# Eye Gaze Correction for Video Conferencing Using Kinect v2

Eunsang Ko, Woo-Seok Jang, and Yo-Sung Ho(✉)

School of Information and Communications,
Gwangju Institute of Science and Technology (GIST),
123 Cheomdangwagi-ro, Buk-gu, Gwangju 500-712, Republic of Korea
{esko, jws, hoyo}@gist.ac.kr

**Abstract.** In video conferencing, eye gaze correction is beneficial for effective communication. In this era, video conferencing at homes using laptops is straightforward. In this paper, we propose an eye gaze correction method with a low-cost simple setup using Kinect v2. Our method detects an ellipse that connects edge points of the face after identifying several feature points within the face using Kinect v2 SDK. Then, we apply a 3D affine transform that allows eye gaze correction using camera space points that are acquired from depth information. Thus, in the preprocessing step, an ellipse model should be extracted when the user gazes the camera and display, respectively. Also, we fill holes that are caused by the affine transform using color inpainting. As a result, we produced a natural eye gaze-corrected image in real-time.

**Keywords:** Eye gaze correction · Eye contact · Video conferencing · Kinect v2

## 1 Introduction

When we are talking with other people, eye contact provides important information. Eye contact is looking at each other's eyes at the same time. By contacting other's eyes, we can guess what they are thinking, or whether they are interested. Thus, eye contact is significant in video conferencing. Video conferencing is one of telecommunication technologies which allows people communicate in two or more locations simultaneous video transmission. In these days, as the telecommunication technologies have grown, many people use the video conferencing in their homes as well as work places. In particular, people who is personal user use free video conferencing program such as Skype using their laptop or webcam. However, general video conferencing system occurs lack of eye contact due to the disparity between the locations of the subject and the camera [1]. These lack of eye contact cause unnatural communication and negative signs to other users.

For solving the lack of eye contact, various eye gaze correction methods have been proposed. One method is a remote collaboration system based on a semi-transparent see-through display. This method creates an experience where local and remote users are seemingly separated only by a vertical sheet of glass [2]. However, this method cannot be widely used due to expensive customized hardware. In this paper,


Y.-S. Ho et al. (Eds.): PCM 2015, Part II, LNCS 9315, pp. 571–578, 2015.
DOI: 10.1007/978-3-319-24078-7_58

we propose an eye gaze correction method using Kinect v2. The proposed method can be used cost-efficient and simple.

The proposed method bases a gaze correction approach using a single Kinect v1 [1]. Kinect v1 is composed of a color camera and a depth camera. They produce images at 640 × 480 resolution. However, the depth image of Kinect v1 is inaccurate with silhouette of the color image due to structure of Kinect v1. Thus, Kuster's method applies smoothing and filling holes on the depth image using Laplacian smoothing. Then, they make a novel view where the gaze is corrected. This is accomplished by applying a rigid transform that allows eye gaze correction. A matrix of the rigid transform is computed only once during the calibration stage. Next, they extract a user's face to track facial feature points in the original color image. They compute 66 feature points along the chin, nose, eyes and eyebrows. Then, they apply a seam optimization that is an optimal stencil to cut the face from a transformed image using the feature points. As a result, they generate a natural result of eye gaze correction by extending the seam of face by 5–10 pixels. Also, they solve a problem of flickering artifacts in a sequence result of eye gaze correction by optimizing the face tracker vertices. Their method runs at about 20 frames per second (fps) on a consumer computer.

## 2 Proposed Method

### 2.1 System Design

We use 27 inches display monitor and Kinect v2. Kinect v2 is combined color camera with depth camera. In Fig. 1(a), the color camera is on the left of Kinect v2, and the depth camera is on the center of Kinect v2 [3]. As we use depth information for eye gaze correction, we align the depth camera with center of the display monitor. Then we set Kinect v2 on the bottom of the display monitor. And, we raise an angle of Kinect v2 to face to the user's face for effective eye gaze correction, but the user's spine must be shown in the color image for face detection using Kinect v2 SDK. Thus, moderate the angle of Kinect v2 and distance between the users and Kinect v2 are required when the user sets the proposed system. Figure 1(b) displays an example of the proposed eye gaze correction system configuration.

