Realizing the Real-time Gaze Redirection System with Convolutional Neural Network

Chih-Fan Hsu[12], Yu-Cheng Chen[1], Yu-Shuen Wang[3], Chin-Laung Lei[2], and Kuan-Ta Chen[1]
{chihfan,uchen}@iis.sinica.edu.tw,yushuen@cs.nctu.edu.tw,cllei@ntu.edu.tw,swc@iis.sinica.edu.tw

[1]Institute of Information Science, Academia Sinica
[2]Department of Electrical Engineering, National Taiwan University
[3]Department of Computer Science, National Chiao Tung University

ABSTRACT

Retaining eye contact of remote users is a critical issue in video conferencing systems because of parallax caused by the physical distance between a screen and a camera. To achieve this objective, we present a real-time gaze redirection system called *Flx-gaze* to post-process each video frame before sending it to the remote end. Specifically, we relocate and relight the pixels representing eyes by using a convolutional neural network (CNN). To prevent visual artifacts during manipulation, we minimize not only the L2 loss function but also four novel loss functions when training the network. Two of them retain the rigidity of eyeballs and eyelids; and the other two prevent color discontinuity on the eye peripheries. By leveraging the CPU and the GPU resources, our implementation achieves real-time performance (i.e., 31 frames per second). Experimental results show that the gazes redirected by our system are of high quality under this restrict time constraint. We also conducted an objective evaluation of our system by measuring the peak signal-to-noise ratio (PSNR) between the real and the synthesized images.

CCS CONCEPTS

• **Computing methodologies → Computational photography**; **Image processing**;

KEYWORDS

Gaze Manipulation, Convolutional Neural Network

ACM Reference Format:
Chih-Fan Hsu[12], Yu-Cheng Chen[1], Yu-Shuen Wang[3], Chin-Laung Lei[2], and Kuan-Ta Chen[1]. 2018. Realizing the Real-time Gaze Redirection System with Convolutional Neural Network. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3204949.3209618

1 INTRODUCTION

Eye contact is one of essential body languages in communication. However, due to different positions of the screen and the camera on a mobile phone or a desktop, remote users are difficult to achieve eye contact during the communication. Specifically, when someone is looking at the remote user's face on the screen, he/she would be perceived as looking at another position by the remote user. This well-known issue is called *parallax*. Since solving this eye contact problem by a physical way is dramatically difficult, we redirect the eye gaze of users by image post processing techniques.

Quality and efficiency are the two challenges in building a gaze redirection system. To address these two issues, a recent research, called DeepWarp [1], trains a neural network to manipulate human gaze by estimating pixel flow fields. However, visual artifacts, such as distortions on the pupil and eyelids, frequently appear because of inconsistent movements of neighboring pixels. To improve the system, we modify the network architecture and add four novel loss functions to train the network in order to minimize visual artifacts when warping an eye gaze. Specifically, two of them retain rigidity of eyeballs and eyelids; and the other two prevent color discontinuity on the eye peripheries. Experiment results show that the eye gazes redirected by our system are visually realistic. We also compute the peak signal-to-noise ratio (PSNR) to measure the similarity between the real and the synthesized images. To implement the idea, we leverage the CPU to detect face regions and the GPU to redirect eye gaze. We also prove that our gaze redirection system can achieve real-time performance on a laptop with a consumer level graphics card.

2 RELATED WORK

Redirecting eye gaze is a popular research topic in recent years. To achieve the aim, GazeDirector [2] segments the eyes in an image and then replaces each eye region by a 3D model. Accordingly, the system rotates the 3D model to redirect eye gaze. Although this strategy achieves both high quality and accurate gaze direction, it inevitably changes the color of users' iris because of the synthesized eyeball textures. In addition, 3D eyes [3] are often too sharp and glossy so that they can appear less realistic. [1, 4] presented a different strategy, by image processing techniques, to change eye gaze. They applied a warping based neural network to generate a pixel flow field and relocate eye pixels. The network is trained by a dataset in which all directions of the eye gaze are recorded. Since the original image is manipulated, the redirected gaze can be photorealistic if the eye structure is well preserved. Although the current state-of-the-art, DeepWarp, can synthesize fairly good results by manipulating the forward gaze, it cannot perform in real-time and potentially distorts users' eyes when relocating pixels. We overcome the above-mentioned problems by simplifying the network structure, and by minimizing the loss functions that penalize structure distortions and color discontinuities when training the network.

