i. When every fixation has one target mapped, it is possible to create a pair: $(E_i(e), T_i^{(c_i)})$ for each F_i. A list of such pairs may be used to calculate parameters of a calibration function in the same way as during a classic point of regard calibration.

The question is how to find a correct mapping from a list of all possible mappings. Without ground truth information the only data that may be used is the uncalibrated output from an eye tracker. It is necessary to define criteria that may be used to compare two distinct mappings. Such the criteria function is called mappings comparison function (*MCF*).

2.2.1. Regression based MCF

One of the possible solutions is to evaluate the specific mapping by: (1) building a linear function that recalculates eye tracker output to target coordinates (g_x, g_y) and (2) checking the function error for the same target points.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

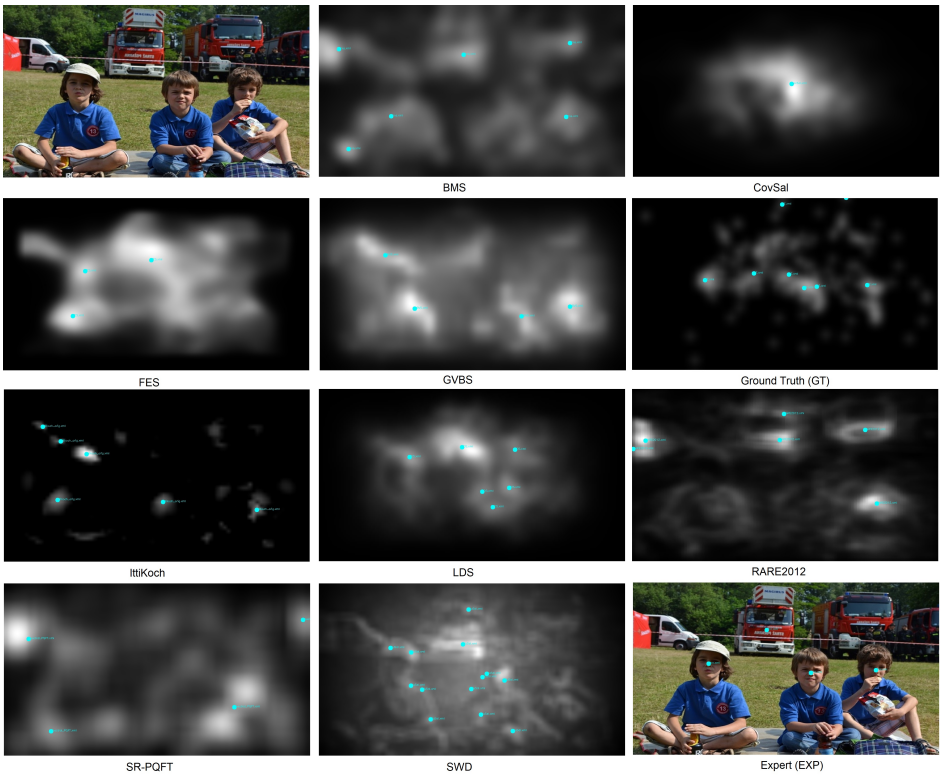

Figure 3. Exemplary image with corresponding saliency maps obtained with different methods. Probable fixation targets (PFTs) presented as cyan dots with labels on each image.

The linear function that recalculates eye tracker output separately to vertical and horizontal coordinates of the gaze may be defined as follows:

$$g_x = \beta_{0x} + \beta_{1x} * e_1 + \beta_{1x} * e_2 + ... + \beta_{nx} * e_n \tag{4}$$

$$g_y = \beta_{0y} + \beta_{1y} * e_1 + \beta_{1y} * e_2 + ... + \beta_{ny} * e_n \tag{5}$$

where g_x are horizontal and g_y vertical gaze coordinates and parameters β_{ij} are regression coefficients calculated using Levenberg-Marquardt algorithm based on known eye tracker output $E_i(e_1...e_n)$ and the corresponding PFT values $T_i(t_x, t_y)$.

The next step is the calculation of coefficient of determination R^2 that compares function's results with target values:

$$R_x^2(t_x, g_x) = 1 - \frac{\sum_i^M (g_x(i) - t_x(i))^2}{\sum_i^M (t_x(i) - \overline{t_x})^2} \tag{6}$$

where M is the number of fixations and $\overline{t_x}$ is the average of all reference values $t_x(i)$. R_x^2 is equal to 1 when the model fits perfectly, i.e. every $g_x(i)$ point is exactly at the same location as the corresponding $t_x(i)$ point.

The same formula may be used for vertical axis to calculate $R_y^2(t_y, g_y)$. Both results are then combined together to form one value that is treated as 'quality' of the mapping:

$$MCF_{REG}(mapping) = (R_x^2(t_x, g_x) * R_y^2(t_y, g_y))^2 \tag{7}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

It may be assumed that the higher value of MCF_{REG} means that is was easier to build a linear function recalculating eye tracker output to the given mapping and therefore the probability that this mapping is correct (i.e. a person really looked at this targets) is higher.

2.2.2. MCF based on relative distances

The MCF_{REG} function is applicable in any general case, however it is not very efficient as it requires calculation of a regression model and its error for each mapping. Another option is to utilize the fact that some information of a spatial correlation between gazes may exist even in uncalibrated signal.

For instance, we can assume that if we calculate Euclidean distance between eye tracker data points (E_i) it should be correlated with distances between gaze points (T_i) corresponding with that outputs. So, when $\|E_i, E_j\| < \|E_j, E_k\|$, we can expect that for the chosen targets the distance $\|T_i, T_j\|$ will also be lower than $\|T_j, T_k\|$.

This assumption was used to calculate another MCF. For each three fixations i, j, k the proportion of distances was calculated using the formula (8):

$$P_{ijk} = \left(\frac{\|E_i, E_j\|}{\|E_i, E_k\|} - 1\right) / \left(\frac{\|T_i, T_j\|}{\|T_i, T_k\|} - 1\right) \tag{8}$$

and then the P values for all triples in the mapping were used to calculate the value of MCF_{DIST} for the mapping:

$$MCF_{DIST}(mapping) = \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} \begin{cases} 1, & P_{ijk} > 0 \\ 0, & otherwise \end{cases} \tag{9}$$

2.2.3. MCF based on correlation between distances

The MCF defined in the previous section is time consuming as it requires comparison of every triple of fixations. Therefore, a faster method was also proposed and tested that works based on the same principle (distances comparison) but is less complex.

