GazeDirector: Fully Articulated Eye Gaze Redirection in Video

Erroll Wood[1,4] Tadas Baltrušaitis[2] Louis-Philippe Morency[2] Peter Robinson[1] Andreas Bulling[3]
[1]University of Cambridge, UK [2]Carnegie Mellon University, USA
[3]Max Planck Institute for Informatics, Germany [4]Microsoft

Abstract

We present GazeDirector, a new approach for eye gaze redirection that uses model-fitting. Our method first tracks the eyes by fitting a multi-part eye region model to video frames using analysis-by-synthesis, thereby recovering eye region shape, texture, pose, and gaze simultaneously. It then redirects gaze by 1) warping the eyelids from the original image using a model-derived flow field, and 2) rendering and compositing synthesized 3D eyeballs onto the output image in a photorealistic manner. GazeDirector allows us to change where people are looking without person-specific training data, and with full articulation, i.e. we can precisely specify new gaze directions in 3D. Quantitatively, we evaluate both model-fitting and gaze synthesis, with experiments for gaze estimation and redirection on the Columbia gaze dataset. Qualitatively, we compare GazeDirector against recent work on gaze redirection, showing better results especially for large redirection angles. Finally, we demonstrate gaze redirection on YouTube videos by introducing new 3D gaze targets and by manipulating visual behavior.[1]

1. Introduction

Gaze redirection is an upcoming research topic in computer vision where the goal is to alter an image to change where someone appears to be looking (see Figure 1) [12, 34]. This is an important generalization of the classic gaze correction problem [10, 43], in which someone's gaze is adjusted along a single direction to simulate eye contact. With gaze redirection, gaze can be adjusted in any direction.

The ability to freely change where someone is looking paves the way for a variety of compelling new applications (see Figure 2). For example, taking a group picture with everyone is looking at the camera at the same time can be difficult [30]. Imagine a gaze-correcting camera that could always enforce eye contact, no matter where people are actually looking. Also, one challenge for actors nowadays is performing alone before other computer-generated characters are composited in. Where are they supposed to look? With gaze redirection their apparent point-of-regard could be controlled in post-production, ensuring they look at virtual characters. Gaze direction is also an important social signal [11] – the ability to redirect gaze or even impose specific visual behaviours on video content in real-time could serve as a useful experimental tool, e.g. to study gaze following or joint attention in autism research [18].

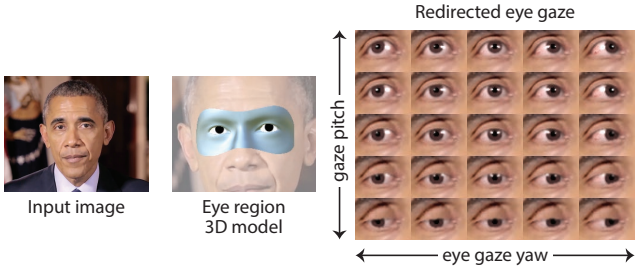

Figure 1. GazeDirector is a new 3D model based approach for gaze redirection. We first recover the shape and appearance of the eyes by fitting a 3D eye region model. We then redirect gaze by warping the eyelids and rendering new redirected eyeballs. Examples of redirected gaze can be seen on the right.

A reliable and robust gaze redirection algorithm should work with previously unseen people and handle desired gaze directions which differ significantly from the original gaze. Thies et al. [34] recently proposed an approach which requires per-user calibration – a tedious process that is unsuitable for many scenarios. More relevant to our goal of user-independent gaze redirection is DeepWarp [12], an approach that uses a deep neural network to directly predict an image-warping flow field between two eye images with a known gaze "correction" angular offset between them. This flow field is applied to the original image to redirect gaze. In this way, DeepWarp can only redirect gaze by shifting it by an angular offset; it cannot specify new gaze directions explicitly. Furthermore, this approach is prone to producing unsightly artefacts when redirecting gaze over large angles. This problem is fundamental in any purely warping-based approach since it is impossible to warp parts of the eye that were occluded in the original image.

In this work we present *GazeDirector*, a new approach for

[1] https://www.youtube.com/watch?v=-tDaZk9V1Nw

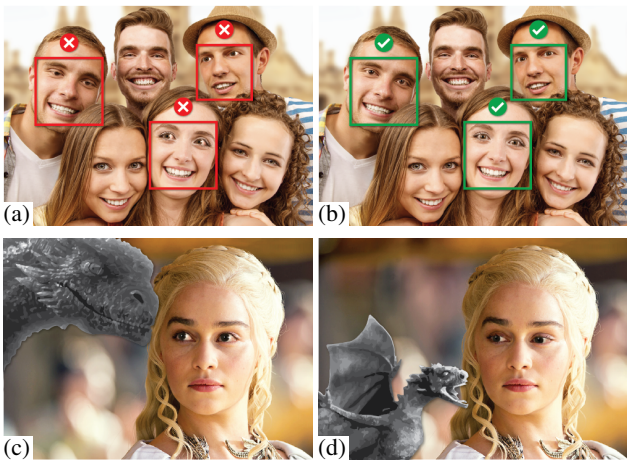

Figure 2. GazeDirector enables new applications that were previously impossible. (a) Taking group pictures with everyone looking at the camera can be tricky. (b) A gaze correcting camera can ensure this is always the case. (c,d) A challenge for actors is knowing where to look before visual effects are added to a scene. This can be modified in post-production, so if a CGI character is changed, the actor's gaze can be adjusted accordingly. The highlighted faces in (a,b) and the face in (c,d) have been modified by GazeDirector.

person-independent gaze redirection. The main idea of our approach is to model the eye region in 3D instead of trying to predict a flow field directly from an input image [12]. Since we recover the shape and pose of the eyes in 3D, our approach can redirect gaze with *full articulation*: GazeDirector can precisely specify new desired gaze targets or directions in 3D instead of using gaze angle correction offsets [12]. To model the eye in 3D, we extend a recently proposed method [40] to fit a 3D morphable model of the eye region to both eyes in an input image using analysis-by-synthesis. Once we have recovered the 3D shape, pose, and appearance of the eyes we redirect gaze in two steps. First, we compute a dense model-derived flow field corresponding to eyelid motion between the original and desired gaze directions. This dense flow field is efficiently extrapolated from sparse per-vertex flow values using GPU rasterization. We apply this flow field to the input image to warp the eyelids. Second, we render and composite our redirected eyeball models onto the output image in a photorealistic manner.

