A Statistical Approach to Continuous Self-Calibrating Eye Gaze Tracking for Head-Mounted Virtual Reality Systems

Subarna Tripathi
UC San Diego
stripathi@ucsd.edu

Brian Guenter
Microsoft Research
bguenter@microsoft.com

Abstract

We present a novel, automatic eye gaze tracking scheme inspired by smooth pursuit eye motion while playing mobile games or watching virtual reality contents. Our algorithm continuously calibrates an eye tracking system for a head mounted display. This eliminates the need for an explicit calibration step and automatically compensates for small movements of the headset with respect to the head. The algorithm finds correspondences between corneal motion and screen space motion, and uses these to generate Gaussian Process Regression models. A combination of those models provides a continuous mapping from corneal position to screen space position. Accuracy is nearly as good as achieved with an explicit calibration step.

1. Introduction

Head mounted eye trackers require precise knowledge of the intrinsic parameters of the eye tracking camera, principally focal length and distortion, as well as the extrinsic parameters, the position and orientation of the camera with respect to the display and illumination sources.

Maintaining precise calibration in a consumer HMD device is difficult; laboratory equipment is handled with care but consumer devices may be dropped, sat upon, or left in the sun. Injection molded plastic is not very stiff so even a modest torque or force can distort the headset. Plastics also have a high coefficient of thermal expansion so parts displace relative to each other as the electronics heat up and cool down.

Most eye tracking systems also require an initial calibration phase, where the user fixates their gaze at target points. This is inconvenient since it must be done every time the headset is put on. The calibration also rapidly drifts due to movement of the headset with respect to the head.

Our algorithm continuously and automatically calibrates the eye tracking system by detecting correspondence between corneal motion and the motion of objects seen in the display. Only very approximate camera intrinsic and extrinsic parameters are needed, so the system is immune to calibration errors caused by small physical distortions of the headset. Through experimental evaluations, we show that the proposed algorithm can achieve gaze estimation accuracy competitive with that of a calibrated eye-tracker, without any manual calibration.

Our primary contribution is that the proposed approach of real-time eye gaze tracking system is able to auto calibrate through smooth pursuit and allow people to look wherever they want rather than requiring them to track a single object. The system creates and maintains both a global and a local statistical model for eye-gaze tracking and can automatically select the best prediction among these two models. The local model can respond much more quickly to changes in headset position and the global model predicts the eye gaze more accurately while the system is in a stable state.

2. Related Work

Gaze estimation techniques have long been targeted for improving human-computer interaction, and advancement of assistive technologies for impaired. Traditionally, gaze estimation techniques use a system where the user can move in front of a computer screen such as [9, 8]. One of such state-of-the-art remote set-up methods [8] exploiting distant camera and nominal head-motion setup reports gaze estimation error about 2.27°.

Alnajar *et al.* [1] presented a method to automatically calibrate gaze in an un-calibrated setup by using the gaze patterns of individuals to estimate the gaze points for new viewers without active calibration using saliency maps and reported an average accuracy of 4.3°. Other saliency map based auto-calibration methods include [20, 3]. Hansen *et al.* [6] compared several different approaches for estimating user's point of regard (PoR). Moving target based or smooth pursuit based calibration methods such as [12, 24] exist for remote setup, but remains unexplored for VR mobile gaming.

With wearable devices being more widely used, gaze es-

timation has also been explored for virtual reality systems and systems with see-through displays. In the arena of see-through displays, Pirri *et al.* [15, 14] proposed a procedure for calibrating a scene-facing camera's pose with respect to the users gaze. In spite of being effective, its dependence on artificial markers in the scene limits the applicability.

Tsukada *et al.* [23] presented a single-eye based gaze detection system that leverages an appearance code book for the gaze mapping. The appearance code book is sensitive to the calibration, and assumed to be constant throughout use. Another recent method [11] proposed adaptive eye-camera calibration for head-worn devices. The method is able to work well with changes in calibration during the time of use with locally-optimal eye-device transformation by leveraging salient regions. However, their results are reported on a simulated dataset and human face centered dataset, both are restrictive compared to the real-world cases.

Although state-of-the-art gaze estimation techniques claim to achieve less than one degree error, in practical settings there are several different error sources such as disparity and physiological differences [2] that affect gaze quality over time. The challenge of compensating the drift from the initial calibration has been analyzed in [21]. Drift analysis is performed on a dataset of natural gaze recorded using synchronized video-based and Electrooculugraphy-based eye trackers of 20 users performing everyday activity in a mobile setting.

By analyzing the drift, [21] proposed a method to automatically self-calibrate head-mounted eye trackers based on a computational model of bottom-up visual saliency such as Boolean Map Saliency, Face Detection, Pedestrian Detection. The high-level saliency detection runs at 17 frames per second. In spite of being a robust mapping approach based on error analysis on real-world first-person video in practical settings which supports HMD movement, the method is far from being real-time.

