Eye Gaze Correction with a Single Webcam Based on Eye-Replacement

Yalun Qin, Kuo-Chin Lien, Matthew Turk, Tobias Höllerer

University of California, Santa Barbara

Abstract. In traditional video conferencing systems, it is impossible for users to have eye contact when looking at the conversation partner's face displayed on the screen, due to the disparity between the locations of the camera and the screen. In this work, we implemented a gaze correction system that can automatically maintain eye contact by replacing the eyes of the user with the direct looking eyes (looking directly into the camera) captured in the initialization stage. Our real-time system has good robustness against different lighting conditions and head poses, and it provides visually convincing and natural results while relying only on a single webcam that can be positioned almost anywhere around the screen.

1 Introduction

With the introduction of both consumer and enterprise video communication technologies, videoconferencing is becoming more and more widely used, allowing people in different locations to meet face-to-face with each other without having to travel to a common location. However, most current videoconferencing systems are not able to maintain eye contact between the conversation participants due to the disparity between the locations of the videoconferencing window and the camera. This problem has been addressed in some high-end level videoconferencing systems by using special hardware, such as semitransparent mirrors/screens [1, 2], cameras with depth sensors [3, 4], or stereo cameras [5, 6]. However due to the extra hardware dependence, these solutions cannot be applied to consumer level videoconferencing systems, which typically only use webcams.

In this paper, we present a gaze correction system that relies only on a single webcam, and thus can be integrated in current videoconferencing systems running on various consumer level devices, ranging from desktop computers to smartphones. Our system corrects the gaze by replacing the eyes of a person in the video with images of eyes looking directly at the camera; these are captured during an initialization session. By further applying several computer vision and computer graphics techniques we make the result look very natural under a range of lighting conditions and head poses. The system supports cameras placed above, below, or to either side of the display screen. The speed of our system is about 30 fps on 640×480 streams from a laptop camera. With minor changes this approach can be easily extended to correct the gaze of multiple users simultaneously.

G. Bebis et al. (Eds.): ISVC 2015, Part I, LNCS 9474, pp. 599–609, 2015.
DOI: 10.1007/978-3-319-27857-5 54

2 Related Work

Previous research using a single webcam for gaze correction has taken two main approaches. The first category of approaches aims to perform the gaze correction by synthesizing a new view of the user's face from a virtual point behind the screen in the center, which is equivalent to performing a rotation of head/face or the entire image by some degrees [7–11]. However these approaches exhibit distortions on the contour of the head, occluded areas and glasses frames. In addition, most of these methods do not perform any explicit changes in the shapes of the eyes, which we believe is very critical to gaze correction and might not be modified sufficiently by merely rotating the head model.

The second category of approaches performs gaze correction by only processing the eyes [12–16]. Most of these methods aim to reposition the iris using computer vision techniques like segmenting and warping [12–14], which all suffer some robustness issues under various lighting conditions and do not have very satisfactory real-time performance. The eye-replacement method proposed by Wolf et al. [15] can establish very natural looking eye contact and is quite robust under various lighting conditions; however due to the quality of the tracker, their system will fail to work when there are large head motions. Their method is also unable to detect whether the user is looking sideways or not and simply replaces the eyes regardless of the actual gaze direction. A recent approach proposed by Kononenko et al. [16] aims to synthesize eyes looking up just by modifying the original image, which turns out to have a good real-time performance, and the result looks natural even when the user is looking sideways. However, due to the limitation of training data, the system only supports vertical gaze redirection by 10 to 15 degrees.

3 Eye-Gaze Correction Method

3.1 System Overview

Our work is based on the eye-replacement method proposed by Wolf et al. [15]. We improved it to make it more robust to changes in lighting conditions and head poses. In contrast to Wolf's approach, ours does not require training and the quick initialization could potentially be done automatically. To ensure the results look good, we also defined the *zone of correction*, which is the range of head poses and eye openness when the gaze correction should be applied. We made the results look natural not only inside the zone but also in the transition areas between inside the zone of correction and outside. Ideally the actual gaze direction should also be taken into account, so that gaze correction is performed only when the user is looking at the video participant. We further extended our system to support the case when the camera is positioned lateral to the screen.