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06.
https://doi.org/10.1145/3204949.3209618

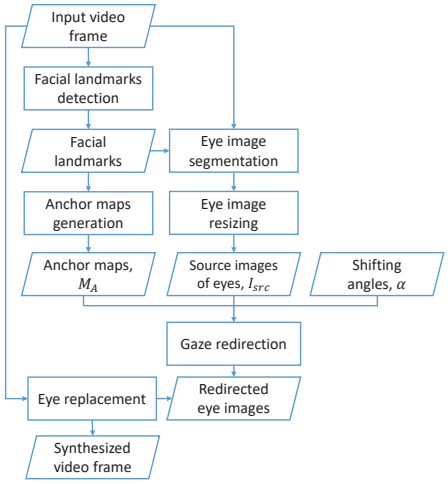


Figure 1: The system framework of Flx-gaze.

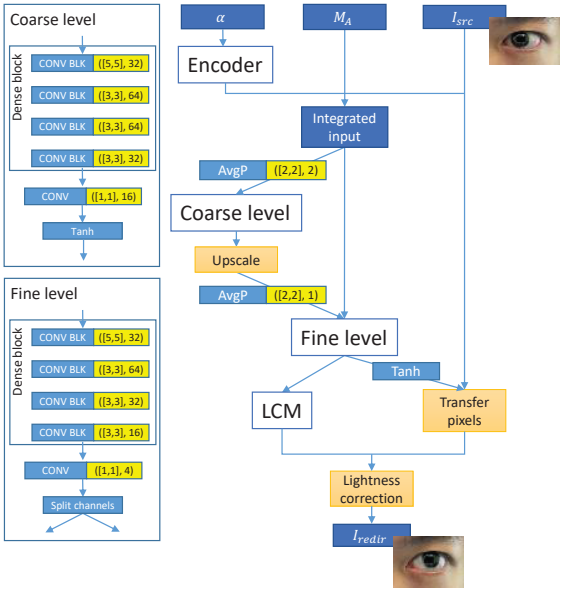


Figure 2: The structure of the proposed network. AvgP represents the average pooling layer. The elements in yellow boxes indicate the size and the number of kernels.

3 FLX-GAZE: A GAZE REDIRECTION SYSTEM

We present a real-time system, called Flx-gaze, which can synthesize realistic gaze redirected images. Figure 1 shows the system framework. In the beginning, we obtain streaming video frames with 640×480 resolution by using a webcam. 68 facial features are then detected by the dlib library [5], in which Histogram of Oriented Gradients (HOG) and a linear classifier are implemented. Among the 68 features, 12 of them (named anchors) that describe the eyes are used to crop the eye regions. To reduce the computation cost, we down sample the input frames before the detection of facial features. After the detection, we project the obtained anchor coordinates back to the original frame resolution. Considering that the input of a neural network should be of a predefined resolution, we resize each eye region to contain 64×48 pixels.

Our system takes three inputs to redirect the gaze of an eye: 1) the user's gaze (denoted by I_{src}), 2) a set of anchor maps (denoted by M_A), and 3) a 2D vector (denoted by α) that describes the angle shift in horizontal and vertical directions. Once the gaze redirected images are synthesized, we replace the original eyes by the synthetic eyes in each video frame to fix the eye contact problem.

3.1 The Region of an Eye

By observing the shape of human eyes, we set the aspect ratio of an eye region to 4:3. Let the leftmost and the rightmost anchors for an eye be a_ℓ and a_r, respectively. We set the width of the eye to $1.5 \times L$, where L is the horizontal distance between a_ℓ and a_r. Since the upper eyelid is more stretchable than the lower eyelid, we set the eye's center E_c to the position slightly higher than the center of a_ℓ and a_r. Namely, $E_c = (0, 0.09375 \times L) + 0.5(a_\ell + a_r)$.