Contributions 1) Our primary contribution is GazeDirector – a new method that demonstrates how eye-region model fitting using analysis-by-synthesis enables superior gaze redirection compared to previous approaches (§3). In addition, we present the following secondary contributions: 2) A practical approach for rapid synthesis of dense model-derived optical flow fields using GPU rasterization (§5.1). 3) Improvements over the state-of-the-art in gaze estimation using our dataset-independent model fitting approach (§6.1).

2. Related Work

Eye gaze manipulation The lack of eye contact during video-conferencing is a well-known problem. In computer vision, there are three main approaches to tackle it: 1) novel-view synthesis, 2) eye-replacement, and 3) eye-warping.

Novel-view synthesis methods re-render the subject's face so they appear to be looking at the camera. The first step is recovering a dense depthmap of the face – this can been done with stereo vision [10, 42], RGB-D (color with depth) cameras [24], and monocular RGB cameras [15]. This facial depthmap is then rotated and re-rendered from a new viewpoint along a frontal gaze path. However, as these methods distort the face as a whole, they are not suitable for more general forms of of gaze manipulation.

Eye-replacement methods replace eyes in the original image with new eye images representing different gaze. The most realistic approaches collect a set of person-specific images of eyes looking at a camera, and composite them into the original face [26, 29, 39]. These methods require person-specific eye images to pick from, and encounter issues when compositing eyes across different head poses or illumination conditions. Other eye-replacement approaches synthesize new eyeballs with graphics [14, 37]. However, these methods do not move the eyelids – an important cue for vertical gaze, and only use rudimentary 2D graphics techniques that ignore iris color, head pose, and scene illumination. Our method instead synthesizes new eyeballs taking eyelid motion, iris color, and illumination into account.

Warping-based methods can redirect gaze without requiring person-specific training data. These methods learn to generate a flow field from one eye image to another using training pairs of eye images with known gaze offsets between them. This flow field is used to warp pixels in the original image, thus modifying gaze [12, 22]. However, purely warping-based methods suffer three major limitations: First, they can only offset the original unknown gaze direction, so cannot specify a new gaze direction explicitly. Second, the range of possible redirection is limited by the gaze directions in the training set. Third, warping artefacts appear for large redirection angles as parts of the eye that were originally occluded cannot be synthesized correctly. Using 3D models, GazeDirector can explicitly specify new gaze directions in 3D, without training data, and without introducing artefacts.

Like us, Banf and Blanz [2] used morphable models to redirect gaze. They fit a single-part face model to an image, and redirect gaze by deforming the eyelids using an example-based approach, and sliding the iris across the model surface using texture-coordinate interpolation. Since they use a mesh where the face and eyes are joined, their method only works when people look straight ahead. GazeDirector instead models the face and eyeballs as separate parts, letting it work for non-frontal input gaze and allowing the eyeball to rotate separately from the eyelids, as it does in real life.

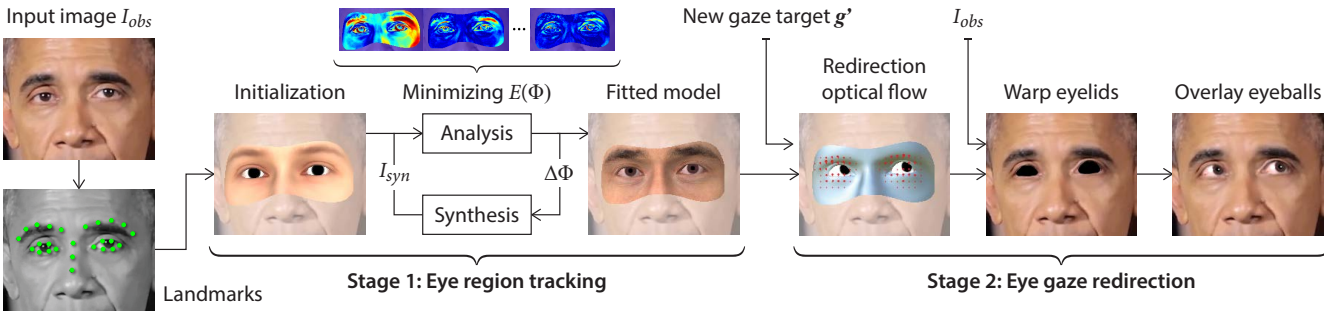

Figure 3. Given observed image I_{obs}, we first initialize our model using 25 facial landmarks from a face tracker [1]. We then find optimal model parameters $\mathbf{\Phi}^*$ using analysis-by-synthesis, minimizing a reconstruction energy $E(\mathbf{\Phi})$. We then modify $\mathbf{\Phi}^*$ with the desired gaze and eyelid behaviour, resulting in a new I_{syn} which we blend onto I_{obs}, giving a redirected gaze image.

Facial performance capture

Since GazeDirector recovers the shape, texture, pose, and gaze of the facial eye region, it is also related to work on monocular facial perfomance capture – a well established research topic [21]. The goal is to recover dynamic facial geometry and appearance using commodity cameras alone.

Monocular facial performance capture is a highly under-constrained problem, so a parametric face model [6] is often used as a prior to help recover shape and albedo. Such models can then be fit to either RGB-D data [33, 38] or RGB data [7, 31, 32]. However, these approaches generally avoid the eyes, cutting them out of the mesh [8, 32]. This is because the parametric face model they use only represents the surface of the skin, and has reduced fidelity around the eye due to poor correspondences in the source head scan data. For GazeDirector, we extended a previous model that was built using high quality scans [40], with care taken to maintain correspondences around the eyelids and eye corners. Critically, this model treats the eyeballs as separate parts that move independently from the face.

Some previous work tracked the eyes as a part of the face. Garrido et al. [13] include eyeball geometry in a "detail" layer of their facial mesh. Though this can lead to acceptable re-rendering, it does not allow gaze redirection as the eyeballs and face are joined in a single mesh. Suwajanakorn et al. [31] model eyeball movement by interpolating between facial textures. This does not allow smooth arbitrary eyeball motion, and requires a large training set of person-specific images with eye movement. Recent work has combined a facial skin surface capture system with a separate gaze tracker [32, 36, 9]. Our approach instead captures the facial eye region and eyeball simultaneously. This lets us reliably recover eyeball shape and texture parameters – important for realistic gaze redirection.

There have been recent breakthroughs in capturing the eyeballs and eyelids in extreme detail using special equipment [3, 4, 5]. Our work does not come close to this level of detail. Instead, we focus on capturing the eye for gaze redirection in commodity monocular images and video.

