Efficiency in Real-time Webcam Gaze Tracking

Gudi, Amogh; li, Xin; van Gemert, Jan

DOI
10.1007/978-3-030-66415-2_34

Publication date
2020

Document Version
Accepted author manuscript

Published in
Computer Vision – ECCV 2020 Workshops

Efficiency in Real-time Webcam $\overrightarrow{\text{Gaze}}$ Tracking

Amogh Gudi[1,2], Xin Li[1,2], and Jan van Gemert[2]

[1] Vicarious Perception Technologies [VicarVision], Amsterdam, The Netherlands
[2] Delft University of Technology [TU Delft], Delft, The Netherlands
{amogh,xin}@vicarvision.nl, j.c.vangemert@tudelft.nl

Abstract. Efficiency and ease of use are essential for practical applications of camera based eye/gaze-tracking. Gaze tracking involves estimating where a person is looking on a screen based on face images from a computer-facing camera. In this paper we investigate two complementary forms of efficiency in gaze tracking: 1. The computational efficiency of the system which is dominated by the inference speed of a CNN predicting gaze-vectors; 2. The usability efficiency which is determined by the tediousness of the mandatory calibration of the gaze-vector to a computer screen. To do so, we evaluate the computational speed/accuracy trade-off for the CNN and the calibration effort/accuracy trade-off for screen calibration. For the CNN, we evaluate the full face, two-eyes, and single eye input. For screen calibration, we measure the number of calibration points needed and evaluate three types of calibration: 1. pure geometry, 2. pure machine learning, and 3. hybrid geometric regression. Results suggest that a single eye input and geometric regression calibration achieve the best trade-off.

1 Introduction

In a typical computer-facing scenario, the task of gaze-tracking involves estimating where a subject's gaze is pointing based on images of the subject captured via the webcam. This is commonly in the form of a gaze vector, which determines the pitch and yaw of the gaze with respect to the camera [30]. A more complete form of gaze tracking further extends this by also computing at which specific point the subject is looking at on a screen in front of the subject [11,28]. This is achieved by estimating the position of the said screen w.r.t. the camera (a.k.a. screen calibration), which is not precisely known beforehand. We present a study of some core choices in the design of gaze estimation methods in combination with screen calibration techniques (see Figure 1), leaning towards an efficient real-time camera-to-screen gaze-tracking system.

Computational efficiency: Input size. Deep networks, and CNNs in particular, improved accuracy in gaze estimation where CNN inference speed is to a large extent determined by the input image size. The input size for gaze estimation can vary beyond just the image of the eye(s) [16,30], but also include the whole eye region [28], the whole face, and even the full camera image [11].

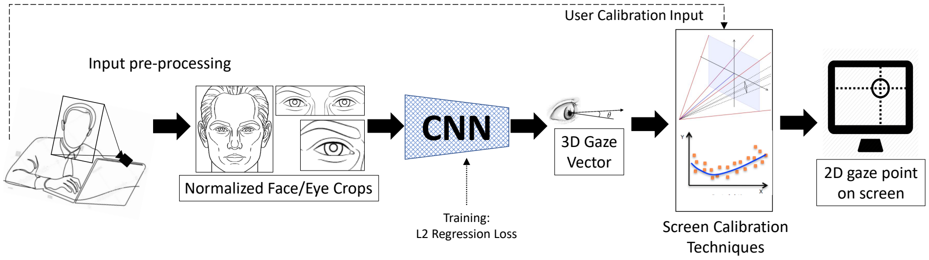


Fig. 1. An overview illustration of our camera-to-screen gaze tracking pipeline under study. (Left to right) Images captured by the webcam are first pre-processed to create a normalized image of face and eyes. These images are used for training a convolutional neural network (using L2 loss) to predict the 3D gaze vector. With the cooperation of the user, the predicted gaze vectors can finally be projected on to the screen where he/she is looking using proposed screen calibration techniques.

Yet, the larger the input image, the slower the inference speed. We study the impact of various input types and sizes with varying amounts of facial contextual information to determine their speed/accuracy trade-off.

Usability efficiency: Manual effort in screen calibration. Most work focus on gaze vectors estimation [2,5,13,15,27]. However, predicting the *gaze-point*, point on a screen in front of the subject where he/she is looking, is a more intuitive and directly useful result for gaze tracking based applications, especially in a computer-facing/human-computer interaction scenario. If the relative locations and pose of the camera w.r.t to the screen were exactly known, projecting the gaze-vector to a point on screen would be straightforward. However, this transformation is typically not known in real-world scenarios, and hence must also be implicitly or explicitly estimated through an additional calibration step. This calibration step needs to be repeated for every setup. Unlike gaze-vector prediction, you cannot have a "pre-trained" screen calibration method. In practice, every-time a new eye-tracking session starts, the first step would be to ask the user to look at and annotate predefined points for calibration. Therefore, obtaining calibration data is a major usability bottleneck since it requires cooperation of the user every time, which in practice varies. Here, we study usability efficiency as a trade-off between the number of calibration points and accuracy.

