Real-time 3D Eyelids Tracking from Semantic Edges

QUAN WEN, FENG XU, MING LU, and JUN-HAI YONG, Tsinghua University

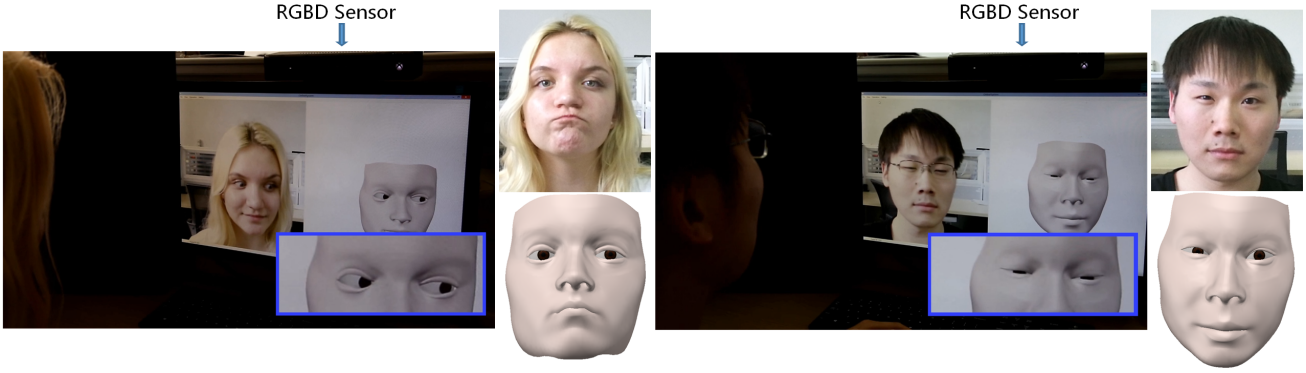

Fig. 1. We present an approach to reconstruct and track 3D eyelids in real time. This technique is integrated into a face and eyeball tracking system to obtain full face results with more detailed eye regions, again in real time (blue rectangles show closeups of the eye region). Our technique successfully reconstructs both eyelid shapes and poses, for instance the shapes of eye contours, double-folds and bulges in the center-left result and the pose differences between the two eyes in the right-most result.

State-of-the-art real-time face tracking systems still lack the ability to realistically portray subtle details of various aspects of the face, particularly the region surrounding the eyes. To improve this situation, we propose a technique to reconstruct the 3D shape and motion of eyelids in real time. By combining these results with the full facial expression and gaze direction, our system generates complete face tracking sequences with more detailed eye regions than existing solutions in real-time. To achieve this goal, we propose a generative eyelid model which decomposes eyelid variation into two low-dimensional linear spaces which efficiently represent the shape and motion of eyelids. Then, we modify a holistically-nested DNN model to jointly perform semantic eyelid edge detection and identification on images. Next, we correspond vertices of the eyelid model to 2D image edges, and employ polynomial curve fitting and a search scheme to handle incorrect and partial edge detections. Finally, we use the correspondences in a 3D-to-2D edge fitting scheme to reconstruct eyelid shape and pose. By integrating our fast fitting method into a face tracking system, the estimated eyelid results are seamlessly fused with the face and eyeball results in real time. Experiments show that our technique applies to different human races, eyelid shapes, and eyelid motions, and is robust to changes in head pose, expression and gaze direction.

CCS Concepts: • **Computing methodologies** → **Motion capture**;

Additional Key Words and Phrases: facial performance capture, eyelid modeling, eyelid tracking, semantic edge detection

This work was supported by the NSFC (No. 61671268, 61672307, 61727808) and the National Key Technologies R&D Program of China (No. 2015BAF23B03). Quan Wen, Feng Xu (corresponding author), Jun-Hai Yong are from TNList and School of Software. Ming Lu is from TNList and Department of Electronic Engineering.

0730-0301/2017/11-ART193 $15.00
https://doi.org/10.1145/3130800.3130837

ACM Reference Format:
Quan Wen, Feng Xu, Ming Lu, and Jun-Hai Yong. 2017. Real-time 3D Eyelids Tracking from Semantic Edges. *ACM Trans. Graph.* 36, 6, Article 193 (November 2017), 11 pages.
https://doi.org/10.1145/3130800.3130837

1 INTRODUCTION

The human face is often the most important body part for a computer to track as it conveys identity and emotion. Thus, facial capture and animation is an important research topic in computer graphics, with applications across movies, computer games, and online communications. Existing techniques reconstruct the face in real time using consumer-level RGB or RGBD sensors, which makes obtaining facial expression cheap and fast [Bouaziz et al. 2013; Cao et al. 2014a, 2013; Li et al. 2013; Weise et al. 2011]. However, state-of-the-art methods majorly focus on face skin regions and are still unable to realistically convey many subtle expressions, such as the shape and motion of the eyes—the window to the soul.

Many recent efforts have improved tracking for specific facial organs, including eyes [Bérard et al. 2016, 2014], eyelids [Bermano et al. 2015], lips [Edwards et al. 2016; Garrido et al. 2016], and teeth [Wu et al. 2016]. These techniques produce high-quality modeling, but are too complex to be applied in real-time. Recently, real-time eyeball modeling and tracking has been achieved [Thies et al. 2016b; Wang et al. 2016; Wen et al. 2016] in face tracking and animation systems. However, this alone is often insufficient for realistic capture. Modeling eyelid shape and motion, including folds and bulges, is still required to generate realistic eye regions in real time.

It is difficult to model the shape and motion of local eye regions because they are small and their motions involve heavy occlusions in fold regions. This is unlike capturing the full facial expression where surfaces are larger and less often occluded. Recent works have used principle component analysis (PCA) to model the overall shape

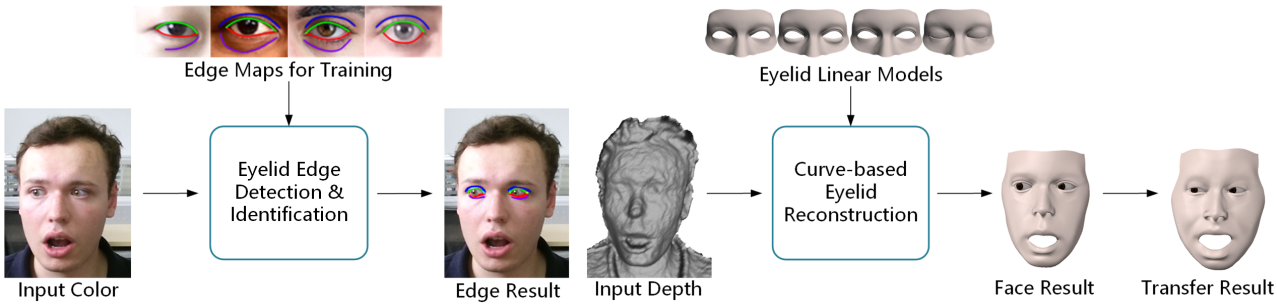

Fig. 2. Overview of our system. Note that the depth input is required by the face fitting technique, not our eyelid reconstruction. We show it here as it contributes to the full face results shown in the figure.

of the eye region from 22 scans [Wood et al. 2016a,b]. However, eye reconstruction should include subtle details like folds, eye contours, and bulges, and real-time performance requires efficient estimation of their motions. Shape from shading can be used to reconstruct these details [Garrido et al. 2013; Richardson et al. 2016; Shi et al. 2014], but these systems are offline. In online systems, only mid-scale facial features, like strong wrinkles, can be reconstructed [Cao et al. 2015].