3. Overview

As shown in Figure 3, our approach consists of two main stages: eye region tracking and eye gaze redirection.

Tracking Given a monocular RGB image frame, we first capture the eyes by fitting our eye region model. This model consists of two parts: a generative facial part and an articulated eyeball part. It is defined by a set of parameters $\mathbf{\Phi}$ that describe shape, texture, pose, and scene illumination. We fit our model to the image using analysis-by-synthesis, searching for optimal parameters $\mathbf{\Phi}^*$ by minimizing a photometric reconstruction energy.

Redirection We redirect gaze in two steps: 1) We warp the eyelids in the original image using a flow field derived from our 3D model. We efficiently calculate this flow field by re-posing our eye region model to change gaze, and rendering the image-space flow between tracked and re-posed eye regions. 2) We then render the redirected eyeballs and composite them back into the image. We blur the boundary between the skin and eyeball to soften the transition.they "fit in" better.

4. Eye region tracking

For our gaze redirection to look plausible, we must first recover the original shape and texture of the eye region. Given an image frame I_{obs}, we therefore wish to recover a set of optimal parameters $\mathbf{\Phi}^*$ that best explains it in terms of our eye region model. We search for $\mathbf{\Phi}^*$ using analysis-by-synthesis: iteratively rendering a synthetic eye region image I_{syn}, comparing it to I_{obs} using our reconstruction energy E (defined in Equation 4), and updating $\mathbf{\Phi}$ accordingly.

4.1. Eye region model

At the heart of our method lies a multi-part eye region model based on that by Wood et al. [40]. For GazeDirector, we extended it to model two eyes rather than one, simplified the iris color model to improve robustness, and added aesthetic improvements (subdivision surfaces, ambient occlusion) to improve realism. Our model contains four main

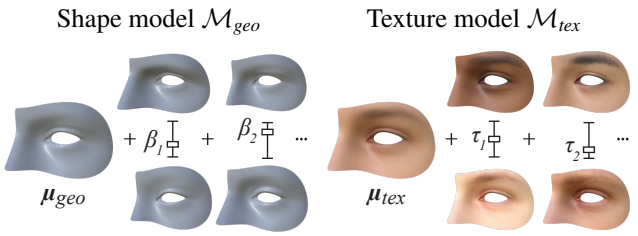

Figure 4. The average facial shape $\boldsymbol{\mu}_{geo}$ and texture $\boldsymbol{\mu}_{tex}$, along with the top modes of variation. The first mode of shape variation moves between hooded and protruding eyes, and the first mode of texture variation moves between dark and light skin.

parts: the left and right facial eye regions, and the left and right eyeballs. It is parameterized by $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} = \{\boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\theta}, \boldsymbol{\iota}\}, \tag{1}$$

where $\boldsymbol{\beta}$ are the set of shape parameters, $\boldsymbol{\tau}$ the texture parameters, $\boldsymbol{\theta}$ the pose parameters, and $\boldsymbol{\iota}$ the illumination parameters. We now describe each parameter below.

Shape $\boldsymbol{\beta}$ The geometric shape of each eye region is described by a linear Principal Component Analysis (PCA) model $\mathcal{M}_{geo} \in \mathbb{R}^{3n}$ in the style of previous work [6]. This comprises $n = 229$ vertices and was built from a collection of 22 high resolution scans acquired online [41]. We assume faces are symmetrical, so the shapes of both eye regions are controlled with a single set of coefficients $\boldsymbol{\beta}_{face} \in \mathbb{R}^{16}$,

$$\mathcal{M}_{geo}(\boldsymbol{\beta}_{face}) = \boldsymbol{\mu}_{geo} + \mathbf{U}\text{diag}(\boldsymbol{\sigma}_{geo})\boldsymbol{\beta}_{face} \tag{2}$$

where $\boldsymbol{\mu}_{geo}$ is the average face shape, $\mathbf{U}$ the modes of shape variation, and $\boldsymbol{\sigma}_{geo}$ the standard deviations of these modes (see Figure 4). For simplicity, each $\beta_i \in \boldsymbol{\beta}_{face}$ is scaled so that $\beta_i = 1$ represents one standard deviation's worth of variation in that dimension. For the eyeball we use a standard two-sphere model based off physiological averages [27]. We also include a parameter β_{iris} that controls iris size by scaling vertices on the iris boundary about the pupil.

Texture $\boldsymbol{\tau}$ We use a linear PCA texture model $\mathcal{M}_{tex} \in \mathbb{R}^{3m}$ of the facial eye region, built from the same set of scans. Rather than model the color of each vertex [6], $\mathcal{M}_{tex}$ generates RGB texture maps sized $m = 512 \times 512$px that we apply to both eye regions. This linear texture model is controlled with texture coefficients $\boldsymbol{\tau}_{face} \in \mathbb{R}^8$,

$$\mathcal{M}_{tex}(\boldsymbol{\tau}_{face}) = \boldsymbol{\mu}_{tex} + \mathbf{V}\text{diag}(\boldsymbol{\sigma}_{tex})\boldsymbol{\tau}_{face} \tag{3}$$

where $\boldsymbol{\mu}_{tex}$ is the average face texture, $\mathbf{V}$ the modes of texture variation, and $\boldsymbol{\sigma}_{tex}$ the respective standard deviations. Each coefficient is scaled in a similar way to $\mathcal{M}_{geo}$, so it represents one standard deviation in its dimension. As shown in Figure 5, we vary the iris by multiplying the iris region of the base eyeball texture with an RGB color $\boldsymbol{\tau}_{iris}$. Since the

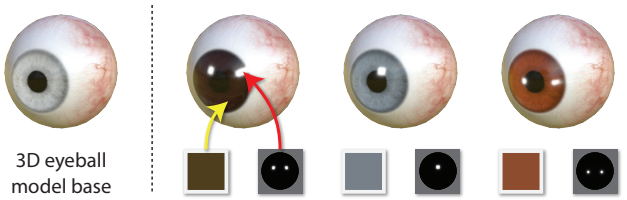

Figure 5. Our eyeball model captures iris color variation with an RGB color $\boldsymbol{\tau}_{iris}$ (yellow arrow). Environmental reflections are added with spherical environment maps (red arrow).

