FaceVR: Real-Time Gaze-Aware Facial Reenactment in Virtual Reality

JUSTUS THIES, Technical University Munich
MICHAEL ZOLLHÖFER, Stanford University
MARC STAMMINGER, University of Erlangen-Nuremberg
CHRISTIAN THEOBALT, Max-Planck-Institute for Informatics
MATTHIAS NIEßNER, Technical University Munich

We propose *FaceVR*, a novel image-based method that enables video teleconferencing in VR based on self-reenactment. State-of-the-art face tracking methods in the VR context are focused on the animation of rigged 3D avatars (Li et al. 2015; Olszewski et al. 2016). Although they achieve good tracking performance, the results look cartoonish and not real. In contrast to these model-based approaches, FaceVR enables VR teleconferencing using an image-based technique that results in nearly photo-realistic outputs. The key component of FaceVR is a robust algorithm to perform real-time facial motion capture of an actor who is wearing a head-mounted display (HMD), as well as a new data-driven approach for eye tracking from monocular videos. Based on reenactment of a prerecorded stereo video of the person without the HMD, FaceVR incorporates photo-realistic re-rendering in real time, thus allowing artificial modifications of face and eye appearances. For instance, we can alter facial expressions or change gaze directions in the prerecorded target video. In a live setup, we apply these newly introduced algorithmic components.

CCS Concepts: • **Computing methodologies** → **Computer vision**; **Computer graphics**;

Additional Key Words and Phrases: Face tracking, virtual reality, eye tracking

1 INTRODUCTION

Modern head-mounted virtual reality displays, such as the Oculus Rift or HTC Vive, are able to provide very believable and highly immersive stereo renderings of virtual environments to a user. In particular, for teleconferencing scenarios, where two or more people at distant locations meet (virtually) face-to-face in a virtual meeting room, VR displays can provide a far more immersive and connected atmosphere than today's teleconferencing systems. These teleconferencing systems usually employ one or several video cameras at each end to film the participants, whose video(s) are then shown on one or several standard displays at the other end. Imagine one could take this to the next level, and two people in a VR teleconference would each see a photo-realistic 3D rendering of their actual conversational partner, not simply an avatar, but in their own head-mounted display (HMD). The biggest obstacle in making this a reality is that while the HMD allows for very immersive rendering, it is a large physical device which occludes the majority of the face. In other words, even if each participant of a teleconference was recorded with a 3D video rig, whose feed is streamed to the other end's HMD, natural conversation is not possible due to the display occluding most of the face. Recent advancements in VR displays are flanked by great progress in face performance capture methods. State-of-the-art approaches enable dense reconstruction of dynamic face geometry in real time, from RGB-D (Bouaziz et al. 2013; Hsieh et al. 2015; Li et al. 2013; Siegl et al. 2017; Weise et al. 2011; Zollhöfer et al. 2014) or even RGB cameras (Cao et al. 2014a, 2015; Thies et al. 2016). A further step has been taken by recent RGB-D (Thies et al. 2015) or RGB-only (Thies et al. 2016) real-time facial re-enactment methods. In the aforementioned VR teleconferencing setting, a facial self-reenactment approach can be used to show the unoccluded face of each participant on the VR display at the other end. Unfortunately, the stability of many real-time face capture methods suffers if the tracked person wears an HMD. Furthermore, existing reenactment approaches cannot transfer the appearance of eyes, including blinking and eye gaze—yet exact reproduction of the facial expression, including the eye region, is crucial for conversations in VR.

In our work, we therefore propose *FaceVR*, a new real-time facial reenactment approach that can transfer facial expressions and realistic eye appearance between a source and a target actor video. Eye movements are tracked using an infrared camera inside the HMD, in addition to outside-in cameras tracking the unoccluded face regions (see Figure 1). Using the self-reenactment described above, where the target video shows the source actor without the HMD, the proposed approach, for the first time, enables live VR teleconferencing. To achieve this goal, we make several algorithmic contributions:

—Robust real-time facial performance capture of a person wearing an HMD, using an outside-in RGB-D camera stream,

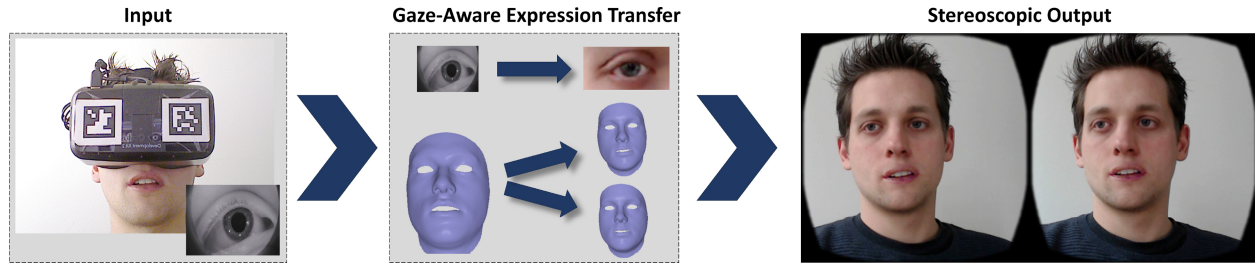**Input**   **Gaze-Aware Expression Transfer**   **Stereoscopic Output**

Fig. 1. We present FaceVR, a novel method to perform real-time gaze-aware facial reenactment with a virtual reality device (left). To capture a face, we use a commodity RGB-D sensor with a frontal view; the eye region is tracked using a new data-driven approach based on data from an IR camera located inside the HMD. Using the 3D reconstructed face as an intermediate, we can modify and edit the face in the target video, as well as re-render it in a photo-realistic fashion, allowing for a variety of applications (e.g., removal of VR goggles or gaze re-targeting). In addition, we render our output in stereo (right), which enables display on stereo devices such as other VR headsets.

with rigid and non-rigid degrees of freedom, and an HMD-internal camera.

— Real-time eye-gaze tracking with a novel classification approach based on random ferns, for video streams of an HMD-internal camera or a regular webcam.

— Facial reenactment with photo-realistic re-rendering of the face region including the mouth and the eyes, using model-based shape, appearance, and lighting capture.

— An end-to-end system for facial reenactment in VR, where the source actor is wearing an HMD and the target actor is recorded in stereo.

2   RELATED WORK

A variety of methods exist to capture detailed static and dynamic face geometry with specialized controlled acquisition setups (Klehm et al. 2015). Some methods use passive multi-view reconstruction in a studio setup (Beeler et al. 2011; Borshukov et al. 2003; Fyffe et al. 2014; Pighin and Lewis 2006), optionally with the support of invisible makeup (Williams 1990) or face markers (Huang et al. 2011). Methods using active scanners for capture were also developed (Weise et al. 2009; Zhang et al. 2004).

Many approaches employ a parametric identity model (Blanz et al. 2003; Blanz and Vetter 1999), and face expression (Tena et al. 2011). Blend shape models are widely used for representing the expression space (Lewis et al. 2014; Pighin et al. 1998), and multi-linear models jointly represent the identity and expression space (Shi et al. 2014; Vlasic et al. 2005). Newer methods enable dense face performance capture in more general scenes with more lightweight setups, such as a stereo camera (Valgaerts et al. 2012), or even just a single RGB video at offline frame rates (Fyffe et al. 2014; Garrido et al. 2013; Shi et al. 2014; Suwajanakorn et al. 2014). Garrido et al. (2016) reconstruct a fully controllable parametric face rig including reflectance and fine scale detail, and Suwajanakorn et al. (2015) build a modifiable mesh model of the face. Ichim et al. (2015) reconstruct a game-type 3D face avatar from static multi-view images and a video sequence of face expressions. More recently, methods reconstructing dense dynamic face geometry in real time from a single RGB-D camera (Bouaziz et al. 2013; Hsieh et al. 2015; Li et al. 2013; Weise et al. 2011; Zollhöfer et al. 2014) were proposed. Some of them estimate appearance and illumination along with geometry (Thies et al. 2015). Using

trained regressors (Cao et al. 2014a, 2015), or parametric model fitting, dense dynamic face geometry can be reconstructed from monocular RGB video (Thies et al. 2016). Recently, Cao et al. (2016) proposed an image-based representation for dynamic 3D avatars that supports various hairstyles and parts of the upper body.

The ability to reconstruct face models from monocular input data enables advanced image and video editing effects. Given a portrait of a person, a limitless number of appearances can be synthesized (Kemelmacher-Shlizerman 2016) based on face replacement and internet image search. Examples for video editing effects are re-arranging a database of video frames (Li et al. 2012) such that mouth motions match a new audio stream (Bregler et al. 1997; Taylor et al. 2015), face puppetry by reshuffling a database of video frames (Kemelmacher-Shlizerman et al. 2010), or re-rendering of an entire captured face model to make mouth motion match a dubbed audio-track (Garrido et al. 2015). Other approaches replace the face identity in a target video (Dale et al. 2011; Garrido et al. 2014). When face expressions are modified, it is often necessary to re-synthesize the mouth and its interior under new or unseen expressions, for which image-based (Kawai et al. 2014; Thies et al. 2016) or 3D template-based (Thies et al. 2015) methods were examined. Recently, Suwajanakorn et al. (2017) presented a system that learns the mapping between audio and lip motion. This learning-based approach requires a large amount of person specific training data and control the gaze direction. Vlasic et al. (2005) describe a model-based approach for expression mapping onto a target face video, enabling offline reenactment of faces under controlled recording conditions. While Thies et al. (2015) enable real-time dense tracking and photo-realistic expression mapping between source and target RGB-D video, Face2Face (Thies et al. 2016) enables real time between captured RGB video of one actor and an arbitrary target face video. Under the hood, they use a real-time tracker capturing dense shape, appearance, and lighting. Expression mapping and image-based mouth re-rendering enables photo-realistic target appearance.

None of the aforementioned capture and reenactment approaches succeeds under strong face occlusion by a VR headset, nor can combine data from several cameras—inside and outside the display—and thus cannot realistically re-render the eye region and appearance, including correct gaze direction. Parts of our method

are related to image-based eye-gaze estimation approaches. Commercial systems exist for eye-gaze tracking of the unoccluded face using special externally placed cameras (e.g., from Tobii[1]), or IR cameras placed inside a VR headset (e.g., from Pupil Labs,[2] FOVE,[3] or SMI[4]).

Appearance-based methods for gaze detection of the unoccluded face from standard externally placed cameras were also researched (Sugano et al. 2014; Zhang et al. 2015). Wang et al. (2016) simultaneously capture 3D eye gaze, head pose, and facial expressions using a single RGB camera at real-time rates. However, they solve a different problem from ours; we need to reenact (i.e., photo-realistically synthesize) the entire eye region appearance in a target video of either a different actor, or the same actor under different illumination, from input video of an in-display camera. Parts of our method are related to gaze correction algorithms for teleconferencing where the eyes are re-rendered such that they look into the webcam, which is typically displaced from the video display (Criminisi et al. 2003; Kononenko and Lempitsky 2015; Kuster et al. 2012). Again, this setting is different from ours, as we need to realistically synthesize arbitrary eye region motions and gazes, and not only correct the gaze direction.

Related to our article is the work by Li et al. (2015), who capture moving facial geometry while wearing an HMD with a rigidly attached depth sensor. In addition, they measure strain signals with electronic sensors to estimate facial expressions of regions hidden by the display. As a result, they obtain the expression coefficients of the face model which are used to animate virtual avatars. Recently, Olszewski et al. (2016) proposed an approach for HMD users to control a digital avatar in real time based on RGB data. The user's mouth is captured by a camera that is rigidly attached to the HMD and a convolutional neural network is used to regress from the images to the parameters that control a digital avatar. They also track eyebrow motion based on a camera that is integrated into the HMD. Both of these approaches only allow one to control a virtual avatar—rather than a real video—and do not capture the eye motion. Our approach takes this a step further and captures facial performance as well as the eye motion of a person using an HMD. In addition, we allow one to re-render and reenact the face, mouth, and eye motion of a target stereo stream photo-realistically and in real time.

