Eye Gaze Tracking Using an RGBD Camera: A Comparison with an RGB Solution

Xuehan Xiong
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
xxiong@andrew.cmu.edu
Qin Cai
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
qincai@microsoft.com
Zicheng Liu
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
zliu@microsoft.com
Zhengyou Zhang
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
zhang@microsoft.com

UbiComp '14, September 13 - 17 2014, Seattle, WA, USA
Copyright 2014 ACM 978-1-4503-3047-3/14/09$15.00.
http://dx.doi.org/10.1145/2638728.2641694

Abstract

Most commercial eye gaze tracking systems are based on the use of infrared lights. However, such systems may not work outdoor or may have a very limited head box for them to work. This paper proposes a non-infrared based approach to track one's eye gaze with an RGBD camera (in our case, Kinect). The proposed method adopts a personalized 3D face model constructed off-line. To detect the eye gaze, our system tracks the iris center and a set of 2D facial landmarks whose 3D locations are provided by the RGBD camera. A simple onetime calibration procedure is used to obtain the parameters of the personalized eye gaze model. We compare the performance of the proposed method against the 2D approach using only RGB input on the same images, and find that the use of depth information directly from Kinect achieves more accurate tracking. As expected, the results from the proposed method are not as accurate as the ones from infrared-based approaches. However, this method has the potential for practical use with upcoming better and cheaper depth cameras.

Author Keywords

Pervasive eye-based interaction, RGBD-sensor-based eye gaze tracking, Multi-modal sensor fusion, Gaze and eye movement analysis methods

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous.

General Terms

HCI, eye gaze, RGBD Sensor, Kinect

Introduction

Eye gaze tracking methods can be classified into two categories based on whether infrared (IR) lights are used or not. Most existing commercial systems rely on IR lights and ship with their specific hardware. Non-IR based methods are less common and the associated techniques are relatively immature and less accurate. However, they still remain as a popular research topic because of their less-strict requirement for hardware and their easiness to integrate with consumer products, such as laptops and tablets. In this paper, we mainly focus on non-IR based approaches, and we provide a brief overview of previous work below. For a complete survey of eye gaze tracking techniques, please refer to [3]. Non-IR based methods can be grouped into the following three categories: appearance-based approach, iris-based approach, and face-model-based approach.

Appearance-based approaches [5] attempt to build a regressor that maps the appearance of the eye to the screen coordinates where the user is looking at. The underlining assumption for these methods is that the appearance changes can be only caused by pupil movement. However, the appearance may vary due to other factors such as illumination changes and head movement. Also, appearance features are usually high dimensional. This results in the need of more calibration points to train the regressor, which implies less pleasant user experience.

Iris-based method [9] first detects iris through an ellipse fitting procedure. The shape of the ellipse can be used for determining the normal of 3D iris. The gaze is approximated by this normal vector. This method is not accurate because extracting the exact shape of iris is difficult due to occlusion by the eyelids, specular reflections in the iris, and noises in the image.

Face-model-based methods [6, 1] first locate facial landmarks on the image. Their 3D locations are obtained either through a stereo camera [6] or using a 3D generic face model [1]. An initial estimation of eyeball center is made based on the predicted landmarks. Then it is further refined by a personalized calibration step. The optical axis is defined by a line crossing the eyeball center and the 3D pupil center. The use of a *generic* 3D face model provides inaccurate 3D locations of the facial landmarks. The depth from stereo may not be accurate enough for gaze estimation. Even small errors in the 3D landmarks will result in large error in gaze estimation. In theory, this type of approaches is robust to head movement, but their accuracy highly depends on the results from head pose estimation and facial feature detection.

Thanks to the recent advances in facial feature detection research. Its accuracy and efficiency have been improved rapidly over the past few years. In this paper, we follow the face-model-based approach by first detecting facial landmarks using a recent published work [10]. The depth information in our approach can either come from a depth camera (a Kinect in this paper) or, if a depth camera is not available, by minimizing the projection error of a *person-specific* 3D face model and the tracked 2D landmarks. The contributions of this paper are the following. We give a comparison between the above two approaches evaluated on simulated data and real-world data. We propose a method to measure the lower bound of gaze error using face-model-based approaches. Also, using simulated data we give an analysis of how the results of pupil detection and facial feature detection affect the performance of gaze estimation.