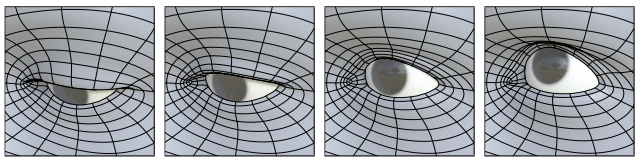

Figure 6. Our eyelid posed using procedural animation for eyelid gaze pitch angles θ_{lid} between $-20°$ and $+20°$.

"white of the eye" is rarely purely white, we also tint it with another color $\boldsymbol{\tau}_{tint}$

Pose $\boldsymbol{\theta}$ Our pose parameters describe both global and local pose. Globally, the eye regions are positioned with rotation $\theta_{\boldsymbol{R}}$ and translation $\theta_{\boldsymbol{T}}$. The interocular distance is controlled via θ_{iod} The eyeball positions are fixed in relation to the eye regions. Our local pose parameters allow the eyeballs to rotate independently from the face, controlling gaze. The general gaze direction is given by pitch and yaw angles θ_p and θ_y, and vergence is controlled with θ_v. When the eyeball looks up or down, the eyelids follow it. We use procedural animation to pose the eyelids in the facial mesh by rotational ammount θ_{lid} [41].

Illumination $\boldsymbol{\iota}$ We assume a simple illumination model of ambient light coupled with a single directional light. The ambient light has intensity $\boldsymbol{\iota}_{amb} \in \mathbb{R}^3$, and the directional light has intensity $\boldsymbol{\iota}_{dir} \in \mathbb{R}^3$ and direction defined by rotation $\boldsymbol{\iota}_R \in \mathbb{R}^2$ (pitch and yaw angles). We assume all surfaces are Lambertian. Though $\boldsymbol{\iota}$ cannot describe complex scene illumination, we found it was sufficient in many cases considering the small facial region that we consider.

In total we have $17 + 14 + 11 + 9 = 51$ parameters of $\boldsymbol{\Phi}$ to optimize over.

Rendering the model Once our model has been configured with parameters $\boldsymbol{\Phi}$, we render synthetic images $I_{syn}(\boldsymbol{\Phi})$ using a DirectX-based rasterizer. We fix our virtual camera location at the world origin, and assume knoweldge (or estimate) of camera intrinsic parameters.

Realistically rendering eyes is a challenge [27]. We implement three additional effects to improve the realism of our output. First, as our model is low-resolution, it appears blocky when rendered. We therefore smooth the skin's surface using a single step of Loop subdivision [25] with pre-

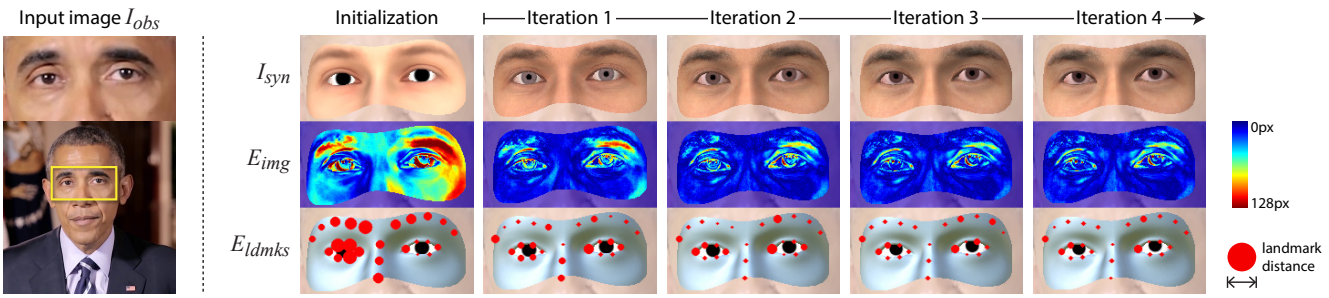

Figure 7. We fit our 3D eye region model to an image my minimizing a reconstruction energy $E(\mathbf{\Phi})$. Our two main energy terms are a dense photometric error term E_{img} and a sparse landmark similarity term E_{ldmks}. This figure shows the energies decreasing over four iterations of the Gauss Newton algorithm.

computed stencils for efficiency. Second, we use physically correct corneal refraction techniques in the eyeball shader to better model its layered transparent structure [17]. Third, we approximate ambient occlusion shadowing on the eyeball using a single-pass analytic techniqe: we project the positions of eyelid vertices into eyeball uv space, fit a 2D cubic polynomial to them, and apply per-pixel ambient occlusion as a function of distance to each eyelid polynomial.

4.2. Energy formulation

A good energy function is critical to the success of any analysis-by-synthesis method. Our proposed energy $E(\mathbf{\Phi})$ is a weighted sum of several terms, each encoding a different requirement of our model fit. Each term is expressable as a sum-of-squares, allowing us to minimize $E(\mathbf{\Phi})$ using the Gauss-Newton algorithm.

$$E(\mathbf{\Phi}) = \underbrace{E_{img}(\mathbf{\Phi}) + E_{ldmks}(\mathbf{\Phi})}_{\text{Data terms}} + \underbrace{E_{stats}(\mathbf{\Phi}) + E_{pose}(\mathbf{\Phi})}_{\text{Prior terms}} \quad (4)$$

Our data terms (see Figure 7) guide our model fit using image pixels and facial landmarks, while our priors penalize unlikely facial shape and texture, and eyeball orientations. We now describe each term in detail.

Image similarity E_{img} Our primary goal is to minimize the photometric reconstruction error between I_{syn} and I_{obs}. The data term E_{img} expresses how well the fitted model explains I_{obs} by densely measuring pixel-wise differences across the images using a robust mean squared error. We promote image similarity with the term

$$E_{img}(\mathbf{\Phi}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \rho\big(|I_{syn}(p) - I_{obs}(p)| \big)^2 \quad (5)$$

where $\mathcal{P} \subset I_{syn}$ represents the set of rendered foreground pixels belonging to our 3D model. The background pixels are ignored. The robust function $\rho(e) = \min(\sqrt{T}, e)$, for threshold T, alleviates the effects of outliers; this is important for recovering iris color in the presence of strong specular highlights on the eye.