Recently, Google presented an approach for HMD removal in the virtual/mixed reality setting (Frueh et al. 2017), which shows the great interest in such technology. Instead of removing the entire HMD, they use translucent rendering techniques to reveal the occluded eye region. They synthesize the eye region similar to our method, based on the gaze estimation of an HMD-integrated SMI eye tracker and static face geometry. In contrast, our approach based on self-reenactment produces a stereo video of the person completely without the HMD. Furthermore, we present a lightweight eye tracking approach that is able to track eye motions and enables us to synthesize new eye motions in a photo-realistic fashion.

[1]http://www.tobii.com.
[2]http://www.pupil-labs.com.
[3]https://www.getfove.com.
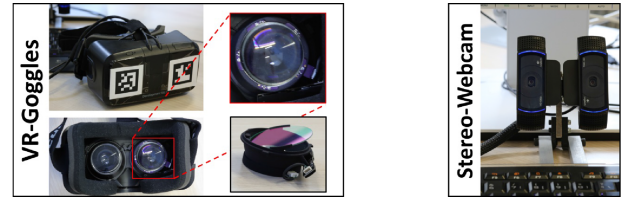[4]http://www.smivision.com.

Fig. 2. Hardware setups. A source actor experiences VR wearing an Oculus DK2 headset (left). We track the source actor using a commodity RGB-D sensor (front-facing), and augment the HMD with ArUco markers, as well as an IR webcam in the inside (mounted with Add-on Cups[2]). The target actor footage is captured with a lightweight stereo rig, which is composed of two webcams (right).

3 HARDWARE SETUP

Our approach requires two different inputs. One is called *source,* it is the live video feed of the person wearing an HMD. We call the person in this video *source actor*. In addition to this live video, we require a prerecorded stereo video of the person without the HMD. This stereo video is the *target video* and the person in that video is called *target actor*. Note that for self-reenactment the *source* and *target actor* are the same person. The *source actor* is wearing an HMD, and we use a lightweight hardware setup to reconstruct and track the source actor's face. To this end, we augment commodity VR goggles with a simple IR webcam on the inside for tracking one eye. For tracking the rigid pose and facial expressions, we use outside-in tracking based on a real-time RGB-D sensor (Asus Xtion Pro), as well as ArUco AR markers on the front panel of the HMD.

The tracking and reconstruction pipeline for the *target actor* differs. Here, we use a stereo setup which is composed of two commodity webcams. This allows for robust face tracking and generation of 3D video content that we can display on an HMD (which is the case for VR teleconferencing). We prerecord the target actor's video stream, but we modify and replay it in real time. In addition, we assume that the face in the target video is mostly unoccluded.

3.1 HMD for the Source Actor

To enable VR teleconferencing, we use an Oculus Rift DK2 HMD, and we integrate a simple IR webcam to track the source actor's eyes. The camera is integrated inside the HMD with Oculus Rift DK2 Monocular Add-on Cups, which allows us to obtain a close-up camera stream of the right eye (Labs 2016) (see Figure 2, left). Although we present results on this specific setup, our method is agnostic to the HMD, and can be used in combination with any other VR device, such as the VR Box, Samsung Gear VR, or HTC Vive. The monocular camera, which we integrate in the DK2, captures an IR stream of the eye region at a resolution of 640×480 pixels at 120Hz. IR LEDs are used as active light sources such that bright images can be obtained, and the camera latency is 5.7ms. The camera is mounted on the top of the VR device lens and an IR mirror is used to get a frontal view of the eye without interfering with the view on the display. The camera is located close to the lenses (see Figure 2, left), and captures images $\mathbf{I}_{\mathcal{E}}$ of the eye at real-time rates. Note that our prototype has only one internal camera. Thus, we use the stream of the right eye to infer and reenact the motion of
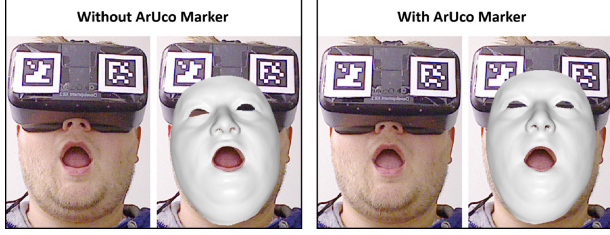
Fig. 3. Tracking with and without ArUco Marker stabilization.

both the left and the right eye. This is feasible as long as we can assume that the focus distance is the same as during calibration—that is, eye vergence (squinting) does not change. If this assumption does not hold, a second internal camera for the left eye can be easily integrated into our design. In addition, we augment the DK2 by attaching two ArUco AR markers to the front of the HMD to robustly track the rigid pose. During face tracking, this allows us to decouple the rigid head pose from the facial expression parameters by introducing additional soft constraints obtained from the markers. The combination of marker tracking and joint optimization allows one to further stabilize the estimates of the rigid head pose, leading to much higher tracking accuracy (see Figure 3).

Tracking of the source actor. For tracking the source actor in real time, we use a commodity RGB-D camera. Specifically, we use an Asus Xtion Pro RGB-D sensor that captures RGB-D frames of 640×480 pixels at 30fps (both color and depth). In every frame, the camera captures an RGB image $\mathbf{I}_\mathcal{I}$ and a depth image $\mathbf{D}_\mathcal{I}$, which we assume to be spatially and temporally aligned. Both images are parameterized by pixel coordinates $\mathbf{p}$, and each RGB value is $\mathbf{I}_\mathcal{I}(\mathbf{p}) \in \mathbb{R}^3$. Depth $\mathbf{D}_\mathcal{I}(\mathbf{p}) \in \mathbb{R}$ is re-projected into the same space as $\mathbf{I}_\mathcal{I}$. Note that we are only considering visible pixel locations $\mathbf{p} \in \mathcal{P}$ on the face that are not occluded by the HMD.

3.2 3D Stereo Rig for Target Actor Tracking

To obtain a 3D reconstruction of the target actor, we use the binocular image stream of a lightweight stereo rig. Our setup is composed of two commodity webcams (Logitech HD Pro Webcam C920), which are rigidly mounted side-by-side and facing the same direction on a stereo bar (see Figure 2, right). The camera rig captures a stereo stream of two RGB pairs $\mathbf{I}_\mathcal{I}^{(c)}$, $c \in \{1, 2\}$ at real-time rates. The two cameras are synchronized up to 33ms and capture images at the resolution of 800×600 pixels at 30Hz. This stereo content is used to capture the target 3D video content. We calibrate the stereo rig intrinsically and extrinsically using standard OpenCV routines.

4 SYNTHESIS OF FACIAL IMAGERY

We parameterize human heads under general uncontrolled illumination based on a multi-linear face and an analytic illumination model. A linear Principle Component Analysis (PCA) basis is used for facial identity (Blanz and Vetter 1999) (geometry and reflectance) and a blend shape basis for the expression variations (Alexander et al. 2009; Cao et al. 2014b). This results in the spatial embedding of the underlying mesh and the associated per-vertex

color information parameterized by linear models, $\mathcal{F}(\mathbf{T}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$ and $C(\boldsymbol{\beta}, \boldsymbol{\gamma})$, respectively. The mesh has 106K faces and 53K vertices. Here, $\mathbf{T} \in \mathbb{R}^{4\times4}$ models the rigid head pose, $\boldsymbol{\alpha} \in \mathbb{R}^{80}$ the geometric identity, $\boldsymbol{\beta} \in \mathbb{R}^{80}$ the surface reflectance properties, $\boldsymbol{\delta} \in \mathbb{R}^{76}$ the facial expression, and $\boldsymbol{\gamma} \in \mathbb{R}^{3\cdot9}$ the incident illumination situation. The 3×9 illumination coefficients encode the RGB illumination based on nine spherical harmonics (SH) (Ramamoorthi and Hanrahan 2001) basis functions. For convenience, we stack all parameters of the model in a vector $\mathcal{X} = (\mathbf{T}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\gamma}) \in \mathbb{R}^{269}$. Synthetic monocular images $\mathbf{I}_\mathcal{S}$ and synthetic stereo pairs $(\mathbf{I}_\mathcal{S}^{(1)}, \mathbf{I}_\mathcal{S}^{(2)})$ of arbitrary virtual heads can be generated by varying the parameters $\mathcal{X}$ and using the GPU rasterization pipeline to simulate the image formation process. To this end, we use a standard pinhole camera model $\Pi (\bullet)$ under a full perspective projection.

Mouth interior. The parametric head model does not contain rigged teeth, a tongue, or a mouth interior, as these facial features are challenging to reconstruct and track from stereo input due to strong occlusions in the input sequence. Instead, we propose two different image-based synthesis approaches (see Section 7). The first is specifically designed for the self-reenactment scenario, where source and target actor are the same person; here we cross project the mouth interior from the source to the target video. For arbitrary source and target actor pairs we improved the retrieval strategy of Thies et al. (2016). This retrieval approach finds the best suitable mouth frame in a mouth database, captured in a short training sequence. In contrast to their approach, our retrieval clusters frames into static and dynamic motion segments leading to temporally more coherent results. The output of this step is then composited with the rendered model using alpha blending (see Section 7).

Eyeball and eyelid. We use a unified image-based strategy to synthesize plausible animated eyes (eyeball and eyelid) that can be used for photo-realistic facial reenactment in VR applications. This novel strategy is one of the core components of this work and is described in more detail in Section 6.

5 PARAMETRIC MODEL FITTING

Our approach uses two different tracking and reconstruction pipelines for each (source and target) actor, respectively. The source actor, who is wearing the HMD, is captured using an RGB-D camera (see Section 3.1). Here, we constrain the face model $\mathcal{F}$ by the visible pixels on the face that are not occluded by the HMD, as well as the attached ArUco AR markers. The target actor reconstruction, which becomes the corresponding VR target content that is animated at runtime, is obtained in a preprocess with the lightweight stereo setup described in Section 3.2. For both tracking pipelines, we use an *analysis-by-synthesis* approach to find the model parameters $\mathcal{X}$ that best explain the input observations. The underlying inverse rendering problem is tackled based on energy minimization as proposed in Thies et al. (2015, 2016).

The tracking for the source and the target actor differ in the energy formulation. The source actor is partly occluded by the HMD; there we measure dense color and depth alignment based on the observations of the RGB-D camera. We restrict the dense reconstruction to the lower part of the face using a predefined visibility

mask. In addition, we use ArUco markers that are attached to the HMD to stabilize the rigid pose of the face (see Figure 3).

As the target videos are recorded in stereo, we adapted the energy formulation to work on binocular RGB data. The results show that our new stereo tracking approach leads to better tracking accuracy than the monocular tracking of Thies et al. (2016).

For simplicity, we first describe the energy formulation for tracking the target actor in Section 5.1. Then, we introduce the objective function for fitting the face model of the source actor in Section 5.2.