We want to remind our readers that Kinect uses IR to estimate depth information. We do not use IR illuminated images in any of our steps. Our approach is not specific to Kinect and other depth sensors can be used as a replacement.

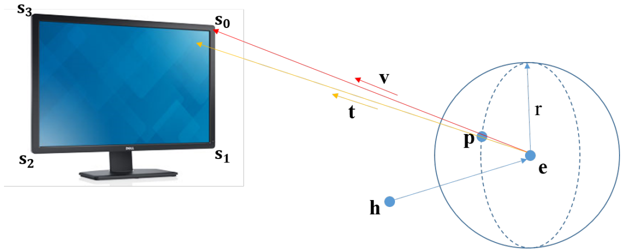


Figure 1: Eye gaze model.

Approach

This section presents the major components in our eye gaze model.

Overview

To define the gaze direction, we use a simple eye gaze model illustrated in Fig.1. The pupil center $\mathbf{p}$ is assumed to lie on the sphere of eyeball. The optical axis $\mathbf{t}$ is defined by a ray crossing the eyeball center $\mathbf{e}$ and the pupil center $\mathbf{p}$. The visual axis $\mathbf{v}$, namely the gaze direction, differs from $\mathbf{t}$ by two angles, α in the horizontal direction and β in the vertical direction. For a particular user, the following parameters are unknown but fixed w.r.t her head coordinate system centered at $\mathbf{h}$: eyeball center $\mathbf{e}$, eyeball radius r, and α, β. In section on calibration, we propose a simple one-time calibration procedure to infer these parameters. (Note, this model is simpler than the one usually used in IR-based techniques where the optical and visual axes of eye gaze are defined with respect to the cornea center, rather than the eyeball center.)

After this calibration, the user's gaze direction can be computed using the following steps under our eye gaze model. First, we transfer the eyeball center from the head coordinate to the world coordinate,

$$\mathbf{e}^t = \mathbf{h}^t + \mathbf{R}_h^t \mathbf{e}, \tag{1}$$

where $\mathbf{h}^t$ and $\mathbf{R}_h^t$ are the head center and head rotation matrix at time t. Section on head pose explains how we obtain the 3D head pose from a RGBD or RGB image. The optical axis direction $\mathbf{t}^t$ can be seen as a normalized vector from $\mathbf{e}^t$ to $\mathbf{p}^t$. The computation of 3D pupil center is explained in section on iris detection. Once the optical axis is known, the gaze direction can be found by rotating $\mathbf{t}$ α degrees horizontally and β degrees vertically. Mathematically speaking,

$$\mathbf{v}^t = \mathbf{R}_h^t \mathbf{R}_{\alpha,\beta} (\mathbf{R}_h^t)^{-1} \mathbf{t}^t, \tag{2}$$

where

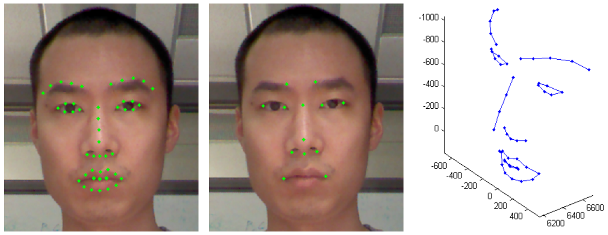
$$\mathbf{R}_{\alpha,\beta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}. \tag{3}$$

Notice that we remove the head rotation before applying the rotation offset between the optical and visual axes.

System calibration

In our eye gaze model, depth camera, color camera, and the monitor screen each have their own coordinate systems. This section describes the calibration steps that transform them into the world coordinate system (color camera). We use a technique described in [11] to calibrate between depth and color cameras. The screen-camera calibration is done by using an auxiliary camera and a calibration pattern in front of the screen so that the auxiliary camera can see both the calibration pattern and the screen while the color camera associated with the screen can see the calibration pattern. The detailed procedure can be found in [12].