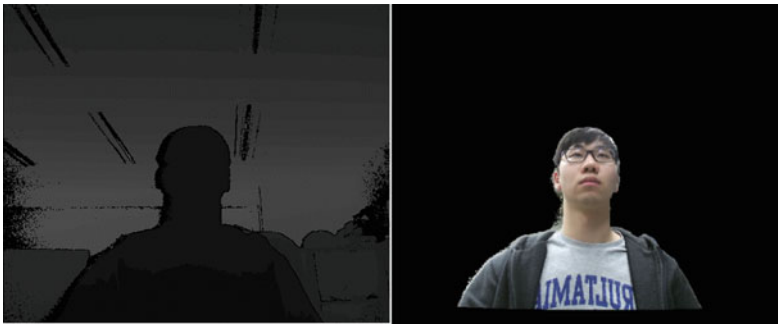
(a) Kinect v2 (b) System setup

**Fig. 1.** Kinect v2 and system configuration

### 2.2 Preprocessing

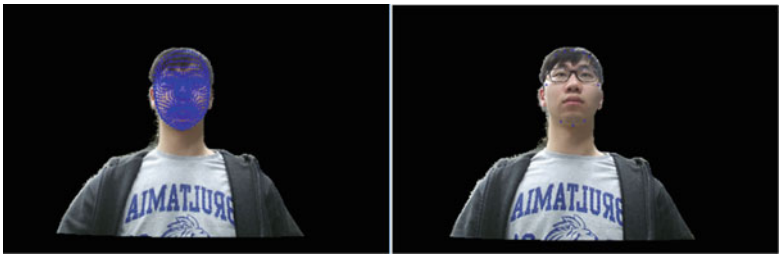
Preprocessing comprises four steps: down-sampling a color image, detecting face feature points, estimating a 3D affine transform matrix that allows eye gaze correction, and creating face mask that is applied 3D affine transform. Although the preprocessing steps are many, some are repeatedly processed each frames, the others take only once in a few seconds by users.

The color camera of Kinect v2 has a resolution of 1920 × 1080 pixels, and the depth camera has a resolution of 512 × 424 pixels. For mapping the color image to depth information, we down-sample the color image using Kinect v2 SDK. We produce eye gaze-corrected image using the down-sampled color image. Thus, the eye gaze-corrected image has a resolution of 512 × 424 pixels. Then, we limit depth value between 450 and 900 for reducing depth noise and subtracting background. The depth value of Kinect v2 means a distance between the object and the camera. The depth camera of Kinect v2 can capture the depth value from 450 mm. Figure 2(a) and (b) shows a real-depth map and the down-sampled color image, respectively.



(a) Real-depth map (b) Down-sampled color image (depth limit: 450-900)

**Fig. 2.** Depth map and down-sampled color image of Kinect v2



(a) 1347 points of face feature (b) Sampled 10 points in 1347 points

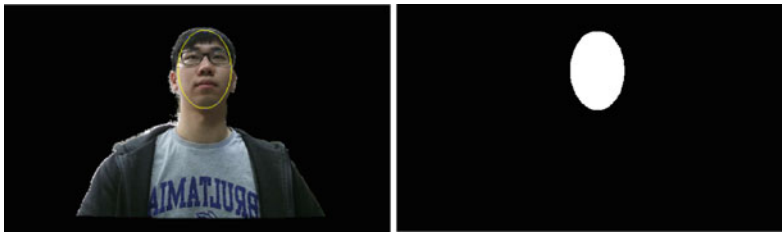
**Fig. 3.** Face feature point detection using Kinect v2 SDK

Second is the detecting face feature points. For detecting the face feature points in Kinect v2 SDK, it needs body tracking data. Thus, the user's upper body like head, neck, shoulder and spine must be shown in field of view of Kinect v2 for detecting skeleton data of body. Once skeleton data is detected, 1347 points of face feature are tracked if face is in the field of view of Kinect v2. And, once tracking of the face feature points are started, Kinect v2 continuously tracks the face feature points in the color image. The face feature points are represented with camera space point. We can display the face feature points on the down-sampled color image by converting camera space point to depth image point using Kinect v2 SDK. Figure 3(a) exhibits a result of detected 1347 points of face feature. However, most of face feature points are convergent in the user's eye, nose and mouth. Thus, we sample 10 points in the 1347 points of face feature for decreasing complexity and improving accuracy of eye gaze correction. Figure 3(b) shows the sampled 10 points in the user's forehead, cheek bone, jaw and chin.