In this paper, we propose a system to reconstruct more detailed eye regions in real time. First, we propose a high-fidelity generative eyelid model with a set of dimensions which independently represent the shapes and motions of eye contours, folds, and bulges. The model is a linear parametric model which efficiently reconstructs eye regions via linear combinations. Second, we propose projective fitting of semantic eye region edges to reconstruct eye regions in real time. The semantic eye region edges are extracted by a multi-channel holistically-nested edge detector, which jointly achieves edge detection and identification. Third, we propose a polynomial curve fitting technique and a correspondence updating technique to handle incorrect and missing edge results, and to achieve real-time performance of the projective fitting.

We list our specific contributions:

- Real-time reconstruction, tracking, and animation of realistic human eyelid shape and motion. It is combined with face and eyeball tracking to generate more complete and vivid result.
- A linear parametric eyelid model for human eyelid shape and motion. It represents details and fits real-time applications.We will release the model to the community.
- A real-time semantic edge-based eyelid fitting solution. The edge detector uses semantic information in a holistically-nested DNN model. Detailed eyelids are reconstructed in real time by a novel 3D-to-2D projective edge fitting algorithm.

1.1 Related Work

We propose a more detailed eyelid model to represent eyelid shape and motion, and use this model to achieve real-time eyelid reconstruction. As there is no previous work focusing on exactly the same goal, we briefly survey models and reconstruction techniques for general faces.

Face Models. To better model, track, and animate human faces, generative 3D face models have been proposed which involve prior knowledge of face geometries and motions. Through these models, a 3D face can be represented and reconstructed in a low dimensional space. Blanz and Vetter [1999] propose the first 3D morphable face model from 200 face scans of different individuals with fixed neutral expression. The morphable face model is further extended with 1000 face scans [Booth et al. 2016] and with both shape and texture [Paysan et al. 2009; Zhu et al. 2015]. Besides facial identity, facial expression is also represented in a low dimensional space. The blendshape model, which consists of a set of key expressions of a particular facial identity, can be used to represent novel expressions by linear combinations of blendshapes. Blendshapes can also be generalized to novel identities by deformation transfer [Sumner and Popović 2004] or example-based rigging [Li et al. 2010]. The functions of morphable model and blendshape model can also be achieved in one model, called a multilinear model [Cao et al. 2014b; Vlasic et al. 2005], in which facial identity and expression are independently modeled by two sets of parameters.

These models focus on the overall face, but are not designed for specific face parts. This is crucial to attaining higher fidelity in face modeling. In [Edwards et al. 2016], a 3D viseme model, called Jali, is proposed to model the speech-related mouth motion. Olszewski et al. [2016] use a new blendshape model with 29 shapes for mouth poses to better model the mouth regions (5 of them for modeling tongue motions). Recently, teeth have been modeled delicately with a tooth row model and a local shape model for individual teeth [Wu et al. 2016]. For eyeballs, a morphable model has been proposed to achieve lightweight eye capture [Bérard et al. 2016]. For modeling eyelid and eye regions, early works focused on the domain of 2D images, which are well surveyed [Ruhland et al. 2014]. In recent years, 3D eye region models have been proposed [Wood et al. 2016a,b], where a PCA model with 8 dimensions is derived to model the overall shape of eye regions in a fully opened eye pose. However, there is no 3D generative model that can represent eye shapes with folds and bulges, let alone model the eyelid dynamics at the same time.

Face Reconstruction. We can generate high quality face shapes and dynamics with stereo [Beeler et al. 2011] and shape from shading (SfS) techniques [Garrido et al. 2013; Shi et al. 2014]. For eye regions, delicate eyeball models are reconstructed by a complex

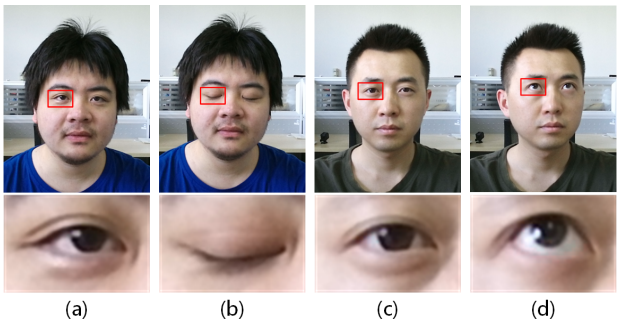

Fig. 3. Folds and bulges with different poses. The bottom image is zoomed in from the left side eye region. (a,b) shows that the fold disappears with the eye-closing motion. (c,d) shows that the bulge disappears with the upwards-looking motion.

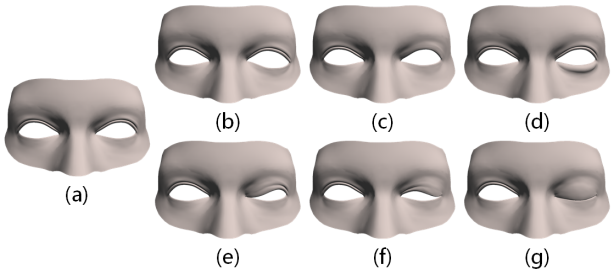

Fig. 4. Example bases in shape and pose rigs. (a) the basic eyelid mesh: b_0^{id} and b_0^{exp}; (b-d) three shape bases with eye contour change, fold change and bulge change on the right-side eye, b_{11}^{id}, b_{21}^{id} and b_{23}^{id}; (e-g) three pose bases with downward motions on the inner part, the outer part and the whole of the upper eyelid of the right-side eye, b_3^{exp}, b_5^{exp} and b_1^{exp}.

capture and processing technique [Bérard et al. 2014]. An eyelid morphable model is also proposed and used for image-based eye region animation [Wood et al. 2017]. Neog et al. [2016] use a cubic spline curve to build an eyelid shape model to learn the correlation between eyeball direction and eyelid motion. Besides these large-scale eyelid modeling and animation techniques, detailed dynamics of eyelid folds are also modeled and reconstructed with multi-view input and offline processing [Bermano et al. 2015]. None of these techniques are real time, which limits their application.