We consider three types of calibration. Geometry based modelling methods have the advantage that maximum expert/geometrical prior knowledge can be embedded into the system. On the other hand, such mathematical models are rigid and based on strong assumptions, which may not always hold. In contrast, calibration methods based on machine learning require no prior domain knowledge and limited hand-crafted modelling. However, they may be more data-dependent in order to learn the underlying geometry. In this paper, we evaluate the efficiency trade-off of various calibration techniques including a hybrid approach between machine learning regression and geometric modelling.

Contributions. We have the following three contributions:

(i) We evaluate computational system efficiency by studying the balance of gains from context-rich inputs vs their drawbacks. We study their individual impact on the system's accuracy w.r.t. their computational load to determine their efficiency and help practitioners find the right trade-off.

(ii) We demonstrate three practical screen calibration techniques that can be use to convert the predicted gaze-vectors to points-on-screen, thereby performing the task of complete camera-to-screen gaze tracking.

(iii) We evaluate the usability efficiency of these calibration methods to determine how well they utilize expensive user-assisted calibration data. This topic has received little attention in literature, and we present one of the first reports on explicit webcam-based screen calibration methods.

2 Related Work

Existing methods for gaze tracking can be roughly categorized into model-based and appearance-based methods. The former [25,26] generates a geometric model for eye to predict gaze, while the latter [23] makes a direct mapping between the raw pixels and the gaze angles. Appearance driven methods have generally surpassed classical model-based methods for gaze estimation.

Appearance-based CNN gaze-tracking. As deep learning methods have shown their potentials in many areas, some appearance-based CNN networks are shown to work effectively for the task of gaze prediction.

Zhang *et al.* [29,30] proposed the first deep learning model for appearance-based gaze prediction. Park *et al.* [16] proposed a combined hourglass [12] and DenseNet [7] network to take advantage of auxiliary supervision based on the gaze-map, which is two 2D projection binary mask of the iris and eyeball. Cheng *et al.* [3] introduced ARE-Net, which is divided into two smaller modules: one is to find directions from each eye individually, and the other is to estimate the reliability of each eye. Deng and Zhu *et al.* [4] define two CNNs to generate head and gaze angles respectively, which are aggregated by a geometrically constrained transform layer. Ranjan *et al.* [17] clustered the head pose into different groups and used a branching structure for different groups. Chen *et al.* [2] proposed Dilated-Nets to extract high level features by adding dilated convolution. We build upon these foundations where we evaluate the speed vs accuracy trade-off in a real-time setting. The image input size has a huge effect on processing speed, and we control the input image size by varying eye/face context.

The seminal work of Zhang *et al.* [29,30] utilized minimal context by only using the grayscale eye image and head pose as input. Krafka *et al.* [11] presented a more context-dependent multi-model CNN to extract information from two single eye images, face image and face grid (a binary mask of the face area in an image). To investigate how the different face region contributes to the gaze prediction, a full-face appearance-based CNN with spatial weights was proposed [28]. Here, we investigate the contribution of context in the real-time setting by explicitly focusing on the speed/accuracy trade-off.

A GPU based real-time gaze tracking method was presented in [5]. This was implemented in a model ensemble fashion, taking two eye patches and head pose vector as input, and achieved good performance on several datasets [5, 22, 30] for person-independent gaze estimation. In addition, [2, 13] have included some results about the improvements that can be obtained from different inputs. In our work, we perform an ablation study and add the dimension of computation load of each input type. Our insight in the cost vs benefit trade-off may help design efficient gaze tracking software that can run real-time beyond expensive GPUs, on regular CPUs which have wider potential in real world applications.

Screen calibration: Estimating point-of-gaze. In a classical geometry-based model, projecting any gaze-vector to a point on a screen requires a fully-calibrated system. This includes knowing the screen position and pose in the camera coordinate system. Using a mirror-based calibration technique [18], the corresponding position of camera and screen can be attained. This method needs to be re-applied for different computer and camera setting, which is non-trivial and time-consuming. During human-computer interactions, information like mouse clicks may also provide useful information for screen calibration [14]. This is, however, strongly based on the assumption that people are always looking at the mouse cursor during the click.

Several machine learning models are free of rigid geometric modelling while showing good performance. Methods like second order polynomial regression [9] and Gaussian process regression [24] have been applied to predict gaze more universally. WebGazer [14] trains regression models to map pupil positions and eye features to 2D screen locations directly without any explicit 3D geometry. As deep learning features have shown robustness in different areas, other inputs can be mixed with CNN-based features for implicit calibration, as done in [11, 28]. CNN features from the eyes and face are used as inputs to a support vector regressor to directly estimate gaze-point coordinates without an explicit calibration step. These methods take advantage of being free of rigid modelling and show good performance. On the other hand, training directly on CNN features makes this calibration technique non-modular since it is designed specific to a particular gaze-prediction CNN. In our work, we evaluate data-efficiency for modular screen calibration techniques that convert gaze-vectors to gaze-points based on geometric modelling, machine learning, and a mix of geometry and regression. We explicitly focus on real world efficiency which for calibration is not determined by processing speed, but measured in how many annotations are required to obtain reasonable accuracy.