Third, for estimating the matrix of 3D affine transform, we need the face feature points of two models when the user gazes the camera and display, respectively. Figure 4(a) and (b) display an example image when the user gazes the camera and display, respectively. Thus, we store couple of 10 points of camera space that are camera model and display model, respectively. Then, we estimate the optimal matrix of 3D affine transform that converts the display model to the camera model using the

(a) When gazes the camera (b) When gazes the display

**Fig. 4.** An example image when the user gazes the camera and display



(a) Ellipse fitting (b) Ellipse mask

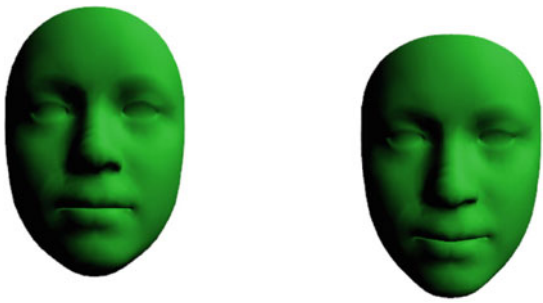
**Fig. 5.** An example image of fit ellipse and created mask

random sample consensus (RANSAC) algorithm for eliminating outliers. We can get converted camera space points by multiplying the matrix of 3D affine transform by original camera space points.

Finally, we create a mask that is applied eye gaze correction using the matrix of 3D affine transform. The mask is ellipse area that connects the sampled 10 face feature points. For finding an optimal ellipse model using the 10 face feature points, we use ellipse fitting function within OpenCV that algorithm [Fitzgibbon95] is used [4, 5]. The mask can be largely changed by coordinates of the sampled face feature points in the user's forehead, cheek bone, jaw and chin, respectively. Thus, the users can adjust the coordinates of the face feature points within Kinect v2 SDK to generate the optimal ellipse mask. Then, we draw the ellipse mask on an empty black image using a simple OpenCV flag [6]. Figure 5(a) and (b) exhibit an example image of fitting ellipse and creating the filled ellipse mask.

### 2.3 Eye Gaze Correction

The proposed eye gaze correction method converts a camera space point of Kinect v2. *X*, *Y*, and *Z* of the camera space point means yaw, pitch, and depth, respectively. Thus, we have to change the *Y* value of the camera space point to make eye gaze correction. Figure 6(a) and (b) simulate a face model that is original model and decreased model by 0.1 the *Y* value of the camera space point, respectively. The matrix of 3D affine transform that we estimated in the processing step allows eye gaze correction by changing the camera space point. The *Y* value of the camera space point is more changed than the *X* or *Z* value. Equation 1 shows the changing camera space point by multiplying the matrix of 3D affine transform by original camera space point. The matrix of 3D affine transform is 3 by 4, *m* means an element of the transform matrix. The *X*, *Y*, and *Z* denote the original camera space point, and the *X*`, *Y*`, and *Z*` represent the converted camera space point. When multiplying the matrix by the original camera space point, we make the original camera space point to a homogeneous matrix.



(a) Original face model (b) Updated face model

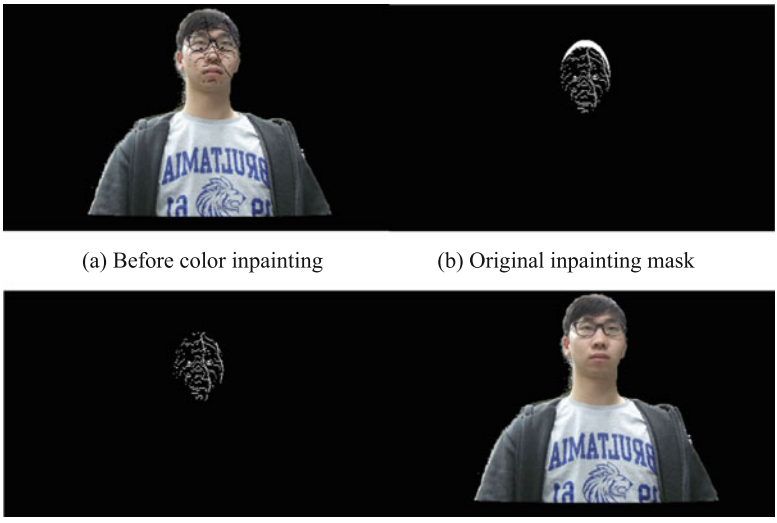
**Fig. 6.** Simulation result of eye gaze correction

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

To apply eye gaze correction, we convert a depth point that is in the face mask to a camera space point using Kinect v2 SDK. The camera space point can be converted using a pair both the depth image point and depth value of the depth point. This camera space point is corrected to make eye contact using the 3D affine transform. The corrected camera space point is converted to depth image point. Thus, $y$ coordinates of the depth point are increased by depth value of the depth point.