Real-time 3D face tracking and animation was first demonstrated by Weise et al. [2011] for a specific user with a RGBD sensor. This work was extended by Li et al. [2013] and Bouaziz et al. [2013] to remove the specific user requirement, and by Cao et al. [2013] to require only RGB input. Later, Cao et al. [2014a] combined these two extensions and reconstructed medium-scale face features like strong wrinkles [Cao et al. 2015]. Occlusion handling can improve the robustness of facial tracking systems [Hsieh et al. 2015; Liu et al. 2015; Saito et al. 2016]. To improve realism, attention has now turned to face parts. Wang et al. [2016] and Wen et al. [2016] track eyeball rotations from RGB and RGBD input in real time. In addition, following the works of real-time face reenactment on RGBD and RGB inputs [Thies et al. 2015, 2016a], Thies et al. [2016b] achieve gaze retargeting for face reenactment with virtual reality (VR) headsets. This is similar to other research on VR-based face tracking [Li et al. 2015; Olszewski et al. 2016]. All the aforementioned real-time systems handle neither precise eyelid shapes nor motions.

1.2 Overview

Our system (fig. 2) first proposes two linear models for representing the eyelid shape and pose in low dimensional subspaces (section 2). Meanwhile, we train an improved holistically-nested network to jointly achieve detection and identification of four semantic edges in the eye regions (section 3). In the online tracking stage, with the input color and depth sequence, we integrate our eyelid fitting technique into a face and eyeball tracking system introduced by Wen et al. [2016]. As the eyelid models have been pre-aligned to the face models, we directly perform eyelid shape and pose tracking (section 4) in the face tracking system. By iteratively solving the optimizations, we recover the eyelid model parameters that best fit the four edges detected on the image. Finally, as our eyelid models are generative, we can transfer the reconstructed eyelid motions to a novel eyelid identity in real time.

2 LINEAR EYELID MODELS

For 3D face modeling, previous works use morphable models to represent the identity/shape changes of a face, and blendshape models to represent expression/pose changes. Similarly, we also use two linear models, with two sets of 3D mesh bases, to represent the shape and pose variations of eyelids independently. In this manner, users can control the shape and pose more freely. However, for faces, the shape and pose correspond to totally different changes on face geometry, while for eyelids, they may cause the same changes. For example, different identities may or may not have folds and bulges, and different poses (e.g., opened/closed, looking forwards/upwards) may also cause the folds and bulges to appear and disappear, as shown in fig. 3. Our two linear rigs share identical bases to handle this issue.

2.1 Shape Linear Rig

Eyelids vary among genders, races, and ages. The characteristics of an eyelid can be categorized as follows:

- *Position* stands for the relative positions of the eyes on a face. It is represented by the vertical location of the eye pair and the horizontal distance between the two eyes.
- *Contour shape* of the eye is affected by whether the eyes are round or flat, wide or narrow in the horizontal direction, upturned (the outer eye corner is higher than the inner eye corner) or downturned (the outer eye corner is lower than the inner eye corner).
- *Double-fold* is described by the distances between the fold and the upper eyelid on different parts, and the strength of the fold. Note that a single-fold eyelid, commonly seen in Asian populations, is an extreme case of the double-fold eyelid. It can be represented as a zero-strength double-fold eyelid.
- *Bulge* indicates the shape and strength of a bulge beneath an eye. 'No bulge' is represented by a zero-strength bulge.

Based on the observations above, we asked an artist to build a set of eyelids to cover the characteristic variations. We asked the artist

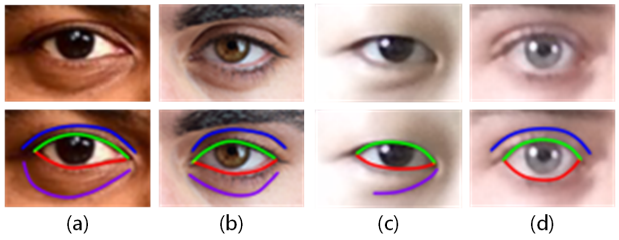

Fig. 5. Detection and identification of the four semantic edges. Each color corresponds to one kind of semantic edge (blue: fold edge; green: top edge; red: bottom edge; purple: bulge edge). (a,b) examples with all four edges; (c) an example without fold edge; (d) an example without bulge edge.

to first build a basic 3D eyelid mesh with a neutral identity in a neutral, open pose, and then modify the basic mesh to generate a set of new meshes. The artist edited these meshes to cover the eyelid space as much as possible, based on her experience and photographic reference: https://imgur.com/a/gitnp. The difference between each new mesh and the base mesh represents one dimension of one characteristic, with the topology and the semantic meaning of the vertices kept between all meshes. We use this set of eyelids to construct the bases of our linear shape model:

$$B^{id} = \{b_k^{id} | k = 0, ..., N^{id} - 1\}, N^{id} = 29, \tag{1}$$

where b_0^{id} is a pair of neutral eyelids. Details of the bases in the shape rig, along with the following pose rig, are described in the appendix, and we show a few eyelid bases in fig. 4. Note that for position, the two eyes always move together (upper or lower, closer or farther), but for other characteristics, the two eyes could change independently as they may be slightly different for most people. Thus, our bases either relate to two eyes or to one eye.

With the shape bases and a set of blending weights, the eyelid model of a specific user in a neutral, open pose is synthesized by:

$$E_N = b_0^{id} + \sum_{k=1}^{N^{id}-1} w_k^{id}(b_k^{id} - b_0^{id}), \tag{2}$$

where $w^{id} = (w_0^{id}, ..., w_{N^{id}-1}^{id})^{\mathrm{T}}$ is the shape weight.

2.2 Pose Linear Rig

We build the pose rig by manually generating a set of bases to cover all possible geometric changes caused by eyelid poses:

$$B^{exp} = \{b_k^{exp} | k = 0, ..., N^{exp} - 1\}, N^{exp} = 23, \tag{3}$$

where b_0^{exp} is the same as b_0^{id}. Note that eyelid fold and bulge may change with either shape or pose. Thus, there are some shared bases for both shape and pose rigs to control the eyelid fold and bulge.

In practice, we first use the shape rig to estimate E_N for a particular user. However, since b_0^{exp} is different from E_N, B^{exp} cannot be directly used to construct the pose rig for E_N. To overcome this drawback, we use deformation transfer [Sumner and Popović 2004] to recover $B^{exp'}$ by transferring the deformation gradient between

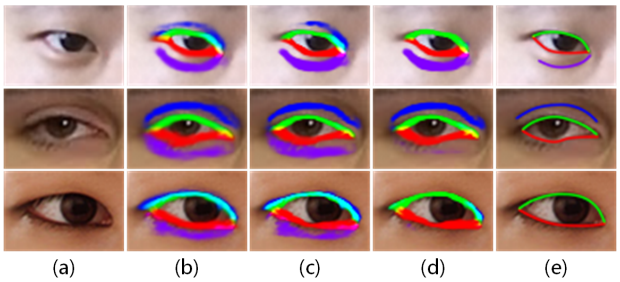

Fig. 6. Comparisons of different solutions on edge detection and identification. (a) input images; (b) results of four separate HEDs; (c) results of our network with separate loss defined in eq. (8); (d) results of our network with uniform loss defined in eq. (6); (e) ground truth. Note that as both (b) and (c) separately consider the four edges, they always detect all the four edges no matter whether or not the images contain fold edges or bulge edges.

b_0^{exp} and b_i^{exp} to E_N. Correspondingly, the pose rig is used as:

$$E_P = b_0^{exp'} + \sum_{k=1}^{N^{exp}-1} w_k^{exp}(b_k^{exp'} - b_0^{exp'}). \tag{4}$$

Here, $'$ stands for the transferred bases.