Our system consists of two stages (Fig. 1):

1. Initialization stage, in which the user is asked to look at the screen first(this step is used for setting the ratio of vertical eye enlargement, which is detailed

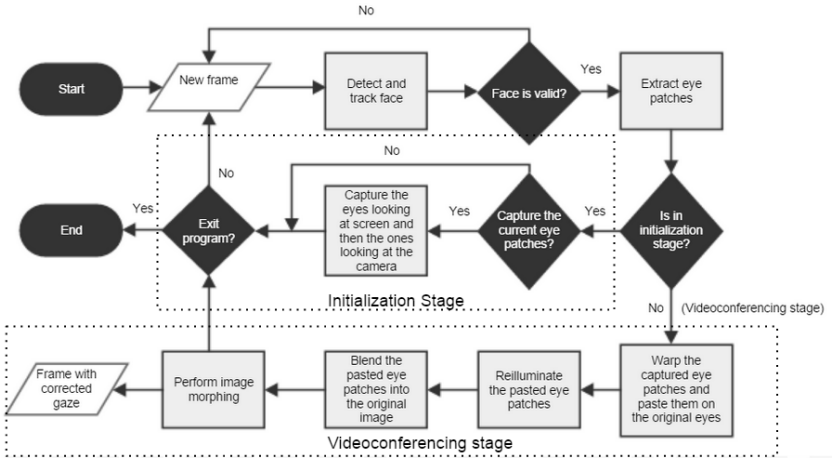


Fig. 1. Work flow of our system

in Section 3.3) and then look straight at the camera. The system captures a pair of eyes that are looking at the camera, used for replacement. Eye contact detection method like [17] can be applied in this stage to make the capture of direct looking eyes more automatic.

2. Videoconferencing stage, in which the user's eyes are tracked and the previous captured eyes are warped and pasted on the original eyes. Reillumination, blending and morphing techniques are then applied to improve the gaze correction results, which are displayed in real time.

Our system is implemented in C++ using OpenCV and OpenGL libraries.

3.2 Face Tracking

The face tracker used in our work is the IntraFace tracker [18]. As is shown in Fig. 2, the IntraFace tracker outputs 49 facial feature points in total, of which the 12 eye points are used by our system. We compute the bounding rotated box for each eye, and enlarge the width of it with a scale factor 3.5 and the height of it with a scale factor 2 to create the bounding rotated box of the eye patch. The areas within the outer rotated rectangles are what we refer as "eye patches". The purpose of this enlargement step is to make sure the pasted eyes completely cover the original eyes. The two rectangles also define the area iterated in the reillumination step, which will be explained in Section 3.3. The two scale factors are selected through experimentation to achieve the best visual result.

3.3 Eye Replacement

Warping and pasting of eyes In order to replace the eyes with the direct looking ones captured in the initialization stage, we need to first warp the captured

eyes so that they have similar shapes to the original eyes, which we approximate by an affine transformation. To achieve this we use four pairs of points (four corners of the enlarged bounding rotated box) in the source image and destination image to establish the mapping. Note that in the destination image we enlarge the eyes by some ratio in the vertical direction if the gaze direction needs to be corrected vertically, which is equal to the ratio of width of the direct looking eyes to the width of the eyes looking at the screen. The enlargement step is important to make sure the direct looking eyes look real after warping.

Reillumination The pasted eyes look very unnatural due to the fixed illumination of the pasted eye images. Ideally the illumination of the pasted eye patches should be very similar to the original eye patches, changing with the actual lighting condition. We use Wolf's [15] reillumination method due to its low computational complexity and good results. The idea is that assuming the user's face is Lambertian, then the intensity of a pixel in each of the RGB channels can be approximated using a third order polynomial [19, 20]. Unlike Wolf's method, we do not iterate through every pixel in the eye patch. Instead, we only iterate through the pixels that are in between the original rotated bounding box and the enlarged one (the area between the two rectangles around each eye in Fig. 2). This is because we found that the pixel intensities in the eye area introduce some