3.2 Gaze Redirection System

We apply a warping based CNN to redirect eye gaze. Figure 2 shows the network architecture. As indicated, the network contains four sub-structures: 1) encoder, 2) coarse level, 3) final level, and 4) the light correction module (LCM). The encoder is used to extend the dimensions of a view angle α to 64×48 so as to concatenate with the eye image and the eye feature maps. Specifically, the encoder consists of a FC(16)-ReLU-FC(16)-ReLU-FC(16)-ReLU-DUP layer structure, where FC(x) represents the fully connection layer, x is the number of hidden nodes in the layer, ReLU is an activation function, and DUP is a duplication layer to duplicate the 16D vector to a $64 \times 48 \times 16$ map. The concatenated maps are then fed into the coarse and the fine level, which consist of a dense block and a bottleneck convolutional layer (CONV). The dense block contains a set of CONV BLK layers, and the output of each CONV BLK is fed to all later CONV BLKs. Each layer contains a CONV-ReLU-BN layer structure, where BN indicates the batch normalization. Our system changes only the eye regions, whereas the remaining areas of each video frame are untouched. Pasting the gaze redirected eyes back to the original video frame would easily cause discontinuity artifacts. Therefore, LCM is added to the back of the network. The LCM contains a CONV BLK-CONV BLK-CONV-SSM layer structure, where SSM is a spatial softmax layer. The outputs of the LCM are two weight maps, which are used to linearly interpolate a white color and the gaze redirected image. Note that the sum of each pixel in these two maps equals to one.

3.3 Additional Loss Functions

The goal of the gaze redirection network is minimizing the distance between I_{redir} and the target gaze image I_{tar}. In our implementation, we apply L2 distance to determine the deviation between I_{redir} and I_{tar} because it is sensitive to outliers. To improve the network by human knowledge, we present two shape-based loss functions to preserve the rigidity of eye structures when warping the eye image. We also introduce two LCM guiding loss functions to guild the LCM adjusting pixel brightness in a proper way.

3.3.1 Shape-based loss functions. Observing that humans can only rotate the eyeballs and move the eyelids up and down with little degree of freedom, we add the rigidity constraints to these two regions when warping the eyes. In other words, pixels in the same region should have similar motions. Let p be a pixel coordinate. We define the flow of pixel p by $flow(p)$. We also apply $R(p)$ and $L(p)$ to indicate whether the pixel p represents the pupil ($R(p) = 1$) or not ($R(p) = 0$) and the pixel's brightness, respectively. To retain the pupil's shape, we minimize the term

$$loss_{eyeball} = \sum_{p \in I} R(p) \cdot (1 - L(p)) \cdot (|\frac{\partial flow(p)}{\partial x}| + |\frac{\partial flow(p)}{\partial y}|). \quad (1)$$

The discrete partial derivative $\partial flow(p)/\partial x$ and $\partial flow(p)/\partial y$ can be calculated by subtracting the flows of neighboring pixels. Namely, $flow(p_{x+1,y}) - flow(p_{x,y})$ and $flow(p_{x,y+1}) - flow(p_{x,y})$. In the case of eyelids, the constraints are formulated in a similar way, except the pupil mask. That is, we minimize the term

$$loss_{eyelid} = \sum_{p \in I} (1 - R(p)) \cdot (|\frac{\partial flow(p)}{\partial x}| + |\frac{\partial flow(p)}{\partial y}|). \quad (2)$$

We point out that the loss function $loss_{eyelid}$ is also helpful to prevent discontinuity problems that appear at the periphery of an eye region I because of the small pixel movements.