3 Setup

The pipeline contains three parts, as illustrated in Figure 1:

1. Input pre-processing by finding and normalizing the facial images;
2. A CNN that takes these facial images as input to predict the gaze vector;
3. Screen calibration and converting gaze-vectors to points on the screen.

3.1 Input Pre-processing

The input to the system is obtained from facial images of subjects. Through a face finding and facial landmark detection algorithm [1], the face and its key parts are localized in the image. Following the procedure described by Sugano *et al.* [22], the detected 2D landmarks are fitted onto a 3D model of the face. This way, the facial landmarks are roughly localized in the 3D camera coordinate space. By comparing the 3D face model and 2D landmarks, the head rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$, and the 3D eye locations $\mathbf{e}$ are obtained in 3D camera coordinate space. A standardized view of the face is now obtained by defining a fixed distance d between the eye centres and the camera centre and using a scale matrix $\mathbf{S} = \text{diag}(1, 1, \frac{d}{||\mathbf{e}||})$. The obtained conversion matrix $\mathbf{M} = \mathbf{S} \cdot \mathbf{R}$ is used to apply perspective warping to obtain a normalized image without roll (in-plane rotation). For training, the corresponding ground truth vector $\mathbf{g}$ is similarly transformed: $\mathbf{M} \cdot \mathbf{g}$.

3.2 CNN Prediction of Gaze Vectors

We use a VGG16 [20] network architecture with BatchNorm [8] to predict the pitch and yaw angles of the gaze vector with respect to the camera from the normalized pre-processed images.

Training. Following the prior work in [30], the network was pre-trained on ImageNet [19]. For all the experiments conducted in this work, we set the following hyperparameters for the training of the network for gaze-vector prediction:

(i) Adam optimizer with default settings [10];
(ii) a validation error based stopping criteria with a patience of 5 epochs;
(iii) learning rate of 10^{-5}, decaying by 0.1 if validation error plateaus;
(iv) simple data augmentation with mirroring and gaussian noise ($\sigma = 0.01$).

Inference. This trained deep neural network can now make prediction of the gaze vector. The predicted gaze-vector (in the form of pitch and yaw angles) are with respect to the 'virtual' camera corresponding to the normalized images. The predicted virtual gaze vectors can be transformed back to the actual gaze vector with respect to the real camera using the transformation parameters obtained during image pre-processing. These vectors can then be projected onto a point on the screen after screen calibration.

3.3 Screen Calibration: Gaze Vectors to Gaze Points

To project the predicted 3D gaze vectors (in the camera coordinate space) to 2D gaze-points on a screen, the position of the screen with respect to the camera must be known which is difficult to obtain in real world settings. The aim of screen calibration is to estimate this geometric relation between the camera and the screen coordinate systems such that the predicted gaze vectors in camera coordinates are calibrated to gaze-points in screen coordinates. Because we focus on the task of eye-tracking in a computer-facing scenario, we can simplify the setup by making some assumptions based on typical webcam-monitor placement (such as for built-in laptop webcams or external webcams mounted on monitors):

(i) the roll and yaw angles between the camera and the screen are 0°,
(ii) the intrinsic camera matrix parameters are known, and
(iii) the 3D location of the eye is roughly known w.r.t the camera (estimated by the eye landmarks in the face modelling step in camera coordinate space).

With these assumptions in place, we can design user-aided calibration techniques where the user cooperates by looking at predefined positions on the screen.

4 Screen Calibration Methods

As calibration is tedious and needs to be performed multiple times, we evaluate efficiency in terms of how much manual effort is required for three calibration versions:

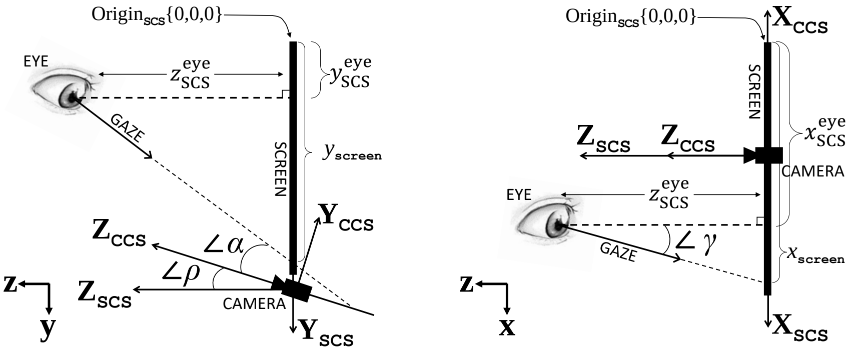
1. calibration by geometry;
2. calibration by machine learning;
3. calibration by a hybrid: geometry and regression.