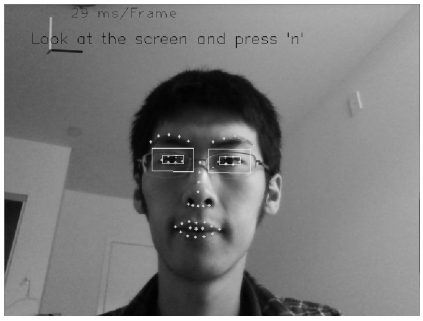
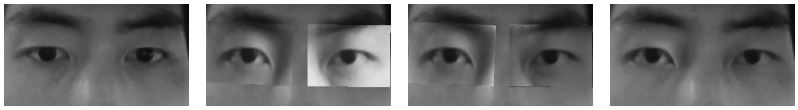


Fig. 2. Face and eye tracking result. The dots represent the feature points returned by the IntraFace tracker. Inner rectangles around the eyes are the rotated rectangles of the minimum area enclosing the eye points. Outer rectangles are the enlarged rectangles. The coordinate system at the top left represents the head pose direction.



(a) (b) (c) (d)

Fig. 3. Comparison of intermediate results: (a) original uncorrected eyes, (b) pasted eyes without any adjustment, (c) reilluminated eyes, (d) final result after Laplacian blending

(a) Original eye image (b) Pasted eye image (c) Mask (d) Blended eye image

Fig. 4. Input and output of Laplacian pyramid blending

unwanted noise to the solution, which as a result makes the reilluminated eye image noisy, having some unwanted slim shadows in the eyes. The reilluminated result is shown in Fig. 3(c).

Laplacian pyramid blending After reillumination, the contrast between the pasted eye patch and the surrounding area is greatly reduced. However the border of the eye patch is still noticeable (Fig. 4(b)). To smooth these artifacts out, we apply Laplacian pyramid blending on the eye images, which is a simple and efficient algorithm to blend one image seamlessly into another [21].

3.4 Smoothing of Transition

Due to the limitation of the face tracker, face tracking fails for some head poses. And since our eye-replacement method doesn't work well when the rotation angle of the user's head is too large, it is necessary to define a zone of correction so that gaze correction is performed only when both processes work. Specifically, the zone of correction is defined as follows:

1. The pitch and yaw angles of head rotation are within some ranges when both face tracking and eye replacement processes work well (Fig. 5). According to our tests, they are $\pm 10^\circ$ for the pitch angle and $\pm 19^\circ$ for the yaw angle.
2. Correction should be applied only when the eyes are open. The degree of openness is measured by the ratio of the height of eye box to the width of eye box. If it is larger than a threshold, a blink is detected and the correction will not be applied.

In order to provide a good user experience, we also define a transition zone (Fig. 5) in which alpha blending (cross-dissolving) is performed to make the transition smooth. To achieve this, the alpha should be changed gradually from 1.0 inside to 0.0 outside the transition zone. The algorithm that determines the value of alpha works as follows:

1. Check if the head pose direction is within the transition range of pitch angles. If it is, then alphaPitch = (maxOuterPitch - pitchAngle)/(maxOuterPitch - maxInnerPitch) (or (pitchAngle - minOuterPitch)/(minInnerPitch - minOuterPitch) depending on the side). If it is outside both the correction and transition zones, then alphaPitch = 0; Otherwise(inside the correction zone), alphaPitch = 1.

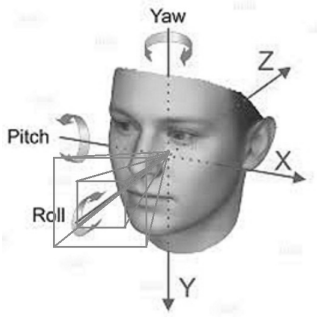


Fig. 5. Correction zone (indicated by the inner square pyramid) and transition zone (indicated by the space between the inner and outer square pyramids) of head pose. Each square pyramid, which is normal to the face plane, denotes the range of possible head pose vectors. The boundary of the outer square pyramid is determined by four angles: maxOuterPitch, minOuterPitch, maxOuterYaw, minOuterYaw. Similarly, the angles maxInnerPitch, minInnerPitch, maxInnerYaw, minInnerYaw determine the boundary of the inner square pyramid. These values are set as constants.