3 EYELID EDGE DETECTION AND IDENTIFICATION

We discuss how to detect and identify the four edges for eyelid motion capture. The four edges have different semantic meanings representing the double-fold, the upper eyelid, the lower eyelid and the lower boundary of the bulge, respectively, as shown in fig. 5. For simplification, we name the four edges as fold edge, top edge, bottom edge and bulge edge correspondingly. Note that the top edge and the bottom edge always exist in all facial images (coincident with each other when the eye is closed), but the fold edge and the bulge edge may not exist for some eye shapes or poses.

Detecting and identifying the four edges is not a trivial task. Recently, DNN-based edge detection techniques have demonstrated noticeable improvements in both detection accuracy and performance. However, in this paper, we need to not only detect but also distinguish the four edges to perform eyelid fitting, which is not considered in previous edge detection techniques. A naive extension to identify the four edges is to train four edge detectors, each of which is individually trained to detect only one of the four edges. However, for the four edges of each eye region, their relative positions, shapes, and motions are highly correlated. Training four detectors separately ignores those correlations and does not achieve good results, as shown in fig. 6(b).

3.1 Network

To exploit the correlation among the four edges, we modify the holistically-nested edge detection (HED) proposed by Xie et al. [2015] to train a uniform network that jointly detects and identifies the four edges. We first formulate the edge detection and identification problem as a multi-channel edge detection problem and then propose a unified energy metric to jointly consider the four edges together in the network training, which learns the correlations of the four edges.

The HED is based on the VGG-16 net [Simonyan and Zisserman 2014]. It connects five side-output layers to each stage of the VGG-16 net. Each side-output layer generates an edge detection result from the VGG features in the corresponding stage. Each result is supervised by the ground-truth edge map, and all results are fused together to generate the final output, which is also supervised by the ground truth. In our system, since identifying the four edges is required, we represent our output and also the ground truth as four-channel binary edge maps, where 1 in each channel stands for pixels on one of the four edges and 0 is used to label other pixels. Comparing with the network in the original HED, we use four convolution kernels, each corresponding to one channel in each side-output layer and also in the final fused layer. In this manner, we create a modified network that fits the representation of the output and jointly achieves detection and identification of the four edges.

3.2 Loss

To train our network to learn optimal network parameters θ, we formulate the loss function as follows:

$$\arg\min_{\theta}(\alpha_f L(\psi_f(I,\theta),G)+\sum_{k=1}^{M}\alpha_{s_k}L(\psi_{s_k}(I,\theta),G)). \tag{5}$$

In this function, I and G are the input eye region image and the four-channel ground truth edge map. ψ_f and ψ_{s_k} stand for the fused output and the side-outputs of the network. The loss function L computes the pixel-wise sigmoid cross-entropy loss between an output edge map and the ground truth. M represents the number of side-outputs in the network (5 in our experiments). α_f and α_{s_k} are the weight parameters for the fused loss and the side losses.

We define a unified loss L that integrates the detection and identification errors of the four edges together:

$$\begin{aligned}L(\psi(I,\theta),G) = &-\beta\sum_{j\in G_+}\log\Pr(g_j=0|I;\theta)\\ &-(1-\beta)\sum_{i=1}^{4}\sum_{j\in G_-^i}\log\Pr(g_j^i=1|I;\theta),\end{aligned} \tag{6}$$

where

$$\beta=\sum_{i=1}^{4}\left|G_-^i\right|/|G|\,. \tag{7}$$

G_-^i is the set of pixels belonging to the ith edge in the ground truth, and thus β stands for the ratio of the number of edge pixels in the ground truth to the number of all pixels in the eye region image. G_+ is the union of all the four set of non-edge pixels in the ground truth and each set is defined as G_+^i. $\Pr(g_j^i=1|I;\theta)$ stands for the probability of pixel j belonging to the ith edge in an output, while $\Pr(g_j=0|I;\theta)$ stands for the probability of belonging to non-edges. $\Pr=\sigma(a_j)$, where σ is the sigmoid function and a_j is the activation value of pixel j in our DNN. With the definition of eq. (6), minimizing eq. (5) leads to a network generating desired output. One example of the input, ground truth, and the output of our network is shown in fig. 6.

There is a more straightforward way to define L by the individual detection errors of the four edges:

$$\begin{aligned}L(\psi(I,\theta),G) = \sum_{i=1}^{4}(&-\beta^i\sum_{j\in G_+^i}\log\Pr(g_j^i=0|I;\theta)\\ &-(1-\beta^i)\sum_{j\in G_-^i}\log\Pr(g_j^i=1|I;\theta)),\end{aligned} \tag{8}$$

where

$$\beta^i=\left|G_-^i\right|/|G|\,. \tag{9}$$

However, in this definition, the four edges independently contribute to different terms, thus their correlations are not considered in the loss. As a consequence, compared with eq. (6), eq. (8) generates incorrect results as shown in fig. 6(c).

3.3 Training

Our training set contains 194 eye region images from 48 identities and the corresponding four-channel ground truth edge maps (manually labeled). The images of 5 identities are recorded by ourselves and another 3 are from the Eyediap database [Mora et al. 2014]. These 8 identities contribute 57 facial images, each of which provides two eye region images. For each identity, our data set contains images with different eyelid poses. The pose changes are caused by eyelid motions, as well as gaze motions, because eyelids change with eyeball movements. The remaining 40 identities are collected from the Internet, each of which has one image. These have no eyelid pose change, and they contribute 80 training samples in total. Note that the fold edge and the bulge edge do not exist in some images: some subjects may not have double eyelids or eye bulges; the top edge may disappear when subjects look downwards or close their eyes. In these cases, we do not label fold edges or bulge edges.

In the training, the network is fine-tuned from an initialization of the pre-trained VGG-16 net model, and we use a standard stochastic gradient descent algorithm in the training. The parameters in the training are set as follows: learning rate (1e-6), momentum (0.9), weight decay (0.0002). The weight parameters α_f and α_{s_k} are all 1.