At first the distances are calculated for each neighboring pairs (E_i, E_{i+1}) to build a variable $\|E\|$. Then all neighboring distances (T_i, T_{i+1}) are used to form another variable $\|T\|$. We assume that relations between distances in both variables will be similar, therefore the final value of MCF_{DISTC} is calculated as a correlation coefficient between both variables:

$$MCF_{DISTC}(mapping) = \rho(\|E\|, \|T\|) \tag{10}$$

Once again the assumption is that higher value of the MCF (i.e. higher correlation between distances) means the better mapping.

2.2.4. MCF based on direction

Another assumption taken into account was that the 'shape' of eye tracker output signal should be similar to the resulting gaze path. This assumption may be used when eye tracker produces only two attributes. It seems to be a difficult constraint, however, it is the case of many eye trackers such as head worn eye trackers that output eye center coordinates, remote eye trackers that output eye center - glint vector or EOG eye trackers that produce two voltage signals.

For instance, when VOG eye tracker is used and an output signal consists of eye center coordinates as visible in an eye camera, it may be assumed that a correlation exists between the direction of eye center changes and gaze positions. Generally, when eye center moves down in the camera image, it may be expected that gaze should move down as well. Similarly, when eye center moves right it may be expected that gaze position moves left (due to the mirror effect) and so on. Therefore, the 'shape' of eye center points should be similar to the shape of gaze points (see Figure 4).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

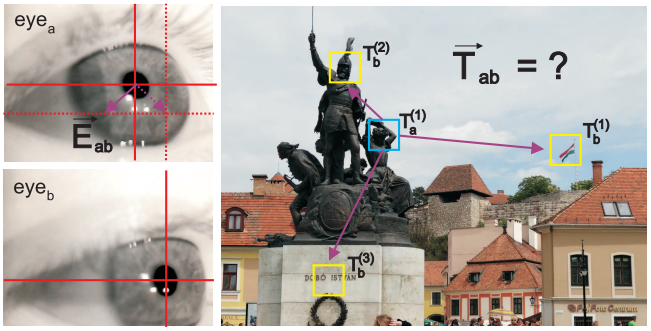

Figure 4. Participant was looking at $T_a^{(1)}$ during fixation a and eye center was in the point shown in eye_a image. Having fixation b with eye center in the point shown in eye_b image it may be assumed that the correct gaze target should be $T_b^{(3)}$, because it results in the lowest angle between $\vec{E_{ab}}$ flipped horizontally (Eq. 12) and $\vec{T_{ab}}$ (Eq. 13).

Taking this assumption into account it was possible to define another MCF to evaluate a mapping. At first, for two fixations F_i and F_j it is possible to calculate a vector between eye center positions for these fixations:

$$\vec{E_{ij}} = (e_x(j) - e_x(i), e_y(j) - e_y(i)) \tag{11}$$

In case of eye camera and eye center coordinates the vector has to be flipped horizontally:

$$\vec{E_{ij}} = (e_x(i) - e_x(j), e_y(j) - e_y(i)) \tag{12}$$

It is also possible to calculate a vector between two targets:

$$\vec{T_{ij}} = (t_x(j) - t_x(i), t_y(j) - t_y(i)) \tag{13}$$

Because of the supposed similarity between eye centers and gaze points it may be assumed that both vectors should be more or less parallel to each other (see Figure 4).

When a cosine of angle between $\vec{E_{ij}}$ and $\vec{T_{ij}}$ vectors will be calculated for every two fixations, it may be used to establish a mapping comparison function (MCF):

$$MCF_{DIR}(mapping) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} cos(\vec{E_{ij}}, \vec{T_{ij}}) \tag{14}$$

For this MCF it is assumed that the mapping with higher value should give better calibration results (see Figure 4 for visual explanation).

2.2.5. Fusion of more than one MCF

The mapping comparison functions (MCF) presented in the previous sections should give some information about the mapping's quality. It is also possible to combine several MCFs with different weights to create a new - possibly more robust - mapping comparison function according to formula (Equation 15):

$$MCF_{FUS}(mapping) = w_0 * MCF_i + w_1 * MCF_j + ... \tag{15}$$

Such a function with correctly tuned weights may give better results than each of the MCFs separately.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

2.3. Heuristic algorithms for searching the mapping space

One problem is to find the MCF, which enables comparison between mappings, another problem is to find the best mapping according to the given MCF. The problem is not trivial. If there are n fixations and m_i targets for every fixation the number of possible mappings is $m_1 \times m_2 \times ... \times m_n$, so it is not feasible to do an exhaustive search through all mappings. Therefore a heuristic algorithm must be used. Two possible heuristics are proposed in this paper: genetic and incremental.

2.3.1. Genetic algorithm

The genetic algorithm treats each target as a gene and each mapping as a chromosome (set of genes). The task is to find the best possible chromosome. At first the algorithm randomly chooses some number of mappings (e.g. 200). Then it evaluates every mapping and mixes some of the best of them to build new mappings (chromosomes) using cross over. Additionally, each chromosome is mutated - some number of genes (targets) is changed randomly. The process is done in many iterations (e.g. 1000) to find the best possible mappings. The algorithm is described in details in Algorithm 1.

Algorithm 1 Heuristic genetic algorithm searching for the best mapping

```
procedure FINDBESTMAPPPING
    pop ← 200 random mappings
    iterations = 0
    noProgress = 0
    while iterations < 1000 and noProgress < 200 do
        newPop ← ∅
        newPop.add(pop.bestMapping)
        for i = 2 → 200 do
            mapping1 = chooseGoodRandomMapping(pop)
            mapping2 = chooseGoodRandomMapping(pop)
            newMapping = crossOver(mapping1, mapping2)
            newPop.add(newMapping)
        for i = 2 → 200 do
            mutate(newPop.mapping(i))
        if newPop.bestMapping > pop.bestMapping then
            noProgress = 0
        else
            noProgress + +
        pop = newPop
    return pop.bestMapping
procedure CHOOSEGOODRANDOMMAPPING
    Choose 5 random mappings and return the one with the best fitness
procedure CROSSOVER(mapping1,mapping2)
    Return a new mapping with values randomly chosen from the first or the second mapping
procedure MUTATE(mapping)
    Randomly change 1.5% indexes in the mapping
```

2.3.2. Incremental mapping

The genetic method explained in the previous section has two important disadvantages: it needs all data to start optimization and it is quite slow. To use the implicit calibration in a real time scenario it is necessary to propose an algorithm that is able to calculate calibration model on-the-fly and is able to improve the model when new data arrives.