5.1 Target Actor Energy Formulation

To process the stereo video stream of the target actor, we introduce a model-based stereo reconstruction pipeline that constrains the face model according to both RGB views per frame. In other words, we aim to find the optimal model parameters $\mathcal{X}$ constrained by the input stereo pair $\{\mathbf{I}_{\mathcal{I}}^{(c)}\}_{c=1}^{2}$. Our model-based stereo reconstruction and tracking energy E_{target} is a weighted combination of alignment and regularization constraints:

$$E_{\text{target}}(\mathcal{X}) = \underbrace{\left[w_{\text{ste}} E_{\text{ste}}(\mathcal{X}) + w_{\text{lan}} E_{\text{lan}}(\mathcal{X}) \right]}_{\text{alignment}} + \underbrace{\left[w_{\text{reg}} E_{\text{reg}}(\mathcal{X}) \right]}_{\text{regularizer}}. \quad (1)$$

We use dense photometric stereo alignment E_{ste} and sparse stereo landmark alignment E_{lan} in combination with a robust regularization strategy E_{reg}. The sub-objectives of E_{target} are scaled based on empirically determined, but constant, weights $w_{\text{ste}} = 100$, $w_{\text{lan}} = 0.0005$, and $w_{\text{reg}} = 0.0025$ that balance the relative importance.

Dense photometric stereo alignment. We enforce dense photometric alignment of the input $\mathbf{I}_{\mathcal{I}}^{(c)}$ and the synthetic imagery $\mathbf{I}_{\mathcal{S}}^{(c)}$. For robustness against outliers, we use the $\ell_{2,1}$-norm (Ding et al. 2006) instead of a traditional least-squares formulation:

$$E_{\text{ste}}(\mathcal{X}) = \sum_{c=1}^{2} \frac{1}{|\mathcal{P}^{(c)}|} \sum_{\mathbf{p} \in \mathcal{P}^{(c)}} \left\| \mathbf{I}_{\mathcal{S}}^{(c)}(\mathbf{p}) - \mathbf{I}_{\mathcal{I}}^{(c)}(\mathbf{p}) \right\|_{2}. \quad (2)$$

Here, $\mathcal{P}^{(c)}$ is the set of visible model pixels $\mathbf{p}$ from the cth-camera. The visible pixels of the model are determined by a forward rendering pass using the old parameters. We normalize based on the total number of pixels $|\mathcal{P}^{(c)}|$ to guarantee that both views have the same influence. Note that the two sets of visible pixels are updated in every optimization step, and for the forward rendering pass we use the face parameters of the previous iteration or frame.

Sparse stereo landmark alignment. We use sparse point-to-point alignment constraints in 2D image space that are based on per-camera sets $\mathcal{L}^{(c)}$ of 66 automatically detected facial landmarks. The landmarks are obtained by a commercial implementation[5] of the detector of Saragih et al. (2011):

$$E_{\text{lan}}(\mathcal{X}) = \sum_{c=1}^{2} \frac{1}{|\mathcal{L}^{(c)}|} \sum_{(\mathbf{l},k) \in \mathcal{L}^{(c)}} w_{\mathbf{l},k} \left\| \mathbf{l} - \Pi(\mathcal{F}_{k}(\mathbf{T}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})) \right\|_{2}^{2}. \quad (3)$$

The projected vertices $\mathcal{F}_{k}(\mathbf{T}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$ are enforced to be spatially close to the corresponding detected 2D feature $\mathbf{l}$. Constraints are

---

[5]TrueVisionSolutions Pty Ltd.

weighted by the confidence measures $w_{\mathbf{l},k}$, which are provided by the sparse facial landmark detector.

Statistical regularization. To avoid implausible face fits, we apply a statistical regularizer to the unknowns of $\mathcal{X}$ that are based on our parametric face model. We favor plausible faces where parameters are close to the mean with respect to their standard deviations σ_{id}, σ_{alb}, and σ_{exp}:

$$E_{\text{reg}}(\mathcal{X}) = \sum_{i=1}^{80} \left[\left(\frac{\boldsymbol{\alpha}_i}{\sigma_{\text{id},i}} \right)^2 + \left(\frac{\boldsymbol{\beta}_i}{\sigma_{\text{alb},i}} \right)^2 \right] + \sum_{i=1}^{76} \left(\frac{\boldsymbol{\delta}_i}{\sigma_{\text{exp},i}} \right)^2. \quad (4)$$

σ_{id} and σ_{alb} are the standard deviations of the statistical face model, and σ_{exp} is set to a constant value (=1).

5.2 Source Actor Tracking Objective

At runtime, we track the source actor who is wearing the HMD and is captured by the RGB-D sensor. The tracking objective for visible pixels that are not occluded by the HMD is similar to the symmetric point-to-plane tracking energy in Thies et al. (2015). In addition to this, we introduce rigid stabilization constraints which are given by the ArUco AR markers in front of the VR headset. These constraints are crucial to robustly separate the rigid head motion from the face identity and pose parameters (see Figure 3). The total energy for tracking the source actor at runtime is given by the following linear combination of residual terms:

$$E_{\text{source}}(\mathcal{X}) = w_{\text{rgb}} E_{\text{rgb}}(\mathcal{X}) + w_{\text{geo}} E_{\text{geo}}(\mathcal{X}) + w_{\text{sta}} E_{\text{sta}}(\mathcal{X}) + w_{\text{reg}} E_{\text{reg}}(\mathcal{X}). \quad (5)$$

The first term of this objective E_{rgb} measures the photometric alignment of the input RGB image $\mathbf{I}_{\mathcal{I}}$ from the camera and the synthetically generated rendering $\mathbf{I}_{\mathcal{S}}$:

$$E_{\text{rgb}}(\mathcal{X}) = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \left\| \mathbf{I}_{\mathcal{S}}(\mathbf{p}) - \mathbf{I}_{\mathcal{I}}(\mathbf{p}) \right\|_{2}. \quad (6)$$

This color term is defined over all visible pixels $\mathcal{P}$ in the bottom half of the face that are not occluded by the HMD, and we use the same $\ell_{2,1}$-norm as in Equation (2).

In addition to the photometric alignment, we constrain the face model by the captured range data:

$$E_{\text{geo}}(\mathcal{X}) = w_{\text{point}} E_{\text{point}}(\mathcal{X}) + w_{\text{plane}} E_{\text{plane}}(\mathcal{X}). \quad (7)$$

Similar to E_{rgb}, geometric residuals of E_{geo} are defined over the same set of visible pixels on the face. The geometric term is composed of two sub-terms, a point-to-point E_{point} term, where $\mathbf{D}_{\mathcal{I}}$ is the input depth and $\mathbf{D}_{\mathcal{S}}$ is the rendered depth (both are back-projected into camera space),

$$E_{\text{point}}(\mathcal{X}) = \sum_{\mathbf{p} \in \mathcal{P}} \left\| \mathbf{D}_{\mathcal{S}}(\mathbf{p}) - \mathbf{D}_{\mathcal{I}}(\mathbf{p}) \right\|_{2}^{2}, \quad (8)$$

as well as a symmetric point-to-plane term

$$E_{\text{plane}}(\mathcal{X}) = \sum_{\mathbf{p} \in \mathcal{P}} \left[d_{\text{plane}}^2(N_{\mathcal{S}}(\mathbf{p}), \mathbf{p}) + d_{\text{plane}}^2(N_{\mathcal{I}}(\mathbf{p}), \mathbf{p}) \right], \quad (9)$$

where $d_{\text{plane}}(\mathbf{n}, \mathbf{p}) = [(\mathbf{D}_{\mathcal{S}}(\mathbf{p}) - \mathbf{D}_{\mathcal{I}}(\mathbf{p}))^T \cdot \mathbf{n}]$, $N_{\mathcal{I}}(\mathbf{p})$ is the input normal, and $N_{\mathcal{S}}(\mathbf{p})$ the rendered model normal.

In addition to the constraints given by the raw RGB-D sensor data, the total energy of the source actor E_{source} incorporates rigid

head pose stabilization. This is required, since in our VR scenario the upper part of the face is occluded by the HMD. Thus, only the lower part can be tracked and the constraints on the upper part of the face, which normally stabilize the head pose, are missing. To stabilize the rigid head pose, we use the two ArUco markers that are attached to the front of the HMD (see Figure 3).

We first extract a set of eight landmark locations based on the two markers (four landmarks each). To handle noisy depth input, we fit two 3D planes to the frame's point cloud that bound each marker, respectively. We then use the resulting 3D corner positions of the markers, and project them into face model space. Using these stored reference positions A_k, we establish the rigid head stabilization energy E_{sta}:

$$E_{sta}(\mathcal{X}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{l}, k) \in \mathcal{S}} \|\mathbf{l} - \Pi(\mathbf{T}A_k)\|_2^2. \qquad (10)$$

Here, $\mathcal{S}$ defines the correspondences between the detected 2D landmark positions $\mathbf{l}$ in the current frame and the reference positions A_k. In contrast to the other data terms, E_{sta} depends only on the rigid transformation $\mathbf{T}$ of the face and replaces the facial landmark term used by Thies et al. (2015). Note that the Saragih tracker is unable to robustly track landmarks in this scenario since only the lower part of the face is visible. The statistical regularization term E_{reg} is the same as for the target actor (see Equation (4)).

5.3 Data-Parallel Optimization

We find the optimum of both face tracking objectives $\mathcal{X}^*_{source} = \text{argmin}_{\mathcal{X}} E_{source}(\mathcal{X})$ and $\mathcal{X}^*_{target} = \text{argmin}_{\mathcal{X}} E_{target}(\mathcal{X})$ based on variational energy minimization, leading to an unconstrained non-linear optimization problem. Due to the robust $\ell_{2,1}$-norm used to enforce photo-metric alignment, we find the minimum based on a data-parallel *Iteratively Re-Weighted Least-Squares* (IRLS) solver (Thies et al. 2016). At the heart of the IRLS solver, a sequence of non-linear least-squares problems are solved with a GPU-based *Gauss-Newton* approach (DeVito et al. 2016; Thies et al. 2015, 2016; Wu et al. 2014; Zollhöfer et al. 2014, 2015) that builds on an iterative Preconditioned Conjugate Gradient (PCG) solver. The optimization is run in a coarse-to-fine fashion using a hierarchy with three levels. We only run tracking on the two coarser levels using seven IRLS steps on the coarsest and one on the medium level. For each IRLS iteration, we perform one Gauss-Newton (GN) step with four PCG steps. To exploit temporal coherence, we initialize the face model with the optimization results from the previous frame. First, this gives us a good estimate of the visible pixel count in the forward rendering pass, and second, it provides a good starting point for the GN optimization. Note that we never explicitly store $J^T J$, but instead apply the multiplication of the Jacobian (and its transpose) on-the-fly within every PCG step. Thus, the compute cost for each PCG iteration becomes more expensive for multi-view optimization of E_{target}; although materialization is still less efficient, since we only need a small number of PCG iterations.

6 AN IMAGE-BASED EYE AND EYELID MODEL

We propose a novel image-based retrieval approach to track and synthesize the region of the eyes, including eyeballs and eyelids.
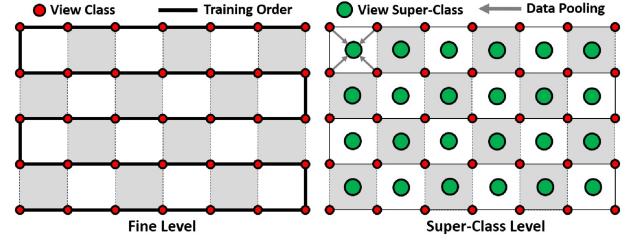
Fig. 4. Left: The eye calibration pattern used to generate training data for learning our image-based eye-gaze retrieval. In the training phase, we progress row-by-row in a zigzag order; each grid point is associated with an eye-gaze direction. Right: To obtain robust results, we perform a hierarchical classification where classes of the finer level are accumulated into a smaller set of superclasses.