Head pose



(a) (b) (c)

Figure 2: a) Facial feature detection results. b). 13 Rigid points used for head pose estimation. c). An example of 3D face model

In this section, we use one of the following two methods to measure user's 3D head pose depending on whether a Kinect is available.

The first method assumes that a Kinect is available. First, we track the 49 facial landmarks on the RGB image using a Supervised Descent Method (SDM) [10]. An example output can be found in Fig. 2(a). Based on the 2D coordinates of the tracked landmarks, we can read out their corresponding 3D coordinates from a calibrated Kinect sensor. For tracking head pose, first we need to build a person-specific 3D face model for each user. We ask the user to keep a frontal pose to the Kinect for one second and during that time 10 sets of 3D facial landmarks are collected and their average is used as the reference 3D face model, $\mathbf{X}_{ref}$. $\mathbf{X}_{ref}$ is a matrix of size $3 \times n$, where n is the number of landmarks and each column represents the 3D position of one facial landmark. Fig. 2(c) shows a 3D face model we built for a particular user. To make the head pose robust to facial expression changes, we only use 13 rigid points on the face (shown in Fig. 2(b)).

A subject's head pose is measured relative to her reference model. The 3D head pose at frame t (head rotation matrix $\mathbf{R}_h^t$, translation vector $\mathbf{t}^t$) is obtained by minimizing the following equation,

$$\arg\min_{\mathbf{R}_t,\mathbf{t}_t} \|\mathbf{R}_h^t \mathbf{X}_{ref} + \mathbf{1}_{1\times n} \otimes \mathbf{t}^t - \mathbf{X}^t\|. \qquad (4)$$

$\otimes$ denotes the Kronecker product and $\mathbf{1}_{1\times n}$ is a row vector of ones of size n. The above formulation is the well-known Procrustes problem [8], which can be solved using Singular Value Decomposition. However, least squares fitting is known to be sensitive to outliers. Kinect occasionally gives zero depth values due to sensor noise. We perform a local neighborhood search for the missing depth values. When a point is occluded, the depth value derived from its neighbor's may deviate from its true value. We remove the points with fitting errors more than two standard deviations away from the mean and perform another minimization on equation 4 using the remaining points.

The second method only requires a calibrated RGB camera and the person-specific face model built off-line. In each frame, after locating facial landmarks on the RGB image we use the POSIT algorithm [2] to estimate user's head pose. POSIT finds object pose by iteratively minimizing the error between the projection of a known 3D model and 2D landmarks tracked.

Iris detection

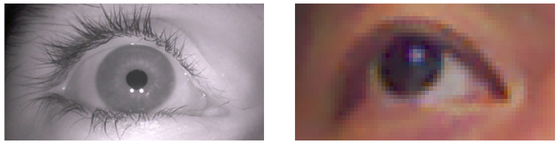


Figure 3: Comparison of eye images taken with IR lighting (left) and without (right).

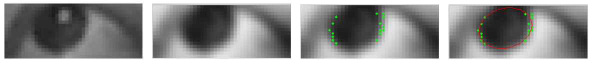


Figure 4: Example outputs of each major step in our iris detection algorithm. 1) Cropped eye image. 2) Eye image after image processing. 3) Detected iris candidate points. 4) Ellipse fitting.

Without IR lighting pupils may not be visible (See Fig. 3) so we replace the task of pupil detection with iris detection. In our approach, iris is modeled as an ellipse and pupil center is inferred as the center of the ellipse. Our iris detection is based on the Starburst algorithm [4] with some modifications. Fig. 4 shows the example outputs of each major step in the algorithm. First, we crop the eye region based on the output of facial feature detection. Histogram equalization is then applied to increase the contrast of the eye image. A binary image is created by thresholding each pixel with the mean pixel value in the image. Connected-component analysis is performed to fill holes (caused by specular reflection) in the iris region followed by a Gaussian blur. 30 rays are emitted from a seed point terminated on the boundary of the polygon that defines the eye region. The direction of the rays is uniformly distributed between $-45°$ to $45°$ and $135°$ to $225°$. Such range is chosen because the top and bottom parts of iris are likely to be occluded by eyelids. The point yielding the highest gradient value along each ray is considered as a candidate of iris boundary. The candidate points with gradient value lower than a predefined threshold are removed. Those remaining points are then used to fit an ellipse. The candidates with fitting residuals greater than two standard deviations away from the mean are considered as outliers and removed. We refit an ellipse on the remaining candidates.