The algorithm for incremental implicit calibration works as follows: It starts when only few fixations are available and creates a list of all possible mappings. When the next fixation (with its own PFTs) arrives, the algorithm creates a new list by adding one of the new PFT indexes to every

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

mapping from the previous list (see Algorithm 2). Every new mapping is evaluated according to the score calculated for the chosen optimization criterion (Line 14). The list of mappings is sorted in descending order (Line 6). The same is done for every new fixation - when fixation $n+1$ is added and there are x mappings (of length n) on the list, a new list is created with $x \times m_{n+1}$ mappings. The number of indexes in every mapping on the new list is $n+1$. The mappings are sorted in descending order according to the given criterion. When the number of mappings on the list exceeds a predefined maximal number $cutoff$ (e.g. 100), the algorithm removes mappings with indexes above $cutoff$ (the ones with the lowest criterion value) (Line 7).

Algorithm 2 Iterative algorithm finding $cutoff$ best mappings after adding a new fixation with $newTargetIndexes$ to mappings created so far

```
1:  procedure NewMappings(oldMappings, newTargetIndexes)
2:      newMappings ← ∅
3:      for all mapping ∈ oldMappings and index ∈ newTargetIndexes do
4:          newMapping ← AddTarget(mapping, index)
5:          newMappings.add(newMapping)
6:      SortDescending(newMappings)
7:      RemoveElementsOverCutoff(newMappings, cutoff)
8:      return newMappings
9:  procedure AddTarget(oldMapping, newIndex)
10:     newMapping ← ∅
11:     for all index ∈ oldMapping do
12:         newMapping.add(index)
13:     newMapping.add(newIndex)
14:     newMapping.calcValue()
15:     return newMapping
```

An outcome of the procedure may be used at every moment and applied to build a calibration model at the early stage of data processing, even after several steps. It may be expected that the calibration error will be high at the beginning, however it will decrease with the number of fixation used.

The main advantage of this solution comparing to heuristic algorithms is that it is considerably faster - the complexity is linear because calculation of a value for a new mapping may utilize the already calculated value of the mapping one index shorter. Moreover, as it was mentioned above, the results are available from the beginning of the experiment (although are really valuable only after some time - see the Results section for details).

2.4. Removing irrelevant fixations

When the best mapping is chosen using the steps explained in the previous sections, it should be additionally processed to remove irrelevant fixations. The rationale behind this step is that even when participants look at the predefined PFTs, there will always be some number of fixations that fall somewhere else.

At first all fixations - pairs $\{E_i(e_1,...e_n), T_i^{(c_i)}(x,y)\}$ - are used to calculate parameters of the linear polynomial calibration function f_l using the Levenberg-Marquardt (LM) damped least-squares algorithm [40]. In the next step gaze coordinates are calculated for every pair based on the following function:

$$gaze_i(x,y) = f_l(E_i(e_1,...e_n)) \tag{16}$$

and the Euclidean distance between the $gaze_i(x,y)$ and $T_i^{(c_i)}(x,y)$ is calculated. Then $removalPercent$ of pairs with the biggest distance is removed from the dataset.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

Removal of outliers is of course a broad topic and many algorithms may be used for this purpose. The RANSAC algorithm [41] is an example of commonly used approaches, which we tested during our studies. However, because the results after its usage were not significantly better than after using the approach presented above, we finally decided to use the latter as it is simpler and faster.

2.5. Classic calibration using ETCAL library

The last step is the usage of the mapping to build a calibration model. When the training set consists of pairs of eye tracker output (E) and the corresponding one target (T) it is possible to use these pairs as a training set and build a calibration model that is able to calculate gaze point (G) for every possible eye tracker output (E).

There are many possible methods that may be used for this purpose, starting from polynomial approximation to nonlinear methods such as Support Vector Regression or Neural Networks [6]. The ETCAL library, which was created by the authors of the paper, implements both polynomial and SVR algorithms. The library allows to choose the specific algorithm and train it parameters. However, for simplicity, we decided to skip the training phase and use a classic quadratic polynomial function. The task was therefore to use the training set to calculate parameters of two quadratic equations that may be later used to find horizontal and vertical gaze coordinates. When there are only two eye tracker output values (like e.g. x and y coordinates of eye center) there are twelve parameters of the equation (Equation 17):

$$\begin{aligned} g_x &= A_x e_x^2 + B_x e_y^2 + C_x e_x e_y + D_x e_x + E_x e_y + F_x \\ g_y &= A_y e_x^2 + B_y e_y^2 + C_y e_x e_y + D_y e_x + E_y e_y + F_y \end{aligned} \tag{17}$$

where g_x and g_y are gaze point coordinates. For more E values the equation becomes more complicated, however the ETCAL library is able to solve it using the Levenberg-Marquardt algorithm [40].

3. Results

Two datasets recorded with two types of eye trackers: the remote one, placed under a display (the Eye Tribe) and the head mounted one (Pupil-Labs), were prepared to verify the correctness and performance of the proposed methods.

For both eye trackers we registered eye movements of participants looking at predefined stimulations consisting of static images. In order to preserve generality of the results there were different sets of images and different groups of participants involved in each experiment.

After collecting data the next step was to compare the implicit calibration with the classic calibration. Our assumption was that, even when we did not expect the implicit calibration to outperform the classic one, the results would be comparable.

We compared 11 ways of producing PFTs: with the usage of expert knowledge (EXP), ground truth (GT) and nine saliency models. For the given PFTs we searched for the best mapping using both genetic and incremental algorithms and four MCFs described above: REG, DIST, DISTC and DIR. Additionally, we checked results for various fusions of these functions and *removalPrecent* values.