Landmark similarity E_{ldmks} The face contains several landmark feature points that can be tracked reliably. We therefore regularize our dense data term (E_{img}) using a sparse set of landmarks $\mathcal{L}$ provided by a face tracker [1]. $\mathcal{L}$ consists of 25 points that describe the eyebrows, nose and eyelids. For each 2D tracked landmark $l \in \mathcal{L}$, we also compute a corresponding synthesized 2D landmark l' as a linear combination of projected vertices in our shape model. Facial landmark similarities are incorporated into our energy using

$$E_{ldmks}(\mathbf{\Phi}) = \lambda_{ldmks} \cdot \frac{1}{|\mathcal{P}|} \sum_{i=0}^{|\mathcal{L}|} \|l_i - l'_i\|^2 \quad (6)$$

As landmark distances $\|l_i - l'_i\|$ are measured in image-space, we normalize the energy by dividing through by foreground area $|\mathcal{P}|$ to avoid bias from eye region size in the image. The importance of E_{ldmks} is controlled with weight λ_{ldmks}.

Statistical prior E_{stats} We penalize unlikely facial shape and texture using a statistical prior [6]. As we assume a normally distributed population, our PCA model parameters should be close to the mean $\mathbf{0}$:

$$E_{stats}(\mathbf{\Phi}) = \lambda_{geo} \cdot \sum_{i=0}^{|\boldsymbol{\beta}|} \beta_i^2 + \lambda_{tex} \cdot \sum_{i=0}^{|\boldsymbol{\tau}|} \tau_i^2 \quad (7)$$

Recall that $\beta_i \in \boldsymbol{\beta}$ and $\tau_i \in \boldsymbol{\tau}$ are scaled by their respective standard deviations in our model. This energy helps our fit avoid degenerate facial shapes and texture, and guides its recovery from poor local minima found in previous frames. The penalties for unlikely shape and texture are weighted separately with λ_{geo} and λ_{tex}.

Pose prior E_{pose} Our final energy penalizes mismatched parameters for eyeball gaze direction and eyelid position. The eyelids follow eye gaze, so if the eyeball is looking upwards, the eyelids should be rotated upwards, and visa versa. We enforce eyelid pose consistency with

$$E_{pose}(\mathbf{\Phi}) = \lambda_{pose} \cdot \|\theta_{lid} - \theta_p\|^2 \quad (8)$$

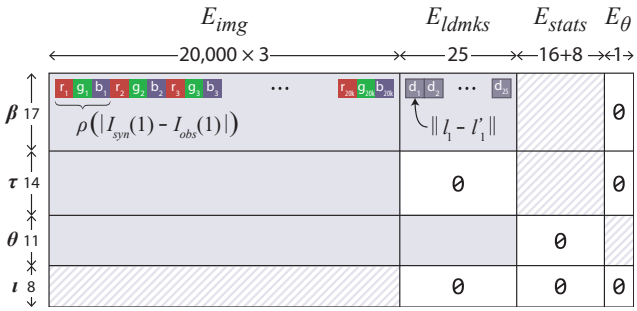

Figure 8. The non-zero structure of our Jacobian $\mathbf{J_r}$ for a 200× 100px eye region. $\mathbf{J_r}$ is calculated entirely on the GPU. Dashed regions represent sparse blocks.

where θ_{lid} is the eyelid pitch angle of our model's face parts, and θ_p is the gaze pitch angle of our eyeball parts. Its relative importance is controlled by weight λ_{pose}.

4.3. Optimization procedure

Minimizing our proposed objective $E(\mathbf{\Phi})$ is a challenging high-dimensional non-convex optimization problem. We use a GPU-assisted, annealed form of the Gauss-Newton algorithm, where the parameter update for $\mathbf{\Phi}$ is as follows:

$$\mathbf{\Phi}^{i+1} = \mathbf{\Phi}^i - \eta^i \left(\mathbf{J_r}^T \mathbf{J_r}\right)^{-1} \cdot \mathbf{J_r}^T \mathbf{r} \qquad (9)$$

where $\mathbf{r}$ is the vector of energy function residuals, $\mathbf{J_r}$ the Jacobian matrix of residuals $\mathbf{r}$ evaluated at $\mathbf{\Phi}^i$, $\mathbf{J_r}^T \mathbf{J_r}$ the approxmation to the Hessian matrix, and η the annealing rate. We perform a variable number of Gauss-Newton iterations, terminating early if no more progress is being made. Figure 7 shows four iterations of our model fit.

To compute the Jacobian we use numerical central derivatives. This is an expensive operation, requiring two images to be rendered for every parameter. We keep our system performant by calculating $\mathbf{J_r}$ and $\mathbf{J_r}^T \mathbf{J_r}$ entirely on the GPU, avoiding expensive pipeline stalls from cross-system data transfer. Additionally, since image rendering is a key operation for our system, we use a tailored DirectX rasterizer that can render I_{syn} over 5000 times per second. To further lighten the computational load of our numerical derivatives, we mask out a subset of $\mathbf{\Phi}$ when tracking in a video, so optimize over a smaller set of parameters frame-to-frame. As a result, GazeDirector can run at interactive rates.

Initialization The energy landscape of $E(\mathbf{\Phi})$ is riddled with local minima, so we must start from a good initializion. Our face tracker provides 3D estimates for the facial landmark positions. We initialize global translation to the mean landmark position and set global rotation parameters using the the Kabsch [19] algorithm. Other parameters are initialized to **0** by default, except for interocular distance and iris size, for which we use anthropomorphic averages, and illumination, for which we experimentally chose a basic setup.

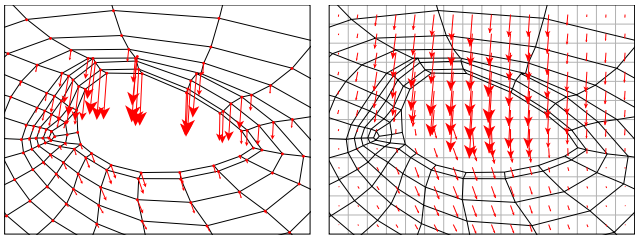

Figure 9. We efficiently convert sparse per-vertex image-space flows (left) to a dense per-pixel flow field (right) using GPU rasterization. We use this dense flow-field to warp the eyelids.

When tracking in video, we exploit temporal similarities by initializing $\mathbf{\Phi}_{init}$ with $\mathbf{\Phi}^*$ from the previous frame.

5. Eye gaze redirection

Once we have obtained a set of fitted model parameters $\mathbf{\Phi}^*$ for an image I_{obs}, our next step is to redirect gaze to point at a new 3D target $\boldsymbol{g}'$.

We first modify $\mathbf{\Phi}^*$ to obtain $\mathbf{\Phi}'$ that represents the redirected gaze. We then calculate the optical flow between eye region models with $\mathbf{\Phi}^*$ and $\mathbf{\Phi}'$, and use this to warp the eyelids in the source image. Finally, we render the redirected eyeballs and seamlessly composite them into the output image.

