Real Time Gaze Estimation with a Consumer Depth Camera

Li Sun[a,*], Zicheng Liu[b], Ming-Ting Sun[c]

[a] *College of Computer Science, Zhejiang University, Hangzhou, China*
[b] *Microsoft Research, Redmond, WA*
[c] *Department of Electrical Engineering, University of Washington, Seattle, WA*

Abstract

Existing eye-gaze-tracking systems typically require multiple infrared (IR) lights and high-quality cameras to achieve good performance and robustness against head movement. This requirement limits the systems' potential for broader applications. In this paper, we present a low-cost, non-intrusive, simple-setup gaze estimation system that can estimate the gaze direction under free head movement. In particular, the proposed system uses only a consumer depth camera (Kinect sensor) positioned at a distance from the subject. We develop a simple procedure to calibrate the geometric relationship between the screen and the camera, and subject-specific parameters. A parameterized iris model is then used to locate the center of the iris for gaze feature extraction, which can handle low-quality eye images. Finally, the gaze direction is determined based on a 3D geometric eye model, where the head movement and deviation of the visual axis from the optical axis are taken into consideration. Experimental results indicate that the system can estimate gaze with an accuracy of $1.4 \sim 2.7^\circ$ and is robust against large

*Corresponding author
Email addresses: lsun@zju.edu.cn (Li Sun), zliu@microsoft.com (Zicheng Liu), mts@uw.edu (Ming-Ting Sun)

head movements. Two real-time human-computer interaction (HCI) applications are presented to demonstrate the potential of the proposed system for wide applications.
Keywords: gaze estimation, eye tracking, gaze direction, system calibration, human-computer interaction

1. Introduction

Eye gaze estimation is used to determine the gaze direction of a person, either the line of sight or the point of regard (PoR). Researchers are vigorously investigating the eye-gaze-tracking technology with the ultimate goal of providing low-cost, non-intrusive, easy-calibration/calibration-free, highly accurate and robust systems that can be applied for general public uses. Gaze-tracking technology is highly valuable, with abundant interactive and diagnostic applications, such as HCI, virtual reality, video oculography, eye disease diagnosis and human behavior studies. For example, a real-time driver vigilance monitoring system is described in [12] for monitoring various visual bio-behaviors. Because gaze direction reflects a person's intention, gaze-tracking systems represent a unique and effective tool for disabled people [16] and also demonstrate great potential for video gaming and gaze-contingent interactive graphic displays [29]. In addition, gaze-tracking technology has been used for image annotation [9] and remote collaboration [19, 24]. Working together with 3D object retrieval technique [7], gaze-tracking can provide comfortable interaction experience.

Despite numerous applications, existing gaze-tracking systems suffer from great system setup complexity, inflexible system configuration, the requirement of expensive devices (e.g., high quality cameras and lenses), cumbersome calibration procedures and low tolerance toward head movement,

which hinders them from being widely used. In particular, gaze tracking is challenging due to the individuality of eyes, occlusion, and variation in scale, head poses, and light conditions.

In this paper, we present a low-cost, non-intrusive, simple-setup gaze estimation system allowing free head movement. The proposed gaze estimation system uses a single consumer depth camera (Kinect), which is easy to setup even for nonprofessional users. Based on a 3D geometric model, the gaze estimation system allows free head movement. With robust facial feature extraction and simple system calibration, the system works in realtime with good accuracy, which can be used for many applications. Our main contributions are as follows:

- A simple-setup real-time gaze estimation system using only a consumer depth camera.
- A 3D model-based gaze estimation method allowing free head movement.
- A simple camera-screen calibration method that is easy to carry out even for nonprofessional users.
- An iris center localization method that can handle relatively low-quality eye images.
- Experimental results demonstrating that the system can estimate gaze directions accurately (error $1.4 \sim 2.7^\circ$) under free head movement.
- Two real time HCI applications validating the effectiveness and efficiency of the proposed technique.

2. Related Work

Most recent work on non-intrusive gaze tracking can be broadly classified into two categories: feature-based and appearance-based. In the following section, we briefly review each type of methods. For a detailed review on recent gaze-tracking techniques, we refer interested readers to [10].

2.1. Feature-Based Gaze Estimation

Feature-based gaze estimation methods typically rely on extracting local features such as pupil/iris contours, eye corners and corneal reflections (glints generated by IR lights), which are related to gaze. Feature-based approaches can be further divided into two distinctive groups: 2D regression-based (mapping-based) [12, 29, 1, 18, 30, 14, 32] and 3D model-based [25, 3, 8, 31, 2, 21]. 2D regression-based approaches focus on calibrating a gaze mapping function from the extracted 2D local eye movement features to the eye gaze (or PoR), whereas 3D model-based approaches calculate the 3D gaze direction by developing a 3D geometric model of the human eye.

For 2D regression-based gaze estimation methods, the most popular 2D local eye movement feature is the pupil center and corneal reflection (PCCR). Point of regard is estimated by tracking the relative position of PCCR, which are usually generated by dedicated IR lights. Many gaze-tracking systems [29, 1, 18, 30] are based on the PCCR technique, most of which require the subjects to keep their heads still. Although these systems could achieve very high accuracy (error less than 1°), the calibrated regression function decays as the head moves away from the original calibration position [18]. The recently developed 2D regression-based methods [29, 1, 18, 30, 31] attempt to address the problem by compensating for the errors caused by the head movement. In [31], Zhu and Ji introduced

a 2D regression-based method that incorporates 3D eye positions to compensate for the head movement. However, their method requires additional hardware. In [32], Zhu and Yang used the iris center and eye corner as regression variables. Its accuracy is in general lower than those of the PCCR systems, and the method is also sensitive to the head movement.

In contrast, 3D model-based gaze estimation methods enjoy the inherent advantage that the head movement is implicitly modeled by the 3D eye location. A 3D geometric eye model is used to determine the gaze direction for 3D model-based gaze-tracking systems, which is based on the anatomical structure of the eye [20] (see Fig. 1). Most 3D model-based methods require the calibration of the geometric relationships between the IR lights, the screen, and the camera. However, few work has described their calibration procedures. In [4], Francken et al. proposed a method to determine the screen's position and orientation using gray code reflections. The setup consists of an LCD screen, a digital camera, and a spherical mirror. The calibration procedure is done for two different sphere locations. In this paper, a simple screen-camera calibration method is described, which is easy to carry out for nonprofessional users.

In most 3D model-based gaze estimation methods, the gaze direction is estimated from the 3D cornea center (center of curvature of the cornea) and the pupil center. The glints generated by IR lights are usually used to derive the 3D cornea center, and the pupil/iris boundary is extracted from an image of the eye to find the 3D pupil/iris center. However, many 3D model-based gaze tracking systems are faced with the dilemma of trading off the head movement range for high-resolution eye images. Basically, a wide field of view is required to allow free head movement, but a narrow field of view is needed to obtain high-resolution eye images. There are generally two types

of approaches to solve this problem, either by using multiple cameras or using stereo and active cameras (cameras mounted on motorized pan and tilt units or cameras together with mirrors mounted on motorized pan and tilt units), both of which increase the complexity and cost of a gaze-tracking system.

Recently, methods using a single camera without IR lights have been proposed [28, 2]. In [28], Yamazoe et al. proposed to estimate gaze directions as 3D vectors connecting both the eyeball and the iris centers. However, their model didn't consider the deviation of the visual axis from the optical axis, and the distance between the eye and the camera was fixed and assumed to be known. The method achieved an accuracy of 5° horizontally and 7° vertically. In [2], Chen and Ji used a 3D eye model to determine the gaze direction, where the deviation angle between the visual axis and the optical axis is modeled. In their implementation, the distance between the eyeball center and the pupil center was fixed as the anthropomorphic average, and the iris center was manually labeled due to low-quality of the eye image. Their method estimated gaze with an accuracy of 2.18° horizontally and 2.53° vertically.

2.2. Appearance-Based Gaze Estimation

Unlike feature-based methods requiring the extraction of local gaze features, appearance-based gaze estimation methods use an entire eye image as a high-dimensional input feature and learn a mapping function from input directly to the screen coordinates, in the hope that some latent gaze features are implicitly modeled. These methods do not require screen-camera

[1]The figure is taken and modified from [20].