2. Compute alphaYaw for the yaw angles in a similar manner.
3. The initial value of alpha is the minimum of alphaPitch and alphaYaw.
4. The transition zone of eye openness is determined by minEllipseRatio and maxEllipseRatio, which are minimum and maximum ratios of the height of eye ellipse to the width of eye ellipse. Compute ratioEye as (maxEllipseRatio - curEyeRatio) / (maxEllipseRatio - minEllipseRatio) if the eye openness is inside the transition zone, 1 if it is inside the correction zone and 0 if otherwise.
5. The final alpha value is equal to the initial alpha value times ratioEye.

After we determine the value of alpha, the final image is computed as:

$$finalImage(x,y) = correctedImage(x,y) \cdot \alpha + originalImage(x,y)(1-\alpha) \quad (1)$$

According to our experiments, when the transition happens during a blink, it does not make much difference whether we apply this blending or not since the corrected eye image is very similar to the original one in appearance when the eye is nearly closed (first row in Fig. 6). However when the transition occurs during a change of head pose, we can see some small artifacts around the eye contours (last three rows in Fig. 6). This is mainly because of the small error of eye point locations returned by the face tracker; given the 22+ fps of the video sequence, these small artifacts are not easily noticed.

4 Results

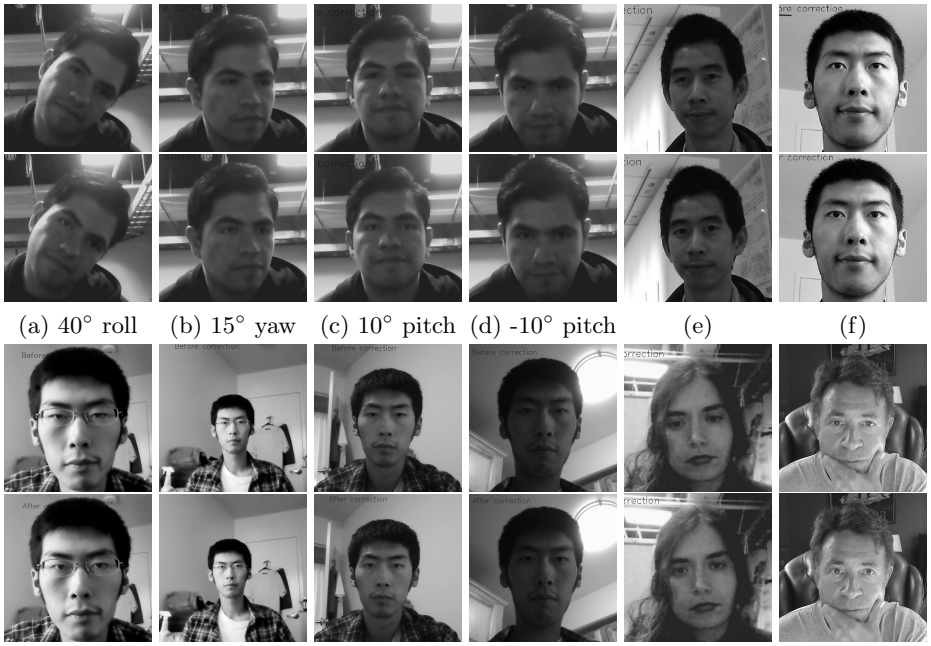
We tested our system on a laptop equipped with an Intel(R) Core(TM) i7-4710HQ CPU, 16GB RAM, and a NVIDIA GeForce GTX 860M (GPU acceleration not used). The frame rate of the built-in webcam is 22 fps for 800 × 600

input videos and 30 fps for 640 × 480 input videos. In typical usage, the face height in the camera image is half the height of the image. About one third of the time is spent on reading the video sequence from the webcam, which might vary if the video resolution changes. Face tracking takes up a very small portion of the processing time, about 5 to 10 milliseconds, which is not very dependent on the video resolution. The remaining time is spent on the eye-replacement process. Since the processing time of warping and pasting of eyes and the reillumination algorithm is dependent on the resolution of eyes and video frame, the entire processing of each frame takes longer when the face is closer or when the image resolution is higher, but for the common use settings like above our system does support real-time interaction.