4 CURVE-BASED EYELID RECONSTRUCTION

We discuss how to reconstruct the 3D eyelids of a user with our linear eyelid models and the four edges identified on each recorded frame. The core idea is to estimate the optimal shape and pose weights in our linear eyelid models, aiming to minimize the inconsistency between the projected eyelid edges and the real eyelid edges on the image. To construct the optimization problem, we need to define the correspondences between the 3D eyelid model and 2D pixels. The edges of the eyelid model are defined by a set of manually-labeled mesh vertices, called 3D eyelid landmarks. Their corresponding pixels, called 2D eyelid landmarks, are extracted by first fitting four polynomial curves to the detected edges on the edge map (section 4.1), and then determining their locations on the curves (section 4.2). With the correspondences, we minimize an energy function that measures the distances between the projected 3D landmarks and their corresponding 2D landmarks (section 4.3). Note that our eyelid reconstruction method is integrated into a real-time face tracking system [Wen et al. 2016], which reconstructs

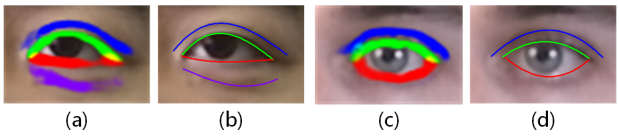

Fig. 7. Results of curve fitting. (a,c) two input eye region images with detected edges; (b,d) the corresponding curve fitting results.

3D global poses, face identities and expressions, as well as the eyeball shapes and gaze directions, all in real time. Combining the 3D eyelid generated by our technique, we recover more complete face reconstruction results with more realistic eye regions.

4.1 Curve Fitting

Our goal of eyelid reconstruction is to fit the four eyelid edges in our eyelid model to the corresponding edge detection results. However, the output of the deep learning network is a pixel-wise four-channel edge map which may label non-edge pixels as edges or miss some true edge pixels. To calculate accurate correspondences in this situation, we use the global information of the edges to extract curves from the edge maps, and the corresponding 2D points of the 3D eyelid landmarks are located on the curves. To be specific, we fit four polynomial curves to the four channels of the edge map by a weighted least square method which solves the following minimization problem:

$$\arg\min_{A} \sum_{k \in C} \omega_k \| P(A, x_k) - y_k \|^2, \tag{10}$$

where $A = \{a_i | i = 0, ..., N_A - 1\}$ stands for the polynomial parameter ($N_A = 3$ is sufficient for all the four curves), P is the polynomial function, C contains all the pixels in a channel of the edge map with intensity values larger than a threshold (set to 0.2 in all our experiments). (x_k, y_k) is the 2D coordinates of pixel k on the edge map, and the weight parameter ω_k is set to the pixel intensity of the channel in the edge map, which indicates that higher possibility of an edge pixel contributes more to the energy. An example of curve fitting is shown in fig. 7.

4.2 Correspondence

We compute the locations of 2D eyelid landmarks corresponding to the 3D eyelid edge vertices. First, as mentioned before, the 3D eyelid edge vertices are manually labeled, but we only need to label the vertices on b_0, as all bases share the same topology and semantic meanings on vertices. To be specific, we select two end vertices for each of the four eyelid edges (the upper eyelid and the low eyelid share the same end points). Then, all vertices in-between the two endpoints are selected as 3D landmarks.

One difficulty in finding the correspondences is that, due to the impact of lighting or head poses, we may not detect the whole edge on the image. For the top and bottom edges, as their end points always overlap, we identify the end points by calculating the intersection points of the polynomial curves. Next, for the in-between pixels, we use the curve length to determine the corresponding points (shown in fig. 8(b)). A straightforward solution here is to keep the relative curve lengths on the 3D eyelid model, determined by the selected vertices, the same as the relative curve lengths on the 2D image, determined by the extracted pixels on the polynomial curves:

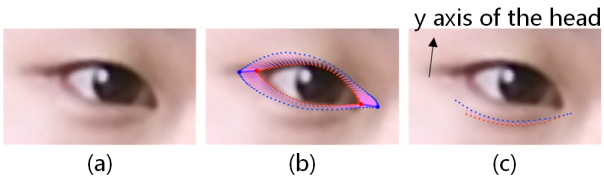

Fig. 8. Correspondences of different edges. (a) input eye region image; (b) correspondences of the top and bottom edges; (c) correspondences of the bulge edge. Blue points indicate the projected vertices of the 3D eyelid model, and the red points indicate their corresponding points on the polynomial curves. In (b), the end points on the 3D model always match the end points on the 2D curve. In (c), the correspondences are obtained by searching along the projected y axis of the head mesh, and some vertices may not find correspondences on the image.

$$s * \ell(v_t, v_{t-1}) = \ell(u_t, u_{t-1}), t = 1, ..., T - 1, \tag{11}$$

where s is a scaling factor $\ell(u_{T-1}, u_0)/\ell(v_{T-1}, v_0)$, l represents the curve length between two points (in 2D or 3D space), v_t denotes a 3D vertex, u_t denotes its corresponding pixel on 2D image and T is the number of 3D landmarks on the edge. However, this solution does not work well when the 3D vertices are not on the same depth layer to the camera. In this case, vertices with uniform intervals should be ideally projected onto 2D locations with nonuniform intervals. To overcome this drawback, we modify eq. (11) to set the following constraint:

$$s * \ell(\pi(v_t), \pi(v_{t-1})) = \ell(u_t, u_{t-1}), t = 1, ..., T - 1, \tag{12}$$

where π denotes the 3D to 2D projection. In this case, curve length is always calculated in 2D space, and thus it holds in all situations. However, since different head poses may cause very different projective curve lengths in 2D space, the left side of eq. (12) should be calculated in the optimization on each frame, rather than using a pre-computation for eq. (11). Therefore, in our method, the correspondences are dynamically updated in the optimization.

To estimate u_t satisfying eq. (12), we need to solve a nonlinear optimization problem, as the 2D curve is a three-order polynomial curve. To achieve real-time performance, we propose an approximate solver. We first calculate all $\ell(\pi(v_t), \pi(v_{t-1}))$ in the current iteration. Then, we densely sample some points on the polynomial curve by $x_p \in \left[x_k^{min}, x_k^{max}\right]$ (x_k^{min} and x_k^{max} denote the minimum and maximum x value in all pixels in C.), and calculate the 2D distance $D(u_p, u_{p-1})$ between each pair of consecutive points $u_p = (x_p, P(A, x_p))$ and $u_{p-1} = (x_{p-1}, P(A, x_{p-1}))$. Finally, for a desired u_t, we use binary search to recover u_{p*} that best satisfies:

$$s * \ell(\pi(v_t), \pi(v_0)) = \sum_{p=1}^{p*} D(u_p, u_{p-1}). \tag{13}$$

From our experiments, this binary search-based method achieves real-time performance and satisfying fitting results.

For fold and bulge edges, we cannot use the described approach to calculate correspondences, as the endpoints cannot be located when

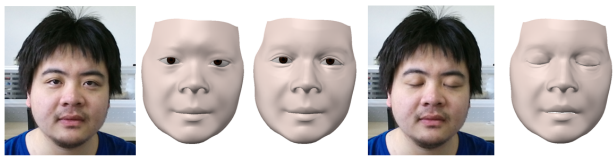

Fig. 9. Results in our fitting system. (a) First input frame with neutral expression; (b) identity fitting result provided by [Wen et al. 2016]; (c) result of our system with eyelid fitting; (d) One input frame in the sequence; (e) the corresponding result of this frame.

only partial edges on the image are detected. In practice, in each iteration, we first project the labeled vertices of the fold and bulge onto the recorded image and then use the curve points with the same y coordinate as their corresponding points. Note that this is the y coordinate in the head model space projected onto image space. Directly using the image space y coordinate will involve incorrect correspondences when the face undergoes in-plane rotation. Also, some vertices may not have correspondences if there are no points on the polynomial curve with the same y coordinates, but the whole edge can still be reconstructed from our eyelid model and the partial correspondences. Fig. 8(c) illustrates the correspondences for a bulge edge.