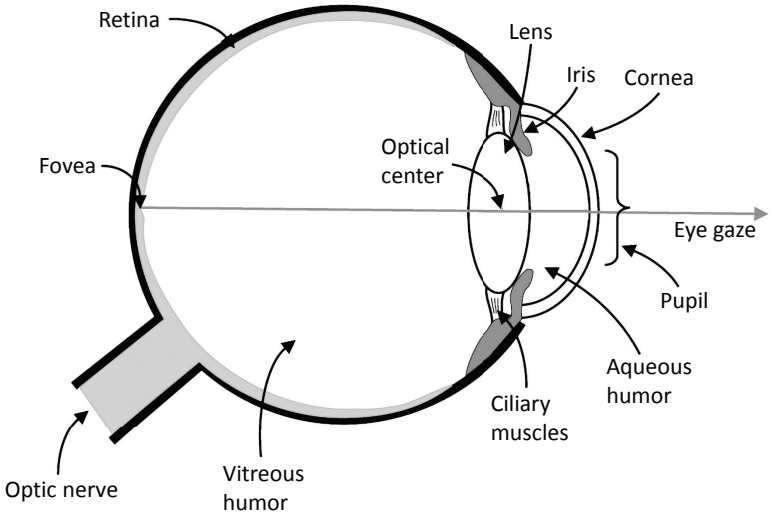

Figure 1: The anatomical structure of the eye[1].

calibration or gaze feature extraction. However, they usually require much more calibration points for personal calibration.

Various methods have been proposed to learn the mapping from an input eye image to screen coordinates, ranging from multilayer neural networks [27], Gaussian processes [26], manifold learning [23], adaptive linear regression [15, 5] and support vector regression [17]. Most of these methods involve a tedious personal calibration procedure to collect enough training data for each user (typically hundreds or even thousands of samples). Recently, Williams et al. [26] proposed to use a semi-supervised Gaussian process regression to reduce the number of labeled training samples, but their method still requires many unlabeled samples. In [15], Feng Lu et al. suggested using an adaptive linear regression on sparsely collected training samples.

Although some appearance-based gaze estimation methods [26, 23, 15] can achieve very high accuracy (error less than 1°), none of them has re-

ported head pose invariance because the appearance of the eye region alone is insufficient to determine the gaze direction since the region may look the same under different head poses. Moreover, the appearance of the eye region is sensitive to illumination variations.

2.3. Gaze Estimation with a Consumer Depth Camera

With the emerging technology of depth sensing, methods [5, 6, 11, 17] using consumer depth cameras (Kinect) have been proposed very recently. Mora and Odobez [5] proposed gaze estimation from multimodal Kinect data. In their method, a 3D face model (3D mesh) is built upon the multimodal Kinect data. Then the head pose and 3D mesh were used to create a frontal pose face image. The gaze direction in the head coordinate system was determined by Active Linear Regression on the eye appearance. Finally, the gaze vector is corrected according to the head pose. Although their method provides robust and accurate head pose tracking, the overall gaze estimation accuracy is relatively low (around 10°). Following their previous work, Mora and Odobez proposed in [6] a person indenpendent appearance-based gaze estimation method that used the coupling constraints between both eyes. The method achieved a gaze estimation accuracy comparable to [5] at lower computational cost. In [11], Jafari and Ziou used a Kinect to acquire head pose and a PTZ camera (characterized by controllable zoom and pointing direction) to obtain eye image. The gaze mapping function was learnt by the variational Bayesian multinomial logistic regression using head pose and eye features as input. The method allowed large head movement without personal calibration procedure. But the gaze estimation accuracy was relatively low (above 10°). In [17], Mansanet et al. proposed to estimate point of regard with a single consumer camera (HD webcam/Kinect

sensor). They took an appearance-based approach and learned the gaze mapping function by non-linear regression on Principal Component Analysis (PCA) coefficients of normalized pixel intensities of the eye image. The precision of the method was approximately 5°, but robustness against head movement was not reported.

In this paper, we propose to estimate gaze with a single Kinect sensor. Different from previous Kinect-based methods [5, 6, 11, 13, 17] that are either 2D regression-based or appearance-based, the proposed method is 3D model-based. Previous 3D model-based methods using a single camera without IR lights [28, 2] assume the distance between the eye and the camera or the distance between the eyeball center and pupil/iris center are fixed and known a priori. In the proposed method, the distance between the eye and the camera is estimated by the Kinect and is allowed to change, and the distance between the eyeball center and iris center is determined through personal calibration. To the best of our knowledge, the proposed method is the first 3D model-based gaze estimation method using Kinect sensor, and it achieves superior performance (accuracy of $1.4 \sim 2.7°$) among the methods using Kinect sensor.

3. 3D Model-based Gaze Estimation

As depicted in Fig. 2, the proposed gaze estimation system consists of three modules: gaze feature extraction, system calibration, and gaze estimation. In gaze feature extraction, 3D face tracking is first conducted to estimate the 3D location and orientation of a subject's head. Then face detection and eye detection are performed to obtain the local region of the subject's eye. Finally the iris center and the inner eye corner are located.

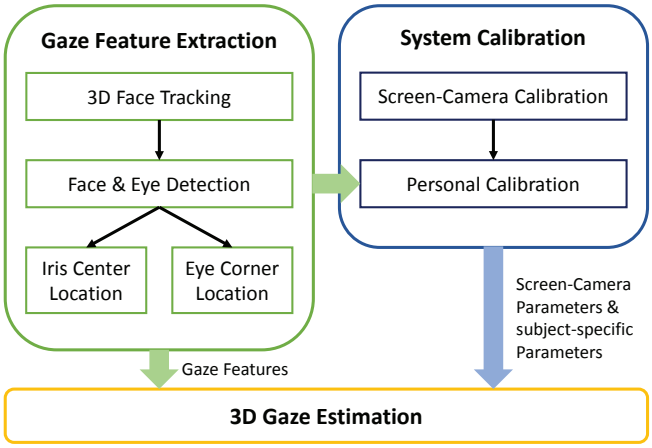

Figure 2: Framework of the proposed gaze estimation system.

The gaze features include: a) the 2D iris center and the inner eye corner coordinates in the image plane, and b) the head orientation and the z-coordinate of the inner eye corner in the camera coordinate system. The system calibration module consists of the person-independent screen-camera calibration, and the personal calibration determines the subject-specific parameters including the angle kappa (the deviation of the visual axis from the optical axis), the eyeball radius (the distance between the eyeball center and the iris center), and the eyeball center location relative to the head coordinate system.

The 3D gaze estimation module computes the 3D locations of the eyeball center and the iris center in the camera coordinate system, and determines the 3D gaze direction. By intersecting the 3D gaze direction with the screen, we obtain the point of regard on the screen.

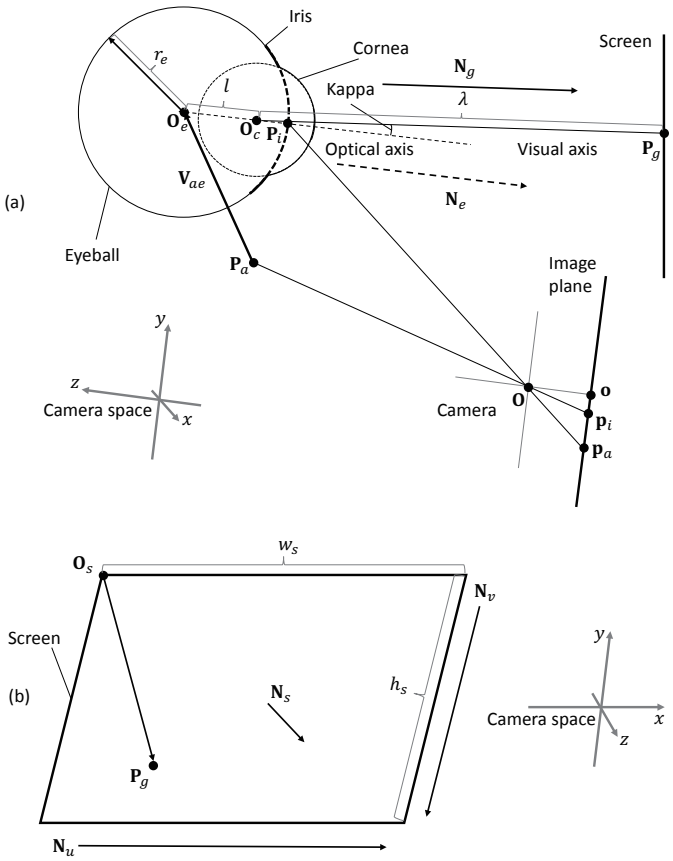

Figure 3: 3D model of the eye and a scene: (a) 3D eye model and its interactions with screen and camera, and (b) 3D point of regard on the screen.