3.1. Head-mounted eye tracker

The first experiment aimed at checking if the proposed implicit calibration algorithms may be used for a headset eye tracker equipped with two cameras: eye camera and scene camera. Twelve participants took part in the experiment. The participants task was to look at the stimulation displayed on the 21 inch display. They were unconstrained and could move their heads freely.

The stimulation presented on the display consisted of the following steps:

- Initial classic nine points calibration

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

Figure 5. Eye trackers used during the experimental part of the research. Left: Pupil Labs head-set eye tracker, right: The Eye Tribe remote eye tracker to be mounted below a screen.

- Reference screen 1
- 34 images - two seconds per image
- Reference screen 2
- 34 images - two seconds per image
- Reference screen 3

There were nine equally distributed points presented on each reference screen. The task for participants was look at the point and click it with mouse. Every point had to be clicked twice. The results were later used to calculate errors of calibration algorithms. The whole stimulus presentation has been published and is available online: https://youtu.be/CLtNx0IVWmU.

The result of every session was a set of frames taken from a scene camera and eye camera. At first every scene frame was processed to find screen corners and the corresponding eye camera frame was processed to find eye center coordinates. As eye camera was always in the same position in relation to participant's eye, the eye center coordinates could be used as eye tracker output sufficient to calculate gaze position. Therefore, eye tracker output E is denoted as $eye(x_e, y_e)$ for this experiment.

Eye center coordinates in eye camera frame were found after a pipeline of transformations: conversion to monochrome, histogram equalization, binarization, dilatation, erosion and finally Canny edge detection. Screen corner coordinates in scene camera frame were found after calculating the frame image integral and searching for corner-like patterns in scene. To accelerate the algorithm, new corners were searched only in the neighborhood of the corners found in the previous frame.

In the next step the coordinates of PFTs were recalculated into scene coordinate system for every scene frame containing an image (see Figure 6). The last step was aggregation of frames into fixations. A sequence of frames was treated as a fixation when there were at least five subsequent eye images (about 150 ms) with eye center locations closer than 5 points (in eye camera coordinates) which is equivalent to less than 0.5 deg. Such frames were aggregated into one fixation. Finally, the list of fixations as defined in Equation 2 was created for every trial. There were from 119 to 191 fixations detected for 12 participants during observation of 68 images.

The k frames that were recorded during the initial calibration were collected separately with information about the calibration point's position $cp(x, y)$ recalculated into scene coordinates and saved as *calPoints*:

$$calPoints : \{(E_1(e_x, e_y), cp_1(x, y)), ..., (E_k(e_x, e_y), cp_k(x, y))\} \tag{18}$$

Similarly, the l frames recorded during the reference screens when the participant task was to look at a specific reference point $rp(x, y)$ were saved as *refPoints*:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

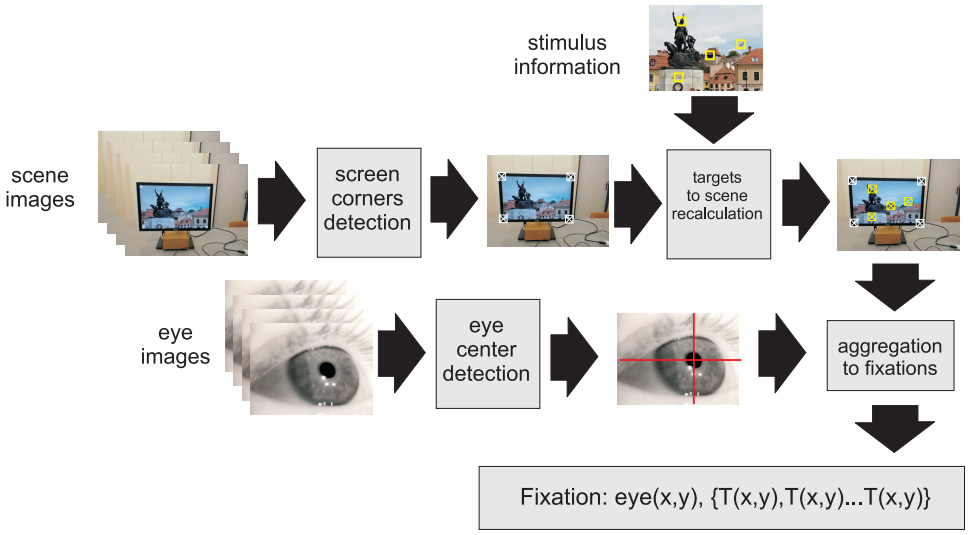

Figure 6. A pipeline of image processing converting scene and eye frames to a list of fixations with eye and targets coordinates

$$refPoints : \{(E_1(e_x, e_y), rp_1(x, y)), ..., (E_l(e_x, e_y), rp_l(x, y))\} \tag{19}$$

The *calPoints* were later used to calculate the reference results of explicit calibration. The *refPoints* were used as the ground truth to calculate errors of different calibration models.

All other frames (e.g. recorded during information screens presentations) were skipped.

To use the gathered data for the implicit calibration it was necessary to define PFTs for each of the 68 images. There were ground truth PFTs found using data taken from genuine users observations (GT), PFTs defined by an expert (EXP) and PFTs calculated by means of nine different saliency models (see Figure 3 as an example).

Incremental and genetic algorithms were used with four different MCFs. To find the best mapping the genetic algorithm was run with 1000 iterations and stop after 200 iterations if no progress was visible. The *cutoff* value for incremental algorithm was set to 100. Additionally, all calculations were performed with various *removalPercent*: 0, 10, 20, 30 and 40.

Every calibration model was used to calculate gaze positions for $refPoints$ on the reference screens. The obtained values $g_i(x, y)$ were then compared with $rp_i(x, y)$ – genuine gaze locations for each $refPoint$ (Equation 19). As the scene camera resolution was 640x480, the difference in scene camera pixels was recalculated to percentage of the scene using the formula:

$$\begin{aligned} Error(x) &= 100 * |rp(x) - g(x)|/640 \\ Error(y) &= 100 * |rp(y) - g(y)|/480 \end{aligned} \tag{20}$$

Errors for all $refPoints$ where then averaged. Reporting error in the percentage of the scene is in our opinion more intuitive than for pixels.