4.1 Geometry-based Calibration

To perfectly project a gaze-vector w.r.t the camera to a point on a screen, we are essentially required to determine the transformation parameters between the camera coordinate system (CCS) and the screen coordinate system (SCS). With our assumptions about roll and yaw in place, this transformation can be expressed by the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{T}$ between the camera and the screen:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\rho) & -\sin(\rho) \\ 0 & \sin(\rho) & \cos(\rho) \end{bmatrix} \quad \& \quad \mathbf{T} = \begin{bmatrix} \Delta x \; \Delta y \; \Delta z \end{bmatrix}^T , \tag{1}$$

where ρ denotes the vertical pitch angle (about the x-axis; along the y-axis) between the camera and the screen norm, and $\Delta x, \Delta y, \Delta z$ represent the translational displacement between the camera and the screen. An illustration of the geometric setup is shown in Figure 2 (2a and 2b).



(a) $y - z$ plane with pitch angles (b) $x - z$ plane with yaw angle

Fig. 2. A illustration of the geometric setup between the eye and the screen in the screen coordinate space. $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}_{\mathsf{CCS}}$ and $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}_{\mathsf{SCS}}$ represent the directions of the $\{x, y, z\}$-axes of the camera and screen coordinate systems respectively; Origin$_{\mathsf{SCS}}$ represents the origin of the screen coordinate system.

Step 1. The first step is to estimate the location of eye in the screen coordinate system. This can be done with the aid of the user, who is asked to sit at a preset distance z from the screen and look perpendicular at the screen plane (such that the angle between the gaze vector and the screen plane becomes 90°). He is then instructed to mark the point of gaze on the screen, denoted by $\{x, y\}$. In this situation, these marked screen coordinates would directly correspond to the x and y coordinates of the eye in the screen coordinate space. Thus, the eye location can be determined as: $\mathbf{e}_{\mathsf{SCS}} = \{x_{\mathsf{SCS}}^{\mathsf{eye}}, y_{\mathsf{SCS}}^{\mathsf{eye}}, z_{\mathsf{SCS}}^{\mathsf{eye}}\} = \{x, y, z\}$.

During this time, the rough 3D location of the eye in the camera coordinate system is also obtained (from the eye landmarks of the face modelling step) and represented as $\mathbf{e}_{\mathsf{CCS}}$. With this pair of corresponding eye locations obtained, the translation vector T can be expressed by:

$$\mathbf{e}_{\mathsf{SCS}} = \mathbf{R} \cdot \mathbf{e}_{\mathsf{CCS}} + \mathbf{T} \quad \Longrightarrow \quad \mathbf{T} = \mathbf{e}_{\mathsf{SCS}} - \mathbf{R} \cdot \mathbf{e}_{\mathsf{CCS}}. \tag{2}$$

Step 2. Next, without (significantly) changing the head/eye position, the user is asked to look at a different pre-determined point on the screen $\{x_{\mathtt{screen}}, y_{\mathtt{screen}}\}$.

During this time, the gaze estimation system is used to obtain the gaze direction vector in the camera coordinate system:

$$\mathbf{g}_{\mathsf{CCS}} = \left[x_{\mathsf{CCS}}^{\mathsf{gaze}} \;\; y_{\mathsf{CCS}}^{\mathsf{gaze}} \;\; z_{\mathsf{CCS}}^{\mathsf{gaze}}\right]^T. \tag{3}$$

This is a normalized direction vector whose values denote a point on a unit sphere. Both the pitch α (about the x-axis) and the yaw γ (about the y-axis) angles of the gaze w.r.t the camera can be re-obtained from this gaze direction vector:

$$\alpha = \arctan 2(-y_{\mathsf{CCS}}^{\mathsf{gaze}}, z_{\mathsf{CCS}}^{\mathsf{gaze}}) \quad \& \quad \gamma = \arctan 2(x_{\mathsf{CCS}}^{\mathsf{gaze}}, z_{\mathsf{CCS}}^{\mathsf{gaze}}). \tag{4}$$

Once α is determined, we can calculate the camera pitch angle ρ between the camera and the screen:

$$\rho = \arctan(\frac{-y^{\texttt{eye}}_{\texttt{SCS}} + y_{\texttt{screen}}}{z^{\texttt{eye}}_{\texttt{SCS}}}) - \alpha. \tag{5}$$

Using this in Equation 1, the rotation matrix $\mathbf{R}$ can be fully determined. This known $\mathbf{R}$ can now be plugged into Equation 2 to also determine the translation vector $\mathbf{T}$. This procedure can be repeated for multiple calibration points in order to obtain a more robust aggregate estimate of the transformation parameters.