3.1. 3D Eye Model and the Proposed Gaze Estimation Approach

In the proposed gaze estimation system, a 3D geometric eye model is used to determine the gaze direction (i.e., the visual axis), as shown in Fig. 3. In the 3D eye model, the eyeball is a sphere with a radius r_e and the cornea is a segment of an imaginary sphere with a smaller radius. The iris is located at the front of the eyeball, whose boundary is assumed to be a circle. The optical axis is defined by the line passing through eyeball center $\mathbf{O}_e$ (rotation center) and iris center $\mathbf{P}_i$. We denote the unit vector along the optical axis by $\mathbf{N}_e$. The visual axis is the line passing through cornea center $\mathbf{O}_c$ and the point on the screen $\mathbf{P}_g$ where the eye is actually looking at. The unit vector along the visual axis is denoted by $\mathbf{N}_g$. Notably, there is an angle of deviation of the visual axis from the optical axis, which is known as *angle kappa*. The angle kappa is subject-specific and requires personal calibration to determine its value.

As the subject changes gaze direction, the eyeball rotates around its center $\mathbf{O}_e$. Because $\mathbf{O}_e$ is inside the subject's face, we cannot directly estimate its position. However, the position of $\mathbf{O}_e$ is constant relative to the coordinate system of the subject's head. Let $\mathbf{Q}_h$ denote the origin of the head coordinate system. Let $\mathbf{X}_h$, $\mathbf{Y}_h$, $\mathbf{Z}_h$ denote the coordinate axes, and let $\mathbf{Q}_h^C$, $\mathbf{X}_h^C$, $\mathbf{Y}_h^C$, $\mathbf{Z}_h^C$ denote the coordinates of $\mathbf{Q}_h$, $\mathbf{X}_h$, $\mathbf{Y}_h$, $\mathbf{Z}_h$ in the camera coordinate system. Denote $\mathbf{R}_h^C = (\mathbf{X}_h^C, \mathbf{Y}_h^C, \mathbf{Z}_h^C)$. For any point $\mathbf{P}$ on the face, let $\mathbf{P}^H$ and $\mathbf{P}^C$ denote its coordinates in the head and camera coordinate systems, respectively. Then

$$\mathbf{P}^C = \mathbf{Q}_h^C + \mathbf{R}_h^C \mathbf{P}^H \ . \tag{1}$$

Denote $\mathbf{O}_e^C$ and $\mathbf{O}_e^H$ as the coordinates of $\mathbf{O}_e$ in the camera and head coor-

dinate systems, respectively. We have

$$\mathbf{O}_e^C = \mathbf{Q}_h^C + \mathbf{R}_h^C \mathbf{O}_e^H \ . \tag{2}$$

$\mathbf{O}_e^H$ does not change over time and can be calibrated. $\mathbf{Q}_h^C$ and $\mathbf{R}_h^C$ are obtained from head pose tracking. Because the head pose tracking of the Microsoft Kinect SDK is not very accurate, we detect the inner eye corner at a higher precision, and call it the anchor point, denoted as $\mathbf{P}_a$. Then

$$\mathbf{P}_a^C = \mathbf{Q}_h^C + \mathbf{R}_h^C \mathbf{P}_a^H \ . \tag{3}$$

From Equation (2–3),

$$\mathbf{O}_e^C - \mathbf{P}_a^C = \mathbf{R}_h^C (\mathbf{O}_e^H - \mathbf{P}_a^H) \ . \tag{4}$$

Denote $\mathbf{V}_{ae}^C = \mathbf{O}_e^C - \mathbf{P}_a^C$, and $\mathbf{V}_{ae}^H = \mathbf{O}_e^H - \mathbf{P}_a^H$, we have

$$\mathbf{V}_{ae}^C = \mathbf{R}_h^C \mathbf{V}_{ae}^H \ . \tag{5}$$

To simplify notations, we will omit the superscript C for the symbols $\mathbf{V}_{ae}^C$, $\mathbf{R}_h^C$, $\mathbf{O}_e^C$, $\mathbf{P}_a^C$, and simply use $\mathbf{V}_{ae}$, $\mathbf{R}_h$,$\mathbf{O}_e$,$\mathbf{P}_a$. Furthermore, all of the 3D coordinates (represented by bold uppercase letters, such as $\mathbf{O}_e$) refer to positions in the *3D camera space* (see Fig. 3), and all of the 2D coordinates (represented by bold lowercase letters, such as $\mathbf{p}_i$) refer to positions in the *image plane* or *screen plane*. As depicted in Fig. 3(a), the camera space origin $\mathbf{O}$ is projected onto the 2D image plane at $\mathbf{o}$, and $\mathbf{p}_i$, $\mathbf{p}_a$ are the projections of $\mathbf{P}_i$, $\mathbf{P}_a$, respectively. According to the pinhole camera model, the mathematical relationship between a 3D point $\mathbf{P} : (x, y, z)$ and its projection $\mathbf{p}$ onto an image plane is given by

$$\mathbf{p} - \mathbf{o} = \frac{f}{z} \begin{bmatrix} x \\ -y \end{bmatrix} \tag{6}$$

where $\mathbf{o}$ is the projection of the camera origin[2] and f is focal length of the camera in pixels. For the anchor point $\mathbf{P}_a$, we have

$$\mathbf{p}_a - \mathbf{o} = \frac{f}{z_a} \begin{bmatrix} x_a \\ -y_a \end{bmatrix} . \tag{7}$$

Again, the relationship between the iris center and its projected position in image plane is given by

$$\mathbf{p}_i - \mathbf{o} = \frac{f}{z_i} \begin{bmatrix} x_i \\ -y_i \end{bmatrix} . \tag{8}$$

Considering the eyeball as a sphere with radius r_e, the unit vector along the optical axis is

$$\mathbf{N}_e = \frac{\mathbf{P}_i - \mathbf{O}_e}{r_e} . \tag{9}$$

As mentioned previously, the visual axis deviates from the optical axis by a fixed angle known as *angle kappa*. By applying the angle kappa to the optical axis, the unit vector along the visual axis can be written as

$$\mathbf{N}_g = \mathbf{R}(\alpha, \beta)\mathbf{N}_e \tag{10}$$

where α, β are the horizontal component and vertical component of angle kappa, respectively, and $\mathbf{R}(\alpha, \beta)$ is the corresponding rotation matrix.

As shown in Fig. 3(a), in the proposed model, the 3D point of regard is determined by the intersection point of the visual axis and the screen. As shown in Fig. 3(b), the screen is considered as a 2D rectangular plane $w_s \times h_s$ with unit normal vector $\mathbf{N}_s$ and top-left corner $\mathbf{O}_s$. For any point $\mathbf{P}_s$ on the screen, $\mathbf{N}_s \cdot \mathbf{P}_s = -d$ (d is a parameter which can be determined

[2]Note that $\mathbf{o}$ is the origin of the image plane with the $\mathbf{X}$ axis from left to right and the $\mathbf{Y}$ axis from top to bottom.

from the camera-screen calibration). Because $\mathbf{P}_g$ is on the plane, we have $\mathbf{N}_s \cdot \mathbf{P}_g = -d$. Finally, as shown in Fig. 3(a), the 3D PoR $\mathbf{P}_g$ can be obtained by adding the two directed line segments as follows:

$$\mathbf{P}_g = \mathbf{O}_e + l\mathbf{N}_e + \lambda\mathbf{N}_g \tag{11}$$

where l is the distance between eyeball center $\mathbf{O}_e$ and cornea center $\mathbf{O}_c$, and λ is the distance between cornea center $\mathbf{O}_c$ and PoR $\mathbf{P}_g$, that is,

$$\lambda = \|\mathbf{O}_c\mathbf{P}_g\|_2 = -\frac{(\mathbf{O}_c - \mathbf{P}_g)\cdot\mathbf{N}_s}{\mathbf{N}_g\cdot\mathbf{N}_s} = -\frac{(\mathbf{O}_e + l\mathbf{N}_e)\cdot\mathbf{N}_s + d}{\mathbf{N}_g\cdot\mathbf{N}_s}\ . \tag{12}$$

Note that l is typically very small, approximately $5.3mm$ [8]. In practice, we ignore the $l\mathbf{N}_e$ term and rewrite Equation (11–12) as

$$\mathbf{P}_g = \mathbf{O}_e + \lambda\mathbf{N}_g \tag{13}$$

$$\lambda = -\frac{\mathbf{O}_e\cdot\mathbf{N}_s + d}{\mathbf{N}_g\cdot\mathbf{N}_s}\ . \tag{14}$$

Given the unit vectors $\mathbf{N}_u$ and $\mathbf{N}_v$ along the horizontal border and the vertical border, respectively, the relationship between 3D PoR $\mathbf{P}_g$ and its projection $\mathbf{p}_g : (u_g, v_g)$ on the 2D screen plane can be derived as follows:

$$\begin{cases} (\mathbf{P}_g - \mathbf{O}_s)\cdot\mathbf{N}_u - \frac{w_s u_g}{w_r} = 0 \\ (\mathbf{P}_g - \mathbf{O}_s)\cdot\mathbf{N}_v - \frac{h_s v_g}{h_r} = 0 \end{cases} \tag{15}$$

where $w_r \times h_r$ is the resolution of the screen.