This approach is later used in all presented applications, especially in the self-reenactment for video conferencing in VR (see Section 8.1). We chose an image-based strategy, since it is specific to a person; it not only models the behavior of the eyeballs but also captures idiosyncrasies of eyelid movement while enabling photo-realistic re-rendering. Our approach uses a hierarchical variant of random ferns (Ozuysal et al. 2010) to robustly track the eye region. To this end, we propose a novel actor-specific and fully automatic training stage. In the following, we describe our fully automatic data generation process, the used classifier, and the optimizations that are required to achieve fast, robust, and temporally stable gaze estimates.

6.1 Training Data Generation

To train our image-based eye regression strategy, we require a sufficiently large set of labeled training data. Since manual data annotation for every new user is practically infeasible, we propose a very efficient approach based on a short eye calibration sequence.

During the training process, we display a small circle at different positions of a 7×5-tiled image grid on the screen in front of the user (see Figure 4, left). This allows us to capture the space of all possible look-at points on the display. In addition, we capture an image of a closed eye for the synthesis of eye blinks. The captured image data $\mathcal{I}_n$ is divided into $36 = 7 \times 5 + 1$ unique classes l_n, where every class is associated with a view direction. The ground truth gaze directions are given by the current position of the *dot* on the screen in the training data. During training, the user focuses on the displayed dot with his eye gaze. We show every dot for 2 seconds for each location. The data captured in the first 0.4 seconds is rejected to allow the user a grace period to adjust his eye gaze to new positions. In the remaining 1.6 seconds, we capture 50 frames which we use to populate the corresponding class. After that, we proceed to the next class, and move the dot to the next position. Note that the dot location for a given class is fixed, but we obtain multiple samples within each class (one for each frame) from the input data. This procedure progresses row-by-row in a zigzag order (see Figure 4, left). Finally, we augment the samples in each class by jittering each captured source image by ± 1 pixels, resulting in 9×50 training frames per class. Each cluster is also associated with a representative image of the eye region

obtained from the captured input data. The representative image of each class is given by the median of the corresponding video clip, which is later used for the synthesis of new eye movements. Finally, we add an additional class which represents eye blinks; this class is obtained by asking the user to close his eyes at the end of the training phase. This calibration sequence is performed for both the source and target actor. Since the calibration sequence is the same for both actors, we obtain one-to-one correspondences between matching classes across actors. Note that for the source actor, we directly use the image data that we observe from the IR camera that is integrated into the HMD as training data. For the target actor, we compute a normalized view of the eye from the stereo video input using the texture space of the parametric face model. These normalized views are later used to re-synthesize the eye motions of the target actor (see Section 7). As detailed in the following subsections, we use the data of the source actor to train an eye-gaze classifier which predicts gaze directions for the source actor at runtime. Once trained, for a given source input frame, the classifier identifies cluster representatives from the target actor eye data. The ability to robustly track the eye direction of the source actors forms the basis for real-time gaze-aware facial reenactment—that is, we are able to photo-realistically animate/modify the eyes of a target actor based on a captured video stream of the source actor. In the following, we detail our eye tracking strategy.

6.2 Random Ferns for Eye-Gaze Classification

The training data $\{\mathcal{I}_n, l_n\}_{n=1}^N$, which is obtained as described in the previous section, is a set of N input images $\mathcal{I}_n$ with associated class labels l_n. Each label $l_n \in \{c_l\}_{l=1}^C$ belongs to one of C classes c_l. In our case, the images of the eye region are clustered based on gaze direction. We tackle the associated supervised learning problem by an ensemble of M random ferns (Ozuysal et al. 2010), where each fern is based on S features. To this end, we define a sequence of $K = MS$ binary intensity features $\mathbf{F} = \{f_k\}_{k=1}^K$, which is split into M independent subsets $\mathbf{F}_m$ of size S. Assuming statistical independence and applying *Bayes rule*, the log-likelihood of the class label posterior can be written as

$$\log P(c_l|\mathbf{F}) \sim \log \left[P(c_l) \cdot \prod_{m=1}^M P(\mathbf{F}_m|c_l) \right]. \tag{11}$$

The class likelihoods $P(\mathbf{F}_m|c_l)$ are learned using random ferns. Each fern performs S binary tests, which discretizes the per-class feature likelihood into $B = 2^S$ bins. At first, we initialize all bins with one to prevent taking the logarithm of zero. In all experiments, we use $M = 800$ ferns with $S = 5$ binary tests. Finally, the class with the highest posterior probability is chosen as the classification result. Training takes only around 4.9ms per labeled image, thus training runs in parallel to the calibration sequence. Once trained, the best class is obtained in less than 1.4ms.

Hierarchical eye-gaze classification. To efficiently handle classification outliers, we perform eye-gaze classification on a two-level hierarchy with a fine and a coarse level. The 35 + 1 classes of the fine level are defined by the grid points of the zigzag calibration pattern shown in the left side of Figure 4. To create the coarse level, we merge neighboring classes of the fine level into
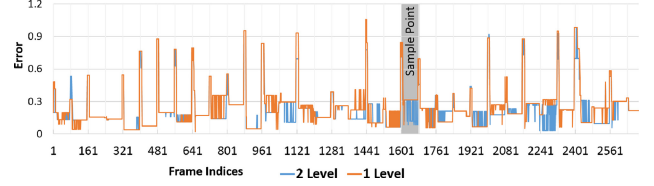

Fig. 5. Comparison of a one- (orange) and a two-level (blue) classifier. Ground truth data is obtained by a test subject looking at a dot that appears every 80 frames (2.6 seconds) at random (Sample Point); error is measured in normalized screen space coordinates in $[0, 1]^2$. As shown by the magnitude of the positional error, the multi-level classifier obtains higher accuracy.

superclasses. For a set of four adjacent classes (overlap of one), we obtain one superclass (see Figure 4, right). This leads to a grid with $25 = 4 \times 6 + 1$ unique classes (rather than the 35 + 1 classes; the class for eye blink is kept the same).

During training, we train the two hierarchy levels independently. The training data for the fine level is directly provided by the calibration pattern, and the data for the coarse level is inferred as described above. At test time, we first run the classifier of the coarse level which provides one of the superclasses. Then the classification on the fine level only considers the four classes of the best matching superclass. The key insight of this coarse-to-fine classification is to break up the task into easier sub-problems. In other words, the classification on the coarse level is more robust and less prone to outliers of the fern predictions since there are fewer classes to distinguish between. The fine level then complements the superclass prediction by increasing the accuracy of the inferred eye-gaze directions. In the end, this multi-level classifier leads to high accuracy results while minimizing the probability of noisy outliers. In Figure 5, we show a comparison between a one- and a two-level classifier. The two-level approach obtains a lower error (mean 0.217973, std. dev. 0.168094) compared to the one-level approach (mean 0.24036, std. dev. 0.18595).

Temporal stabilization. We also introduce a temporal stabilizer that favors the previously retrieved eye-gaze direction. This particularly helps in the case of small eye motions, where the switch to a new class would introduce unwanted jitter. To this end, we adjust the likelihood of a specific class $P(c_l)$ using an empirically determined temporal prior such that the previously predicted eye-gaze direction c_{old} is approximately 1.05× more likely than changing the state and predicting a different class:

$$P(c_l) = \begin{cases} p_1 = \frac{1.05}{1+1.05} \approx 0.512, & \text{if } (c_l = c_{old}) \\ p_2 = 1 - p_1 \approx 0.488, & \text{else} \end{cases} \tag{12}$$

We integrate the temporal stabilization on both levels of the classification hierarchy. First, we *favor* the superclass on the coarse level using the aforementioned temporal prior. If the current and previous prediction on the coarse level is the same, we apply a similar prior to the view within the superclass. Otherwise, we use no temporal bias on the fine level. This allows fast jumps of the eye direction, which is crucial for fast saccade motion that pushes the boundary of the 30Hz temporal resolution of the stereo setup.

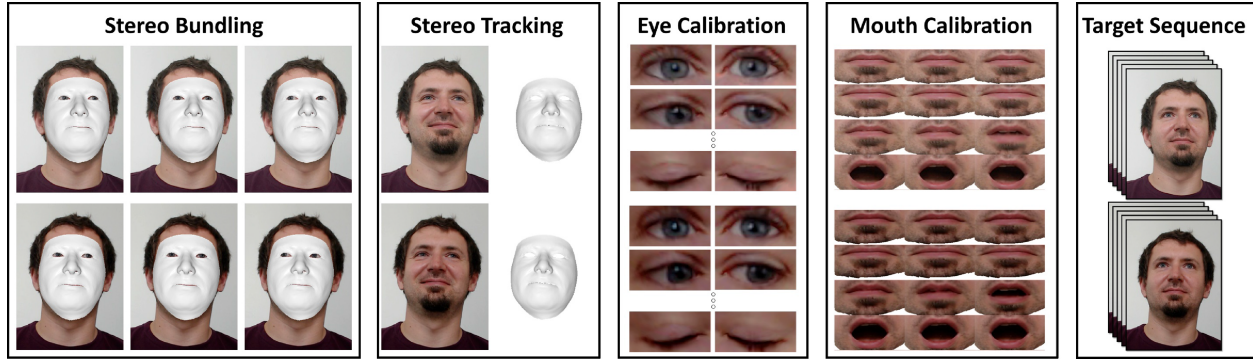**Stereo Bundling** | **Stereo Tracking** | **Eye Calibration** | **Mouth Calibration** | **Target Sequence**

Fig. 6. Building a personalized stereo avatar. From left to right: We first jointly optimize for all unknowns of our parametric face model using a non-rigid bundle adjustment formulation on the input of three stereo pairs. For tracking, we only optimize for expression, lighting, and rigid pose parameters constrained by synchronized stereo input; this optimization runs in real time. Next, we train our data-driven eye tracker with data from an eye-calibration sequence. In addition to eye calibration, we build a database of mouth stereo pairs, which captures the variation of mouth motion. Note that the mouth database is only required if the mouth cross-projection is not used. As a result, we obtain a tracked stereo target, which is used during live reenactment (this is the target actor).

7 FACE RIG AND COMPOSITING

Generation of a personalized face rig. At the beginning of each recording, both of the source and target actor, we compute a person-specific face rig in a short initialization stage. To this end, we capture three keyframes with slightly different head rotations to recover the user's facial geometry and skin reflectance. Given the constraints of these three keyframes, we jointly optimize for all unknowns of our face model $\mathcal{F}$—facial geometry, skin reflectance, illumination, and expression parameters—using our tracking and reconstruction approach. This initialization requires a few seconds to complete; once computed, we maintain a fixed estimate of the facial geometry and replace the reflectance estimate with a person-specific illumination-corrected texture map.

In the stereo case, we compute one reflectance texture for each of the two cameras. This ensures that the two re-projections exactly match the input streams, even if the two used cameras have slightly different color response functions. In the following steps, we use this high-quality stereo albedo map for tracking, and we restrict the optimizer to only compute the per-frame expression and illumination parameters. All other unknowns (the facial identity) are person specific and can remain fixed for a given user.

To track and synthesize new eye motions in both videos (source and target), we capture the person-specific appearance and motion of the eyes and eyelids during a short eye-calibration sequence in the initialization stage as described in Section 6.1.

Reenactment and real-time compositing. At runtime, we use the reconstructed face model along with its calibration data (eye and mouth; see Figure 6) to photo-realistically re-render the face of the target actor. We first modify the facial expression parameters of the reconstructed face model of the target actor to match the face expression of the source actor. The expressions are transferred from source to target using the subspace deformation transfer approach of Thies et al. (2016).