Given pupil center in the image, $[u, v]$, its 3D position in the world can easily be obtained following the geometry illustrated in Fig. 5. The 3D pupil center is the intersection point between the eyeball sphere and line $\vec{\mathbf{ou}}$. The camera center $\mathbf{o}$ is at the origin. $\mathbf{u} = [u - u_0, v - v_0, f]$ is the 3D coordinate of the 2D pupil center in the world, where $[u_0, v_0]$ is the image center from camera intrinsic parameters, f is the camera focal length in pixels.

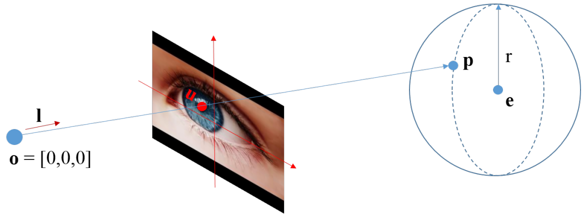


Figure 5: 3D pupil center localization.

Personal parameter calibration
All eye gaze systems require a user calibration step due to the unknown offset between the optical and visual axes. Besides such offset, our calibration step also automatically computes the optimal eyeball center and eyeball radius.

During the calibration, the user is asked to look at predefined 9 points on the monitor screen, from which we can compute the ground truth gaze direction. The user calibration is achieved by minimizing the sum of angles between predicted gaze directions and the ground truth ones. The initial user parameters are set to be the human average. For example, the initial estimation of eye radius is set to be 12mm. We use COBYLA algorithm [7] for optimization.

Experiments

In this section, we report the experimental results of the proposed methods on iris detection and gaze estimation. We are not able to report the results on head pose estimation because the ground truth is difficult to obtain.

Iris detection

First, we present the results of iris detection method described in section on Iris Detection. The test data is collected as follows. We ask each subject to look at nine points (shown in Fig. 8(a)) on the monitor screen and one or more images are taken for each point. Eye appearance may vary across people from different ethnicity background. In total, we collect 157 images from 13 different individuals. The subjects are from three ethnic groups: Asian, Indian, and Caucasian (See Fig. 7(b)). We flip the images to double the test set. The ground truth iris center is found by manually selecting some points along the iris boundary and then fitting an ellipse on the selected points. The error (in pixels) is computed as the distance between the ground truth iris center and the predicted one. Out of 628 eyes 555 are detected. The example results and cumulative error distribution can be found in Fig. 6 and Fig. 7(a). We found that 55% of the data has an error under 1 pixel, 75% under 1.5 pixel, and 86% under 2 pixels.

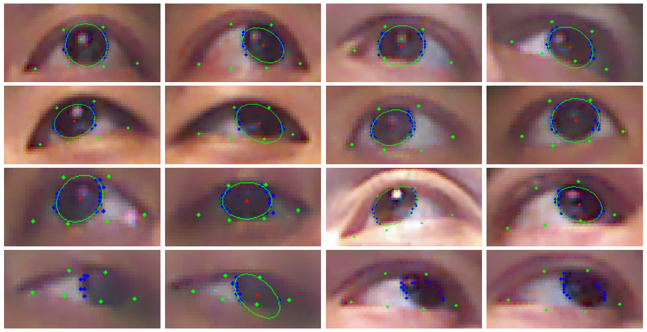
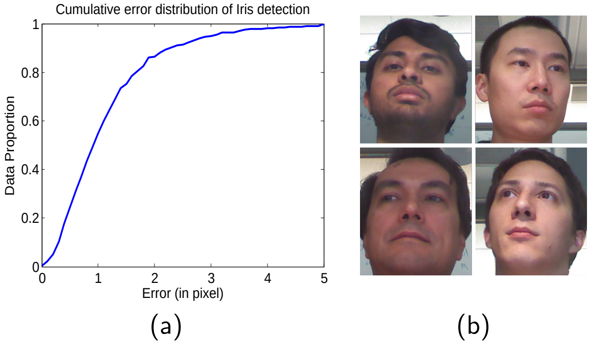


Figure 6: Example outputs from our iris detection. Last row shows some failure cases. Fitted ellipses are drawn in green line. Detected iris boundary points are drawn in blue. Green dots are the facial feature points.