3.2. Proposed Gaze Estimation Algorithm

Based on the foregoing discussion, our approach to 3D gaze estimation is: 1) computing the 3D positions of $\mathbf{O}_e$ and $\mathbf{P}_i$ to obtain the optical axis; 2) adding angle kappa to the optical axis to obtain the visual axis; 3) intersecting the visual axis with the screen to obtain the point of regard.

Several unknown constant parameters need to be determined beforehand for 3D gaze estimation. These parameters are either system-specific or subject-specific, both of which can be determined by calibration. In particular, by screen-camera calibration, we obtain the size of the screen $w_s \times h_s$ in mm, top-left corner $\mathbf{O}_s$, and the unit normal $\mathbf{N}_s$ vector of the screen, unit vector $\mathbf{N}_u$ along the horizontal screen border, unit vector $\mathbf{N}_v$ along the vertical screen border, the resolution of the screen $w_r \times h_r$ in pixels, the focal length f of the camera in pixels, and the projected position $\mathbf{o}$ of the camera origin. Through personal calibration, the eyeball radius r_e, offset vector $\mathbf{V}_{ae}^H$ and deviation angles (α, β) are determined. To estimate 3D gaze direction, we extract several gaze features including head rotation matrix $\mathbf{R}_h$, depth value z_a of $\mathbf{P}_a$, and 2D image positions $\mathbf{p}_i$ and $\mathbf{p}_a$.

The proposed gaze estimation algorithm is summarized in Algorithm 1.

Algorithm 1 The proposed gaze estimation algorithm.

Require:

$\mathbf{R}_h, \mathbf{p}_i, \mathbf{p}_a, z_a, w_s, h_s, \mathbf{O}_s, \mathbf{N}_s, \mathbf{N}_u, \mathbf{N}_v, d, w_r, h_r, f, \mathbf{o}, r_e, \mathbf{V}_{ae}^H, \alpha, \beta$

Ensure:

$\mathbf{P}_g, \mathbf{p}_g$

1: Obtain $\mathbf{P}_a$ by using Equation (7);
2: Calculate $\mathbf{V}_{ae}$ using Equation (5), and obtain $\mathbf{O}_e$ using Equation (4);
3: Solve Equation (8) together with $\|\mathbf{P}_i - \mathbf{O}_e\|_2 = r_e$, choose the unique solution that satisfies $z_i < z_e$;
4: Compute $\mathbf{N}_e$ using Equation (9) and then $\mathbf{N}_g$ using Equation (10);
5: Obtain $\mathbf{P}_g$ in the camera space using Equation (13–14);
6: Obtain $\mathbf{p}_g$ on the screen plane using Equation (15).

4. Gaze Feature Extraction

In the proposed method, some gaze features are required to determine the 3D gaze direction, including: a) head rotation matrix $\mathbf{R}_h$, b) iris center position $\mathbf{p}_i$ on the image plane, c) anchor point position $\mathbf{p}_a$ on the image plane and d) depth value z_a of the anchor point $\mathbf{P}_a$ in the camera space.

The Kinect sensor that we use is quite convenient for collecting data for gaze feature extraction. The head rotation matrix $\mathbf{R}_h$ is provided by the face-tracking SDK[3]. We use a technique described in Section 4.2 to detect the inner eye corner location on the RGB image, and use its depth value to obtain its 3D position in the camera space.

At this point, we can obtain the head rotation matrix $\mathbf{R}_h$ and depth value z_a of the anchor point $\mathbf{P}_a$. We next discuss face and eye detection, and iris center and eye corner localization to extract the remaining gaze features.

4.1. Face and eye detection

Before locating the iris center and eye corner in the image plane, face and eye detection are first performed to find the eye region. In the proposed gaze estimation system, visual context boosting [22] is used for eye detection. As proposed in [22], the eye is modeled by its surrounding visual context using a visual context pattern (VCP), which describes the relation between the region of the eye and its region of reference both in space and appearance. The context feature (encoded VCP) is integrated with Haar-like features,

[3]Microsoft Kinect SDK, `http:/www.microsoft.com/en-us/kinectforwindows/develop/`, 2013.

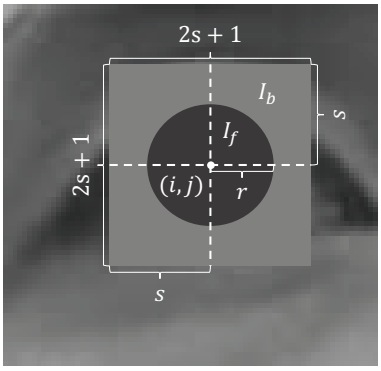

Figure 4: Iris template for iris detection.

followed by semi-supervised boosting to construct a classifier for precise eye detection.

4.2. Iris center and eye corner location

Iris center position on the image plane is a very important gaze feature for the proposed gaze estimation method. The most popular iris center localization method fits an eclipse to the extracted iris contour. However, because the eye image captured by the Kinect sensor is of low contrast and low resolution, edge-based iris center location methods tend to work quite poorly.

To address this problem, we present an iris-fitting algorithm based on a parameterized iris model. Although the iris contour may be blurry, the intensity of the iris region is lower than that of the rest of eye, and it is thereby natural to consider the local contrast between the pixels inside and outside the iris region. In particular, the proposed iris model assumes the iris is a circular area that is darker than its surrounding area. Alternatively, one could use an ellipse instead of a circle to fit the iris region. We found in our experiments that fitting an ellipse is not as robust as fitting a circle

when there are occlusions.

As depicted in Fig. 4, we scan the detected eye region with a square window of iris template and find the iris center and radius that minimizes the discrepancies between the image and the iris model, which is given by

$$(i^*, j^*, r^*) = \arg\min_{i,j,r} \sum\nolimits_{m\in[i-s,i+s],n\in[j-s,j+s]} f_c(i, j, r, m, n, \mathbf{I}) \tag{16}$$

where s is a constant determining the size of the sliding window, (i, j, r) is the hypothesized iris center and radius, and f_c is the cost of individual pixels in the sliding window. Intuitively, the iris template is based on two observations: the iris pixels form a circular region, and the iris pixels (foreground pixels) are darker than other pixels (background pixels) in the eye region. Then the cost function for a pixel inside the sliding window is defined by

$$f_c(i, j, r, m, n, I) = \begin{cases} \max(I_{mn} - I_f, 0), & (m-i)^2 + (n-j)^2 \le r^2 \\ \max(I_b - I_{mn}, 0), & (m-i)^2 + (n-j)^2 > r^2 \end{cases} \tag{17}$$

where I_f, I_b are the intensity threshold for the foreground pixels (inside the iris circle) and that for the background pixels (outside the iris circle), respectively, and I_{mn} is the intensity value of pixel (m, n) of the image $\mathbf{I}$. Particularly, the cost function favors pixels inside the hypothesized iris region with lower intensity than I_f, and outside pixels with higher intensity than I_b. In our implementation, I_f is set using the first peak location of the intensity histogram of the eye image, and $I_b = I_f + 35$, and $s = 18$. As shown in Fig. 5, the iris location algorithm operates quite well on low-resolution, low-contrast eye images.