In the final compositing stage, we render the mouth texture, the eye textures, and the (potentially modified) 3D face model on top of the target video using alpha blending. Instead of a static face texture, we use a per-frame texture based on the current frame of the target video. This leads to results of higher resolution, since slight misalignments during the generation of the personalized face rig have no influence on the final texture quality.

Synthesis of mouth interior. To enable high-quality reenactment of the mouth in the target video, we propose two different approaches. The method of choice depends on the specific use-case. In the self-reenactment scenario, which is the case for HMD removal (see Section 8.1), we directly project the mouth interior of the source video to the target video. We use Poisson image blending (Pérez et al. 2003) to seamlessly blend the mouth texture into the modified target video. This ensures an accurate reproduction of the correct mouth shape and interior in the case of identical source and target identity. The Poisson equation is solved on the GPU using the Jacobi iterative method.

In the case of stereo reenactment, where the source and the target actor differ, we built a database of target mouth interiors using a short calibration sequence as proposed by Thies et al. (2016). In this scenario, cross-projection cannot be applied, as this would change the identity of the target actor. The mouth motion database is clustered into static and dynamic motion segments based on the space-time trajectory of the sparse 2D landmark detections. We select the mouth frame from the database that has the most similar spatial distribution of 3D marker positions. In contrast to Thies et al. (2016), we prefer frames that belong to the same motion segment as the previously retrieved one. This leads to higher temporal coherence and hence less visual artifacts. The retrieved mouth frames do not exactly match the transferred facial expression. To account for this, Thies et al. (2016) stretch the texture based on the face parameterization leading to visual artifacts—that is, unnaturally stretched teeth, which are temporally unstable. To alleviate this problem, we propose a new strategy and match the retrieved texture to the outer mouth contour of the target expression using a saliency preserving image warp (Wang et al. 2008). For a comparison of both approaches, we refer to the accompanying video. We use a modified as-rigid-as-possible
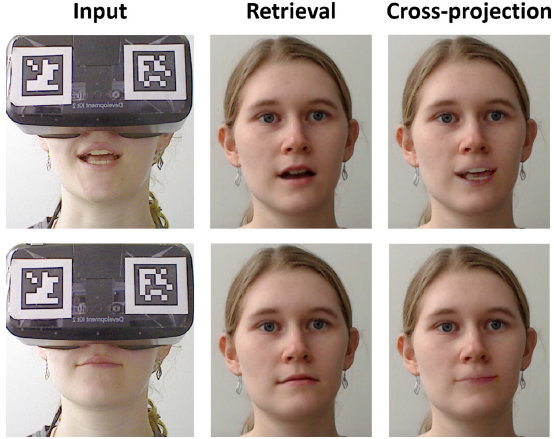
Input   Retrieval   Cross-projection

Fig. 7. Comparison in the case of self-reenactment between the two proposed image-based mouth interior synthesis strategies. Cross-projection (right) leads to more natural and higher quality mouth interiors than the retrieval-based approach.

regularizer that takes local saliency of image pixels into account. The idea is to deform the mouth texture predominantly in regions that will not lead to visual artifacts. Stretching is most noticeable for the bright teeth, since they are perfectly rigid in the physical world, while it is harder to detect in the darker regions that correspond to the mouth interior. Therefore, we use pixel intensity as a proxy to determine local rigidity weights (a high value for bright and low value for dark pixels) that control the amount of warping in different texture regions. This is based on the assumption that the teeth are the predominant white pixels in the mouth region.

As can be seen in Figure 7, the mouth cross-projection approach leads to more natural results and captures more details such as the movement of the tongue compared to the retrieval-based approach.

Synthesis of the eye region. Our eye gaze estimator is specifically developed to allow a one-to-one correspondence between the source and the target actor (see Section 6). Thus, after tracking the source actor, we know the index of the gaze class in the eye database of the target actor. To synthesize temporally coherent and plausible eye motion, we temporally filter the eye motion by averaging the retrieved view direction of the gaze class in a small window of frames. Afterward, we use the average view direction to perform the texture lookup. As described earlier (Section 6.2), we use an additional class in our eye gaze classification strategy to represent lid closure. To obtain temporally smoother transitions between an open and closed eye, we temporally filter the eye texture based on an exponential average (a factor 0.8 for the retrieved texture and 0.2 for the last result). Figure 8 shows an exemplary eye blink transition. Since the eye images of the target live in the space of the face model texture space, they can directly be used in the final rendering process.

8 RESULTS

In this section, we evaluate our gaze-aware facial reenactment approach in detail and compare against state-of-the-art tracking methods. All experiments run on a desktop computer with an
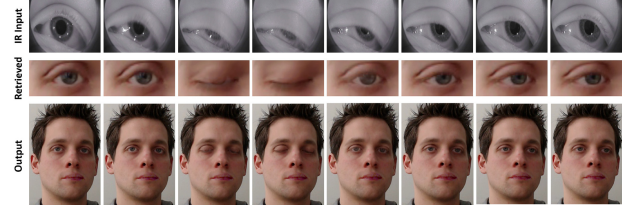

Fig. 8. Eye blinking.Consecutive frames from left to right. The IR input image captured by the camera mounted inside the HMD (top row) is used to retrieve realistic eye textures (middle row). In the final compositing stage, the texture is seamlessly blended with the target face (bottom row).

Nvidia GTX1080 and a 3.3GHz Intel Core i7-5820K processor. For tracking the source and target actor, we use our hardware setup as described in Section 3. Our approach is robust to the specific choice of parameters, and we use a fixed parameter set in all experiments. For stereo tracking, we set the following weights in our energy formulation: $w_{ste} = 100.0$, $w_{lan} = 0.0005$, $w_{reg} = 0.0025$. Our RGB-D tracking approach uses $w_{rgb} = 100.0$, $w_{geo} = 10000.0$, $w_{sta} = 1.0$, and $w_{reg} = 0.0025$.

As our main result, we demonstrate self-reenactment for VR goggles removal. In Appendix A, we also show gaze correction in monocular live video footage and gaze-aware facial reenactment. All three applications share a common initialization stage that is required for the construction of a personalized face and eye/eyelid model of the users (see Section 7). The source video content is always captured using the Asus Xtion depth sensor. Depending on the application, we use our lightweight stereo rig or the RGB-D sensor to capture the target actor.

8.1 Self-Reenactment for VR Video Conferencing

Our real-time facial reenactment approach can be used to facilitate natural video chats in virtual reality. The major challenge for video conferencing in the VR context is that the majority of the face is occluded by the HMD; therefore, the other person in a VR conversation is unable to see the eye region. Using self-reenactment, the users can alter both the facial expression and the eye/eyelid motion of the prerecorded video stream. This virtually removes the HMD from the face and allows users to appear as themselves in VR without suffering from occlusions due to the HMD (see Figure 9). In addition, the output video stream mimics the eye motion, which is crucial since natural eye contact is essential in conversations. Additionally, we show HMD removal examples with a matching audio stream in the supplemental video. This shows that the final result is well aligned with the voice of the source actor.

Although compression is not the main focus of this article, it is interesting to note that the reenactment results can be easily transferred over a network with low bandwidth. To transmit the 3D video content at runtime to the other participants in a video chat, we only have to send the model parameters, as well as the eye and mouth class indices. The final modified stereo video can be directly synthesized on the receiver side using our photo-realistic re-rendering. Given that current video chat software, such as Skype, still struggles under poor network connections, our approach may be able to boost visual quality.
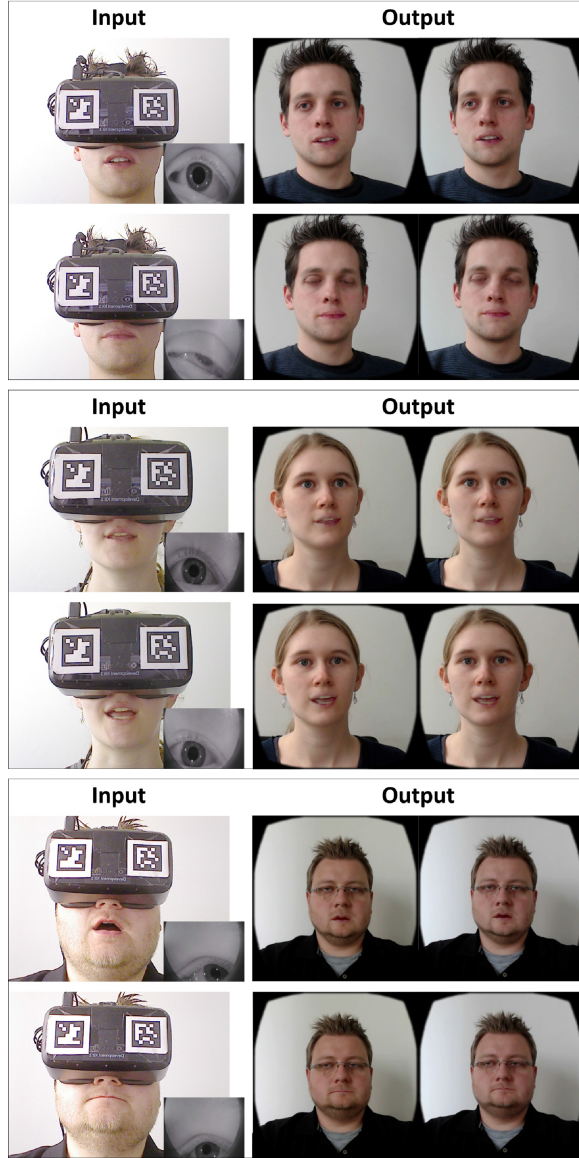
Input	Output

Fig. 9. Self-reenactment for VR video conferences. Our real-time facial reenactment approach allows one to virtually remove the HMD by driving a prerecorded target video of the same person. Note that these results employ the mouth cross-projection strategy to fill in the mouth interior.

Evaluation of face identity estimation. The identity of the target actor is obtained using our model-based stereo bundle adjustment strategy. We compare our identity estimate with the approach of Thies et al. (2016) (Face2Face) (see Figure 10). As a reference, we use a high-quality structured light scan of the same person taken with a David 3D scanner. Our approach obtains a better reconstruction of the identity; especially the chin, nose, and cheek regions are of higher quality. Note that we estimate the identity by model-based bundle adjustment over three stereo pairs. Face2Face uses only the three images of one of the two RGB cameras.
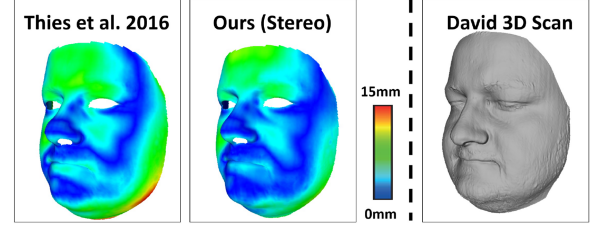

Fig. 10. Accuracy of reconstructed identity. We compare our result against Face2Face (Thies et al. 2016). Note that our approach obtains a better shape estimate of the chin, nose, and cheek regions. For reference, we use a structured light reconstruction from a David 3D scanner. The mean Hausdorff Distance of Face2Face is $3.751mm$ (RMSE $4.738mm$). Our approach has a mean distance of $2.672mm$ (RMSE $3.384mm$).
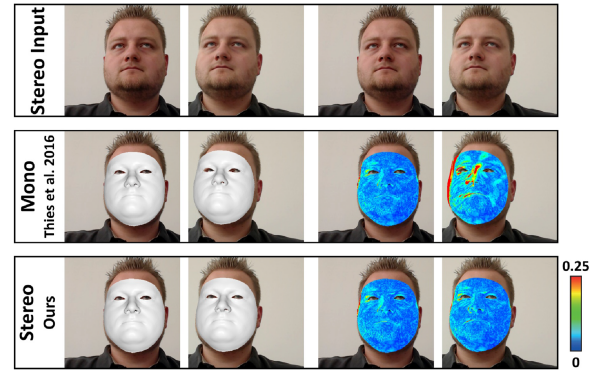

Fig. 11. Stereo alignment. We compare the photometric alignment accuracy of our approach to Face2Face (2016). Face2Face only obtains a good fit to the image captured by the left camera (average error of 0.011), but the re-projection to the right camera suffers from strong misalignments (average error of 0.019). In contrast, our stereo tracking method obtains consistently low errors for both views (average error of 0.011 left and 0.012 right).