Re-posing our model The first step of gaze redirection is straightforward: given a new target $\boldsymbol{g}'$, we calculate new values for eye gaze pitch θ'_p, yaw θ'_y, and vergence θ'_v so each eyeball points towards $\boldsymbol{g}'$. Furthermore, we calculate θ'_{lid} to match the new gaze direction. Altogether, these new gaze parameters are encoded in $\mathbf{\Phi}'$.

5.1. Warping the eyelids

When the eyeball rotates, the eyelids move with it. To simulate this, we warp the eyelids from the original image using a model-derived optical flow field $\boldsymbol{O}$. To calculate $\boldsymbol{O}$, we first calculate the sparse screen-space flow $\boldsymbol{o}_i \in \mathbb{R}^2$ for each vertex $\boldsymbol{v}_i \in \mathbb{R}^3$ in both facial parts of the eye region:

$$\boldsymbol{o}_i = \Pi\left(\Theta'(\boldsymbol{v}_i)\right) - \Pi\left(\Theta^*(\boldsymbol{v}_i)\right) \quad i \in [0, 458] \qquad (10)$$

where Π is the projection defined by our camera parameters, and $\Theta^{*|\prime}$ are the transforms that combine eyelid motion (θ_{lid}) with model-to-world transforms $\theta_{\boldsymbol{R}}$ and $\theta_{\boldsymbol{T}}$. It is common for analysis-by-synthesis methods to use GPU rasterization to evaluate an objective function [28, 32]. We propose a simple and efficient approach for computing dense flow-fields using the same framework. To efficiently distribute sparse flow values across image space, we load per-vertex flows $\boldsymbol{o}_i$ into our renderer as vertex attributes and let the rasterization stage interpolate between them and handle occlusions between different model parts (see Figure 9). This takes ∼5ms. The result is a dense flow field $\boldsymbol{O}$ that we use to remap source image pixels to simulate eyelid motion.

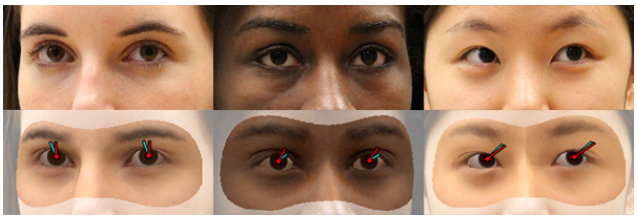

Figure 10. Eye region model fits on the Columbia gaze dataset [30] showing true gaze (red) and estimated gaze (cyan).

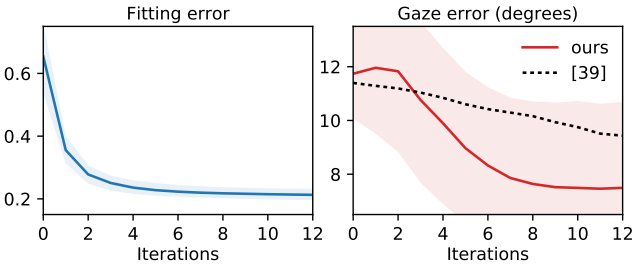

Figure 11. Fitting error and gaze error for the Columbia dataset [30] decrease with the number of fitting iterations. Line is median, filled region is interquartile range. Our second-order optimization strategy converges faster than previous first-order methods [40].

5.2. Compositing redirected eyeballs

Once the eyelids have been warped, we render the portion of the eyeballs between the eyelids and composite them onto the output image. Following rasterization, the eyelid edges will be perfectly sharp and unlikely to match the observed image. We therefore follow the approach adopted by the real-time rendering community [17, 20], and blur the seam where the eyeballs meet the eyelids with a small Gaussian.

A shortcoming of our underlying scene model is the lack of specular reflections on the eyeball surface. Real world eye images often exhibit strong highlights or glints. We decided not to explicitly model multiple light sources in Φ because of the additional computational cost with numerical derivatives. We instead pre-rendered a set of five spherical reflection maps that model common environmental lighting scenarios (see Figure 5), and use them to apply specular reflections on the eyeball at runtime. This choice is made by seeking the reflection map that minimizes image error. While this cannot model complex environmental reflections, it improves the perceived quality of the eyeball re-rendering.

6. Evaluations

In this section we evaluate GazeDirector. Quantitatively, we evaluate our model fitting stage with a gaze estimation experiment, and our gaze synthesis stage with a gaze redirection experiment. Qualitatively, we compare our method against recent work and demonstrate gaze redirection and visual behaviour manipulation on YouTube videos.

Figure 12. DeepWarp (top rows) [12] and GazeDirector (bottom rows) showing horizontal gaze redirection up to 45°. Our model based approach avoids the smudging artefacts encountered from large redirection angles with DeepWarp.

6.1. Model fitting performance

We performed an experiment to assess our fitting strategy. We measured two factors: 1) photometric error to determine how well we reconstructed the image, and 2) gaze estimation error to see if we can correctly recover eyeball pose. We used the Columbia gaze dataset [30], which contains images of 56 people looking at a target grid on the wall. The participants were constrained by a head-clamp, and images were taken from five different head orientations. In our experiments we used a subset of 34 people (excluding those with eyeglasses) with 20 images per person.

Results of our experiment can be seen in Figure 11, and example model fits can be seen in Figure 10. Photometric error and gaze estimation error decrease with the number of model fitting iterations. This confirms the effectiveness of our fitting strategy. If we examine the pitch and yaw components of gaze separately, we outperform recent work [16] in terms of gaze yaw (3.13° vs 3.51°), though perform worse in terms of gaze pitch (6.92° vs 4.27°). This result is promising since GazeDirector operates in a dataset agnostic manner, while previous work [16] was trained on the Columbia dataset specifically. Furthermore, our second-order optimization strategy leads to faster convergence than first-order methods used in previous work [40], despite performing a similar amount of computation per iteration.