Given the region of the eye obtained by eye detection, we locate the eye corner by template filtering, as proposed in [32]. Specifically, a 2D convolution is performed on the potential area of the eye corner with a

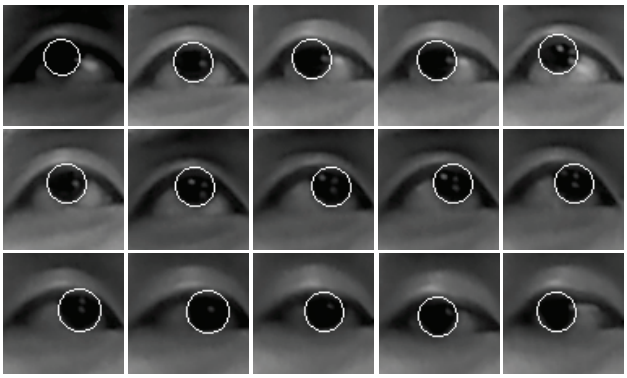

Figure 5: Examples of iris center location.

pre-set corner filter, and the eye corner is then determined to be the pixel location with the maximum filter response.

5. System Calibration

For the proposed gaze estimation system, system calibration determines the geometric relationship between the screen and the camera, and the subject-specific parameters. It can be generally divided into two steps: screen-camera calibration and personal calibration.

5.1. Screen-camera calibration

Screen-camera calibration mainly determines the size, location and orientation of the screen in the camera space. We assume that the intrinsic camera parameters (focal lengths of the depth camera and the RGB camera of the Kinect sensor, the transformation matrix from depth camera space to RGB camera space, projected position of the camera origin on the image plane, and the size of the image plane) are known. These parameters can be easily retrieved from Kinect's face-tracking SDK.

A stick (approximately $1800mm$ in length and $10mm$ in radius) is used to point to the screen corner. The end of the stick that is close to the screen cannot be seen by the camera, but the far end of the stick is visible. We then fit a 3D line to the 3D point cloud of the stick.

By keeping the stick pointed at the screen corner and changing its direction, a different 3D line going through the screen corner can be determined. By repeating the process several times, many 3D lines that intersect with the screen corner can be generated. The 3D position of the screen corner is the intersection point (point with the smallest square sum of distances to all of the lines) of those 3D lines. In this way, we can determine all of the four screen corners. From the four screen corners, we obtain the following parameters: a) the size of the screen $w_s \times h_s$ in the camera space, b) the top-left corner $\mathbf{O}_s$, and unit normal vector $\mathbf{N}_s$ of the screen and c) the unit vector $\mathbf{N}_u$ along the direction from the top-left corner of the screen to the top-right corner and the unit vector $\mathbf{N}_v$ along the direction from top-left corner to the bottom-left corner. The overall algorithm is summarized in Algorithm 2.

Although the depth information obtain from the Kinect sensor is not very accurate, the proposed screen-camera calibration works quite well with an accuracy comparable to the results in [4]. In fact, for a screen measuring $376 \times 301mm$, the calibrated dimensions are $371.3 \times 295.7mm$, suggesting an accuracy of up to a few millimeters. Moreover, the entire screen-camera calibration procedure takes only a few minutes (usually $5 \sim 8$ minutes), and all of the manual interventions are easy to carry out and do not require special skill or knowledge. Thus, the nonprofessional users can perform the screen-camera calibration easily.

Algorithm 2 Screen-camera calibration algorithm.

Require:

Intrinsic parameters of the Kinect sensor

Ensure:

$w_s, h_s, \mathbf{O}_s, \mathbf{N}_s, \mathbf{N}_u, \mathbf{N}_v, d$

1: Collect depth map of a stick pointed at the screen corner, obtain the 3D point cloud of the stick and fit a 3D line to the cloud points;

2: Transform the 3D line from the depth camera space to the camera (RGB camera) space;

3: Change the direction of the stick and repeat 1–2 several times;

4: Compute the position of the screen corner by finding the optimal intersection point of the set of 3D lines;

5: Repeat 1–4 until all screen corners are located in the camera space;

6: Fit a rectangle to the 3D points of the screen corners, and obtain $w_s, h_s, \mathbf{O}_s, \mathbf{N}_s, \mathbf{N}_u, \mathbf{N}_v, d$.

5.2. Personal calibration

In personal calibration, the subject is asked to look at N calibration points on the screen one at a time, and each calibration point is looked at with M different head poses. Personal calibration is essential for the proposed gaze estimation system to determine the subject-specific parameters, including a) eyeball radius r_e, b) offset vector $\mathbf{V}_{ae}^{H}$ from anchor point $\mathbf{P}_a$ to eyeball center $\mathbf{O}_e$ in the frontal face, and c) deviation angle (α, β) of the visual axis from the optical axis. These subject-specific parameters cannot be estimated directly. Instead, by examining the geometric relationship of the eye and the scene in the 3D model, the subject-specific parameters can be determined by solving a linear system of equations.

During personal calibration, the subject looks at a target point (calibration point) on the screen. Because the 2D position of the target point $\mathbf{p}_t : (u_t, v_t)$ on the screen plane is known, the corresponding 3D position $\mathbf{P}_t$ can be recovered by the following expression:

$$\mathbf{P}_t = \mathbf{O}_s + \frac{u_t w_s}{w_r}\mathbf{N}_u + \frac{v_t h_s}{h_r}\mathbf{N}_v \ . \tag{18}$$

The 3D iris center $\mathbf{P}_i$ is obtained by Equation (8), where the depth value z_i is estimated by using the Kinect SDK. Then, the ground truth unit vector $\mathbf{N}_g$ in the direction of the visual axis is calculated by

$$\mathbf{N}_g = \frac{\mathbf{P}_g - \mathbf{P}_i}{\|\mathbf{P}_g - \mathbf{P}_i\|_2} \tag{19}$$

where $\mathbf{P}_g$ is estimated using Equation (18) with ground truth position $\mathbf{p}_t$. Assuming that the deviation angle (α, β) is known, the unit vector along optical axis can be obtained by

$$\mathbf{N}_e = \mathbf{R}(-\alpha, -\beta)\mathbf{N}_g \ . \tag{20}$$

We can split the rotation matrix into three row vectors as $\mathbf{R}_h = \begin{bmatrix} \mathbf{R}_x^T; \mathbf{R}_y^T; \mathbf{R}_z^T \end{bmatrix}$. Finally, we obtain two linear equations using the mathematic relationship (see Equation (6)) between the 3D iris center and its projection on the image plane, which are expressed as follows:

$$\begin{cases} f \cdot (x_a + \mathbf{R}_x \cdot \mathbf{V}_{ae}^H + r_e \mathbf{N}_e \cdot \mathbf{V}_x) + (u_0 - u_i)(z_a + \mathbf{R}_z \cdot \mathbf{V}_{ae}^H + r_e \mathbf{N}_e \cdot \mathbf{V}_z) = 0 \\ f \cdot (y_a + \mathbf{R}_y \cdot \mathbf{V}_{ae}^H + r_e \mathbf{N}_e \cdot \mathbf{V}_y) + (v_i - v_0)(z_a + \mathbf{R}_z \cdot \mathbf{V}_{ae}^H + r_e \mathbf{N}_e \cdot \mathbf{V}_z) = 0 \end{cases} \tag{21}$$

where $\mathbf{V}_x = (1, 0, 0)^T$, $\mathbf{V}_y = (0, 1, 0)^T$, $\mathbf{V}_z = (0, 0, 1)^T$ and $\mathbf{P}_a : (x_a, y_a, z_a)$ is the 3D position of the anchor point.