Other recent methods which exploit video signals either use appearance-based methods with eye-image as a descriptor to obtain gaze [22] position or Shape-based methods by tracking parts of actual eye-anatomy such as corneal reflection, pupil contour, and iris contour [9, 10, 5]. Corneal reflection and pupil contour methods need infrared ray (IR) active illumination [13, 6]. SensoMotoric Instruments (SMI) [19] has its prototype eye-tracking VR HMD and showed its eye-tracked Rift at the 2014 Augmented World Expo. That prototype shares the similar kind of architecture as of ours except they use two 3D eye-facing cameras. Their eye-tracking HMD system measures the 3D eye model which is used as input for gaze based interaction schemes. The latency is high (between 50 ms to 100 ms) for applications such as foveated rendering.

Apart from being computationally-heavy geometric model estimator, the system is not truly calibration-free. The utility contains a default explicit one-point calibration and a three-points calibration procedure which asks the user to look at each dot and pressing a button. In spite of being highly accurate (0.5 to $1\,\mathrm{deg}$) at the cost of latency and power, the tracking accuracy is quite sensitive to HMD placement. Even a little pressure on the HMD without really moving it, can cause noticeable shift and the system needs to go through the explicit calibration utility again as per users feedback [18].

Very recently convolutional neural network based methods such as [16, 7] are proposed. The authors collect a new dataset containing a set of images as well as annotation of the gaze of each person inside an image. In spite of providing an interesting research frontier, these methods are not yet foveated rendering ready since the angular errors are larger than 10°.

Our proposed method of real-time eye gaze tracking system is able to auto calibrate and allow people to look wherever they want rather than requiring them to track a single object. The proposed approach relies on simple and efficient method of exploiting smooth pursuit for appropriate visual signals present in mobile games or Virtual Reality video content.

3. System Overview

The eye tracking prototype is a modified Oculus Rift Dk2, as shown in fig. 3.1. Two custom prototype high speed cameras are mounted on the top of the headset and see the eye reflected in a hot mirror tilted at 67 degrees. The eye is seen through the lens of the Oculus.

The prototype cameras use a custom image sensor developed in house. It has 13um square responsive pixels with a unique ADC per pixel architecture. The imager is capable of high-frame rates, up to 1000 fps with an exposure time of 500us, and low-power operation ranging from 60mW at 1000 fps to 3mW at 60fps.

All except one of the prototype cameras failed before we could gather results so the system used for this paper has a single camera. The left eye looks at the display through the Oculus lens and the right eye is tracked by the camera. A black screen is presented to the right eye to avoid vergence problems.

An infrared illuminator shines on the eye, creating bright glints on the surface of the cornea. The closeup image, fig. 3.2, shows the IR led illuminator array more clearly. These glints are used to compute the center of the three-dimensional corneal sphere based on similar principles described in [25]. The x,y,z coordinates of the corneal center are the input to the learning algorithm.

The camera view of the eye is shown in fig. 3.3. Because only glints are tracked the exposure is dark, as seen in the raw camera image on the left. On the right the brightness levels have been raised so the eye and illuminator circuitry

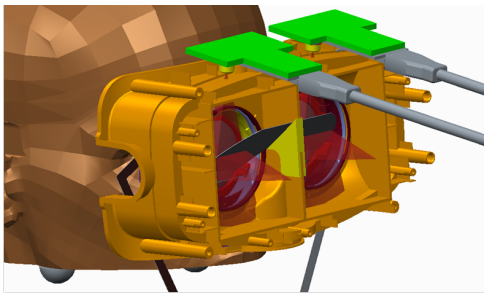


Figure 3.1. Cutaway CAD view of the prototype HMD made from a modified Oculus Rift Dk2. We had the Oculus shell professionally 3D scanned and changed the CAD file to insert custom prototype high speed cameras and an IR reflecting hot mirror. Each camera (the green T shapes) looks down from the top of the HMD and sees the reflected image of the eye in the hot mirror. The user looks through the hot mirror to the display, not shown in this figure. The user cannot see through the front of the HMD. (best viewed in color)

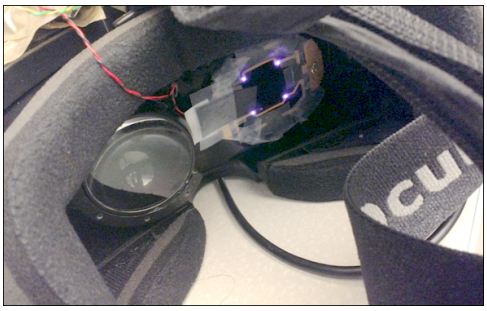


Figure 3.2. Closeup of the interior of the prototype showing the led illuminator array.

are both visible.

Tracking glints alone, rather than also tracking the pupil, is attractive for several reasons. Far less illumination power is required, a big plus for mobile HMD's. The brightness of the glints also makes it possible to capture at very high frame rates. Our system works at up to 500Hz. Computation is also low; bright spots are much easier to track than the pupil and many fewer pixels have to be processed. There is far less variation across people in corneal reflectance than in the contrast between pupil and iris so tracking should be more robust.

4. Background

The regression function which maps the cornea location in the camera coordinate system to screen space gaze location can either be model-based or statistical in nature. In case of a parametric model, a common practice is to fit a bi-quadratic function to estimate the mapping where the corneal location and corresponding screen gaze points are collected by asking the user to fixate the gaze at some predefined screen points while keeping the head as still as possible.