3.3.2 LCM guiding loss function. We expect the LCM not to adjust the color/brightness of the pixels close to the periphery of an eye region I because the gaze redirected eye will be pasted back to the original frame. Therefore, we add the term

$$loss_{adj} = \sum_{p \in I} G(p) \cdot lcm_{wc}(p), \quad (3)$$

where $lcm(p)$ represents the weight maps which are inferred from LCM and wc represents the white color one, and $G(p)$ is the guidance, in which the value increases from the eye center to the periphery. In other words, by minimizing the $loss_{adj}$, the weight $lcm_{wc}(p)$ would be close to 0 if p is close to the periphery of I. We also expect the brightness adjustment to be smooth so that sharp dots do not appear in the synthesized images. Therefore, we present the term

$$loss_{sgl} = \sum_{p \in I} G(p) \cdot (|\frac{\partial lcm(p)}{\partial x}| + |\frac{\partial lcm(p)}{\partial y}|). \quad (4)$$

In our implementation, we set $G(p)$ to

$$G(p) = \beta \times \sqrt[2]{(p_x - E_{c,x})^8 + (p_y - E_{c,y})^8} + \gamma, \quad (5)$$

where $p = (p_x, p_y)$, the eye's center $E_c = (E_{c,x}, E_{c,y})$, and we set $\beta = 3$ and $\gamma = 0.05$ to define the influence region.

By minimizing the weighted sum of the aforementioned five loss functions, Flx-gaze achieves high quality gaze redirected images, and we set the weights of these functions to one in our experiment.

4 IMPLEMENTATION, RESULTS, AND EVALUATION

4.1 Network Training

We train the presented network by Tensorflow on a desktop with Intel® Core™ i7 and two NVIDIA GeForce GTX 1080 graphics cards. Figure 3 shows how the two graphics cards speed up the training process. Specifically, networks on the two graphics cards have the same parameters. In each step, different training batches are fed into the two networks, respectively. Since the obtained gradients for back propagation on these two networks are different, we first average the gradients and then use the gradients to update the two networks. In addition, we set the optimizer to Adam, [6], batch size to 256, and learning rate to 10^{-3}.

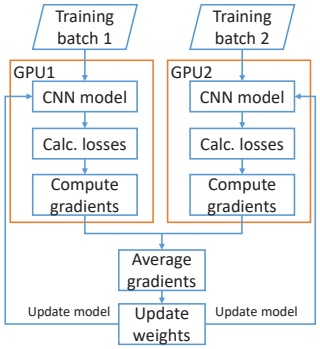


Figure 3: The training procedure on two graphics cards.

We collect our own training data set in the laboratory environment because there are no publicly available data sets that contain accurate features for defining eye regions. Specifically, an identical light condition and a green background are set when recording eye gazes from volunteers. The dataset was collected from 37 volunteers. Each of them was asked to gaze at 100 different directions when recording. Considering the physical limitation, we recorded the volunteers' gaze directions from -40° to 40° in horizontal and from -30° to 30° in vertical. Among the 100 recorded gazes, 63 of them are of fixed directions and the remaining 37 are randomly sampled. To ensure the data quality, we manually removed the image if the volunteer's eyes were closed. Overall, our collected data set contains 3,650 gaze images (360,208 gaze pairs). We partitioned the gaze images by separating the volunteers into the training and the validation groups which contain 35 and 2 volunteers (3,451 and 199 gaze images), respectively. We note here that the training and the validation sets have 340,407 and 19,801 pairs, respectively.

4.2 Real-time Eye Gaze Redirection System

We applied the trained neural network to redirect eye gaze of the input image. Figure 4 shows two examples. On the left are the original eye gaze images and on the right are the redirected versions. In each example, the upper row shows the gaze from left to right, and the lower row shows the gaze from bottom to top, respectively. Figure 5 shows the synthesized images which the redirected eyes are pasted back to the original images. As can be seen, the results generated by our system are visually realistic.

4.3 Evaluations

We tested our system on a laptop with Intel® Core™ i7 and a NVIDIA GeForce GTX 960M GPU. The test video is obtained by a built-in webcam with 640x480 resolution.