Evaluation of face tracking accuracy. In Figure 11, we evaluate the stereo alignment accuracy of our approach and compare to the monocular face tracker of Face2Face (Thies et al. 2016). As input, we use the binocular image stream captured by our custom stereo setup (see Section 3). We measure the photometric error between the input frames and the re-projection of the tracked face model. The tracking of Face2Face is based on the left camera stream, since this approach uses only monocular input data. Thus, Face2Face obtains a good fit with respect to the left camera (average error of 0.011), but the re-projection regarding the right camera suffers from strong misalignments (average error of 0.019). In contrast, our stereo tracking approach obtains consistently low errors for both views (average error of 0.011 left and 0.012 right), since we directly optimize for the best stereo overlap. For the aforementioned reenactment applications in VR, it is crucial to obtain high-quality alignment with respect to both camera streams of the stereo setup.

We evaluate the accuracy of our approach on ground truth data (see Figure 12). As ground truth, we use high-quality stereo reconstructions obtained by Valgaerts et al. (2012). To this end, we
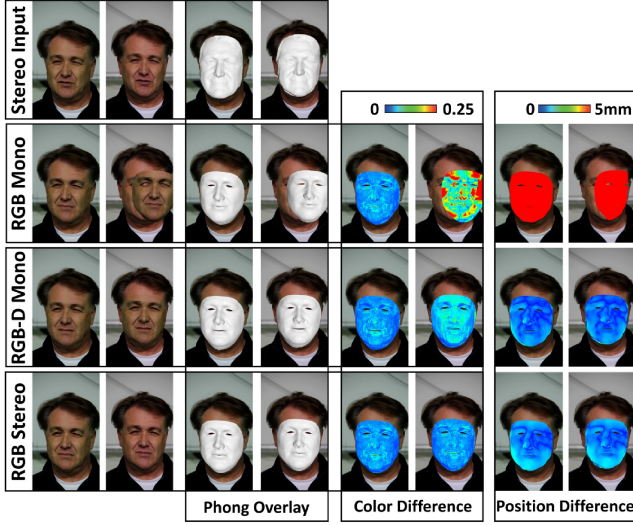
Fig. 12. Ground truth comparison. We evaluate the photometric and geometric accuracy of our stereo tracking approach (RGB Stereo). As ground truth, we employ the high-quality stereo reconstructions of Valgaerts et al. (2012). Our approach achieves consistently low photometric and geometric error for both views. We also compare to Thies et al. (2015) (RGB-D Mono) and Face2Face (2016) (RGB Mono). Both approaches show consistently higher photometric error, since they do not optimize for stereo alignment. Note that the RGB-D tracker uses the ground truth depth as input.

Table 1. Tracking Accuracy of Our Approach (RGB Stereo) Compared to Thies et al. (2015) (RGB-D Mono) and Face2Face (2016) (RGB Mono)

	Photometric		Geometric	
	Left	Right	Left	Right
RGB Mono	**0.0130**	0.0574	0.2028	0.1994
RGB-D Mono	**0.0123**	0.0183	**0.0031**	**0.0031**
RGB Stereo (Ours)	**0.0118**	**0.0116**	**0.0046**	**0.0046**

Note: Our approach achieves low photometric and geometric errors for both views since we directly optimize for stereo alignment.

synthetically generate a high-quality binocular RGB-D stream from the reference data. Our approach achieves consistently low photometric and geometric errors. We also compare against the state-of-the-art face trackers of Thies et al. (2015) (RGB-D Mono) and Face2Face (Thies et al. 2016) (RGB Mono) on the same dataset. All three approaches are initialized using model-based RGB-(D) bundling of three (stereo) frames. The RGB Mono and RGB-D Mono trackers show consistently higher photometric errors for the right input stream, since they do not optimize for stereo alignment (see also Table 1). Given that Face2Face (Thies et al. 2016) only uses monocular color input, it suffers from depth ambiguity, which results in high geometric errors. Due to the wrong depth estimate, the re-projection to the right camera image does not correctly fit the input. The RGB-D-based tracking approach of Thies et al. (2015) resolves this ambiguity and therefore obtains the highest depth accuracy. Note, however, that this approach has access to the ground truth depth data for the sake of this evaluation. Since
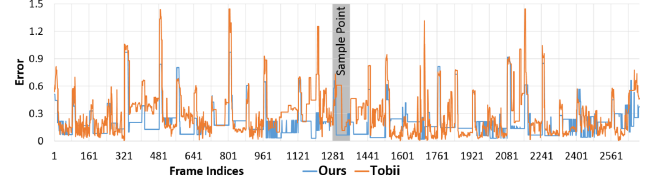

Fig. 13. Comparison to the commercial Tobii EyeX eye tracking solution. The ground truth data is obtained by a test subject looking at a dot on the screen that appears every 80 frames (2.6 seconds) at random (Sample Point); error is measured in normalized screen space coordinates in $[0, 1]^2$. We plot the magnitude of the positional error of Tobii EyeX (orange) and our approach (blue). Our approach obtains a consistently lower error.

the two cameras have slightly different response functions, the reconstructed model colors do not match the right image, leading to high photometric error. Only our model-based stereo tracker is able to obtain high-accuracy geometric and photometric alignment in both views. This is crucial for the creation of 3D stereo output for VR applications, as demonstrated earlier. Neither of the two other approaches achieves this goal.

8.2 Evaluation of Eye Tracking Accuracy

We evaluate the accuracy of our monocular eye-gaze classification strategy on ground truth data and compare to the commercial Tobii EyeX eye tracker.[6] To this end, a test subject looks at a video sequence of a dot that is displayed at random screen positions for 80 successive frames (2.6 seconds given 30Hz input)—this provides a ground truth dataset. During this test sequence, we capture the eye motion using both the Tobii EyeX tracker and our approach. We measure the per-frame magnitude of the positional 2D error of Tobii and our approach with respect to the known ground truth screen positions (see Figure 13). Note that screen positions are normalized to $[0, 1]^2$ before comparison. As can be seen, we obtain consistently lower errors. On the complete test sequence (more than 74 seconds), our approach has a mean error of 0.206 (std. dev. 0.178). In contrast, the Tobii EyeX tracker has a higher error of 0.284 (std. dev. 0.245). The high accuracy of our approach is crucial for realistic and convincing eye reenactment results. Note that the outside-in tracking of Tobii EyeX does not generalize to the VR context, since both eyes are fully occluded by the HMD. In the supplemental video we also evaluate the influence of head motion on the retrieved eye texture. As can be seen in the video sequence, the head motion has less impact on the eye texture retrieval.

We also compare our reconstructions to the state-of-the-art approach of Wang et al. (2016) (see Figure 14, left). For the complete sequence, we refer to the supplemental video. Our reconstructions are of similar quality in terms of the obtained facial shape and the retrieved gaze direction. Note that in contrast to Wang et al. (2016), our approach additionally enables realistic re-rendering of the actor (see Figure 14, right), which is the foundation for VR goggles removal and reenactment in virtual reality at the cost of a short person specific calibration sequence.

---

[6]http://www.tobii.com/xperience/.

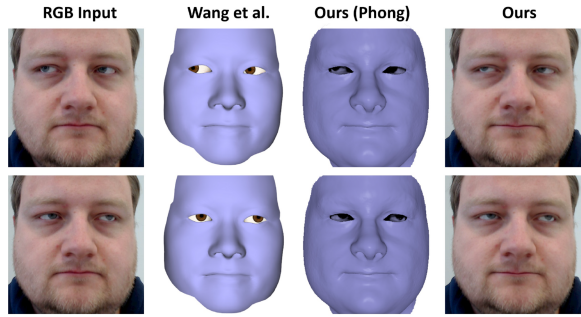**RGB Input** **Wang et al.** **Ours (Phong)** **Ours**

Fig. 14. Comparison to Wang et al. (2016). From left to right: RGB input, output of Wang et al., our Phong rendered output with retrieved eyes, and final realistic face re-rendering using our approach.
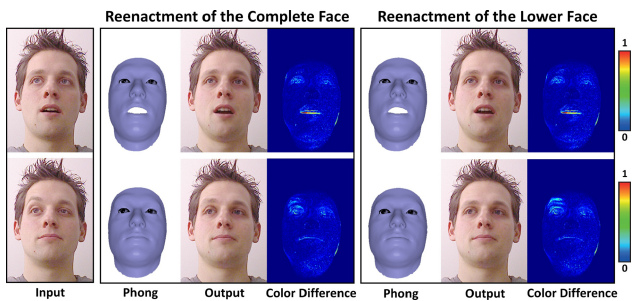

Reenactment of the Complete Face   **Reenactment of the Lower Face**

Input **Phong** **Output** **Color Difference**   **Phong** **Output** **Color Difference**

Fig. 15. Perceptual side-by-side comparison for the self-reenactment scenario.

8.3 Perceptual Evaluation

To quantify the quality of our approach, we perform a side-by-side ground truth comparison for the self-reenactment scenario (see Figure 15). To this end, we employ the same sequence as source and as target. This enables us to measure the color difference between the real video and the synthesized output. In the VR scenario, the source is wearing an HMD, thus we are only able to track and transfer the expressions of the lower part of the face. To measure the loss of information, we evaluate both scenarios, full reenactment and reenactment of only the lower part of the face. We refer to the supplemental video for the complete video sequence. Full facial reenactment results in a mean error of 0.01067 measured in RGB color space. Due to the lack of eyebrow motion, the reenactment of only the lower part of the face has a slightly higher error of 0.01086.

We also conducted a pilot study with 18 participants (working in the field of computer graphics) to evaluate the realism of our results. A variety of different stereoscopic videos were shown. The first video is a real video of an actor wearing an HMD, followed by result videos of our approach. The participants were asked to rate the realism and the impression of sitting face-to-face to a person (from 1 (very good) to 6 (very bad)). The original video achieved a score of 1.75 and a score of 2.5625, respectively. The videos created with our stereoscopic reenactment method achieved a score of 2.281 and 2.09. Our approach produces good quality and the preliminary perceptual evaluation shows that we improved the
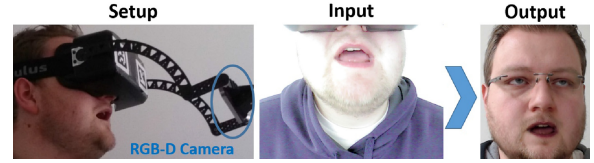
Setup   **Input**   **Output**

Fig. 16. A setup with a rigidly mounted RGB-D camera (Intel Realsense F200) allows for cross-projection of the mouth independently of the head rotation.

impression of sitting face-to-face to a person, which is of paramount importance for making VR teleconferencing viable.