Step 3. Once calibration is complete, any new eye location $\hat{\mathbf{e}}_{\texttt{CCS}}$ can be converted to the screen coordinate space:

$$\hat{\mathbf{e}}_{\texttt{SCS}} = [\hat{x}^{\texttt{eye}}_{\texttt{SCS}}, \hat{y}^{\texttt{eye}}_{\texttt{SCS}}, \hat{z}^{\texttt{eye}}_{\texttt{SCS}}] = \mathbf{R} \cdot \hat{\mathbf{e}}_{\texttt{CCS}} + \mathbf{T}. \tag{6}$$

Using the associated new gaze angles $\hat{\alpha}$ and $\hat{\gamma}$, the point of gaze on the screen can be obtained:

$$\hat{x}_{\texttt{screen}} = \hat{z}^{\texttt{eye}}_{\texttt{SCS}} \cdot tan(\hat{\gamma}) + \hat{x}^{\texttt{eye}}_{\texttt{SCS}} \quad \& \quad \hat{y}_{\texttt{screen}} = \hat{z}^{\texttt{eye}}_{\texttt{SCS}} \cdot tan(\hat{\alpha} + \rho) + \hat{y}^{\texttt{eye}}_{\texttt{SCS}}, \tag{7}$$

4.2 Machine Learning (ML)-based Calibration.

Since the task of gaze vector to gaze point calibration requires learning the mapping between two sets of coordinates, this can be treated as a regression problem. In our implementation, we use a linear ridge regression model for this task for it's ability to avoid overfitting when training samples are scarce. The input to this calibration model includes the predicted gaze-vector angles and the 3D location of the eye, all in the camera coordinate system. The outputs are the 2D coordinates of the gaze-point on the screen in the screen coordinate system.

During calibration, the user is asked to look at a number of predefined points on the screen (such that they span the full region of the screen) while their gaze and eye locations are estimated and recorded for each of these points. These calibration samples are then used to train the model. Given enough training/calibration points, this model is expected to implicitly learn the mapping between the two coordinate systems.

4.3 'Hybrid' Geometric Regression Calibration.

To combine the benefits of geometry based prior knowledge with ML based regression, a hybrid geometric regression technique can be derived where machine learning is used to infer the required geometric transformation parameters.

As before, we assume the roll and yaw angles between the camera and the screen are 0°. The only unknown between the pose of the camera w.r.t the screen is the pitch angle ρ. The rotation and translation matrices are the same as given by Equation 1, and the formulations of gaze pitch and yaw angles α and γ stay the same as defined by Equation 4.

Again, during calibration the user is asked to look at a number of varied predefined points on the screen while their gaze directions and eye locations are recorded. These data samples are then used to jointly minimize the reprojection errors (squared Euclidean distance) to learn the required transformation parameters $\rho, \mathit{\Delta} x, \mathit{\Delta} y, \mathit{\Delta} z$:

$$\underset{\rho,\mathit{\Delta}x,\mathit{\Delta}y,\mathit{\Delta}z}{\arg\min} \sum_{i=1}^{N} \left((x^i_{\texttt{point}} - \hat{x}^i_{\texttt{screen}})^2 + (y^i_{\texttt{point}} - \hat{y}^i_{\texttt{screen}})^2 \right) , \tag{8}$$

where N is the number of training/calibration points; $\{x_{\texttt{point}}, y_{\texttt{point}}\}$ denote the ground truth screen points, while $\{\hat{x}_{\texttt{screen}}, \hat{y}_{\texttt{screen}}\}$ are the predicted gaze points on screen as estimated using Equation 7.
We solve this minimization problem by differential evolution [21].

5 Experiments and Results

5.1 Datasets

We perform all our experiments on two publicly available gaze-tracking datasets:

MPIIFaceGaze [28]. This dataset is an extended version of MPIIGaze [30] with available human face region. It contains 37,667 images from 15 different participants. The images have variations in illumination, personal appearance, head pose and camera-screen settings. The ground truth gaze target on the screen is given as a 3D point in the camera coordinate system.

EYEDIAP [6]. This dataset provides 94 video clips recorded in different environments from 16 participants. It has two kinds of gaze targets: screen points and 3D floating targets. It also has two types of head movement conditions: static and moving head poses. For our experiments, we choose the screen point target with both the static and moving head poses, which contains 218,812 images.

5.2 Exp 1: Speed/Accuracy Trade-off for Varying Input Sizes

Setup. A gaze vector can be predicted based on various input image sizes, as illustrated in Figure 3:

- Full face image: The largest and most informative input;
- Two eyes: Medium sized and informative;
- Single eye: Minimal information and smallest size.

To assess the performance gains of different input sizes vs their computational loads and accuracy, we setup an experiment where we vary the input training and testing data to the neural network while keeping all other settings fixed. We then measure the accuracy of the system and compute their individual inference-time computational loads.