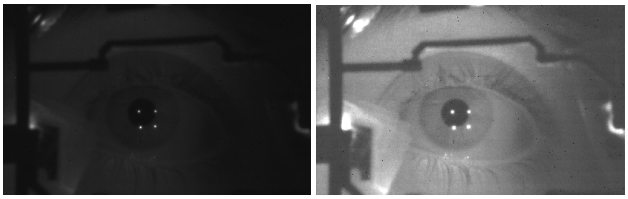


Figure 3.3. The eye as seen by the eye tracking camera. On the left is the raw image. Because we track only glints the exposure is very dark. This reduces illuminator power and allows higher frame rates at any given power. On the right the brightness level of the image has been raised to show more detail.

We explore on Gaussian Process Regression (GPR) for a statistical model. Here each observation $\mathbf{y}$ can be thought of as related to an underlying function $f(\mathbf{x})$ through a Gaussian noise model as:

$$\mathbf{y} = f(\mathbf{x}) + \mathcal{N}(0, {\sigma_n}^2) \tag{1}$$

One observation is related to another through the *covariance function*, $\mathcal{K}(\mathbf{x}, \mathbf{x}')$. We fold the noise into $\mathcal{K}(x, x')$ as in [4]:

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = {\sigma_f}^2 \exp[\frac{-(\mathbf{x} - \mathbf{x}')^2}{2l^2}] + {\sigma_n}^2 \delta(\mathbf{x}, \mathbf{x}') \tag{2}$$

where $\delta(\mathbf{x}, \mathbf{x}')$ is the Kronecker delta function. Thus, given n observations $\bar{\mathbf{y}}$, our objective becomes predicting $\mathbf{y}_*$, and not the actual value of f_*. To prepare for the GPR, we calculate the covariance function between all possible combinations of the points. Extreme off-diagonal elements tend to be zero when $\mathbf{x}$ spans a large enough domain. For, 9-point baseline calibration setup, the covariance function takes the following form:

$$\mathcal{K} = \begin{bmatrix} k(\mathbf{x_1}, \mathbf{x_1}) & k(\mathbf{x_1}, \mathbf{x_2}) & \cdots & k(\mathbf{x_1}, \mathbf{x_9}) \\ k(\mathbf{x_2}, \mathbf{x_1}) & k(\mathbf{x_2}, \mathbf{x_2}) & \cdots & k(\mathbf{x_2}, \mathbf{x_9}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x_9}, \mathbf{x_1}) & k(\mathbf{x_9}, \mathbf{x_2}) & \cdots & k(\mathbf{x_9}, \mathbf{x_9}) \end{bmatrix}$$

According to the key assumption in GP modeling that the data can be represented as a sample from a multivariate Gaussian distribution, we have

$$\begin{pmatrix} \bar{\mathbf{y}} \\ \mathbf{y}_* \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{pmatrix} \mathcal{K} & {\mathcal{K}_*}^T \\ \mathcal{K}_* & \mathcal{K}_{**} \end{pmatrix}) \tag{3}$$

The conditional probability $\mathbf{y}_* | \bar{\mathbf{y}}$ follows a Gaussian distribution:

$$\mathbf{y}_* | \bar{\mathbf{y}} \sim \mathcal{N}(\mathcal{K}_* K^{-1} \bar{\mathbf{y}}, \mathcal{K}_{**} - \mathcal{K}_* K^{-1} {\mathcal{K}_*}^T) \tag{4}$$

Thus, the best estimate for $\mathbf{y}_*$ is the mean of the distribution $\mathcal{K}_* K^{-1} \bar{\mathbf{y}}$ and the uncertainty in the estimate is captured in its variance: $var(\mathbf{y}_*) = \mathcal{K}_{**} - \mathcal{K}_* K^{-1} {\mathcal{K}_*}^T$

where, $\mathbf{x}_*$ is the input test point, $\mathcal{K}_* = [k(\mathbf{x}_*, \mathbf{x_1}), k(\mathbf{x}_*, \mathbf{x_1}), ..., k(\mathbf{x}_*, \mathbf{x_9})]$ and $\mathcal{K}_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$. We note that the maximum allowable covariance defined as ${\sigma_f}^2$ should be high for functions which cover a broad range on the $\mathbf{y}$ axes. Thus, the values of the hyper-parameters i.e. ${\sigma_f}^2$, ${\sigma_n}^2$ and the kernel width, l should be selected based on the nature of the input and output data. For the baseline system, we use mean-centered unit-norm data distribution and select the values of these hyper-parameters accordingly.

5. Auto-Calibration

The proposed auto-calibration is about learning the mapping from corneal location $\mathbf{x}$ (x_c, y_c, z_c) to screen gaze location $\mathbf{y}$ (x_g, y_g) through Gaussian Process Regression. However, unlike explicit 9-point calibration method the correspondence between corneal location and screen gaze point data is not available directly. In addition, the system needs to be able to recover from minor HMD movement with respect to the head.

A typical scenario of mobile video games where 2D icons are moving over the screens with perceivable speed, smooth pursuit becomes the most significant eye motion while playing those games. Our auto-calibration algorithm exploits smooth pursuit eye movement to dispense with the need of explicit calibration and is able to continuously calibrate the wearable system.