9 LIMITATIONS

Although FaceVR is able to facilitate a wide range of face appearance manipulations in VR, it is one of the early methods in a new field. As such, it is a first step and thus constrained by several limitations. While our eye tracking solution provides great accuracy with little compute cost, it is specifically designed for the VR scenario. In contrast to Wang et al. (2016) our approach is person specific, but allows us to re-synthesize eye motion photo-realistically. Since our eye tracking approach is only based on one eye in the VR device, we correctly capture vergence and squinting; one would need to add a second IR camera to the HMD, which is a straightforward modification. As discussed in Section 7, we only employ one class for lid closure and apply a simple blending between open and closed eyes, explicitly modeling in-between states can further improve the results (Bermano et al. 2015). The cross-projection of the mouth interior, which is used in the self-reenactment scenario, requires a similar head rotation in the source and target sequence. If the head rotations differ too much, noticeable distortions might occur in the final output. Therefore, we also tested a setup similar to Li et al. (2015), where the camera is rigidly attached to the HMD (see Figure 16). Note that the original system of Li et al. is only able to animate a digital avatar and it does not allow for photo-realistic gaze-aware self-reenactment of a person. The setup decreases the ergonomics of the HMD, but ensures a frontal view of the mouth that can be easily transferred to a front facing virtual stereoscopic avatar.

The major limitation of our approach is that we cannot modify the rigid head pose of the target videos. This would require a reconstruction of the background and the upper body of the actor including hair and so forth, which we believe is an interesting research direction.

Our VR face tracking is based on the rigid head pose estimates and the unoccluded face regions. Unfortunately, the field of view of the IR camera attached to the inside of the device is not large enough to cover the entire occluded face region. Thus, we cannot track most of the upper face except the eyeballs. Here, our method is complementary to the approach of Li et al. (2015); they use additional sensor input from electronic strain measurements to fill in this missing data. The resulting constraints could be easily included in our face tracking objective; note, however, that their approach does not enable gaze-aware facial reenact. In the context of facial reenactment, we have similar limitations as Thies et al. (2015) and Face2Face (Thies et al. 2016)—that is, we cannot

handle occlusions in the target video such as those caused by microphones or waving hands. We believe that this could be addressed by computing an explicit foreground-face segmentation; the work by Saito et al. (2016) already shows promising results to specifically detect such cases.

10 CONCLUSION

In this work, we have presented FaceVR, a novel approach for real-time gaze-aware facial reenactment in the context of virtual reality. The key components of FaceVR are robust face reconstruction and tracking, data-driven eye tracking, and photo-realistic re-rendering of facial content on stereo displays. Therefore, we are able to show a variety of exciting applications, especially self-reenactment for teleconferencing in VR. We believe that this work is a stepping stone in this new field, demonstrating some of the possibilities of the upcoming virtual reality technology. In addition, we are convinced that this is not the end of the line, and we believe that there will be even more exciting future work targeting photo-realistic video editing to improve the VR experience, as well as many other related applications.

APPENDIX

In this appendix, we show additional use-cases of FaceVR. Besides self-reenactment for video conferences in VR, FaceVR produces compelling results for a variety of other applications, such as gaze-aware facial reenactment, reenactment in virtual reality, and re-targeting of somebody's gaze direction in a video conferencing call.

A.1 Gaze-Aware Facial Reenactment

Our approach enables real-time photo-realistic and gaze-aware facial reenactment of monocular RGB-D and 3D stereo videos (see Figures 19, 17 and 18).

In both scenarios, we track the facial expressions of a source actor using an external Asus Xtion RGB-D sensor, and transfer the facial expressions—including eye motion—to the video stream of a target actor. The eye motion is tracked using our eye-gaze classifier based on the data captured by the external camera (monocular RGB-D reenactment) or the internal IR camera which is integrated into the HMD (stereo reenactment). We transfer the tracked facial motion to a RGB-D or stereo target video stream using the presented facial reenactment approach. The modified eye region is synthesized using our unified image-based eye and eyelid model (see the main article for more details). This allows the source actor to take full control of the face expression and eye gaze of the target video stream at real-time frame rates. Our approach leads to plausible reenactment results even for greatly differing head poses in the target video (see Figure 20).
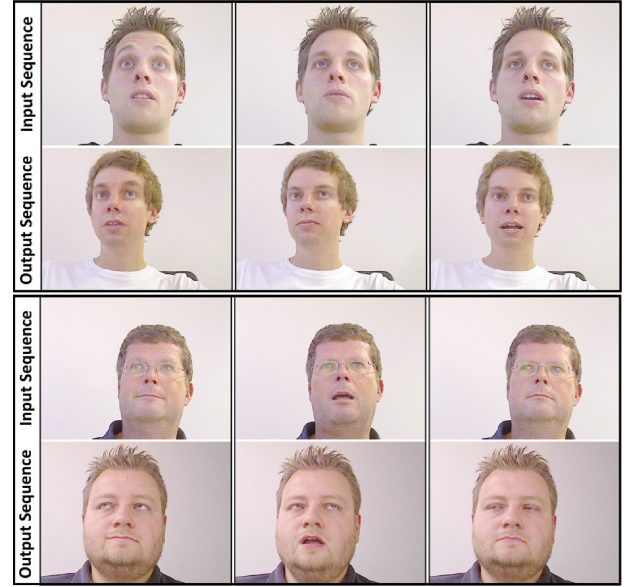

Fig. 17. Gaze-aware facial reenactment of monocular RGB-D video streams. We employ our real-time performance capture and eye tracking approach to modify the facial expressions and eye motion of a target video. In each sequence, the source actor's performance (top) is used to drive the animation of the corresponding target video (bottom). Note that these results employ the mouth retrieval strategy to fill in the mouth interior.
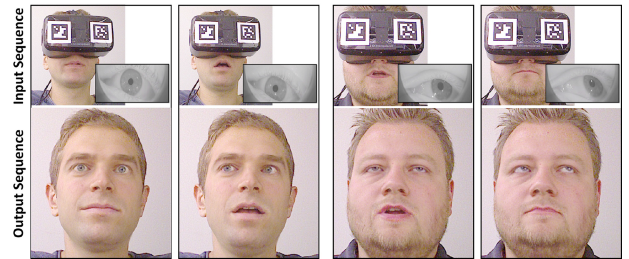

Fig. 18. Self-reenactment for VR video conferences. Our real-time facial reenactment approach allows one to virtually remove the HMD by driving a prerecorded target video of the same person. Note that these results employ the mouth retrieval strategy to fill in the mouth interior.

A.2 Gaze Correction for Video Conferencing

Video conference calls, such as Skype chats, suffer from a lack of eye contact between participants due to the discrepancy between the physical location of the camera and the screen. To address this common problem, we apply our face tracking and reenactment approach to the task of online gaze correction for monocular live video footage (see Figure 21). Our goal is the photo-realistic modification of the eye motion in the input video stream using our image-based eye and eyelid model. To this end, we densely track the face of the user, and our eye-gaze classifier provides us with an estimate of the gaze direction—that is, we determine the 2D screen position where the user is currently looking. Given the eye tracking result, we modify the look-at point by applying a delta offset to the gaze direction which corrects for the different
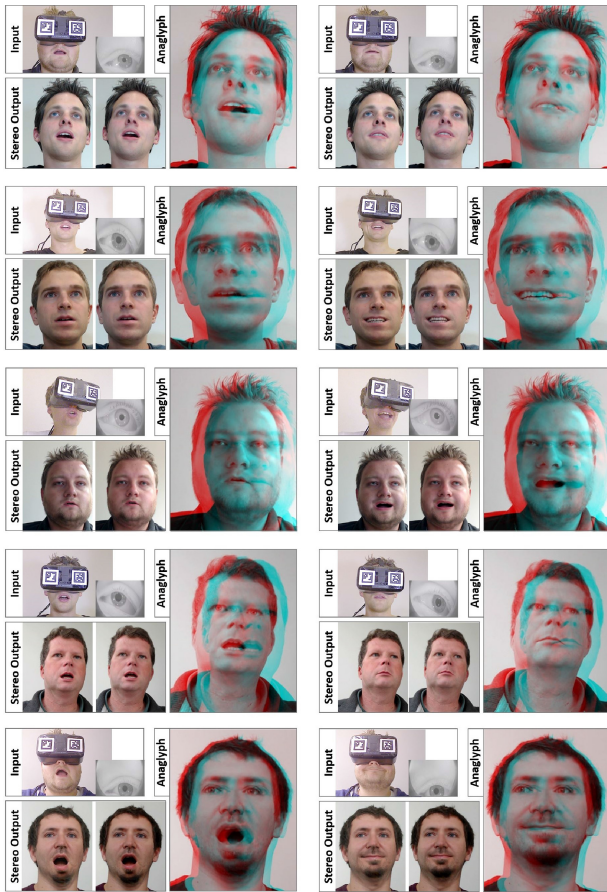
Fig. 19. Gaze-aware facial reenactment of stereo target video content. We employ our real-time gaze-aware facial reenactment approach to modify the facial expressions and eye motion of stereo 3D content. The input (i.e., source actor) is captured with a frontal view and an internal IR camera. With our method, we can drive the facial animation of the stereo output videos shown below the input—the facial regions in these images are synthetically generated. We employ the mouth retrieval strategy to fill in the mouth interior. The final results are visualized as anaglyph images on the right.
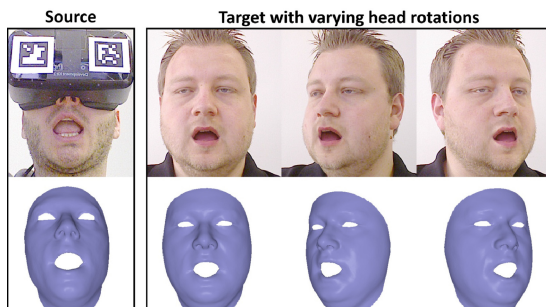

Fig. 20. Reenactment results for different rigid head poses of the target actor. The mouth interior in the frontal view is of highest quality, since the mouth database consists of front facing mouth textures. Rigid rotations of the target actor's face still lead to plausible results with only minor distortions.
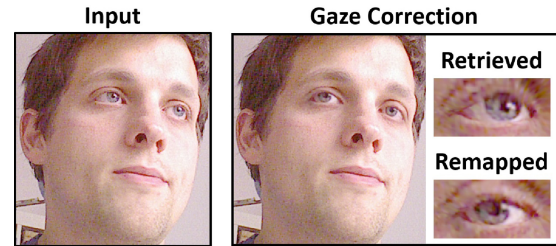
Fig. 21. Gaze correction. A common problem in video chats is the discrepancy between the physical location of the webcam and the screen, which leads to unnatural eye appearance (left). We use our eye tracking and retrieval strategy to correct the gaze direction in such a scenario, thus enabling realistic video conversations with natural eye contact (right).

positions of the camera and screen. Finally, we retrieve a suitable eye texture that matches the new look-at point and composite it with the monocular input video stream to produce the final output. A gaze correction example is shown in Figure 21.

REFERENCES

Oleg Alexander, Mike Rogers, William Lambeth, Matt Chiang, and Paul Debevec. 2009. The digital emily project: Photoreal facial modeling and animation. In *Proceedings of ACM SIGGRAPH 2009 Courses*. Article 12, 15 pages.

Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. 2011. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph.* 30, 4, 75:1–75:10.

Amit Bermano, Thabo Beeler, Yeara Kozlov, Derek Bradley, Bernd Bickel, and Markus Gross. 2015. Detailed spatio-temporal reconstruction of eyelids. *ACM Trans. Graph.* 34, 4, Article 44, 11 pages.