4.3 Eyelid Reconstruction

We discuss the estimation of eyelid shape and pose weights in our real-time tracking system. The weights $\{w\}$ are obtained by minimizing the distances between the projections of the 3D eyelid landmarks and their corresponding 2D eyelid landmarks, formulated as:

$$\arg\min_{w} \sum_{i}^{4} \sum_{t \in S^i} \alpha_t^i \| \pi(v_t^i(w, B)) - u_t^i \|_2^2, \tag{14}$$

where i indicates different edges, S^i contains all the correspondence pairs of edge i and α_t^i is the weight of the correspondence, which is set to 2 for end points ($t = 0, T-1$) and 1 for others.

In general, the fitting of eyelid shape and pose is integrated into the face fitting pipeline [Wen et al. 2016] by following the face identity and expression fitting in each iteration. To achieve integration, we also need to match our 3D eyelid mesh model to the multilinear model used by Wen et al. [2016]. Otherwise, the face result and the eyelid result cannot be fused together to recover one consistent result. First, we manually select the eyelid boundary on the multilinear model. Then we apply Laplacian deformation [Sorkine et al. 2004] to deform our eyelid bases to fit the multilinear bases. Thus our eyelid meshes can be seamlessly connected to the face meshes, and the eyelid fitting can seamlessly follow face fitting to generate more realistic results on eye regions.

In the first frame of a sequence, where users are asked to keep a neutral expression, we perform eyelid shape fitting after face identity fitting in each iteration. Here, the shape rig B^{id} is used in eq. (14) and the correspondences are updated by the method in section 4.2. Then, the optimal w^{id} is estimated and fixed, and the eyelid model E_N with neutral expression is synthesized. Next, we construct the user-specific eyelid pose rig B^{exp} based on E_N, as described in section 2.2. After the first frame, eyelid pose fitting is iteratively performed with head pose estimation and facial expression fitting on the following frames in real time, i.e., w^{exp} is estimated using B^{exp}. Finally, as the eyeball performance is also reconstructed by Wen et al. [2016], we recover a near-complete face reconstruction with realistic eye regions (but without teeth and detailed lip motions). Some results of this procedure are shown in fig. 9.

5 EXPERIMENTS

We discuss experiments performed using the described approach. First, since we add a new eyelid tracking module to an existing real-time face tracking system, the performance of the new system is discussed. Second, as the eyelid linear models and the four edge detection are our key techniques to achieve real-time eyelid tracking, we evaluate the two techniques respectively. Third, we compare our system with the existing system. As we generate more details in the eye regions, our results are more vivid and realistic. Next, to demonstrate the power of our whole technique, we show our results on various eyelid shapes and poses and we show eyelid motion transfer results for facial animation applications. Finally, we discuss the limitations of our techniques.

Performance. Our system runs on a computer with a 3.60 GHz eight-core CPU, 16 GB RAM, and an NVIDIA Geforce GTX 980 graphics card. For each input frame, face and eyeball tracking takes ≈30ms. For eyelid tracking, our semantic edge detection, identification, and curve fitting takes 11ms. As this part only requires eye region images as input, it is performed on another thread on GPU in parallel with the face tracking method. Our eyelid correspondence update and energy minimization take 7ms, and are pipelined with the head pose and facial expression fitting steps. Thus, our system requires a total of 37ms to process each frame.

5.1 Evaluations

First, we evaluate our proposed generative eyelid model. As our model is constructed by a set of shape and pose bases, we show the effectiveness of some bases individually in the accompanying video. It demonstrates that different bases control different eyelid variation, including eye region positions, eye contour shapes, fold shapes and strengths, bulge shapes and strengths, and closing-eye motions. Note that except the eye region positions, all the other characteristics can be separately controlled on each of the eyes. For simplification, we control them together in the video.

Second, we evaluate our edge detection and identification scheme. Although we have not collected a large dataset to train our DNN model, we still find that the model shows good generalization capability on various test data downloaded from the Internet. As shown in fig. 10, various Internet eyelid images with different eye shapes, folds, bulges, skin colors, lighting conditions, long or short eyelashes, and pupil colors are all correctly handled by our method. The strong structural information of the four semantic edges contributes to the generalization capability of our model, which learns the information by the network with the designed uniform loss.

Besides the visual comparison in fig. 6, we also numerically compare our detection and identification scheme with the alternative solutions discussed in section 3. As shown in fig. 11, our solution

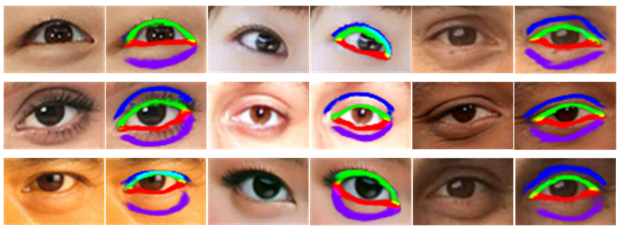

Fig. 10. Results of edge detection and identification on internet images. Different colors indicate different semantic edges used in our eyelid fitting method.

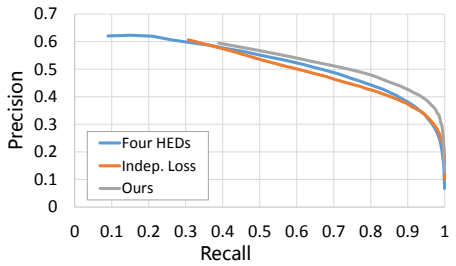

Fig. 11. Numerical comparison of different edge extraction methods. Note that our method outperforms the two compared methods as for the same recall, we always give higher precision, and for the same precision, we always give higher recall.

outperforms the approach using four HEDs and the one using independent losses. Note that the numbers in fig. 11 are calculated from 100 Internet images which were manually labeled. Note that it is difficult to obtain high precisions (close to 1) with our method, because our detected edges are always much wider than the ground truth (fig. 6). However, it does not affect our whole method because we have a curve fitting scheme to extract the true edges more precisely.

5.2 Comparisons

Since there is no previous work focusing on real-time shape and pose reconstruction of 3D eyelids, we compare our system with the most related previous approach which tracks the motions of face regions and eyeballs [Wen et al. 2016]. As shown in fig. 12, with the eyelid tracking, our reconstructed eye regions are more vivid and consistent with the input images, while Wen et al. [2016] generates eyelids with fixed shape and unnatural poses as their multilinear model does not cover the variations of eye regions and there are not enough features on the eye regions extracted to perform more delicate eyelid fitting. Further comparisons are shown in the accompanying video, better demonstrating that our technique improves the visual quality of the results.