5.1. Tracklet

We have the corneal location measurement as explained in section 3. To identify the corresponding object on screen, we find trajectory similarity between cornea and one of the multiple moving objects on the screen space. Since it is natural for a user to follow a moving object for some tangible duration of time and then change gaze and follow another one, we look for similarity between corneal trajectory and screen-space motion trajectories within a temporal neighborhood. We define temporal locality in terms of tracklets. A query tracklet is a collection of consecutive n samples in corneal trajectory. In our experiments, $n = 40$ to 50. Apart from the trajectory appearance similarity between the query and candidate tracklets, an informed guess about the possible spatial locations of screen-space candidates leads to computational tractability and improved eye tracking accuracy. We describe the tracklet matching strategy in subsection 5.2.

5.2. Tracklet Matching

In the process of learning the mapping between $3D$ mapping from corneal location to $2D$ screen-space location, we observe that the 2D corneal trajectory (assuming constant depth of corneal sphere with respect to the camera) and the motion of the object the user follows on screen space has appearance similarity in a small spatio-temporal neighborhood. In the beginning, without any knowledge of the possible mapping from corneal location to screen gaze points, we rely on discrete normalized velocity similarity between the corneal trajectory and the best matched object's motion. Limited number of independently moving 2D icons over screen space makes it possible to evaluate candidate tracklets efficiently. Figure 5.1 show some matched tracklets.

The horizontal and the vertical direction of motion is assumed to be independent. Tracklet matching algorithm takes the inputs of synchronized time series of horizontal and vertical corneal locations and the coordinates of all objects on the screen space and outputs the single object whose trajectory is most similar to that of the eye or nothing if no object is being followed. The process is continuously performed over small temporal windows. With informed guess on search-space, directional independence hypothesis, and smooth pursuit in small temporal window - initialization of tracklet-matching reduces from complex 2D/3D trajectory shape matching to finding minimum euclidean distances between horizontal and vertical velocity vectors independently.

Initialization Let, $\mathbf{C_X}$ and $\mathbf{C_Y}$ denote the corneal X and Y position vectors for the query tracklet in temporal order. We begin with normalizing $\mathbf{C_X}$ and $\mathbf{C_Y}$ so that the sum of the individual elements in both the vectors becomes unity. All the candidate screen space tracklets, $\mathbf{S^i_X}$ and $\mathbf{S^i_Y}$ are similarly normalized where i denotes the index of the candidate tracklet. The direction of the horizontal component of the corneal velocity vector with respect to the eye-facing camera coordinate system is opposite to that of the corresponding object's horizontal velocity. Therefore, direction of corneal horizontal velocity is reversed before tracklet appearance matching process. Let, D^j_X be the euclidean distance between normalized sign-reversed vector of horizontal motion and j-th screen space trajectory's horizontal component and D^j_Y be the euclidean distance between normalized corneal vertical motion and j-th screen space trajectory's vertical component for valid temporal indices. Then the appearance similarity S_j with j-th candidate tracklet is calculated as:

$$\begin{aligned} D^j_X &= \Sigma(-\mathbf{NC_X} - \mathbf{NS^j_X}) \\ D^j_Y &= \Sigma(\mathbf{NC_Y} - \mathbf{NS^j_Y}) \\ D_j &= \sqrt{{D^j_X}^2 + {D^j_Y}^2} \\ S_j &= \exp(-D_j/s_n) \end{aligned} \tag{5}$$

where prefix N denotes normalized vectors and s_n is the cardinality of the query tracklet. If the horizontal coordinate value of the object gets bigger when moving to the left or right of the screen, horizontal location of the eye

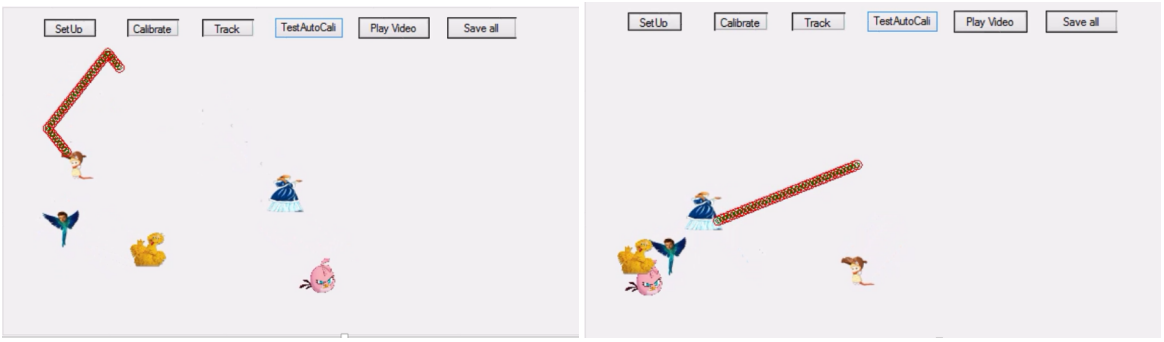


Figure 5.1. Visualization of best candidate tracklet selection. At two different time instant, the best matched corresponding screen-space tracklets are shown. The left one corresponds to following the path of flying baby and the right one corresponds to following the path of princess for about half a second in each case. Green circles are the selected screen-space tracklet and the red circles demonstrate the correctness of the local model learned on the those data-pairs i.e. predicting gaze point for those associated corneal locations. (best viewed in color)

should also increase. This implies that the screen should be located orthogonal to the line of sight which is mostly true for HMD setup. However, some relative movement of HMD can invalidate the constraint and there arises the need of tracklet matching with other efficient method.