For this experiment, we individually train our deep network on each of the multiple types and sizes of the pre-processed inputs shown in Figure 3. In order to obtain a reliable error metric, we perform 5-fold cross-validation training. This experiment is repeated for both the MPIIFaceGaze and EYEDIAP dataset.

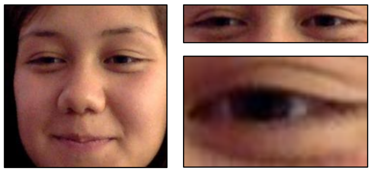
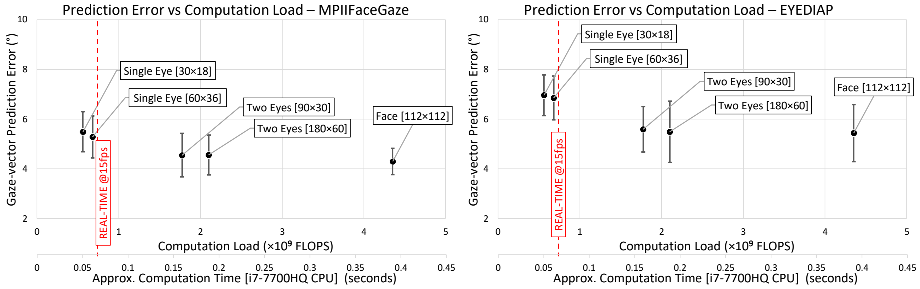


Fig. 3. Examples of three input types used in the experiments: (left) face crop, sized 224×224 and 112×112; (right top) two eyes region crop, sized 180×60 and 90×30; (right bottom) single eye crop, sized 60×32 and 30×18.

Results. The results of this experiment can be seen in Figure 4. As expected, we observed that the lowest error rates are obtained by the largest size of input data with the maximum amount of context: the full face image. We also observe that using this input type results in the highest amount of computation load.



(a) MPIIFaceGaze

(b) EYEDIAP

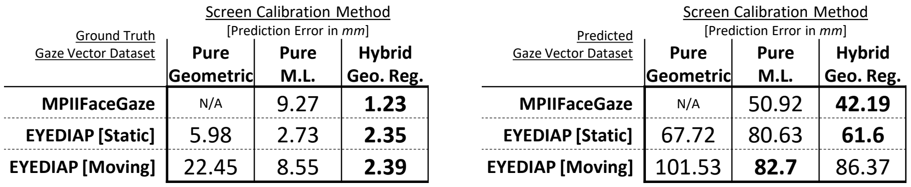
Fig. 4. Scatter plots of the performance of a VGG16 based gaze tracking network trained on different input types vs their computation load/time in FLOPS/seconds. The error bars represent the standard deviation of the errors (5-fold cross-validation). While the computation cost of these inputs vastly vary, they all perform in roughly the same range of accuracy. The red dashed line represents approximate real-time computation at 15 fps on an *Intel i7* CPU.

As we reduce the input sizes, the accuracy only marginally degrades while the computation load gets cut down severely. In fact, even if we simply use a crop of the eye region or just the crop of a single eye, we obtain accuracies comparable to that from full face input albeit with a fraction of the computation.

5.3 Exp 2: Usability/Accuracy Trade-off for Screen Calibration

Setup. To evaluate the three screen calibration techniques proposed, we train and test them individually using calibration data samples from MPIIFaceGaze and EYEDIAP. This data for screen calibration consists of pairs of gaze-vectors and their corresponding ground truth screen points. Using this, calibration methods are trained to predict the 2D screen points from the 3D gaze-vectors. We evaluate on noise-free ground truth gaze-vectors and on realistic predicted gaze-vectors (using 30×18 eye crop as input) so as to assess the accuracy of the complete camera-to-screen eye-tracking pipeline. As training data, we obtain calibration data pairs of gaze-vectors and points such that they are spread out evenly over the screen area. This is done by dividing the screen in an evenly-spaced grid and extracting the same number of points from each grid region.

Results. The results of these experiments can be seen in Table 1 for a fixed calibration training set size of 100 samples. For the 'theoretical' task of predicting gaze-points from noise-free ground truth gaze-vectors, we see in Table 1a that the hybrid geometric regression method outperforms others. We see that the gap in performance is smaller when head poses are static, while the hybrid method does better for moving head poses. This suggests that for the simplest evaluation on static head poses with noise-free gaze-vectors, all methods perform well; however, as movement is introduced, the limitations of the purely geometric method and the advantage of hybrid method becomes clearer.