6.2. Gaze redirection

We performed an experiment to evaluate our gaze redirection stages. We prepared another subset of the Columbia gaze dataset [30] with neutral head pose. We aligned images of each participant using facial landmarks [1], and used the aligned images with different gaze as ground truth for "redirected gaze". Following model fitting on the frontal gaze image, we produced three output images for each different gaze image: a) with no gaze redirection, b) with gaze redirection with the eyeballs only, and c) with gaze redirection with eyeballs and eyelids. We measured the per-pixel im-

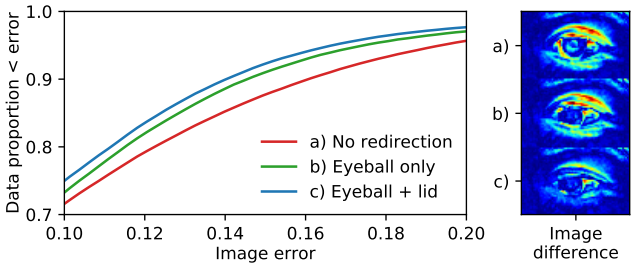

Figure 13. Redirection error decreases as we enable more parts of our redirection pipeline. The x-axis represents image error, and the y-axis represents the proportion of data under that error.

age difference between GazeDirector images and the ground truth redirected gaze images (see Figure 13). The benefits of both eyeball redirection and eyelid redirection are clear.

Comparison to DeepWarp [12] Previous work produces unsightly smudging artefacts when starting from non-central gaze, and redirecting gaze over large angles. This is because their method fails to correctly hallucinate parts of the eyeball that were originally occluded. As can be seen in Figure 12, these issues do not arise with GazeDirector as we use a 3D model. Furthermore, since DeepWarp can only apply an angular gaze offset to an input gaze direction, it cannot be used to produce results like those in Figure 14 where someone has been made to look at 3D gaze targets. Please see our supplementary video for additional comparisons.

6.3. Redirecting gaze in YouTube videos

We demonstrate GazeDirector on videos with a variety of eye appearances, head pose, and illumination conditions by redirecting gaze in YouTube videos. We downloaded videos from YouTube and resized them to a resolution of 640×480px. New 3D gaze targets were specified through physics simulations and procedural programming using the Unity engine [35]. Figure 14 shows some examples. Please refer to our supplementary video for the full results.

Runtime GazeDirector runs on a commodity desktop machine (3.3Ghz CPU, NVidia GTX 1080). Runtime is split between fitting and redirection. We first process the entire video to recover $\mathbf{\Phi}^*$ for each frame. This model fitting stage ran at 11.6fps, 12.5fps, and 12.1fps for the three YouTube videos in Figure 14. We then redirect gaze for each frame in the video. Gaze redirection is less computationally demanding, and ran at 80fps for each video.

7. Discussion

In this work we described GazeDirector, a novel method for gaze redirection that uses model-fitting. Unlike previous work, GazeDirector does not require person-specific training data, and can redirect eye gaze to new 3D targets explicitly. We fit a parametric eye region model to images using analysis-by-synthesis, minimizing a reconstruction energy to recover shape, texture, pose, gaze, and illumination simultaneously. Gaze redirection is then performed by warping eyelids, and compositing eyeballs onto the output in a photorealistic manner.

Limitations remain. We do not explicitly model a full range of facial expressions such as blinking or squinting. Furthermore, we do not handle occlusions or distortion effects from eyeglasses [23]. Our model does not include the eyelashes – these are hard to model realistically, but can provide an important cue for downwards looking eye gaze. We also do not consider cast shadows from hooded eyes or eyelashes. Despite these limitations, we believe our work will enable a range of interesting and novel applications.

Acknowledgements

This work was funded, in part, by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University, Germany.

References

[1] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "OpenFace: an open source facial behavior analysis toolkit," in *IEEE WACV*, 2016.

[2] M. Banf and V. Blanz, "Example-based rendering of eye movements," in *Computer Graphics Forum*, 2009.

[3] P. Bérard, D. Bradley, M. Nitti, T. Beeler, and M. H. Gross, "High-quality capture of eyes." *ACM Trans. Graph.*, vol. 33, no. 6, pp. 223–1, 2014.

[4] P. Bérard, D. Bradley, M. Gross, and T. Beeler, "Lightweight eye capture using a parametric model," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 117, 2016.

[5] A. Bermano, T. Beeler, Y. Kozlov, D. Bradley, B. Bickel, and M. Gross, "Detailed spatio-temporal reconstruction of eyelids," *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015.

[6] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proc. 26th conf. on Computer graphics and interactive techniques*, 1999.

[7] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," *ACM Transactions on Graphics (TOG)*, 2014.

[8] C. Cao, D. Bradley, K. Zhou, and T. Beeler, "Real-time high-fidelity facial performance capture," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 46, 2015.

[9] C. Cao, H. Wu, Y. Weng, T. Shao, and K. Zhou, "Real-time facial animation with image-based dynamic avatars," *ACM Transactions on Graphics (TOG)*, 2016.

[10] A. Criminisi, J. Shotton, A. Blake, and P. H. Torr, "Gaze manipulation for one-to-one teleconferencing," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 191–198.

[11] N. J. Emery, "The eyes have it: the neuroethology, function and evolution of social gaze," *Neuroscience & Biobehavioral Reviews*, vol. 24, no. 6, pp. 581–604, 2000.

[12] Y. Ganin, D. Kononenko, D. Sungatullina, and V. Lempitsky, "Deepwarp: Photorealistic image resynthesis for gaze