After Initialization: The candidate with the highest similarity is the one we start learning the GPR (4) where the input is 3D-corneal trajectory and the output is 2D screen-space gaze locations. After the first set of samples collected through initial temporal window, we obtain the first set of corresponding corneal ($\bar{\mathbf{x}}_1$) and screen gaze data points ($\bar{\mathbf{y}}_1$). Thus, we have initial estimate of the covariance matrix ($\hat{\mathcal{K}}_1$) of the first local GPR model. The initial appearance similarity based matching requires alignment with line-of-sight *i.e.* at every new frame we update the local mapping model with the newly found valid corneal and screen-space data-pair using the *so far* learned GPR model. Updating the model means updating $\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i$, and $\hat{\mathcal{K}}_i$; where i denotes the index of the local GPR model. The term local model signifies the locality of space and time which a small tracklet satisfies.

If the distance between the gaze prediction on screen space ($\mathcal{K}_{i*} K_i^{-1} \bar{\mathbf{y}}_i$) using the continuously learned GPR model and the best screen space candidate is small enough, we use that data-pair to update the corresponding local GPR model. The local model is able to quickly respond to the changes of the relative pose between the head and the headset.

5.3. Local GPR Models

The pursuit is initiated with every moving signal and under most natural circumstances, a user looks at or follows different objects in different time instant. As every tracklet is localized in space-time, it makes sense to maintain multiple local GPR models to ensure better space coverage for minimizing prediction uncertainty. As GPR involves matrix inversion, using all corresponding data-pairs for all the matched tracklets is prohibitive from computation and memory point of view. We perform the continuous local mapping model update and maintain them in a circular buffer fashion. Every buffer can be thought of as pointing to a block of varying size which is less or equal to the tracklet sample size in the overall system covariance matrix. In the absence of prior information on all the data at once, we approximate the system covariance matrix as a block diagonal matrix assuming each tracklet is independent. The maximum size of the system matrix $\mathcal{K}$ could at most be the number of points present in a tracklet (n) times the number of local GPR models (m) maintained. We use the value of m between 8 to 15 for our experiments and observed that they yield similar accuracy. Size of each local block can vary based on the prediction accuracy based validity criteria. The circular buffer implementation replaces the oldest block in the system covariance matrix with the newest one.

$$\mathcal{K} = \begin{pmatrix} \underbrace{\begin{bmatrix} \hat{K}_1 \end{bmatrix}}_{n_1 \times n_1} & & & \\ & \underbrace{\begin{bmatrix} \hat{K}_2 \end{bmatrix}}_{n_2 \times n_2} & & \\ & & \ddots & \\ & & & \underbrace{\begin{bmatrix} \hat{K}_m \end{bmatrix}}_{n_m \times n_m} \end{pmatrix} \tag{6}$$

Our intuition is that the best so far model learned in the far past can not compensate for a recent possible change

in relative head to HMD pose. System's covariance matrix including all local models looks like equation 6. It always maintains the most recent m number of local models each of which could be of different sizes during online learning. Their respective covariance matrices are denoted by $\hat{K}_i$, where $i \in [1, m]$.

5.4. Global GPR Model

Tracklet independence assumption yields a block-diagonal system covariance matrix which is computationally tractable but not necessarily a good approximation for actual system covariance where data correspondences are spread across a wide range of screen space. Hence, the local GPR based prediction might not be highly confident in the entire screen space region. Thus, we want to have a set of sparse samples of cornea-screen data pairs over the screen space which are valid for relatively longer duration of time. The GPR model learned using those samples is called global model.

For encouraging space-diversity, we divide the screen-space into non-overlapping rectangular grids and use only limited number of best data pairs seen from all matched tracklets per grid. In our experiments, we use 6×4 grids and maintain at most 4 best data pairs per grid. Thus, at any moment the global GPR model deals with inverting a matrix of at most 96×96 size. As the covariance matrix is symmetric and we get data in incremental fashion, we can employ incremental matrix inversion strategy for updating both the global model and the local models. The covariance matrix for the global model, $\hat{K}_g$, is not block-diagonal and captures the essence of eye-screen transformation capable of performing a better prediction through interpolation.

5.5. Overall Framework

We initialize the first local GPR model by establishing the correspondence between corneal and screen-space tracklet by trajectory appearance similarity. Then, we keep updating the first local GPR model as we keep getting more on-the-fly corneal and screen-space correspondences. The covariance matrix corresponding to the first matched-tracklet becomes the first block of the block-diagonal system covariance matrix. We then continue matching tracklets based on the model learned so far. We keep m of such local GPR models created by good correspondences from the last m matched tracklets. At every frame, the current local GPR model is updated with the new corresponding corneal and screen-space tracklet data pair. The space-diverse sampling of all data-pairs is continuously performed for creating a global model.

Online gaze detection involves two gaze predictions: one from the best local model and another from the global model. Once the samples for learning the global model tend to have more space-diversity than the current local model, the prediction from the global model obtains higher confidence in any region of the screen than the one from the local models. Local models are better responsive to relative eye-HMD movement. The prediction from global model is more confident for stable-HMD, specially during saccadic corneal motion. When prediction from global model suddenly becomes significantly less confident compared with the local model, we consider it as a notable relative movement between the head and the HMD. Therefore, we start the process of recreating global GPR model. The test time gaze prediction becomes the selection between the local model-based prediction and the global model-based prediction.