Ground Truth Gaze Vector Dataset	Screen Calibration Method [Prediction Error in *mm*] Pure Geometric	Pure M.L.	Hybrid Geo. Reg.
MPIIFaceGaze	N/A	9.27	**1.23**
EYEDIAP [Static]	5.98	2.73	**2.35**
EYEDIAP [Moving]	22.45	8.55	**2.39**

(a) Groundtruth gaze-vector calibration

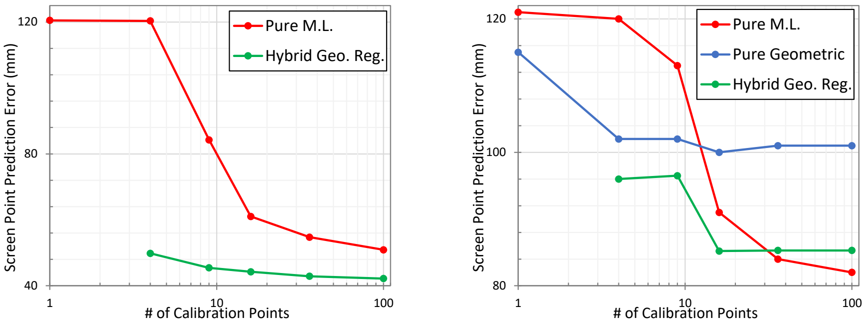
Predicted Gaze Vector Dataset	Screen Calibration Method [Prediction Error in *mm*] Pure Geometric	Pure M.L.	Hybrid Geo. Reg.
MPIIFaceGaze	N/A	50.92	**42.19**
EYEDIAP [Static]	67.72	80.63	**61.6**
EYEDIAP [Moving]	101.53	**82.7**	86.37

(b) Predicted gaze-vector calibration

Table 1. Performance of calibration methods (trained with 100 samples) on different datasets and conditions expressed in gaze-point prediction errors (in *mm*). Hybrid geometric regression technique significantly outperforms both purely geometric and purely machine learning (M.L.) based calibration methods in most conditions.
Legend: [Static] denotes static head poses, [Moving] denotes moving head poses.

When calibration is performed on actual gaze-vectors predicted by the system, overall accuracy deteriorates by one to three orders of magnitude. Comparing the methods, the hybrid geometric regression method also does well compared to others in most conditions, as seen in Table 1b. We observe that the purely geometric method actually copes better than the ML based method when head poses are static. However, it's performance severely degrades with moving head poses. Also, the ML method is able to marginally outperform the hybrid method on moving head poses. This is likely because given sufficient training samples, the ML method is able to learn features from the input that the other—more rigid—methods cannot do. Note that only the EYEDIAP dataset results are reported for the purely geometric technique, since this method can only be trained on static head poses and MPIIFaceGaze does not have any static head poses (the geometric method can still be tested on the moving head poses of EYEDIAP).

To assess the efficiency of these calibration methods, we must ascertain the least amount of calibration samples required with which satisfactory performance can still be attained. This can be assessed by observing the learning curves of the calibration methods, where the prediction errors of the methods are plotted against the number of calibration/training points used. This is shown in Figure 5.



(a) MPIIFaceGaze (b) EYEDIAP (Moving)

Fig. 5. Learning curves of the calibration techniques on MPIIFaceGaze and EYEDIAP dataset (log scale). The purely geometric method performs better than ML method when calibration data is scarce, but does not improve further when more data is available. The ML method improves greatly when calibration data becomes abundant. The hybrid geometric regression method performs the best over a wide range of calibration data points used.

The hybrid method is able to outperform both the other methods even when a low number of calibration points are available. An interesting observation seen in Figure 5b is that the purely geometric method actually performs better than the ML method when the number of calibration points is low ($\lessapprox 9$). This can be due to the rigid and pre-defined nature of the geometric model which has prior knowledge strongly imparted into it. On the other hand, the ML model requires more data points to learn the underlying geometry from scratch. This is also seen in the results: as the number of points increase, the ML model's performance improves while the geometric model stagnates. Overall, the lower error rate of the hybrid model over a broad range of used calibration points affirms its strengths over the overtly rigid geometric model and the purely data-driven ML approach.

6 Discussion

The experiments related to input types and sizes produce some insightful and promising results. The comparison between them with respect to their performance vs their computational load indicate that the heavier processing of larger inputs with more contextual information is not worth the performance gain they produce. Roughly the same accuracies can be obtained by a system that relies only on eye image crops. In contrast, the gap in the computational load between these two input types is a factor of 20. This supports our idea that for an objective measurement task like gaze-vector prediction, the value of context is limited. These results can help in guiding the design of eye tracking systems meant for real-time applications where efficiency is key.

Outputs in the form of gaze-vectors are not always readily useful in a computer-facing scenario: they need to be projected onto the screen to actually determine where the person is looking. This area has received little attention in literature, and our experiments provide some insight. Our comparison of three calibration techniques show that a hybrid geometric regression method gives the overall best performance over a wide range of available calibration data points. Our results show that purely geometric modelling works better when calibration points are very few, while a purely ML method outperforms it when more points become available. However, a hybrid model offers a robust trade-off between them.