For N calibration points, we have $2MN$ equations with 6 unknown subject-specific parameters

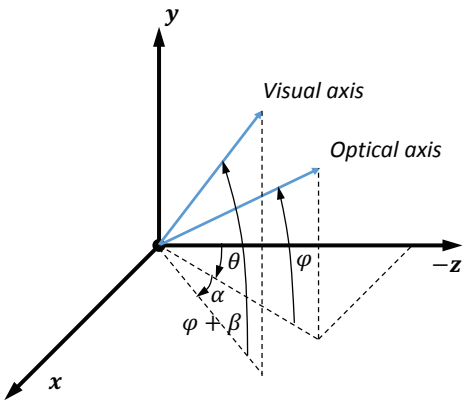

Figure 6: Direction of optical axis and visual axis in camera space.

r_e, $\mathbf{V}_{ae}^H$, α, β as follows:

$$\begin{cases} f \cdot \left(x_a(n) + \mathbf{R}_x(n) \cdot \mathbf{V}_{ae}^H + r_e\left(\mathbf{R}(-\alpha,-\beta)\,\mathbf{N}_g(n)\right) \cdot \mathbf{V}_x\right) + \\ (u_0 - u_i(n)) \left(z_a(n) + \mathbf{R}_z(n) \cdot \mathbf{V}_{ae}^H + r_e\left(\mathbf{R}(-\alpha,-\beta)\,\mathbf{N}_g(n)\right) \cdot \mathbf{V}_z\right) = 0 \\ f \cdot \left(y_a(n) + \mathbf{R}_y(n) \cdot \mathbf{V}_{ae}^H + r_e\left(\mathbf{R}(-\alpha,-\beta)\,\mathbf{N}_g(n)\right) \cdot \mathbf{V}_y\right) + \\ (v_i(n) - v_0) \left(z_a(n) + \mathbf{R}_z(n) \cdot \mathbf{V}_{ae}^H + r_e\left(\mathbf{R}(-\alpha,-\beta)\,\mathbf{N}_g(n)\right) \cdot \mathbf{V}_z\right) = 0 \end{cases} \tag{22}$$

where $x_a(n)$, $y_a(n)$, $y_a(n)$, $\mathbf{R}_x(n)$, $\mathbf{R}_y(n)$, $\mathbf{R}_z(n)$, $\mathbf{N}_g(n)$ vary across the MN observations. In our implementation, we set $M = N = 5$.

To solve Equation (22), we use the iterative algorithm summarized in Table 3. Note that when (α, β) are known and fixed, Equation (22) becomes a system of $2MN$ linear equations, which can be solved easily to obtain r_e, $\mathbf{V}_{ae}^H$. When r_e, $\mathbf{V}_{ae}^H$ are known and fixed, $\mathbf{N}_e(n)$ can be readily computed, and (α, β) can then be obtained using $\mathbf{N}_e(n)$ and $\mathbf{N}_g(n)$.

As shown in Fig. 6, the direction of the optical axis can be divided into two components: a horizontal angle θ and a vertical angle ϕ. The direction of the visual axis can also be represented by a horizontal angle $\theta+\alpha$ and a vertical angle $\phi+\beta$. As in [8], the mathematical relationship between these angles

and directions can be expressed by $\mathbf{N}_e = [\cos\phi \sin\theta, \sin\phi, -\cos\phi\cos\theta]^T$ and $\mathbf{N}_g = [\cos(\phi+\beta)\sin(\theta+\alpha), \sin(\phi+\beta), -\cos(\phi+\beta)\cos(\theta+\alpha)]^T$. Then, (α, β) can be obtained from $\mathbf{N}_e$ and $\mathbf{N}_g$ by

$$\begin{cases} \alpha = \arccos\frac{x_{ng}}{\sqrt{1-y_{ng}^2}} - \arccos\frac{x_{ne}}{\sqrt{1-y_{ne}^2}} \\ \beta = \arcsin y_{ng} - \arcsin y_{ne} \end{cases} . \tag{23}$$

Algorithm 3 Personal calibration algorithm.

Require:

MN observations of $x_a(n)$, $y_a(n)$, $z_a(n)$, $\mathbf{R}_x(n)$, $\mathbf{R}_y(n)$,$\mathbf{R}_z(n)$, $\mathbf{N}_g(n)$

Ensure:

r_e, $\mathbf{V}_{ae}^H$, α, β

1: Initialize $\alpha = 0$, $\beta = 0$;

2: Compute $\mathbf{R}(-\alpha, -\beta)$, and solve Equation (22) for r_e, $\mathbf{V}_{ae}^H$;

3: Compute $\mathbf{N}_e(n)$ using Equation (4–5,7–9);

4: Obtain deviation angle (α', β') using Equation (23);

5: Set $\alpha = \alpha'$, $\beta = \beta'$, and repeat 2–4 to convergence;

6: **return** r_e, $\mathbf{V}_{ae}^H$, α, β.

6. Experimental Evaluation and Application

6.1. Experimental setup

The proposed gaze estimation system consists of a Kinect senor located in front of the bottom border of a computer screen, as shown in Fig. 7. The RGB image resolution is 1280 × 960 pixels, with the eye region of approximately 70 pixels in width. The resolution of the eye image is very low compared to those used by many existing gaze estimation systems [31], where the width of eye region is typically greater than 150 pixels.

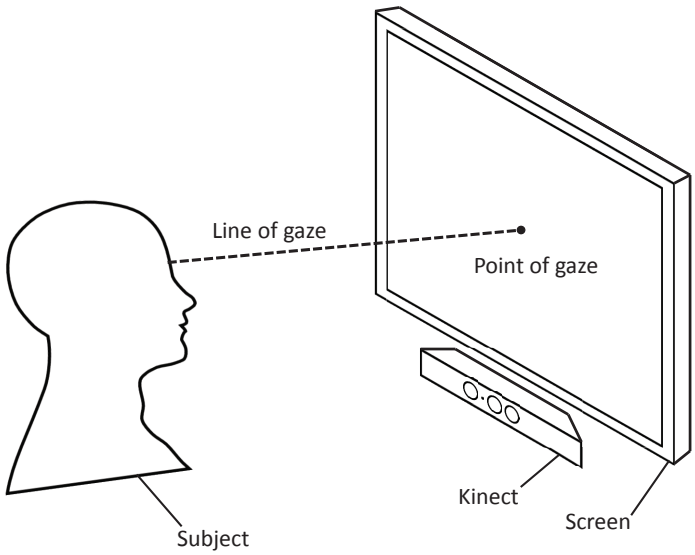

Figure 7: System setup of the proposed 3D gaze estimation system.

To evaluate the performance of the proposed system, we conduct experiments on 8 subjects. The gaze estimation accuracy is represented by gaze estimation error (the angle between the estimated gaze direction and the ground truth gaze direction). The ground truth gaze direction is gathered by computing the line through the 3D iris center and the calibration point on the screen. During personal calibration and testing, the subject moves his/her head purposely within a space volume of approximately $300 \times 300 \times 200mm$ ($width \times height \times depth$) at about $550mm$ from the Kinect sensor.

6.2. Experimental result and analysis

We report the gaze estimation accuracy and calibrated subject-specific parameters for the 8 subjects in Table 1. As shown in Table 1, the mean gaze estimation error ranges from 1.77° to 2.71°. Generally, the calibrated

Table 1: Mean gaze estimation error for 8 subjects.

Subject	Mean error (degree)			Eyeball
	3D	Horizontal	Vertical	radius r_e
1	1.9550	1.0423	1.4832	11.5
2	2.3306	1.2915	1.7116	10.1
3	2.7053	1.4740	1.9776	8.9
4	1.8105	0.9771	1.2289	10.8
5	1.7701	1.0471	1.1999	11.6
6	2.0713	1.0601	1.6062	11.8
7	2.1671	1.1666	1.5874	9.77
8	2.4676	1.6309	1.5903	9.3

eyeball radius closer to the real value indicates higher gaze estimation accuracy. Note that the human eyeball radius is approximately $11 \sim 13mm$, the calibrated subject-specific parameters of Subject 1,4,5,6 are closer to the true values, thus yielding more accurate gaze estimates. It is worth mentioning that the evaluation is performed under free head movement, for each target point, the subject looks at it with 5 different head poses. Moreover, each estimate is the result of averaging over multiple frames, as the depth and head pose information obtained from the Kinect sensor are relatively jittery.

To investigate the effect of different setup configurations on the gaze estimation accuracy, we test on Subject 1 with different system setups. We have considered two factors of the system setup: a) the size of the screen and b) the elevation angle of the Kinect sensor. As shown in Table 2, there is no significant change in the gaze estimation accuracy among different screen sizes. However, with a smaller elevation angle of the Kinect sensor, the sys-

Table 2: Gaze estimation accuracy for Subject 1 under different system setups.

Setup	Mean error (degree)			Setup features
	3D	Horizontal	Vertical	
1	1.9550	1.0423	1.4832	Elevation angle $\sim 19^\circ$, 19" screen
2	2.1210	1.1774	1.5951	Elevation angle $\sim 19^\circ$, 22" screen
3	1.3817	0.9508	0.8235	Elevation angle $\sim 9^\circ$, 19" screen

tem is able to achieve a gaze estimation accuracy of up to 1.38°. Generally, a small elevation angle of the Kinect sensor is preferred. However, an excessively small evaluation angle will lead to a small visible iris area due to the occlusion of eyelids, which will create difficulties in gaze feature extraction (e.g., iris center location) and degrade the gaze estimation performance.