The gaze estimate y_{l*} from the local model and its uncertainty $var(y_{l*})$ are calculated as:

$$\begin{aligned} j &= \arg\min_{i \in C} (\hat{K}_{i**} - \hat{K}_{i*}\hat{K}_i^{-1}\hat{K}_{i*}^T) \\ y_{l*} &= \hat{K}_j * \hat{K}_j^{-1}\mathbf{y} \\ var(y_{l*}) &= \hat{K}_{j**} - \hat{K}_{j*}\hat{K}_j^{-1}\hat{K}_{j*}^T \end{aligned} \tag{7}$$

where C is the set of indices of all local models considered.

And, the gaze estimate, y_{g*}, from the global model and its uncertainty, $var(y_{g*})$, are calculate as :

$$\begin{aligned} y_{g*} &= \hat{K}_g * \hat{K}_g^{-1}\mathbf{y} \\ var(y_{g*}) &= \hat{K}_{g**} - \hat{K}_{g*}\hat{K}_g^{-1}\hat{K}_{g*}^T \end{aligned} \tag{8}$$

If the prediction from the global model has more uncertainty $var(y_{g*})$ than the uncertainty $var(y_{l*})$ of the best local model, either due to less number of training samples generated so far or due to relative head-HMD movement, the global model is disposed and created again.

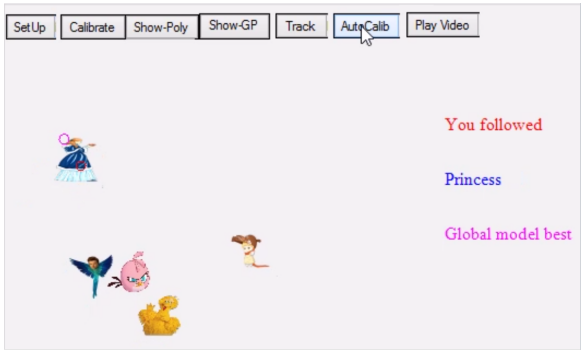
Final gaze prediction, y_* becomes:

$$y_* = \begin{cases} y_{l*} & \text{if } var(y_{g*}) > var(y_{l*}); \\ y_{g*} & \text{otherwise.} \end{cases} \tag{9}$$

On the fly selection between prediction from local and global model has been visually validated in the screen-shots as shown in figure 5.2.

While creating the covariance matrix i.e. the individual blocks in equation 6, we do not consider mean-centered unit-norm distribution since estimating screen-space mean value is difficult for localized data points of a single tracklet. On the contrary, the global model can use mean-centered unit-norm data distribution by virtue of space-diverse sampling of data points. The different nature of the data distribution imposes the need of using different values for the hyper-parameters. Thus, the estimated standard deviation from local and global model has difference in scale which is compensated before comparing the uncertainty of gaze prediction from local and global model (eq. 9).

(a) Local model prediction better than the one from global model



(b) Global model prediction better than the one from local model

Figure 5.2. These screen-shots show that there are two test time gaze predictions generated - one from the global model which is shown in Magenta and another one from the best of all local models which is shown in Red. The final estimated gaze point is selected based on the uncertainty of predictions from both the global and local model. (best viewed in color)

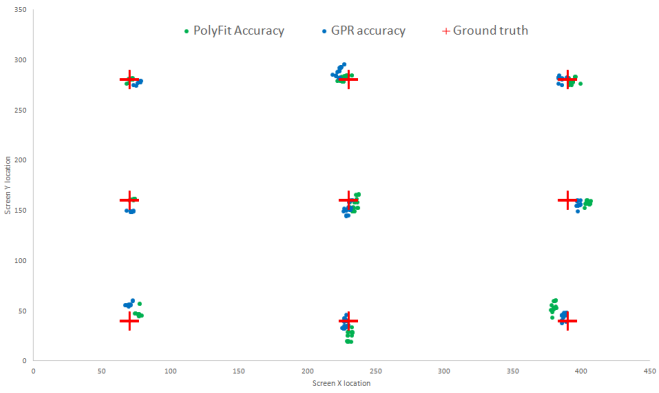


Figure 6.1. 9-Point baseline calibration evaluation. Both the polynomial model and the GPR based methods exhibit similar accuracy. (best viewed in color)

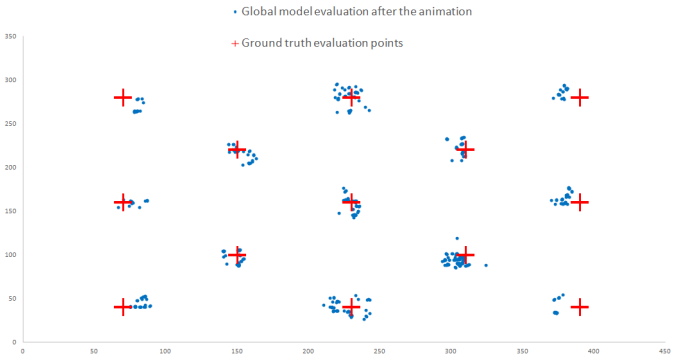


Figure 6.2. 13-point evaluation for the continuous auto-calibration method after the game. (best viewed in color)

6. Experimental Results

We evaluate both the efficiency of the explicit calibration-based GPR baseline and the proposed continuous auto-calibration method. The explicit calibration requires a user to look at a set of predefined targets as shown in Fig 6.1 to establish a mapping from cornea location to gaze positions in the users visual scene. In laboratory settings, this explicit method serves as the golden standard for eye gaze tracking evaluation.