We conducted several experiments to test the robustness and temporal consistency of our system. The hardware settings are similar to the ones listed above but the resolution might vary a bit since we also used an external webcam with a different resolution setting, which does not affect the gaze correction result.

Fig. 7 shows screenshots from testing the robustness against head pose changes. The system gives good correction results as long as the head pose is within the zone of correction, which is set to be $\pm 10^\circ$ of vertical rotation and $\pm 19^\circ$ of horizontal rotation. If the user's head pose is in the transition zone, morphing will be applied and the results are similar to those in Fig. 6, which still look natural. If the head pose is outside the transition zone, then no correction is applied. The test results shown in Fig. 7 and the supplementary video also demonstrate the robustness of our system against facial expressions. Fig. 7(g) and (h) illustrate the results when the user is close to and far from the webcam. As long as the user's face is within the working range of the face tracker, the gaze correction process will also work. Fig. 7(i) and (j) show the results when the lighting and background change. They belong to the same video sequence, in which the user is moving from one place to another while holding the laptop (along with the built-in webcam). Good results are achieved when the facial features are visible and the image quality doesn't change much during the movement. Fig. 7(l) and (k) show the case when the face is partially occluded by hand or hair. As long

Fig. 6. Transition from open eyes to closed eyes (first row from left to right). Transition when the user's head is turning to one side(second row from left to right). Transition when the user is lowering down his head(third row from left to right). Transition when the user is raising up his head(fourth row from left to right)



(a) 40° roll (b) 15° yaw (c) 10° pitch (d) -10° pitch (e) (f)

(g) (h) (i) (j) (k) (l)

Fig. 7. Good results. Top rows are original images and bottom rows are corrected ones. Please refer to our supplementary video for more results.

as the facial features like eyes and mouth are visible, the tracker will work and so will our gaze correction system. Our system also supports the cases when the camera is positioned to the side of the screen (Fig 7(e)) or below the screen (Fig 7(f)). According to our test results, our system works for almost all the camera positions as long as the head pose in the image is within the correction zone, which means that the disparity angle between the locations of the camera and the videoconferencing window should not be larger than 19 degrees horizontally and 10 degrees vertically.

The supplementary video shows how our system performs in real time. Overall the temporal consistency is good, i.e., the corrected eyes do not flicker much from frame to frame. It still flickers a little bit from time to time due to the noise of the image. We find that if a noise reduction algorithm is applied to each frame beforehand the results become much more stable.

The results shown in Fig. 8 demonstrate some of the major limitations of our system. When the user is looking extremely sideways (Fig. 8(a)), the large area of eye white may whiten the reilluminated eye image due to the limitation of our reillumination method. When the user's eyes are widely opened (Fig. 8(b)), the enlarged replacement eyes may differ from the actual eyes quite a lot, because of the difference between the captured direct looking eyes and the actual eyes. Moreover, if there are strong reflection of light on the glasses, the tracked eye

(a) (b) (c) (d)

Fig. 8. Unsatisfactory results. Top rows are original images and bottom rows are corrected ones. Please refer to our supplementary video for more results.

points may be very inaccurate and unstable, and therefore the final correction result will also look very bad (Fig. 8(c)).

One drawback of our gaze correction approach is that by simply replacing the original eyes with the direct looking eyes, it ignores the actual gaze direction, i.e., the gaze direction in the corrected image does not change with the actual gaze direction (Fig. 8(d)). It would be preferable to perform gaze correction only when the user is looking directly at the screen, or design a gaze correction approach that takes into account the actual gaze correction.

5 Conclusion

In this paper we presented our gaze correction system based on eye-replacement, similar to Wolf et al. [15]. Compared to their approach, ours does not require training, and we explicitly define the correction zone and handle the gaze in the transition zone. Our system also supports more camera positions around the display. According to our test results, the system has good robustness against different common lighting conditions and head poses. Our system is also shown to have a good run-time speed and provide visually convincing and natural results.