To evaluate the robustness of the proposed gaze estimation system against head movements, we examine the performance under different head poses. Two kinds of head movements have been considered: head translation and head rotation. For head translation, the position of the subject's head is first set at $(0, 0, 0.53)$ (in meters) in the 3D camera space. Then, with a fixed head orientation subject's head moves along the X axis of the camera space and gaze estimation accuracy is evaluated at five different positions: $x = -0.1, -0.05, 0, 0.05, 0.1$. Similar procedures are repeated for head translations along the Y/Z axis of the camera space. For head rotation, we evaluate the performance under different head orientations with fixed head position at $(0, 0, 0.53)$. Note that when the subject normally looks at the screen with frontal face, the head rotation angles are about $(pitch, yaw, roll) = (10, 0, 0)$ (in degrees), as the Kinect sensor has an angle of elevation.

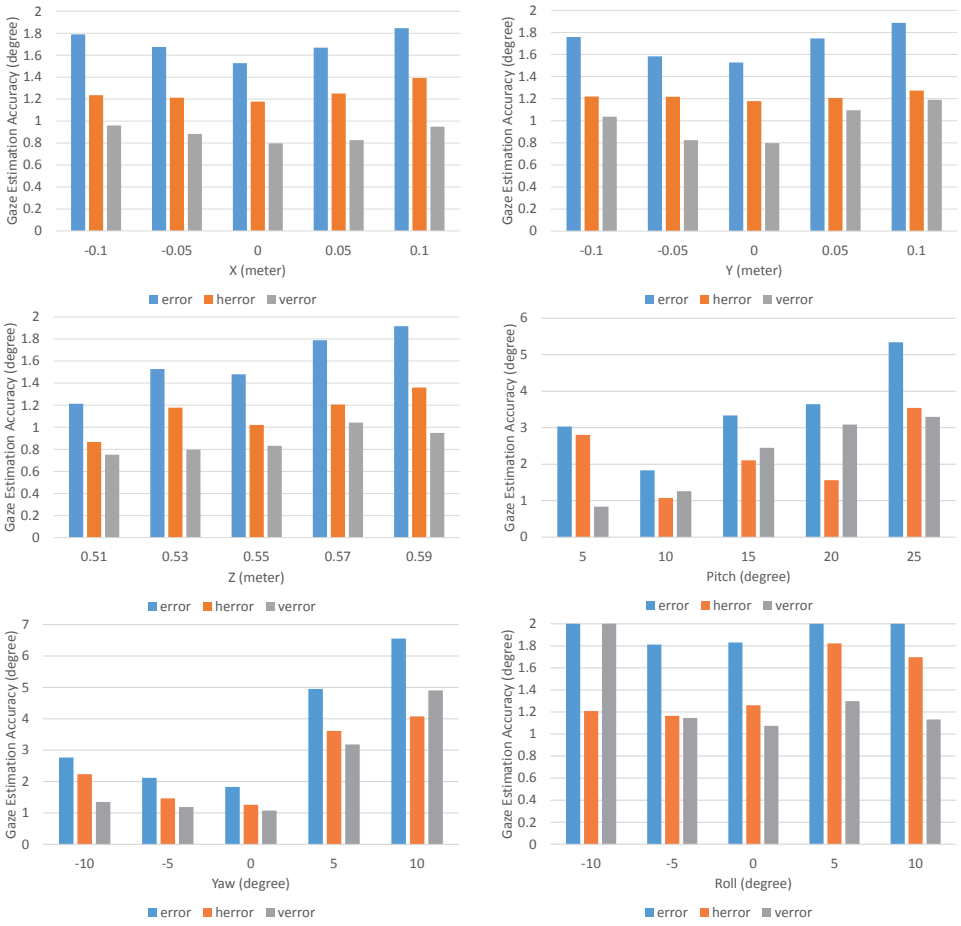

Figure 8: Gaze estimation error under different head poses.

As depicted in Fig. 8, the proposed system exhibits robustness against head movement, especially for head translation and head roll. Specifically, when the subject's head moves along the X/Y axis, the gaze estimation accuracy does not change much (generally within $0.3°$). Better performance is observed when the head is near the central position $(0, 0, 0.53)$, which is mainly because the iris is less likely to be occluded by the eye lids when the subject looks at the screen in such condition. As the subject's head moves away from the camera along the Z axis, the gaze estimation error gradually increases because the captured eye images become smaller. However, the performance degrades dramatically when the head rotation angles of pitch/yaw go away from the frontal face position. This is due to that head pitch/yaw usually changes the appearance of the eye a lot, and the eye feature extraction is affected in turn. In general, the proposed gaze estimation system works well under natural head movements.

We compare the proposed gaze estimation system with several recently developed gaze estimation systems, including ones using Kinect sensor [5, 6, 11, 17] and ones using single camera without IR lights [2, 28]. Table 3 summarizes the performance of our proposed and the related gaze estimation systems, concerning gaze estimation accuracy, setup, running speed and robustness against head movements. Note that there was no clear report in [2, 28] on the robustness against head movements, and in [28] the distance between the eye and the camera is fixed and known. Moreover, all existing gaze estimation systems using Kinect sensor [5, 6, 11, 13, 17] took an appearance-based approach. Unlike them, the proposed gaze estimation system is 3D model-based. In general, the proposed gaze estimation system achieves superior gaze estimation accuracy with simple setup, real-time running speed and robustness against head movements among the

Table 3: Comparison with other gaze estimation systems.

Methods	Average accuracy (degree)	Features
ours	$1.38 \sim 2.71$	1 Kinect sensor, 12fps, robust against head translation and roll
Chen2008[2]	Horizontal 2.18, vertical 2.53	single camera
Yamazoe2008[28]	Horizontal 5.3, vertical 7.7	single camera, 10fps
Mora2012[5]	$7.6 \sim 12.6$	1 Kinect sensor, robust against head rotation
Mora2013[6]	$7.6 \sim 14.5$	1 Kinect sensor, robust against head rotation, person independent
Jafari2012[11]	above 10	1 Kinect sensor and 1 PTZ camera, robust against head translation
Mansanet2013[17]	about 5	1 Kinect sensor

gaze estimation systems using Kinect sensor or single camera without IR lights. Although there exist many other gaze estimation systems that can operate with very high accuracy (less than 1°), they either require complex and expensive setups (e.g., stereo systems, zoomable lenses, high-resolution cameras, IR lights, mirrors and pan-tilt units) or need to constrain the allowable head movement to be very small, which dramatically limits their applications.

6.3. Example applications for HCI

To investigate the feasibility of using the proposed gaze estimation system for HCI applications, we have developed a real-time 3D chess game and a screen keyboard using the estimated gaze to drive the cursor. Because gaze usually reflects a person's intention, it is quite natural to use gaze as

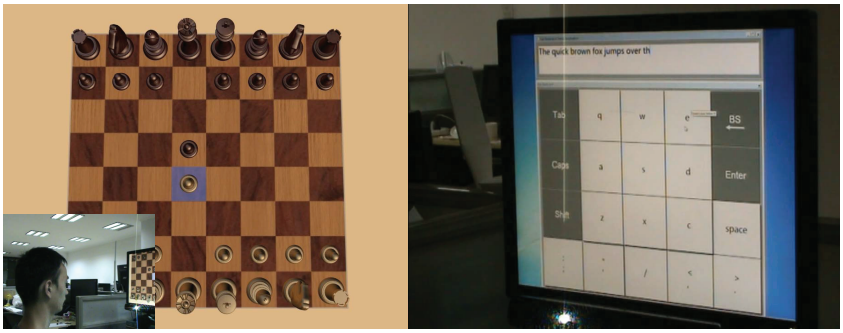

Figure 9: A snapshot of the 3D chess game and Eye Keyboard based on gaze estimation. Demo videos are available at `http://youtu.be/rj8HkUThrm4` and `http://youtu.be/Tq-Pj-5xcyU`.

an HCI interface. In the 3D chess game application, the cursor is driven by the estimated gaze, and all the moves of the chess figures and chessboard are conducted by directing the gaze along the desired direction. The pick-up and put-down of the chess pieces are triggered by hitting the space bar. Fig. 9 shows a snapshot of the 3D chess game. In the screen keyboard application, we use the gaze to drive the cursor and use eye blinks to trigger mouse click events. It is completely hand free. The central 3×3 block is a face on a 3D cube, which can be rotated to show other faces with different keys. In this way, nearly all the common characters and symbols can be input through this interface. Please refer to the supplemental video clips for more details. This application provides a feasible way for a disabled person to enter text.