6.1. Evaluation Method : 9-Point Baseline

We learn the best baseline model assuming the transformation function is going to be the same for the entire duration of use. In the current HMD setup, the values of corneal locations in the camera coordinate system vary from -2.5 to $+2.5$ millimeters in horizontal (x) direction, -2.5 to $+3.5$ in the vertical (y) direction and 100 millimeter or more in z-direction for different users and different relative poses. In case of baseline GPR, we use corneal (x_c, y_c, z_c) location as input and corresponding screen-gaze (x_s, y_s) as labels for learning the regression. The range of the values for screen space gaze locations goes from 0 to 500 in x direction and 0 to 300 in y-direction in the current hardware. We use mean-centered unit-variance data for learning and evaluating the model. The kernel width we use is 10. The hyper-parameters in the GPR model such as the maximum allowable covariance and noise parameters are 1.2 and 0.01 respectively.

Additionally, we perform bi-quadratic polynomial fitting. We see that both the bi-quadratic mapping and Gaussian Process Regression based transformation learning exhibit similar gaze prediction accuracy for the current system. Figure 6.1 shows the evaluation for baseline methods using bi-quadratic and GPR visually. GPR additionally provides confidence of each gaze point prediction.

6.2. Evaluation of Continuous Auto-calibration

Fig 6.2 shows the visual 13-point evaluation result of the proposed auto-calibration. Table 6.1 shows the average

gaze prediction error for VR systems. We are aware of only one commercial system SMI [19] which is made for and evaluated on virtual reality setup. In spite of being highly accurate, [19] has high latency and the tracking is significantly sensitive to HMD placement. SMI is not fully calibrating free. The rest of the entries correspond to the baseline methods in the same hardware with explicit calibration procedure. Given the fact that there are error sources such as chromatic aberration, absence of two cameras and temporal synchronization between the camera and the display in the existing hardware, the results produced with this hardware are promising. The proposed continuous auto-calibration method performs almost as good as the baseline active calibration procedure.

Comparison with VR systems and Baselines

Methods	Setup	Error (°)
SMI [19]	VR	0.5 to 1
9-pt calib, chin rest (Poly fit)	VR	0.75
9-t calib, chin rest (GPR)	VR	0.776
9-pt calib, no chin rest (Poly fit)	VR	1.223
9-pt calib, no chin rest (GPR)	VR	1.248
Proposed. Auto-calib, 13-pt eval	VR	**1.822**

Table 6.1. Comparison with VR systems.

We also compare the results of our method with other auto-calibration methods reported for wearable devices (Table 6.2). Self-calibrating eye gaze tracking methods [11, 21] for head-worn devices deal with see-though displays with scene-facing cameras which exploit 2D/3D saliency maps. Though not specifically meant for VR, methods involving the see-through display system still provide a good reference for our approach.

The accuracy of our method applied to mobile games is comparable to the performance of [11] on artificial dataset. Additionally, in comparison to 4-5 fps speed as reported in [11], our method is able to run in real-time.

Almost immune to HMD movement and amenable to practical settings, the fully-automatic system in [21] is far from being real-time due to the associated high-level saliency detection which reportedly runs at most 17 frames per second. Since our approach tries to find correspondences between corneal and screen-space object tracklets, it allows the selection of very small targets which could be difficult for other saliency based methods.

There are other self-calibrating eye-tracking systems available in the literature (Section 2) that are applied in remote eye distance scenario. The detailed performance analysis of such methods are outside the scope of this work which deals with wearables. However, from the perspective of technical relevance *i.e.* smooth pursuit based calibration, we summarize some methods. The typical pursuit calibration method such as [12] deals with sampling in a specific and restrictive way, by displaying a single smoothly moving target to the user. [24] can deal with only linear or circular object trajectories. Our unified statistical model-based matching and prediction dispenses with the linear trajectory assumption. GPR by its inherent nature is able to match and predict any kind of trajectory shape given the eye-object data-pairs which is incrementally estimated in our tracklet matching approach.

Comparison with auto-calibration AR systems

Methods	Setup	Error (°)
[11] on human dataset	AR	0.168
[11] on simulated dataset	AR	1.494
[21] : immune to HMD movement	AR	5 to 10
Proposed cont. Auto-calib	VR	**1.822**

Table 6.2. Comparison with auto-calibration AR systems.