(a)

(b)

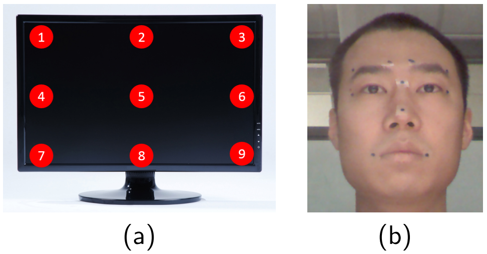
Figure 7: a). Cumulative error distribution of our iris detection method. b). Examples of participants in iris data collection.

Gaze estimation on simulated data

This section gives an analysis on how the results of facial feature detection and pupil detection affect the accuracy of gaze estimation. We build a simulation program that allows us to control the noise level of each component in our eye gaze model. In this simulation, we assume a perfect system calibration and user parameters are known in advance. The only sources of errors are from facial feature detection and pupil detection. The simulation consists of a virtual camera, a virtual screen, and a 3D face model. The parameters of these components are kept similar as they are in the real-world. The ground truth of facial landmarks is obtained by projecting the face model onto the image plane using the virtual camera. We apply the same strategy to pupil center and obtain its ground truth location in the image.

In Fig. 9, we plot the gaze errors against landmark noise for both RGBD and RGB solutions. In the RGBD solution, we add noise directly to the 3D landmarks while in the RGB solution we add noise to the 2D landmarks. To put "pixel" in perspective, the interocular distance of the projected face is 100 pixels in our simulation. For both solutions, gaze error increases linearly with the noise added in landmark localization when pupil detection is almost perfect. The simulation also shows that accurate gaze estimation requires high quality image and depth sensors. For example, in the RGBD solution, to achieve a 2° gaze accuracy the errors in 3D landmark localization and pupil detection need to be kept within 2mm and 0.5 pixel, respectively. Achieving the same accuracy using the RGB solution requires both 2D landmark and pupil localization error to be under 0.5 pixel. Again, those numbers are optimistic since we assume a perfect system calibration and user calibration.

Gaze estimation on real-world data



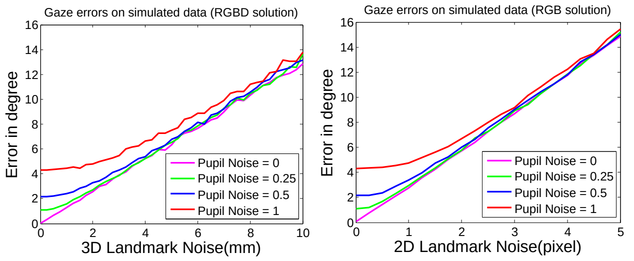
(a) (b)

Figure 8: a) Nine calibration points we used in the experiment. b). A subject wearing colored stickers.

This section presents the results of gaze estimation using real-world data collected from a Kinect. The monitor used in our experiments has a dimension of 520mm by 320mm. The distance between a test subject and the Kinect is between 600mm and 800mm. There are 9 subjects participated in the data collection. We collect three training sessions and two test sessions for each subject. During each training session, nine dots are displayed on the screen (shown in Fig.8 (a)). The subject is required to click on each of them (by clicking it enforces the user to look at the dot) and five images are taken upon each clicking action. During data collection, the subjects are free of head movement. After finishing one session, the subject is asked to move her seating position before starting recording the next one. A testing session is recorded in a similar manner but with 15 images per point. Data collected in training sessions is used for the user calibration described in section on personal parameter calibration. The gaze errors are computed on data from test sessions.