The first step was checking which value of *removalPercent* gives the lowest error for GT saliency. The results are presented in Table 1. The differences were not significant and 20% was chosen for subsequent tests.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

Table 1. Average errors for various *removalPercent* values.

Removal percent	Error X (%)	Error Y (%)	Avg Error (%)
0	8.52	7.17	7.84
10	8.11	7.06	7.59
20	8.06	7.03	7.55
30	8.12	7.08	7.60
40	8.13	7.23	7.68

The next task was checking the results of a classic calibration. The $calPoints$ (Equation 18) were used as input to the LM algorithm calculating calibration function parameters. The errors are presented as CALIB in Table 2.

In order to check the baseline calibration without the PFTs defined we decided to additionally build a calibration model that utilizes 'center location paradigm' stating that most of the time observers look at the center of a screen. So, for each fixation we created an artifical target in the middle of a screen and we built the mapping that takes into account only these targets. The errors of such a calibration are denoted as CENTER in Table 2.

Finally, the calibration models were built using various MCFs and both optimization algorithms. The errors for GT and EXP saliencies are presented in Table 2.

It is important to note that because in this case the image from eye camera is flipped horizontally with relation to scene camera image, it was necessary to use the equation (12) instead of (11) in MCF_{DIR} calculation.

Table 2. Average errors for various MCFs with grount truth (GT) and expert (EXP) saliency models. CALIB stands for classic explicit calibration and CENTER for center target calibration. Algorithm G - genetic optimization, I - incremental optimization.

MCF	Algorithm	Saliency model	Error X (%)	Error Y (%)	Avg Error (%)
CALIB			2.49	2.02	2.26
DIR	G	GT	2.36	2.82	2.59
DIR	I	GT	2.40	3.15	2.77
DIR	I	EXP	2.65	2.97	2.81
DIR	G	EXP	2.71	2.94	2.83
REG	G	EXP	5.76	6.92	6.34
DIST	I	GT	7.72	5.63	6.68
REG	I	EXP	8.09	7.19	7.64
DIST	I	EXP	9.69	5.74	7.71
DC	G	EXP	8.76	7.33	8.05
DIST	G	EXP	9.54	6.94	8.24
DISTC	I	EXP	9.00	7.55	8.28
DISTC	I	GT	8.28	8.52	8.40
DC	G	GT	9.02	8.58	8.80
DIST	G	GT	10.64	8.80	9.72
REG	G	GT	11.53	9.06	10.30
REG	I	GT	12.49	9.72	11.11
CENTER			12.41	10.03	11.22

As the MCFs using DIR occurred to be significantly better, only MCF_{DIR} calculated using genetic algorithm was used in the next step, which involved comparison of different saliency models (Table 3).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

Table 3. Errors for various saliency models when MCF_{DIR} is used.

Saliency model	Error X (%)	Error Y (%)	Avg Error (%)
GT	2.36	2.82	2.59
EXP	2.71	2.94	2.83
RARE2012	3.18	3.36	3.27
SWD	3.48	3.68	3.58
GBVS	3.05	4.21	3.63
BMS	4.03	3.52	3.78
SR-PQFT	4.99	3.64	4.31
IttiKoch	4.12	4.98	4.55
LDS	7.95	7.10	7.52
FES	8.07	7.91	7.99
CovSal	10.63	8.91	9.77

The next step was checking various fusions of MCFs. As MCF_{DIR} appeared to be the best option, all fusions included this type of MCF complemented by one of the other MCFs with various weights from 0.2 to 0.4. The results are presented in Table 4, only for fusions that resulted in average error less than $MCF_{DIR(g)}$. It occurred that only fusions using the genetic algorithm were able to improve the result. Additionally, the fusions of MCF_{DIR} with MCF_{DISTC} occurred to give the best results. It seems that - despite of not good results as a single criterion - the MCF_{DISTC} function is the most valuable addition to MCF_{DIR} criterion when a fusion is taken into account.

Table 4. Average errors for various MCF fusions calculated for GT saliency and *removalPercent*=20%. Only fusions that gave results better than the baseline $MCF_{DIR(g)}$ are listed.

MCF	Algorithm	Error X (%)	Error Y (%)	Avg Error (%)
DIR+0.3*DISTC	G	2.24	2.62	2.43
DIR+0.2*REG	G	2.17	2.69	2.43
DIR+0.2*DISTC	G	2.24	2.69	2.47
DIR+0.4*DISTC	G	2.29	2.66	2.47
DIR+DISTC	G	2.29	2.71	2.50
DIR+0.4*REG	G	2.34	2.71	2.53

3.2. Remote eye tracker

The Eye Tribe eye tracker registering eye movements with frequency of 60Hz was used in the second experiment. As this eye tracker needs to be calibrated to start working, the calibration procedure was initially started with one person who did not took part in the later experiments. Therefore, the signal obtained from the eye tracker could be treated as an uncalibrated signal in the further experiments.

The stimulation consisted of the following steps:

- Initial 9 point calibration
- 6 images observations (3 seconds each)
- Reference screen - user clicks 9 points looking at them
- 6 images observations (3 seconds each)
- Final 9 point calibration

29 participants took part in the experiment. The eye movement signal recorded for each of them was used to check implicit calibration algorithm. The initial and final calibrations were done in order to enable comparison of our method to the classic calibration. All errors were calculated as the difference between the calibrated gazes locations and locations calculated using data from the reference screen. Due to different screen resolutions for each trial, the error is reported as percentage of a screen dimensions independently for X and Y axes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

To use the gathered data for implicit calibration it was necessary to define PFTs for each of the 12 images. As in the previously-described experiment the PFTs were found by means of data taken from: genuine users observations (GT), PFTs defined by an expert (EXP) and PFTs calculated using nine different saliency models. Incremental and genetic algorithms were used with four different MCFs and all calculations were performed with various *removalPercent*: 0, 10, 20, 30 and 40.

At first we checked which value of *removalPercent* gives the best results when taking into account targets defined by genuine gazes (GT).

Table 5. Average errors for various *removalPercent* values for GT saliency.

Removal percent	Error X (%)	Error Y (%)	Avg Error (%)
0	14.81	12.51	13.66
10	14.16	11.80	12.98
20	13.97	11.35	12.66
30	14.05	11.43	12.74
40	14.42	11.48	12.95

The best results were achieved for *removalPercent* equal to 20% so all subsequent results are reported taking into account only tests with such value.