7 Conclusion

In this work, we explored the value of visual context in input for the task of gaze tracking from camera images. Our study gives an overview of the accuracy different types and sizes of inputs can achieve, in relation to the amount of computation their analysis requires. The results strongly showed that the improvement obtained from large input sizes with rich contextual information is limited while their computational load is prohibitively high. Additionally, we explored three screen calibration techniques that project gaze-vectors onto screens without knowing the exact transformations, achieved with the cooperation of the user. We showed that in most cases, a hybrid geometric regression method outperforms a purely geometric or machine learning based calibration while generally requiring less calibration data points and thus being more efficient.

Acknowledgement

The authors are grateful to Messrs. Tom Viering, Nikolaas Steenbergen, Mihail Bazhba, Tim Rietveld, Hans Tangelder for their most valuable inputs.
Special thanks to Sig. Fabio Gatti for critical driving services :)

References

1. Baltrušaitis, T., Robinson, P., Morency, L.P.: Continuous conditional neural fields for structured regression. In: Proceedings of the European Conference on Computer Vision. pp. 593–608 (2014)
2. Chen, Z., Shi, B.E.: Appearance-based gaze estimation using dilated-convolutions. In: Proceedings of the Asian Conference on Computer Vision. pp. 309–324 (2018)
3. Cheng, Y., Lu, F., Zhang, X.: Appearance-based gaze estimation via evaluation-guided asymmetric regression. In: Proceedings of The European Conference on Computer Vision (2018)
4. Deng, H., Zhu, W.: Monocular free-head 3d gaze tracking with deep learning and geometry constraints. In: Proceedings of the International Conference on Computer Vision. pp. 3162–3171 (2017)
5. Fischer, T., Jin Chang, H., Demiris, Y.: Rt-gene: Real-time eye gaze estimation in natural environments. In: Proceedings of the European Conference on Computer Vision (2018)
6. Funes Mora, K.A., Monay, F., Odobez, J.M.: Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras. In: Proceedings of the ACM Symposium on Eye Tracking Research and Applications (2014)
7. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 4700–4708 (2017)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)
9. Kasprowski, P., Harezlak, K., Stasch, M.: Guidelines for the eye tracker calibration using points of regard. In: Information Technologies in Biomedicine, Volume 4. pp. 225–236. Springer International Publishing (2014)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations (2015)
11. Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., Torralba, A.: Eye tracking for everyone. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
12. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European conference on computer vision. pp. 483–499. Springer (2016)
13. Palmero, C., Selva, J., Bagheri, M.A., Escalera, S.: Recurrent CNN for 3d gaze estimation using appearance and shape cues. In: British Machine Vision Conference (BMVC) (2018)
14. Papoutsaki, A., Sangkloy, P., Laskey, J., Daskalova, N., Huang, J., Hays, J.: Webgazer: Scalable webcam eye tracking using user interactions. In: Proceedings of the International Joint Conference on Artificial Intelligence. pp. 3839–3845 (2016)
15. Park, S., Mello, S.D., Molchanov, P., Iqbal, U., Hilliges, O., Kautz, J.: Few-shot adaptive gaze estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9368–9377 (2019)
16. Park, S., Spurr, A., Hilliges, O.: Deep pictorial gaze estimation. In: European conference on computer vision (2018)
17. Ranjan, R., De Mello, S., Kautz, J.: Light-weight head pose invariant gaze tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 2156–2164 (2018)

18. Rodrigues, R., Barreto, J.P., Nunes, U.: Camera pose estimation using images of planar mirror reflections. In: European Conference on Computer Vision (2010)
19. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) pp. 211–252 (2015)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
21. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization pp. 341–359 (1997)
22. Sugano, Y., Matsushita, Y., Sato, Y.: Learning-by-synthesis for appearance-based 3d gaze estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1821–1828 (2014)
23. Tan, K.H., Kriegman, D.J., Ahuja, N.: Appearance-based eye gaze estimation. In: Sixth IEEE Workshop on Applications of Computer Vision, 2002.(WACV 2002). Proceedings. pp. 191–195. IEEE (2002)
24. Tripathi, S., Guenter, B.: A statistical approach to continuous self-calibrating eye gaze tracking for head-mounted virtual reality systems. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 862–870. IEEE (2017)
25. Wood, E., Baltrušaitis, T., Morency, L.P., Robinson, P., Bulling, A.: A 3d morphable model of the eye region. In: Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: Posters. pp. 35–36 (2016)
26. Wood, E., Bulling, A.: Eyetab: Model-based gaze estimation on unmodified tablet computers. In: Proceedings of the Symposium on Eye Tracking Research and Applications. pp. 207–210. ACM (2014)
27. Yu, Y., Liu, G., Odobez, J.M.: Improving few-shot user-specific gaze adaptation via gaze redirection synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11937–11946 (2019)
28. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Its written all over your face: Full-face appearance-based gaze estimation. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 2299–2308 (2017)
29. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Appearance-based gaze estimation in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4511–4520 (2015)
30. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(1), 162–175 (2017)