Volker Blanz, Curzio Basso, Tomaso Poggio, and Thomas Vetter. 2003. Reanimating faces in images and video. In *Proceedings of EUROGRAPHICS*, Vol. 22. 641–650.

Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In *Proceedings of SIGGRAPH*. 187–194.

George Borshukov, Dan Piponi, Oystein Larsen, J. P. Lewis, and Christina Tempelaar-Lietz. 2003. Universal capture: Image-based facial animation for "The Matrix Reloaded." In *Proceedings of SIGGRAPH Sketches*. 16:1.

Sofien Bouaziz, Yangang Wang, and Mark Pauly. 2013. Online modeling for realtime facial animation. *ACM Trans. Graph.* 32, 4, Article 40, 10 pages.

Christoph Bregler, Michele Covell, and Malcolm Slaney. 1997. Video rewrite: Driving visual speech with audio. In *Proceedings of SIGGRAPH*. 353–360.

Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. 2015. Real-time high-fidelity facial performance capture. *ACM Trans. Graph.* 34, 4, Article 46, 9 pages.

Chen Cao, Qiming Hou, and Kun Zhou. 2014a. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Trans. Graph.* 33, 4, Article 43, 10 pages.

Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. 2014b. FaceWarehouse: A 3D facial expression database for visual computing. *IEEE Trans. Vis. Comput. Graph.* 20, 3, 413–425.

Chen Cao, Hongzhi Wu, Yanlin Weng, Tianjia Shao, and Kun Zhou. 2016. Real-time facial animation with image-based dynamic avatars. *ACM Trans. Graph.* 35, 4, Article 16.

Antonio Criminisi, Jamie Shotton, Andrew Blake, and Philip H. S. Torr. 2003. Gaze manipulation for one-to-one teleconferencing. In *Proceedings of ICCV*.

Kevin Dale, Kalyan Sunkavalli, Micah K. Johnson, Daniel Vlasic, Wojciech Matusik, and Hanspeter Pfister. 2011. Video face replacement. *ACM Trans. Graph.* 30, 6, 130:1–130:10.

Zachary DeVito, Michael Mara, Michael Zollhöfer, Gilbert Bernstein, Jonathan Ragan-Kelley, Christian Theobalt, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. 2016. Opt: A domain specific language for non-linear least squares optimization in graphics and imaging. arXiv:1604.06525.

Chris H. Q. Ding, Ding Zhou, Xiaofeng He, and Hongyuan Zha. 2006. R1-PCA: Rotational invariant L1-norm principal component analysis for robust subspace factorization. In *Proceedings of ICML*. ACM, New York, NY, 281–288.

Christian Frueh, Avneesh Sud, and Vivek Kwatra. 2017. Headset removal for virtual and mixed reality. In *Proceedings of ACM SIGGRAPH Talks 2017*.

Graham Fyffe, Andrew Jones, Oleg Alexander, Ryosuke Ichikari, and Paul Debevec. 2014. Driving high-resolution facial scans with video performance capture. *ACM Trans. Graph.* 34, 1, Article 8, 14 pages. DOI : http://dx.doi.org/10.1145/2638549

Pablo Garrido, Levi Valgaerts, Ole Rehmsen, Thorsten Thormaehlen, Patrick Perez, and Christian Theobalt. 2014. Automatic face reenactment. In *Proceedings of CVPR*.

Pablo Garrido, Levi Valgaerts, Hamid Sarmadi, Ingmar Steiner, Kiran Varanasi, Patrick Perez, and Christian Theobalt. 2015. VDub—modifying face video of actors for plausible visual alignment to a dubbed audio track. In *Proceedings of EUROGRAPH-ICS*.

Pablo Garrido, Levi Valgaerts, Chenglei Wu, and Christian Theobalt. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph.* 32, 158:1–158:10.

Pablo Garrido, Michael Zollhöfer, Dan Casas, Levi Valgaerts, Kiran Varanasi, Patrick Pérez, and Christian Theobalt. 2016. Reconstruction of personalized 3D face rigs from monocular video. *ACM Trans. Graph.* 35, 3.

Pei-Lun Hsieh, Chongyang Ma, Jihun Yu, and Hao Li. 2015. Unconstrained realtime facial performance capture. In *Proceedings of CVPR*.

Haoda Huang, Jinxiang Chai, Xin Tong, and Hsiang-Tao Wu. 2011. Leveraging motion capture and 3D scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph.* 30, 4, Article 74, 10 pages. DOI : http://dx.doi.org/10.1145/2010324.1964969

Alexandru Eugen Ichim, Sofien Bouaziz, and Mark Pauly. 2015. Dynamic 3D avatar creation from hand-held video input. *ACM Trans. Graph.* 34, 4, Article 45, 14 pages.

Masahide Kawai, Tomoyori Iwao, Daisuke Mima, Akinobu Maejima, and Shigeo Morishima. 2014. Data-driven speech animation synthesis focusing on realistic inside of the mouth. *J. Inf. Process.* 22, 2, 401–409.

Ira Kemelmacher-Shlizerman. 2016. Transfiguring portraits. *ACM Trans. Graph.* 35, 4, Article 94, 8 pages.

Ira Kemelmacher-Shlizerman, Aditya Sankar, Eli Shechtman, and Steven M. Seitz. 2010. Being John Malkovich. In *Proceedings of ECCV*. 341–353.

Oliver Klehm, Fabrice Rousselle, Marios Papas, Derek Bradley, Christophe Hery, Bernd Bickel, Wojciech Jarosz, and Thabo Beeler. 2015. Recent advances in facial appearance capture. In *Proceedings of EUROGRAPHICS STAR Reports*. DOI : http://dx.doi.org/10.1111/cgf.12594

D. Kononenko and V. Lempitsky. 2015. Learning to look up: Realtime monocular gaze correction using machine learning. In *Proceedings of CVPR*. 4667–4675.

Claudia Kuster, Tiberiu Popa, Jean-Charles Bazin, Craig Gotsman, and Markus Gross. 2012. Gaze correction for home video conferencing. *ACM Trans. Graph.* 31, 6, 164:1–174:6.

Pupil Labs. 2016. Home Page. Retrieved April 4, 2018, from https://pupil-labs.com/pupil/.

J. P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. 2014. Practice and theory of blendshape facial models. In *Proceedings of EUROGRAPHICS STAR Reports*. 199–218.

Hao Li, Laura Trutoiu, Kyle Olszewski, Lingyu Wei, Tristan Trutna, Pei-Lun Hsieh, Aaron Nicholls, and Chongyang Ma. 2015. Facial performance sensing head-mounted display. *ACM Trans. Graph.* 34, 4, Article 47.

Hao Li, Jihun Yu, Yuting Ye, and Chris Bregler. 2013. Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph.* 32, 4, Article 42 .

Kai Li, Feng Xu, Jue Wang, Qionghai Dai, and Yebin Liu. 2012. A data-driven approach for facial expression synthesis in video. In *Proceedings of CVPR*. 57–64.

Kyle Olszewski, Joseph J. Lim, Shunsuke Saito, and Hao Li. 2016. High-fidelity facial and speech animation for VR HMDs. *ACM Trans. Graph.* 35, 6, Article 221.

Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. 2010. Fast keypoint recognition using random ferns. *IEEE Trans. Pattern Anal. Mach. Intell.* 32, 3, 448–461.

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. In *Proceedings of ACM SIGGRAPH*. ACM, New York, NY, 313–318.

F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. 1998. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH*. 75–84.

F. Pighin and J. P. Lewis. 2006. Performance-driven facial animation. In *Proceedings of ACM SIGGRAPH Courses*.

Ravi Ramamoorthi and Pat Hanrahan. 2001. A signal-processing framework for inverse rendering. In *Proceedings of SIGGRAPH*. ACM, New York, NY, 117–128.

Shunsuke Saito, Tianye Li, and Hao Li. 2016. Real-time facial segmentation and performance capture from RGB input. In *Proceedings of ECCV*.

Jason M. Saragih, Simon Lucey, and Jeffrey F. Cohn. 2011. Deformable model fitting by regularized landmark mean-shift. *Int. J. Comput. Vis.* 91, 2, 200–215.

Fuhao Shi, Hsiang-Tao Wu, Xin Tong, and Jinxiang Chai. 2014. Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM Trans. Graph.* 33, 6, Article 222.

Christian Siegl, Vanessa Lange, Marc Stamminger, Frank Bauer, and Justus Thies. 2017. FaceForge: Markerless non-rigid face multi-projection mapping. *IEEE Trans. Vis. Comput. Graph.* 23, 11, 2440–2446.

Y. Sugano, Y. Matsushita, and Y. Sato. 2014. Learning-by-synthesis for appearance-based 3D gaze estimation. In *Proceedings of CVPR*. 1821–1828. DOI : http://dx.doi.org/10.1109/CVPR.2014.235

Supasorn Suwajanakorn, Ira Kemelmacher-Shlizerman, and Steven M. Seitz. 2014. Total moving face reconstruction. In *Proceedings of ECCV*. 796–812.

Supasorn Suwajanakorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. 2015. What makes Tom Hanks look like Tom Hanks. In *Proceedings of ICCV*.

Supasorn Suwajanakorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. 2017. Synthesizing Obama: Learning lip sync from audio. *ACM Trans. Graph.* 36, 4, Article 95, 13 pages. DOI : http://dx.doi.org/10.1145/3072959.3073640

Sarah L. Taylor, Barry-John Theobald, and Iain A. Matthews. 2015. A mouth full of words: Visually consistent acoustic redubbing. In *Proceedings of ICASSP*. IEEE, Los Alamitos, CA, 4904–4908.

J. Rafael Tena, Fernando De la Torre, and Iain Matthews. 2011. Interactive region-based linear 3D face models. *ACM Trans. Graph.* 30, 4, Article 76, 10 pages.

Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. 2015. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.* 34, 6, Article 183, 14 pages.

Justus Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. 2016. Face2Face: Real-time face capture and reenactment of RGB videos. In *Proceedings of CVPR*.

Levi Valgaerts, Chenglei Wu, Andrés Bruhn, Hans-Peter Seidel, and Christian Theobalt. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph.* 31, 6, Article 187.

Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popovic. 2005. Face transfer with multilinear models. *ACM Trans. Graph.* 24, 3, 426–433.

Congyi Wang, Fuhao Shi, Shihong Xia, and Jinxiang Chai. 2016. Realtime 3D eye gaze animation using a single RGB camera. *ACM Trans. Graph.* 35, 4, Article 118.

Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee. 2008. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph.* 27, 5, Article 118.

Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime performance-based facial animation. *ACM Trans. Graph.* 30, 4, Article 77.

Thibaut Weise, Hao Li, Luc J. Van Gool, and Mark Pauly. 2009. Face/Off: Live facial puppetry. In *Proceedings of SCA*. 7–16.

Lance Williams. 1990. Performance-driven facial animation. In *Proceedings of SIGGRAPH*. 235–242.

Chenglei Wu, Michael Zollhöfer, Matthias Nießner, Marc Stamminger, Shahram Izadi, and Christian Theobalt. 2014. Real-time shading-based refinement for consumer depth cameras. *ACM Trans. Graph.* 33, 6, Article 200.

Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. 2004. Spacetime faces: High resolution capture for modeling and animation. *ACM Trans. Graph.* 23, 3, 548–558.

Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2015. Appearance-based gaze estimation in the wild. In *Proceedings of CVPR*.

Michael Zollhöfer, Angela Dai, Matthias Innmann, Chenglei Wu, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2015. Shading-based refinement on volumetric signed distance functions. *ACM Trans. Graph.* 34, 4. Article 96.

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. 2014. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Trans. Graph.* 33, 4, Article 156.