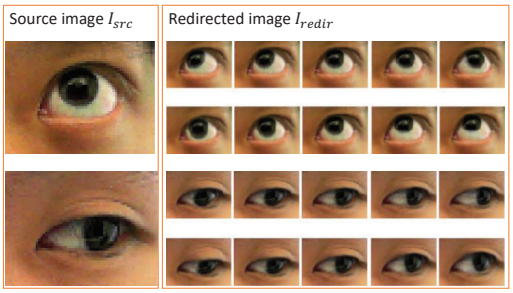
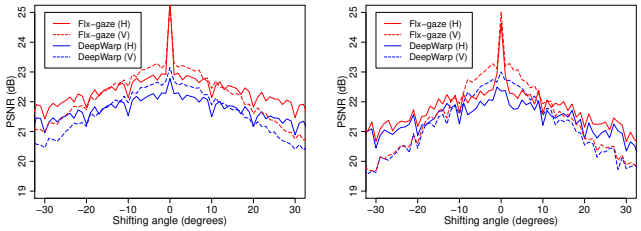


Figure 4: Gaze redirected by our system.

Figure 5: The input (at middle) and the synthesized images. The man's gaze is controlled by a small dot.

4.3.1 Performance Evaluation. We evaluate the performance of our system by estimating the computation time for processing a video frame. To obtain a convincing statistics, we count the number of frames processed in a minute and then determine the time taken by a frame. Specifically, our system takes 31.9 ms to redirect the eye gaze to a frame, including the face detection (15.2 ms), eye segmentation (5.3 ms), gaze redirection (10.8 ms), and other minor steps. The current performance is bounded by the built-in webcam on the tested laptop, which achieves only 30 FPS, and our system can be improved by the parallel processing.

4.3.2 Quality Evaluation. We evaluate the gaze redirected images objectively on the validation set. To achieve the aim, the real and the synthesized eye gaze images (I_{tar} and I_{redir}) are compared and the PSNR is determined. Figure 6(b) shows the results. The value on a line indicates the averaged PSNR value which is calculated by fixed one dimension of the shifting angle α at a certain angle of vertical (V) or horizontal (H) directions. In the training set (Figure 6(a)), the PSNRs of Flx-gaze are all higher than the PSNRs of the DeepWarp. In the validation set (Figure 6(b)), Flx-gaze outperforms DeepWarp as well if the shifting angle is between -20° and 20°. We point out that when an image is processed by the DeepWarp or Flx-Gaze, there is no guarantee that the input image will not be modified with a very small shifting angle, because the image has been passed through the network. However, Flx-gaze actually learns the concept – the input eye images should not be modified as much as possible when the shifting angle near 0°. In addition, we observed that the PSNRs in horizontal direction are higher than the PSNRs in vertical direction. The result is reasonable because redirecting an eye gaze vertically has to warp not only the eyeball but also the eyelids.



(a) Training set. **(b) Validation set.**

Figure 6: Image quality evaluation.

4.4 Demo

We will prepare a consumer level laptop with a built-in camera to demo our gaze redirection system. When users are looking at someone on the screen, the system will compute the shifting angle to redirect the gaze in real time. The redirection can be switched on and off for the comparison.

5 CONCLUDING REMARKS

We presented a real-time gaze redirection system, Flx-gaze, which is based on a warping based CNN. To achieve high quality results, we introduce five loss functions to train the network. Specifically, the rigidity of eyeballs and eyelids is retained, and the color discontinuity on eye peripheries is avoided. To evaluate our system, we compared the Flx-gaze to DeepWarp and show that the PSNRs of Flx-gaze are higher than those of DeepWarp in most situations. We also demonstrated that our system can achieve real-time performance on a consumer level laptop. Accordingly, by using Flx-gaze, solving the eye contact problem in video conferencing can be expected.

REFERENCES

[1] Y. Ganin, D. Kononenko, D. Sungatullina, and V. Lempitsky. *DeepWarp: Photorealistic Image Resynthesis for Gaze Manipulation*, pages 311–326. Springer International Publishing, Cham, 2016.

[2] E. Wood, T. Baltrusaitis, L.P. Morency, P. R., and A. Bulling. Gazedirector: Fully articulated eye gaze redirection in video. *CoRR*, abs/1704.08763, 2017.

[3] E. Wood, T. Baltrušaitis, L.P. Morency, P. Robinson, and A. Bulling. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pages 131–138, 2016.

[4] D. Kononenko and V. Lempitsky. Learning to look up: Real-time monocular gaze correction using machine learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2015.

[5] D.E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[6] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.