The baseline for evaluating the implicit calibration results were errors for two classic calibrations performed at the beginning and end of the trial. The error values for the classic calibrations are presented as CALIB1 and CALIB2 in Table 6.

Similarly to the previous experiment, center location paradigm was used as another baseline to evaluate implicit calibration errors. The mapping that takes into account only targets in the middle of the screen was denoted as CENTER in Table 6.

Table 6 presents error values of different mapping comparison functions (MCFs) in relation to the baselines descibed above. The errors were calculated for GT and EXP defined PFTs.

Table 6. Average errors for GT PFTs and various mapping comparison functions (MCF).

MCF type	Algorithm	Saliency	Error X (%)	Error Y (%)	Avg Error (%)
CAL2			3.04	4.21	3.63
CAL1			3.04	4.59	3.82
DIR	G	EXP	3.44	5.37	4.41
DIR	I	EXP	3.44	5.77	4.60
DIR	G	GT	4.85	6.50	5.68
DIR	I	GT	5.35	6.41	5.88
REG	G	EXP	8.44	7.70	8.07
REG	G	GT	10.15	11.41	10.78
REG	I	EXP	12.65	9.51	11.08
DIST	I	GT	13.64	12.16	12.90
DIST	I	EXP	16.42	10.38	13.40
DISTC	G	EXP	16.91	13.23	15.07
DIST	G	GT	19.88	11.53	15.70
DISTC	I	EXP	17.94	14.08	16.01
DISTC	G	GT	19.48	12.69	16.08
DISTC	I	GT	18.28	14.87	16.58
REG	I	GT	20.33	15.33	17.83
DIST	G	EXP	22.27	15.35	18.81
CENTER			21.37	18.41	19.89

It occurred that MCFs utilizing DIR algorithm are significantly better than other methods. When we chose the best MCF (DIR G) and 20% removal percent it was possible to compare different saliency maps (Table 7).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

Table 7. Average errors for various saliency models taking MCF_{DIR} and *removalPercent* = 20.

MCF type	Error X (%)	Error Y (%)	Average error
EXP	3.44	5.37	4.41
GT	4.85	6.50	5.68
SR-PQFT	5.89	7.78	6.84
BMS	4.87	9.53	7.20
GBVS	5.94	10.19	8.07
RARE2012	4.77	12.29	8.53
IttiKoch	9.18	8.21	8.70
SWD	8.90	8.51	8.71
LDS	12.91	11.46	12.18
CovSal	19.18	18.37	18.78
FES	20.83	17.20	19.02

Similarly to the previous experiment with head-mounted eye tracker, the next step was to search for the best fusions of different criteria. As MCF_{DIR} appeared to be the best option, all fusions used this type of MCF and additionally added one of the other MCFs with various weights from 0.2 to 0.4. The best results are presented in Table 8.

Table 8. Average errors for various fusions of MCFs, saliency EXP and *removalPercent* = 20. Only the best achieved fusions are presented.

MCF type	Algorithm	Number of trials	Error X (%)	Error Y (%)	Average error
DIR+0.2*DIST	G	29	3.00	4.83	3.91
DIR+0.2*REG	G	29	2.97	4.86	3.91
DIR+0.4*REG	G	29	2.93	5.03	3.98
DIR+0.2*DISTC	G	29	3.07	5.00	4.03
DIR+0.3*DISTC	G	29	2.93	5.14	4.03
DIR+0.4*DISTC	G	29	2.79	5.28	4.03
DIR+0.3*REG	G	29	2.93	5.14	4.03
DIR+0.4*DIST	G	29	3.00	5.14	4.07

3.3. Time analysis

The results for implicit calibration algorithms presented in the previous section were calculated offline, after collecting data. Therefore, the next research question was if it is possible to use it online, during the eye movement registration. To check it average times of calibration models calculation for different MCFs and optimization algorithms were evaluated (Table 9).

Table 9. Average times of execution for different MCFs for remote eye tracker.

MCF type	Incremental time [s]	Genetic time [s]
DISTC	0.10	2.32
DIR	3.78	9.52
REG	26.86	62.24
DIST	127.64	492.41

It is visible that only MCF_{DISTC} and MCF_{DIR} algorithms offer efficiency suitable for the online calibration.

The next question was how many fixations are necessary to obtain reliable results. The MCF_{DIR} and incremental algorithm were used to assess how the error changes when the number of fixations increases. The results are presented in Figure 7 for the remote eye tracker and in Figure 8 for the head mounted eye tracker.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

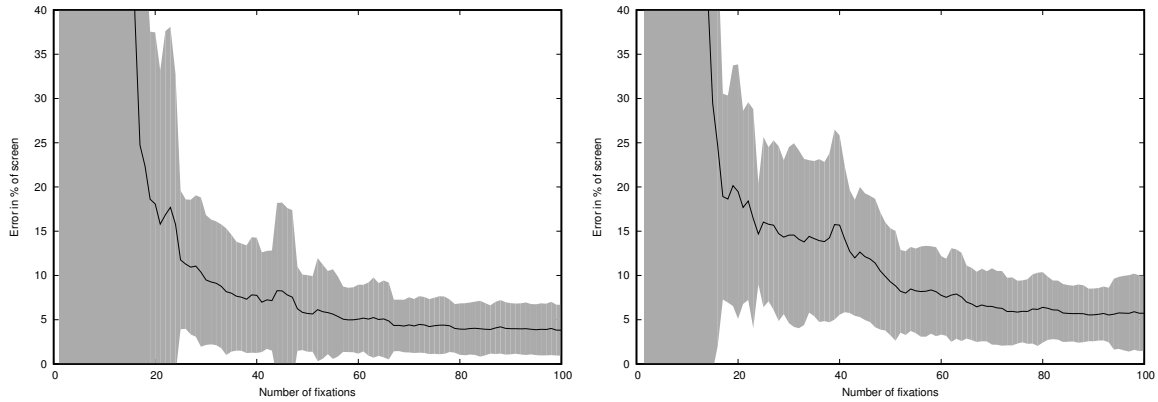

Figure 7. Error for remote eye tracker with MCF_{DIR} and incremental algorithm depending on the number of fixations. Left: horizontal error, right: vertical error. Gray area represents standard deviation.