In the 3D chess game application each cell of the chessboard is a square of side length approximately $35mm$, corresponding to a visual angle of about $3.3°$. In both HCI applications, we average over 8 frames to yield stable estimates, as the Kinect data is relatively jittery. However, this will lead to noticeable latency during the undergoing of eye movements and head move-

ments. The proposed gaze estimation system and the two HCI applications are implemented on a PC with 4GB memory, an Intel Core2 Quad CPU Q9550 2.83GHz and a GT310 graphic adapter. The computational costs of gaze feature extraction and gaze estimation are $29.51ms$ and $0.64ms$ respectively. We can see that nearly all the computational time is spent on feature extraction. Specifically, the gaze estimation is running at $12fps$ in both HCI applications, which is limited by the fact that the Kinect data stream arrives at $12fps$. Considering the accuracy, the running speed, the ease of calibration procedures, the robustness against head movements and the widespread of Kinect sensors, we believe that the proposed technique can find many applications.

7. Conclusion

In this paper, we present a novel low-cost, non-intrusive, simple-setup gaze estimation system operating under natural head movements. Instead of IR lights and high-quality cameras, a consumer depth camera (Kinect sensor) is used. Based on a 3D model of the eye, the gaze direction is determined by computing the 3D position of the eyeball center and iris center. To extract gaze features from low quality eye images, a new parameterized iris model is used to locate the iris center. Before using the system, a simple and effective screen-camera calibration procedure is conducted to determine the geometric relationship between the screen and the camera, and personal calibration is performed to determine the subject-specific parameters. Experimental results show that the proposed system can operate with fairly high accuracy (error $1.4 \sim 2.7^\circ$) under natural head movements. We have developed two real time application systems. One uses the eye gaze tracking

to play chess, and the other uses the eye gaze tracking to enter text. These working prototypes validate the robustness our proposed gaze tracking algorithm, and demonstrate the feasibility of eye gaze tracking with commodity cameras.

In the future, we will build a gaze estimation system based on a single web camera without depth sensor. We are planning on developing additional applications that leverage the 3D eye gaze tracking.

References

[1] X. Brolly and J. Mulligan, "Implicit calibration of a remote gaze tracker," in *CVPRW 2004* , Washington, pp. 134–134, 2004.

[2] J. Chen and Q. Ji, "3D gaze estimation with a single camera without IR illumination," in *Proc. CVPR 2008*, Anchorage, pp. 1–4, 2008.

[3] J. Chen, Y. Tong, W. Gray, and Q. Ji, "A robust 3d eye gaze tracking system using noise reduction," in *Proc. ETRA 2008*, Savannah, pp. 189–196, 2008.

[4] Y. Francken, C. Hermans, and P. Bekaert, "Screen-camera calibration using gray codes," in *Sixth Canadian Conference on Computer and Robot Vision*, Kelowna, pp. 155–161, 2009.

[5] K. A. Funes Mora and J.-M. Odobez, "Gaze estimation from multimodal kinect data," in *Proc. CVPRW 2012*, Providence, pp. 25–30, 2012.

[6] K. A. Funes Mora and J.-M. Odobez, "Person Independent 3D Gaze Estimation From Remote RGB-D Cameras," in *Proc. ICIP 2013*, Melbourne, 2013.

[7] Y. Gao, M. Wang, D. Tao, R. Ji and Q. Dai, "3D Object Retrieval and Recognition with Hypergraph Analysis," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4290–4303, 2012.

[8] E. Guestrin and E. Eizenman, "General theory of remote gaze estimation using the pupil center and corneal reflections," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 6, pp. 1124–1133, 2006.

[9] S. N. Hajimirza, M. J. Proulx, and E. Izquierdo, "Reading users' minds from their eyes: A method for implicit image annotation," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 805–815, 2012.

[10] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 478–500, Mar. 2010.

[11] R. Jafari and D. Ziou, "Gaze estimation using Kinect/PTZ camera," in *Robotic and Sensors Environments (ROSE), 2012 IEEE International Symposium on*, Magdeburg, 2012.

[12] Q. Ji and X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance," *Real-Time Imaging*, vol. 8, no. 5, pp. 357–377, 2002.

[13] Y. Li, D. monaghan, and N. o' Connor, "Real-time gaze estimation using a Kinect and a HD webcam," in *MMM 2014*, Dublin, pp. 506–517, 2014.

[14] H.-C. Lu, G.-L. Fang, C. Wang, and Y.-W. Chen, "A novel method for gaze tracking by local pattern model and support vector regressor," *Signal Process.*, vol. 90, no. 4, pp. 1290–1299, Apr. 2010.

[15] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Inferring human gaze from appearance via adaptive linear regression," in *Proc. ICCV 2011*, Barcelona, pp. 153–160, 2011.

[16] P. Majaranta and K.-J. Räihä, "Twenty years of eye typing: Systems and design issues," in *Proc. ETRA 2002*, New Orleans, pp. 15–22, 2002.

[17] J. Mansanet, A. Albiol, R. Paredes, J. M. Mossi, and A. Albiol, "Estimating point of regard with a consumer camera at a distance," *Pattern Recognition and Image Analysis*, vol. 7887, pp. 881–888, 2013.

[18] C. H. Morimoto and M. R. M. Mimica, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 4–24, Apr. 2005.

[19] J. Ou, L. M. Oh, S. R. Fussell, T. Blum, and J. Yang, "Predicting visual focus of attention from intention in remote collaborative tasks," *IEEE Trans. Multimedia*, vol. 10, no. 6, pp. 1034–1045, 2008.

[20] S. E. Palmer, *Vision science: Photons to phenomenology*, The MIT Press, 1999.

[21] E. Pogalin, A. Redert, I. Patras, and E. A. Hendriks, "Gaze tracking by using factorized likelihoods particle filtering and stereo vision," in *Proc. 3DPVT 2006*, Chapel Hill, pp. 57–64, 2006.

[22] M. Song, D. Tao, Z. Sun, and X. Li, "Visual-context boosting for eye detection," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 6, pp. 1460–1467, Dec. 2010.

[23] K.-H. Tan, D. J. Kriegman, and N. Ahuja, "Appearance-based eye gaze estimation," in *Proc. WACV 2002*, Washingtion, pp. 191–195, 2002.

[24] K.-H. Tan, I. N. Robinson, B. Culbertson, and J. Apostolopoulos, "Connectboard: Enabling genuine eye contact and accurate gaze in remote collaboration," *IEEE Trans. Multimedia*, vol. 13, no. 3, pp. 466–473, 2011.

[25] A. Villanueva and R. Cabeza, "A novel gaze estimation system with one calibration point," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 1123–1138, 2008.

[26] O. Williams, A. Blake, and R. Cipolla, "Sparse and semi-supervised visual mapping with the sˆ 3gp," in *Proc. CVPR 2006*, New York, pp. 230–237, 2006.

[27] L.-Q. Xu, D. Machin, and P. Sheppard, "A novel approach to real-time non-intrusive gaze finding," in *Proc. BMVC 1998*, Southampton, pp. 1–10, 1998.

[28] H. Yamazoe, A. Utsumi, T. Yonezawa, and S. Abe, "Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions," in *Proc. ETRA 2008*, Savannah, , pp. 245–250, 2008.

[29] Z. Zhu and Q. Ji, "Eye and gaze tracking for interactive graphic display," *Machine Vision and Applications*, vol. 15, no. 3, pp. 139–148, 2004.

[30] Z. Zhu and Q. Ji, "Eye gaze tracking under natural head movements," in *Proc. CVPR 2005*, San Diego, pp. 918–923, 2005.

[31] Z. Zhu and Q. Ji, "Novel eye gaze tracking techniques under natural head movement," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 12, pp. 2246–2260, 2007.

[32] J. Zhu and L. Yang, "Subpixel eye gaze tracking," in *Proc. FGR 2002*, Washington, pp. 124–129, 2002.