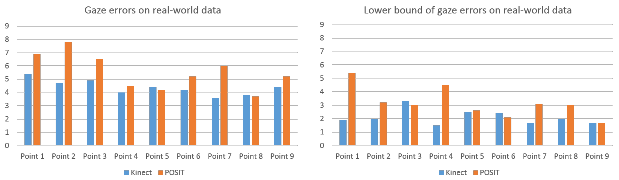
Fig. 10(a) shows the gaze errors from our RGB and RGBD solutions. We remind our readers that the only difference between our RGBD solution and RGB one is the source of depth information. The depth in RGBD solution comes from a Kinect while RGB solution uses POSIT algorithm to obtain depth information. The results are generated by averaging over 18 test sessions. RGBD solution gives a mean error of $4.4°$ while RGB solution gives $5.6°$. It is worth mentioning that RGBD solution consistently outperforms RGB one except for point 5 and 8. When a subject looks at point 5 and 8 her pose is close to the reference model and therefore, POSIT algorithm can give an accurate depth estimate.

It would be interesting to know what is the lower bound of gaze error using our approach. We asked one subject to wear colored stickers on his face during data collection so that those stickers can be treated as facial landmarks and can be tracked perfectly. See Fig. 8(b) for an example. For iris detection, we manually select some points along the boundary of iris and then fit an ellipse on those points. Fig. 10(b) shows the gaze errors that are computed under the above protocol. Again, the RGBD solution outperforms RGB's giving a mean error of $2.1°$ versus $3.2°$ of RGB's. Given the current generation Kinect, the lower bound of gaze error estimated under our eye gaze model is $2.1°$.



(a) (b)

Figure 9: Gaze errors (in degree) on simulated data. a). RGBD solution. b). RGB solution.



(a) (b)

Figure 10: a). Gaze errors (in degree) evaluated on real-world data from our RGB (POSIT) and RGBD (KINECT) solutions. b). The lower bounds of gaze errors on real-world data.

Conclusion

In this paper, we propose a face-model-based approach for eye gaze estimation. By leveraging a Kinect sensor we are able to obtain an accurate estimation of a user's head pose. The 3D location of the eyeball center is then inferred from this estimation. Its final location along with other user parameters are further refined in a calibration step. In the experiments, we found that the use of depth information directly from Kinect provides more accurate gaze estimation compared with the one from only RGB images. The lower bound for gaze error based on our eye gaze model and hardware is around $2°$. The proposed method achieves $4°$ error. The Kinect used in our experiments is the first generation Kinect. With the advance of depth sensing technologies, we can only expect higher accuracy of gaze estimation in the future.

References

[1] Chen, J., and Ji, Q. 3D gaze estimation with a single camera without IR illumination. In *ICPR* (2008), 1–4.

[2] DeMenthon, D. F., and Davis, L. S. Model-based object pose in 25 lines of code. *International Journal*

of Computer Vision 15 (1995), 123–141.

[3] Hansen, D. W., and Ji, Q. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 3 (2010), 478–500.

[4] Li, D., Winfield, D., and Parkhurst, D. J. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Proceedings of the IEEE Vision for Human-Computer Interaction Workshop at CVPR* (Washington, DC, USA, 2005).

[5] Lu, F., Sugano, Y., Okabe, T., and Sato, Y. Inferring human gaze from appearance via adaptive linear regression. In *ICCV* (2011), 153–160.

[6] Newman, R., Matsumoto, Y., Rougeaux, S., and Zelinsky, A. Real-time stereo tracking for head pose and gaze estimation. In *FG*, IEEE Computer Society (2000), 122–128.

[7] Powell, M. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Proceedings of the Sixth Workshop on Optimization and Numerical Analysis* (1994).

[8] Schönemann, P. A generalized solution of the orthogonal procrustes problem. *Psychometrika 31*, 1 (1966), 1–10.

[9] Wang, J., Sung, E., and Venkateswarlu, R. Estimating the eye gaze from one eye. *Computer Vision and Image Understanding 98*, 1 (2005), 83–103.

[10] Xiong, X., and De la Torre, F. Supervised descent method and its application to face alignment. In *CVPR* (2013).

[11] Zhang, C., and Zhang, Z. Calibration between depth and color sensors for commodity depth cameras. In *ICME* (2011), 1–6.

[12] Zhang, Z. Camera-screen calibration in the Viewport system. Microsoft Research Tech Note, 2011.