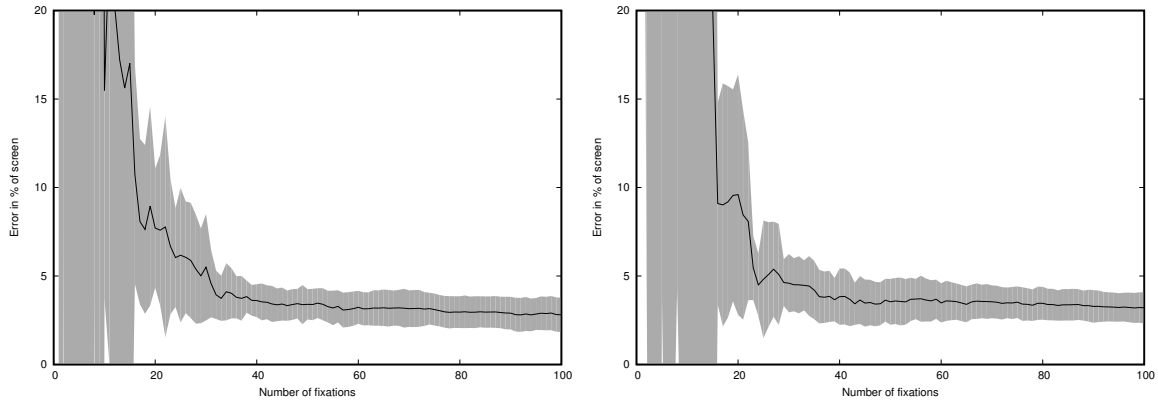

Figure 8. Error for headset eye tracker and MCF_{DIR} incremental algorithm depending on the number of fixations. Left: horizontal error, right: vertical error. Gray area represents standard deviation.

It is visible that after analysis of about 50 fixation (10-20 seconds depending on the trial) the error stabilizes.

4. Discussion

The paper presents an implicit calibration algorithm that uses the idea of Probable Fixation Targets. Using the targets instead of the whole saliency maps as it was typically done in previous research [11,16,18] has several important advantages:

- Data processing is easier and faster, what which makes a real time calibration feasible.
- It is more universal as it may be used with any stimulus: static images, movies but also with computer interfaces (e.g. with buttons or labels as targets) or games (e.g. with avatars as targets what was proposed in [2]).
- Creating a short list of potential targets should be faster than creation of a full saliency map (but it was not tested during this research).
- Definition of targets may be easily done manually, by a human expert (which was checked experimentally) while the creation of a saliency map requires special algorithms.

The results of two experiments described in the previous section show that the implicit calibration using PFTs may be used instead of a classic calibration. Error levels are higher than for the classic calibration, however these levels are comparable for some mapping comparison functions MCFs.

Unsurprisingly, the best results were achieved for ground truth and expert based targets. However, for some fully automatic saliency models, the results were not much worse than the ground truth,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

which gives clue that, after some improvements, the process of implicit calibration may be independent of the type of a scene.

Of course the quality of the PFT based calibration, as for every implicit calibration, depends heavily on the type of a stimulus being presented. If the probable targets of stimuli are always near the center of a screen (which is e.g. typical for movies), quality of calibration model near the edges of the scene will decrease. This is the main limitation of the presented solution. However, the same problem applies to every implicit calibration algorithm and if all probable targets are indeed gathered near the center of the screen, then the calibration model should also focus on the center and its accuracy near the edges may be not so important.

Another limitation is that errors obtained for implicit calibration using saliency models are so far significantly higher than for the classic calibration. However, there is still a lot of room for improvement. As for the future work we plan to investigate how to improve the PFTs generation algorithm that will find more robust PFTs by extracting it from saliency maps. We also plan to test new fusions of already introduced MFCs and search for new MFCs. Last but not least, we would like to test the method using new datasets with different stimuli such as movies or classic user interfaces.

Supplementary Materials: The code of ETCAL library used for experiments and both datasets are available online at www.kasprowski.pl/etcal.

Acknowledgments: Publication supported by Rector Habilitation Grant, Silesian University of Technology, Grant Number: 02/010/RGH18/0131.

Author Contributions: P.K. designed and implemented the algorithm, K.H. took part in the experiments and paper edition, P.S. implemented and executed the code generating saliency maps. All authors took part in paper preparation and edition.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

PFT: Probable Fixation Target
MCF: Mapping Comparison Function

1. Hornof, A.J.; Halverson, T. Cleaning up systematic error in eye-tracking data by using required fixation locations. *Behavior Research Methods, Instruments, & Computers* **2002**, *34*, 592–604.
2. Kasprowski, P.; Harezlak, K. Implicit calibration using predicted gaze targets. Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications. ACM, 2016, pp. 245–248.
3. Sugano, Y.; Matsushita, Y.; Sato, Y. Calibration-free gaze sensing using saliency maps. Computer vision and pattern recognition (cvpr), 2010 ieee conference on. IEEE, 2010, pp. 2667–2674.
4. Wang, K.; Wang, S.; Ji, Q. Deep eye fixation map learning for calibration-free eye gaze tracking. Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications. ACM, 2016, pp. 47–55.
5. Sun, L.; Song, M.; Liu, Z.; Sun, M.T. Real-Time Gaze Estimation with Online Calibration. *IEEE MultiMedia* **2014**, *21*, 28–37.
6. Kasprowski, P.; Harezlak, K.; Stasch, M. Guidelines for the eye tracker calibration using points of regard. *Information Technologies in Biomedicine* **2014**, *4*, 225–236.
7. Vadillo, M.A.; Street, C.N.; Beesley, T.; Shanks, D.R. A simple algorithm for the offline recalibration of eye-tracking data through best-fitting linear transformation. *Behavior research methods* **2015**, *47*, 1365–1376.
8. Kasprowski, P.; Harezlak, K. Study on participant-controlled eye tracker calibration procedure. Proceedings of the 7th Workshop on Eye Gaze in Intelligent Human Machine Interaction: Eye-Gaze & Multimodality. ACM, 2014, pp. 39–41.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