7. Conclusion

We propose an effective continuous auto-calibration for eye tracking in head mounted displays using smooth pursuit eye movement. In most natural circumstances, smooth pursuit is generated from signals commonly present in mobile games in virtual reality environment. For automatically learning and updating the eye to screen mapping in real-time, Gaussian Process Regression models are learned through corneal and screen-space trajectory matching. A set of continuously updated local GPR models which are valid mappings on small temporal windows respond to HMD movement faster than the global GPR model which tracks over the entire screen space with higher confidence. The statistical model based unified matching and prediction dispenses with the need of linear trajectory assumption. A combination of local and global model enables real-time consistent automatic eye tracking system. Our results show that the proposed system achieves nearly as good performance as explicit calibration but without imposing any constraints such as fixating gaze as per given instruction. The method is also immune to minor relative head-HMD movement.

The method can also be extended for the VR world which can enhance user experience. For VR videos, we have forward and/or backward optical flow data at pixel level. The generation of candidate tracklets can potentially be performed through linking the optical flow as in [17] to create pixel level trajectories. A space-diverse sampling of all the tracklets with a bias towards horizontal motion can limit the search space over all the candidate tracklets with some initial guess Our future work lies in exploring continuous auto-calibration for such realistic VR video data.

References

[1] F. Alnajar, T. Gevers, R. Valenti, and S. Ghebreab. Calibration-free gaze estimation using human gaze patterns. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 137–144, Dec 2013. 1

[2] M. Barz, A. Bulling, and F. Daiber. Computational modelling and prediction of gaze estimation error for head-mounted eye trackers. http://perceptual.mpi-inf.mpg.de/files/2015/01/gazequality.pdf, 2015. 2

[3] J. Chen and Q. Ji. Probabilistic gaze estimation without active personal calibration. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 609–616, June 2011. 1

[4] M. Ebden. Gaussian processes for regression: A quick introduction. 2008. 3

[5] E. Guestrin and E. Eizenman. General theory of remote gaze estimation using the pupil center and corneal reflections. *Biomedical Engineering, IEEE Transactions on*, 53(6):1124–1133, June 2006. 2

[6] D. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):478–500, March 2010. 1, 2

[7] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba. Eye tracking for everyone. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[8] Y. ming Cheung and Q. Peng. Eye gaze tracking with a web camera in a desktop environment. *Human-Machine Systems, IEEE Transactions on*, 45(4):419–430, Aug 2015. 1

[9] C. Morimoto, A. Amir, and M. Flickner. Detecting eye position and gaze from a single camera and 2 light sources. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 314–317 vol.4, 2002. 1, 2

[10] A. Nakazawa and C. Nitschke. Point of gaze estimation through corneal surface reflection in an active illumination environment. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision ECCV 2012*, Lecture Notes in Computer Science, pages 159–172. Springer Berlin Heidelberg, 2012. 2

[11] D. Perra, R. Kumar Gupta, and J.-M. Frahm. Adaptive eye-camera calibration for head-worn devices. June 2015. 2, 8

[12] K. Pfeuffer, M. Vidal, J. Turner, A. Bulling, and H. Gellersen. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *ACM Symposium on User Interface Software and Technology (UIST)*, October 2013. 1, 8

[13] B. Pires, M. Devyver, A. Tsukada, and T. Kanade. Unwrapping the eye for visible-spectrum gaze tracking on wearable devices. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 369–376, Jan 2013. 2

[14] F. Pirri, M. Pizzoli, D. Rigato, and R. Shabani. 3d saliency maps. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 9–14, June 2011. 2

[15] F. Pirri, M. Pizzoli, and A. Rudi. A general method for the point of regard estimation in 3d space. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 921–928, June 2011. 2

[16] A. Recasens*, A. Khosla*, C. Vondrick, and A. Torralba. Where are they looking? In *Advances in Neural Information Processing Systems (NIPS)*, 2015. * indicates equal contribution. 2

[17] M. Rubinstein and C. Liu. Towards longer long-range motion trajectories. In *Proceedings of the British Machine Vision Conference*, pages 53.1–53.11. BMVA Press, 2012. 8

[18] O. Sacks. An eye-tracked oculus rift. http://doc-ok.org/?p=1021, 2014. 2

[19] SensoMotoric Instruments. Eye tracking HMD upgrade package: for the oculus rift dk2. 2014. 2, 8

[20] T. Shi, M. Liang, and X. Hu. A reverse hierarchy model for predicting eye fixations. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2822–2829, June 2014. 1

[21] Y. Sugano and A. Bulling. Self-calibrating head-mounted eye trackers using egocentric visual saliency. In *Proc. of the 28th ACM Symposium on User Interface Software and Technology (UIST 2015)*, 2015. 2, 8

[22] K.-H. Tan, D. J. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, WACV '02, Washington, DC, USA, 2002. IEEE Computer Society. 2

[23] A. Tsukada, M. Shino, M. Devyver, and T. Kanade. Illumination-free gaze estimation method for first-person vision wearable device. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2084–2091, Nov 2011. 2

[24] B. A. Vidal, M. and H. Gellersen. Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proc. of UbiComp 2013*, 2013. 1, 8

[25] A. Villanueva, J. J. Cerrolaza, and R. Cabeza. Geometry issues of a gaze tracking system. In C. Stephanidis, editor, *Universal Access in Human-Computer Interaction. Ambient Interaction: 4th International Conference on Universal Access in Human-Computer Interaction, UAHCI 2007 Held as Part of HCI International 2007 Beijing, China, July 22-27, 2007 Proceedings, Part II*, chapter 30, pages 1006–1015. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 2