Acknowledgements. This work was supported in part by Citrix Online and by NSF Grants IIS-1423676 and IIS-1219261.

References

1. Jones, A., Lang, M., Fyffe, G., Yu, X., Busch, J., McDowall, I., Bolas, M., Debevec, P.: Achieving eye contact in a one-to-many 3d video teleconferencing system. In: ACM Transactions on Graphics (TOG). Volume 28., ACM (2009) 64
2. Okada, K.I., Maeda, F., Ichikawaa, Y., Matsushita, Y.: Multiparty videoconferencing at virtual social distance: Majic design. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work, ACM (1994) 385–393

3. Kuster, C., Popa, T., Bazin, J.C., Gotsman, C., Gross, M.: Gaze correction for home video conferencing. ACM Transactions on Graphics (TOG) **31** (2012) 174
4. Zhu, J., Yang, R., Xiang, X.: Eye contact in video conference via fusion of time-of-flight depth sensor and stereo. 3D Research **2** (2011) 1–10
5. Criminisi, A., Shotton, J., Blake, A., Torr, P.H.: Gaze manipulation for one-to-one teleconferencing. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, IEEE (2003) 191–198
6. Yang, R., Zhang, Z.: Eye gaze correction with stereovision for video-teleconferencing. In: Computer VisionECCV 2002. Springer (2002) 479–494
7. Cham, T.J., Krishnamoorthy, S., Jones, M.: Analogous view transfer for gaze correction in video sequences. In: Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on. Volume 3., IEEE (2002) 1415–1420
8. Yip, B., Jin, J.S.: An effective eye gaze correction operation for video conference using antirotation formulas. In: Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on. Volume 2., IEEE (2003) 699–703
9. Gemmell, J., Toyama, K., Zitnick, C.L., Kang, T., Seitz, S.: Gaze awareness for video-conferencing: A software approach. IEEE Multimedia **7** (2000) 26–35
10. Gemmell, J., Zhu, D.: Implementing gaze-corrected videoconferencing. In: Communications, Internet, and Information Technology. (2002) 382–387
11. Giger, D., Bazin, J.C., Kuster, C., Popa, T., Gross, M.: Gaze correction with a single webcam. In: Multimedia and Expo (ICME), 2014 IEEE International Conference on, IEEE (2014) 1–6
12. Hundt, E., Riegel, T., Schwartzel, H., Ziegler, M.: Correction of the gaze direction for a videophone (1996) US Patent 5,499,303.
13. Jerald, J., Daily, M.: Eye gaze correction for videoconferencing. In: Proceedings of the 2002 symposium on Eye tracking research & applications, ACM (2002) 77–81
14. Weiner, D., Kiryati, N.: Virtual gaze redirection in face images. In: Image Analysis and Processing, 2003. Proceedings. 12th International Conference on, IEEE (2003) 76–81
15. Wolf, L., Freund, Z., Avidan, S.: An eye for an eye: A single camera gaze-replacement method. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE (2010) 817–824
16. Kononenko, D., Lempitsky, V.: Learning to look up: Realtime monocular gaze correction using machine learning. In: Computer Vision and Pattern Recognition (CVPR), IEEE (2015)
17. Smith, B.A., Yin, Q., Feiner, S.K., Nayar, S.K.: Gaze locking: Passive eye contact detection for human-object interaction. In: Proceedings of the 26th annual ACM symposium on User interface software and technology, ACM (2013) 271–280
18. Xiong, X., De la Torre, F.: Supervised descent method and its applications to face alignment. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE (2013) 532–539
19. Basri, R., Jacobs, D.W.: Lambertian reflectance and linear subspaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on **25** (2003) 218–233
20. Bitouk, D., Kumar, N., Dhillon, S., Belhumeur, P., Nayar, S.K.: Face swapping: automatically replacing faces in photographs. In: ACM Transactions on Graphics (TOG). Volume 27., ACM (2008) 39
21. Burt, P.J., Adelson, E.H.: A multiresolution spline with application to image mosaics. ACM Transactions on Graphics (TOG) **2** (1983) 217–236