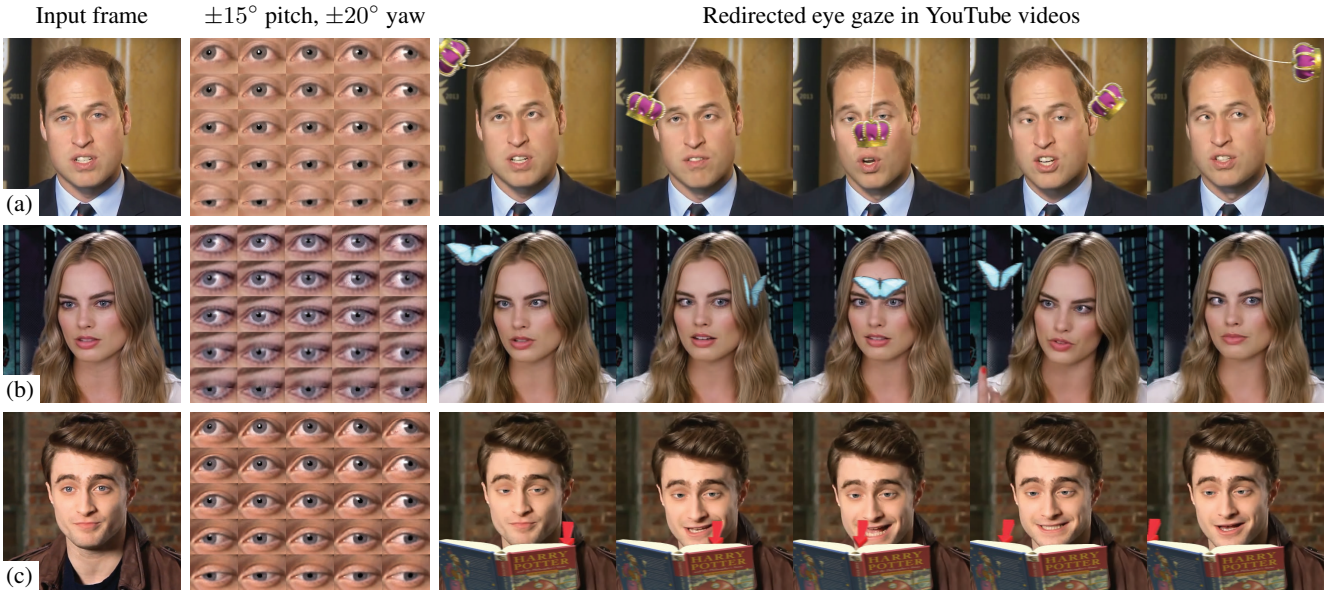

Figure 14. Example input frame, redirected eye gaze grid, and example output frames for three separate YouTube videos. (a,b): gaze has been redirected to new 3D gaze targets. (c): we have modified visual behaviour, making the video subject appear to read a book.

manipulation," in *European Conference on Computer Vision*. Springer, 2016, pp. 311–326.

[13] P. Garrido, M. Zollhoefer, D. Casas, L. Valgaerts, K. Varanasi, P. Perez, and C. Theobalt, "Reconstruction of personalized 3d face rigs from monocular video," 2016.

[14] J. Gemmell, K. Toyama, C. L. Zitnick, T. Kang, and S. Seitz, "Gaze awareness for video-conferencing: A software approach," *IEEE Multimedia*, 2000.

[15] D. Giger, J.-C. Bazin, C. Kuster, T. Popa, and M. Gross, "Gaze correction with a single webcam," in *Intl. Conf. on Multimedia and Expo (ICME)*. IEEE, 2014.

[16] L. A. Jeni and J. F. Cohn, "Person-independent 3d gaze estimation using face frontalization," in *Proc. CVPR Workshops (CVPRW)*. IEEE, 2016.

[17] J. Jimenez, E. Danvoye, and J. von der Pahlen, "Photorealistic eyes rendering," in *SIGGRAPH Talks, Advances in Real-Time Rendering*. ACM, 2012.

[18] E. A. Jones and E. G. Carr, "Joint attention in children with autism theory and intervention," *Focus on autism and other developmental disabilities*, vol. 19, no. 1, pp. 13–26, 2004.

[19] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.

[20] B. Karis, T. Antoniades, S. Caulkin, and V. Mastilovic, "Digital humans: Crossing the uncanny valley in unreal engine 4," in *GDC Talks*. EPIC, 2016.

[21] O. Klehm, F. Rousselle, M. Papas, D. Bradley, C. Hery, B. Bickel, W. Jarosz, and T. Beeler, "Recent advances in facial appearance capture," in *Computer Graphics Forum*. Wiley Online Library, 2015.

[22] D. Kononenko and V. Lempitsky, "Learning to look up: realtime monocular gaze correction using machine learning," in *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.

[23] T. C. Kübler, T. Rittig, E. Kasneci, J. Ungewiss, and C. Krauss, "Rendering refraction and reflection of eyeglasses for synthetic eye tracker images," in *Proc. ETRA*. ACM, 2016.

[24] C. Kuster, T. Popa, J.-C. Bazin, C. Gotsman, and M. Gross, "Gaze correction for home video conferencing," *ACM Transactions on Graphics (TOG)*, 2012.

[25] C. Loop, "Smooth subdivision surfaces based on triangles," 1987.

[26] Y. Qin, K.-C. Lien, M. Turk, and T. Höllerer, "Eye gaze correction with a single webcam based on eye-replacement," in *Advances in Visual Computing*. Springer, 2015.

[27] K. Ruhland, S. Andrist, J. Badler, C. Peters, N. Badler, M. Gleicher, B. Mutlu, and R. Mcdonnell, "Look me in the eyes: A survey of eye and gaze animation for virtual agents and artificial systems," in *Eurographics State-of-the-Art Report*, 2014, pp. 69–91.

[28] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proc. 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015.

[29] Z. Shu, E. Shechtman, D. Samaras, and S. Hadap, "Eyeopener: Editing eyes in the wild," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 1, p. 1, 2016.

[30] B. Smith, Q. Yin, S. Feiner, and S. Nayar, "Gaze Locking: Passive Eye Contact Detection for HumanObject Interaction," in *ACM User Interface Software and Technology (UIST)*, 2013.

[31] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "What makes tom hanks look like tom hanks," in *Proc. International Conference on Computer Vision (ICCV)*. IEEE, 2015.

[32] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and

M. Nießner, "Face2Face: Real-time Face Capture and Reenactment of RGB Videos," in *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[33] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, "Real-time expression transfer for facial reenactment," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 183, 2015.

[34] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Facevr: Real-time facial reenactment and eye gaze control in virtual reality," *arXiv preprint arXiv:1610.03151*, 2016.

[35] Unity, "Game engine," *Online – http://unity3d.com*.

[36] C. Wang, F. Shi, S. Xia, and J. Chai, "Realtime 3d eye gaze animation using a single rgb camera," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 118, 2016.

[37] D. Weiner and N. Kiryati, "Virtual gaze redirection in face images," in *Proc. Conf. Image Analysis and Processing*. IEEE, 2003.

[38] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4. ACM, 2011, p. 77.

[39] L. Wolf, Z. Freund, and S. Avidan, "An eye for an eye: A single camera gaze-replacement method," in *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010.

[40] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "A 3d morphable eye region model for gaze estimation," in *European Conference on Computer Vision*. Springer, 2016.

[41] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Learning an appearance-based gaze estimator from one million synthesised images," in *Proc. ETRA*. ACM, 2016.

[42] R. Yang and Z. Zhang, "Eye gaze correction with stereovision for video-teleconferencing," in *European Conference on Computer Vision*. Springer, 2002, pp. 479–494.

[43] C. L. Zitnick, J. Gemmell, and K. Toyama, "Manipulation of video eye gaze and head orientation for video teleconferencing," *Microsoft Research MSR-TR-99-46*, 1999.