9. Pfeuffer, K.; Vidal, M.; Turner, J.; Bulling, A.; Gellersen, H. Pursuit calibration: Making gaze calibration less tedious and more flexible. Proceedings of the 26th annual ACM symposium on User interface software and technology. ACM, 2013, pp. 261–270.
10. Hirvonen, T.; Aalto, H.; Juhola, M.; Pyykkö, I. A comparison of static and dynamic calibration techniques for the vestibulo-ocular reflex signal. *International journal of clinical monitoring and computing* **1995**, *12*, 97–102.
11. Chen, J.; Ji, Q. A probabilistic approach to online eye gaze tracking without explicit personal calibration. *IEEE Transactions on Image Processing* **2015**, *24*, 1076–1086.
12. Perra, D.; Gupta, R.K.; Frahm, J.M. Adaptive eye-camera calibration for head-worn devices. CVPR, 2015, pp. 4146–4155.
13. Zhang, Y.; Hornof, A.J. Easy post-hoc spatial recalibration of eye tracking data. Proceedings of the symposium on eye tracking research and applications. ACM, 2014, pp. 95–98.
14. Sugano, Y.; Matsushita, Y.; Sato, Y.; Koike, H. An incremental learning method for unconstrained gaze estimation. European Conference on Computer Vision. Springer, 2008, pp. 656–667.
15. Itti, L.; Koch, C.; Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1998**, *20*, 1254–1259.
16. Sugano, Y.; Matsushita, Y.; Sato, Y. Appearance-based gaze estimation using visual saliency. *IEEE transactions on pattern analysis and machine intelligence* **2013**, *35*, 329–341.
17. Harel, J.; Koch, C.; Pietro Perona. Graph-Based Visual Saliency. Proceedings of Neural Information Processing Systems (NIPS), 2006.
18. Alnajar, F.; Gevers, T.; Valenti, R.; Ghebreab, S. Auto-Calibrated Gaze Estimation Using Human Gaze Patterns. *International Journal of Computer Vision* **2017**, pp. 1–14.
19. Nguyen, P.; Fleureau, J.; Chamaret, C.; Guillotel, P. Calibration-free gaze tracking using particle filter. Multimedia and Expo (ICME), 2013 IEEE International Conference on. IEEE, 2013, pp. 1–6.
20. Wang, K.; Ji, Q. 3D gaze estimation without explicit personal calibration. *Pattern Recognition* **2018**, *79*, 216–227.
21. Kasprowski, P. Mining of eye movement data to discover people intentions. International Conference: Beyond Databases, Architectures and Structures. Springer, 2014, pp. 355–363.
22. Sugano, Y.; Bulling, A. Self-calibrating head-mounted eye trackers using egocentric visual saliency. Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology. ACM, 2015, pp. 363–372.
23. Kasprowski, P.; Harezlak, K. Implicit calibration using predicted gaze targets. Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications. ACM, 2016, pp. 245–248.
24. Kasprowski, P.; Harezlak, K. Comparison of mapping algorithms for implicit calibration using probable fixation targets. Proceedings of the 10th Biennial ACM Symposium on Eye Tracking Research & Applications. ACM, 2018.
25. Kasprowski, P.; Harezlak, K. ETCAL–a versatile and extendable library for eye tracker calibration. *Digital Signal Processing* **2017**.
26. Judd, T.; Ehinger, K.; Durand, F.; Torralba, A. Learning to predict where humans look. 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 2106–2113.
27. Koch, C.; Ullman, S. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology* **1985**, *4*, 219–227.
28. Harezlak, K.; Kasprowski, P. Chaotic Nature of Eye Movement Signal. International Conference on Intelligent Decision Technologies. Springer, 2017, pp. 120–129.
29. Judd, T.; Durand, F.; Torralba, A. A benchmark of computational models of saliency to predict human fixations **2012**.
30. Le Meur, O.; Liu, Z. Saccadic model of eye movements for free-viewing condition. *Vision Research* **2015**, *116, Part B*, 152–164.
31. Zhang, J.; Sclaroff, S. Exploiting Surroundedness for Saliency Detection: A Boolean Map Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2016**, *38*.
32. Erdem, E.; Erdem, A. Visual saliency estimation by nonlinearly integrating features using region covariances. *Journal of Vision* **2013**, *13*, 11–11.
33. Tavakoli, H.R.; Rahtu, E.; Heikkila, J. Fast and Efficient Saliency Detection Using Sparse Sampling and Kernel Density Estimation. In *Image Analysis*; Springer Berlin Heidelberg, 2011; pp. 666–675.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 November 2018 doi:10.20944/preprints201811.0253.v1

Peer-reviewed version available at Sensors 2019, 19, 216; doi:10.3390/s19010216

34. Fang, S.; Li, J.; Tian, Y.; Huang, T.; Chen, X. Learning Discriminative Subspaces on Random Contrasts for Image Saliency Analysis. *IEEE Transactions on Neural Networks and Learning Systems* **2017**, *28*, 1095–1108.
35. Riche, N.; Mancas, M.; Duvinage, M.; Mibulumukini, M.; Gosselin, B.; Dutoit, T. RARE2012: A multi-scale rarity-based saliency detection with its comparative statistical analysis. *Signal Processing: Image Communication* **2013**, *28*, 642–658.
36. Hou, X.; Zhang, L. Saliency Detection: A Spectral Residual Approach. 2007 IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.
37. Schauerte, B.; Stiefelhagen, R. Quaternion-Based Spectral Saliency Detection for Eye Fixation Prediction. In *Computer Vision - ECCV 2012*; Springer Berlin Heidelberg, 2012; pp. 116–129.
38. Duan, L.; Wu, C.; Miao, J.; Qing, L.; Fu, Y. Visual saliency detection by spatially weighted dissimilarity. CVPR 2011, 2011, pp. 473–480.
39. Meyer, F. Color image segmentation. International Conference on Image Processing and its Applications, 1992. IET, 1992, pp. 303–306.
40. Moré, J.J. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*; Springer, 1978; pp. 105–116.
41. Fischler, M.A.; Bolles, R.C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*; Elsevier, 1987; pp. 726–740.

Sample Availability: Datasets and code are available from the authors www.kasprowski.pl/etcal.