### 2.4 Color Inpainting

Figure 7(a) shows an example of eye gaze correction result. There are many holes due to a pixel rounding error when applying the 3D affine transform. Also, there is one large hole in the user's forehead that represents eye gaze correction is applied well. Figure 7(b) displays an error map of the result of eye gaze correction. For removing the error, we fill hole using color inpainting using [Telea04] method [7]. However, the large hole causes an unnatural result after color inpainting. Thus, we create a color inpainting mask by updating the face mask. Figure 7(c) and (d) represent the updated color inpainting mask and a result of color inpainting, respectively.



(a) Before color inpainting (b) Original inpainting mask

(c) Updated inpainting mask (d) After color inpainting

**Fig. 7.** A result of color inpainting

## 3 Experiment Result

To evaluate our proposed eye gaze correction method, we capture several results of eye gaze correction when the user watches around the monitor. Figure 8(a)–(c) display the results of eye gaze correction, respectively. Figure 8(a) is a result of eye gaze correction when the user watches center of the monitor, and Fig. 8(b)–(c) are results of eye gaze correction when the user watches left and right of the monitor, respectively. As watching the results, there are three problems. First, there is a depth hole in area of the user's glasses due to reflected light. Second, there is boundary noise due to time-of-flight (ToF) depth error. The boundary noise occurs while the color image is

(a) When the user gazes center of the monitor

(b) When the user gazes left edge of the monitor

(c) When the user gazes right edge of the monitor

**Fig. 8.** Experiment result of eye gaze correction

down-sampled in the processing step. Finally, a result of eye gaze correction is out of shape if the user's depth value is largely changed like shifting the user's position.

We processed the proposed eye gaze correction method on a desktop computer that uses a CPU: Intel Core i7 4960X @ 3.6 GHz. To achieve processing of the eye gaze correction method in real-time, we handled a multi-thread scheduling when capturing the color and depth image, down-sampling the color image, applying eye gaze correction. As a result, a result of eye gaze-corrected is generated at a rate of 28 fps.

## 4 Conclusion

In this paper, we proposed an eye gaze correction method using Kinect v2. We down-sampled a color image and detected face feature points using its SDK. Then, we estimated the matrix of 3D affine transform that allows eye gaze correction by means of a face model as the user gazes the camera and display, respectively. Finally, we applied the 3D affine transform and color inpainting within the face mask. As a result, we acquired an eye gaze-corrected natural image in real-time. There exists depth holes and boundary noise due to the limitation of Kinect v2 depth camera. Hence, if we process a depth up-sampling using high-performance computer and GPU processing, we can generate a result of eye gaze-corrected at 1920 × 1080 resolution as well as solve depth holes and boundary noise of the result image.

**Acknowledgement.** This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2011-0030079)

## References

1. Kuster, C., Popa, T., Bazin, J.C., Gotsman, C., Gross, M.: Gaze correction for home video conferencing. ACM Trans. Graphics **31**(6), 1–6 (2012)
2. Tan, K.H., Robinson, I.N., Culbertson, B., Apostolopoulos, J.: ConnectBoard: Enabling Genuine Eye Contact and Accurate Gaze in Remote Collaboration. IEEE Trans. Multimedia **13**(3), 466–473 (2011)
3. http://www.microsoftstore.com/store/mssg/en_SG/pdp/Kinect-for-Windows-v2-Sensor/productID.299057000
4. Fitzgibbon, A.W., Fisher, R.B.: A Buyer's guide to conic fitting. In: Proceedings of the British Machine Vision Conference (BMVC), Birmingham, England, vol. 2, pp. 513–522 (1995)
5. http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=fitellipse#fitellipse
6. http://docs.opencv.org/modules/core/doc/drawing_functions.html?highlight=ellipse#ellipse
7. Telea, A.: An image inpainting technique based on the fast marching method. J. Graphics, GPU, Game Tools **9**(1), 25–36 (2004)