5.3 Results

We execute our system on facial sequences with various identities and eyelid motions. Some selected frames are shown in fig. 13. We can also observe subtle pose changes reconstructed by our method. For example, the middle two results in the first row have different strengths on double-folds. The middle two results in the second row show the generation of bulges caused by the smile motion. The third result in the fourth row shows the slightly different poses of the two eyes. Please refer to the accompanying video for sequence results. Besides the subtle motions, our technique is robust to eye glasses (Live Demo I) or hair occlusions (Result III). This is contributed by the robustness of our edge detection and identification method.

Table 1. 2D numerical errors of the tracked eyelid landmarks. The error is the average distance between the projected eyelid edge vertices and the corresponding 2D landmarks normalized by inter-pupil distance.

	fold	top	bottom	bulge
Seq. Lina	1.179%	0.879%	0.524%	0.522%
Seq. Sun	0.933%	0.652%	0.73%	0.515%
Seq. Zhang	0.893%	0.756%	0.679%	0.603%

To measure the accuracy of the final reconstructed eyelids, we calculate the 2D numerical errors of the eyelid landmarks, i.e., the distances between the projected eyelid edge vertices and their corresponding 2D landmarks on the 2D curves. From table 1, we can observe that the eyelid edge vertices match the 2D landmarks well after the energy minimization.

Animation. As our reconstructed motions are represented by weights of bases of our eyelid model, we are able to transfer motion across identities. One example is shown in the accompanying video, where we transfer a sequence of facial motions to two identities with different face and eyelid shapes. Since our shape and pose models share the same bases for folds and bulges, directly transferring the source weights onto these bases may cause shape changes in the target. To overcome this drawback, we transfer the weight changes to the weights in the rest pose of the source to that of the target. As the weight changes represent the dynamics in the source sequence, they are likely to preserve the target shape but generate the source motions.

Note that our transfer technique is not similar to blendshape-based facial motion transfer where blendshapes with consistent semantic meanings are available for both source and target. In our transfer technique, since we do not have any prior knowledge on the characteristic of the dynamics of the target, the transferred motions always contain some source dynamics that may not match the target shape. This problem may be solved with a better pose rig of the target. We could use techniques like those form Li et al. [2010] to replace the simple deformation transfer in constructing the pose bases.

5.4 Limitations

First, as the artist-designed bases do not cover all the variations in real eyelids, the fitting attained using them still has errors. Also, our model focuses on eye contour shape, double-folds and bulges, thus we do not consider other details like wrinkles near eye regions. Generating arbitrary fine-scale wrinkles in real time is still an open problem (though medium-scale wrinkle capture has been solved by Cao et al. [2015]). Second, we only collect a small dataset for training our eyelid edge detector, so input images largely different from our training set will not be processed well, for example, images recorded

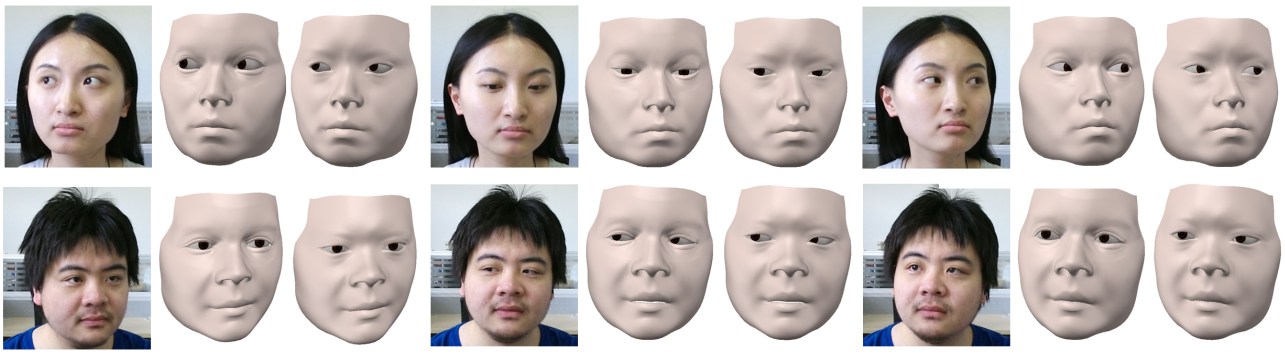

Fig. 12. Comparisons with Wen et al. [2016] on six selected poses of two users. In each comparison, we provide the input image, the result of our method and the result of Wen et al. [2016] from left to right.

in challenging lighting conditions. The accompanying video shows an example of this case, where the bulge detection is not consistent over time due to the lighting change. Involving more training images under the target environment may solve this problem. Third, our system still requires depth information in face tracking. However, as face fitting could be achieved by RGB input and our eyelid fitting is based on RGB information, our method could also be ported to RGB-based systems.

6 CONCLUSION

We propose a real-time system that achieves 3D shape and motion reconstruction and animation of eyelids. Combined with a face and eyeball tracking technique, our system generates full face results in real time from a single view RGBD input, where the more vivid eye regions largely improve the realism of the reconstructed face. Technically, the proposed generative eyelid model represents the complex eyelid shape and motion variations by two linear models in low dimensional space, which can not only be used for fast eyelid fitting tasks but also for more applications in eyelid modeling and animation. The proposed DNN model is modified from a holistically-nested network, but it jointly detects and identifies multiple semantic edges by learning their structural correlations from the training data. Finally, the projective edge fitting method achieves real-time eyelid reconstruction by estimating the parameters of our eyelid models from 2D edges. To achieve this goal, our method extracts semantic points information from edge maps by a polynomial curve fitting technique and handles partial edge detection by a novel correspondence updating scheme.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions, and thank James Tompkin and Kyle Olszewski for proofreading this paper.

REFERENCES

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. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 75.

Pascal Bérard, Derek Bradley, Markus Gross, and Thabo Beeler. 2016. Lightweight eye capture using a parametric model. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 117.

Pascal Bérard, Derek Bradley, Maurizio Nitti, Thabo Beeler, and Markus H Gross. 2014. High-quality capture of eyes. *ACM Trans. Graph.* 33, 6 (2014), 223–1.

Amit Bermano, Thabo Beeler, Yeara Kozlov, Derek Bradley, Bernd Bickel, and Markus Gross. 2015. Detailed spatio-temporal reconstruction of eyelids. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 44.

Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 187–194.

James Booth, Anastasios Roussos, Stefanos Zafeiriou, Allan Ponniah, and David Dunaway. 2016. A 3D morphable model learnt from 10,000 faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5543–5552.

Sofien Bouaziz, Yangang Wang, and Mark Pauly. 2013. Online modeling for realtime facial animation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 40.

Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. 2015. Real-time high-fidelity facial performance capture. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 46.

Chen Cao, Qiming Hou, and Kun Zhou. 2014a. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 43.

Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 2013. 3D Shape Regression for Real-time Facial Animation. *ACM Trans. Graph.* 32, 4, Article 41 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2462012

Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. 2014b. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (2014), 413–425.

Pif Edwards, Chris Landreth, Eugene Fiume, and Karan Singh. 2016. JALI: an animator-centric viseme model for expressive lip synchronization. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 127.

Pablo Garrido, Levi Valgaerts, Chenglei Wu, and Christian Theobalt. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph.* 32, 6 (2013), 158–1.

Pablo Garrido, Michael Zollhöfer, Chenglei Wu, Derek Bradley, Patrick Pérez, Thabo Beeler, and Christian Theobalt. 2016. Corrective 3D reconstruction of lips from monocular video. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 219.

Pei-Lun Hsieh, Chongyang Ma, Jihun Yu, and Hao Li. 2015. Unconstrained Realtime Facial Performance Capture. In *Computer Vision and Pattern Recognition (CVPR)*.

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 Transactions on Graphics (Proceedings SIGGRAPH 2015)* 34, 4 (July 2015).

Hao Li, Thibaut Weise, and Mark Pauly. 2010. Example-based facial rigging. In *Acm transactions on graphics (tog)*, Vol. 29. ACM, 32.

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

Yilong Liu, Feng Xu, Jinxiang Chai, Xin Tong, Lijuan Wang, and Qiang Huo. 2015. Video-audio Driven Real-time Facial Animation. *ACM Trans. Graph.* 34, 6, Article 182 (Oct. 2015), 10 pages. https://doi.org/10.1145/2816795.2818122

Kenneth Alberto Funes Mora, Florent Monay, and Jean-Marc Odobez. 2014. Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 255–258.

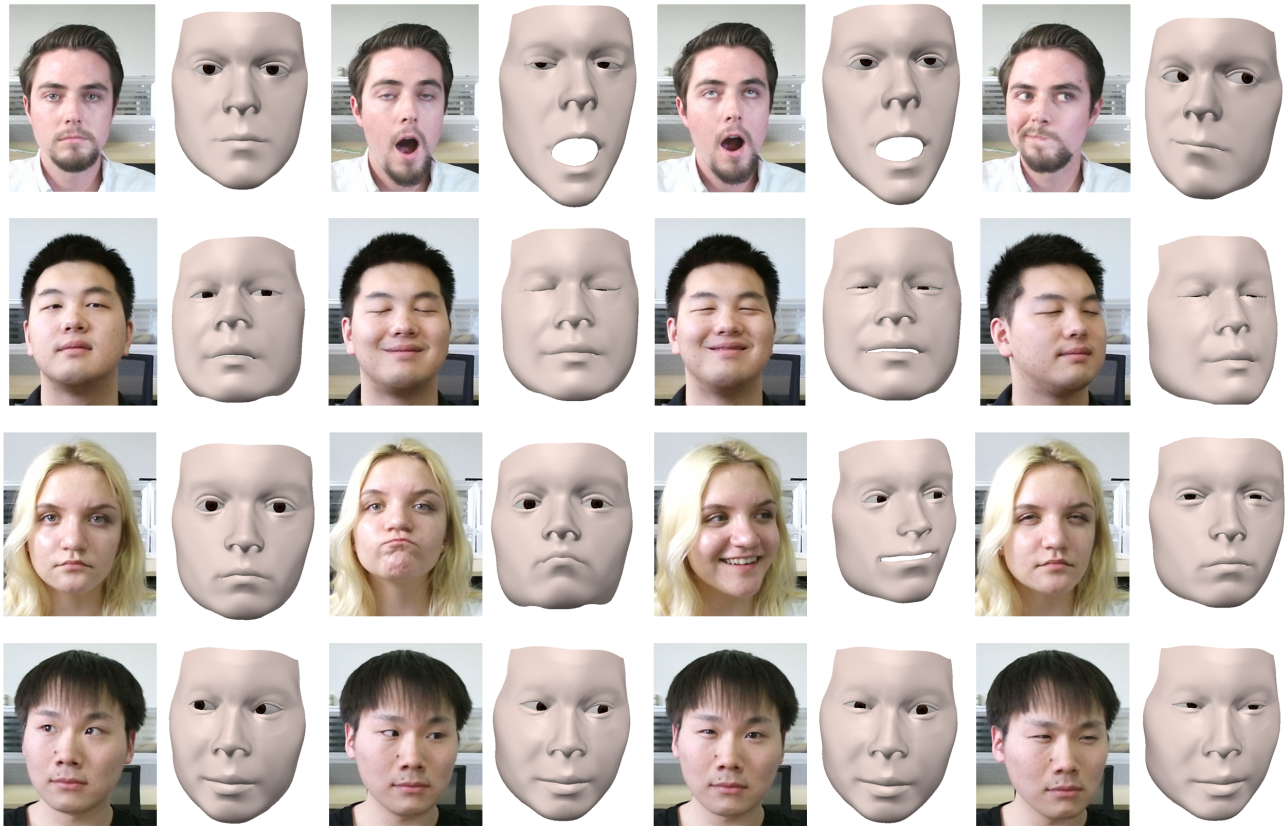

Fig. 13. Results on motion sequences. Each row shows some selected poses in one sequence of a user.

Debanga R Neog, João L Cardoso, Anurag Ranjan, and Dinesh K Pai. 2016. Interactive gaze driven animation of the eye region. In *Proceedings of the 21st International Conference on Web3D Technology*. ACM, 51–59.

Kyle Olszewski, Joseph J. Lim, Shunsuke Saito, and Hao Li. 2016. High-Fidelity Facial and Speech Animation for VR HMDs. *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia 2016)* 35, 6 (December 2016).

Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. 2009. A 3D face model for pose and illumination invariant face recognition. In *Advanced video and signal based surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*. IEEE, 296–301.

Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. 2016. Learning Detailed Face Reconstruction from a Single Image. *arXiv preprint arXiv:1611.05053* (2016).

Kerstin Ruhland, Sean Andrist, Jeremy Badler, Christopher Peters, Norman Badler, Michael Gleicher, Bilge Mutlu, and Rachel Mcdonnell. 2014. 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*. 69–91.

Shunsuke Saito, Tianye Li, and Hao Li. 2016. Real-Time Facial Segmentation and Performance Capture from RGB Input. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Fuhao Shi, Hsiang-Tao Wu, Xin Tong, and Jinxiang Chai. 2014. Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 222.

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014).

Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, 175–184.

Robert W Sumner and Jovan Popović. 2004. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 399–405.

J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt. 2015. Real-time Expression Transfer for Facial Reenactment. *ACM Transactions on Graphics (TOG)* 34, 6 (2015).

Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016a. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2387–2395.

Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016b. FaceVR: Real-Time Facial Reenactment and Eye Gaze Control in Virtual Reality. *arXiv preprint arXiv:1610.03151* (2016).

Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. 2005. Face transfer with multilinear models. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 426–433.

Congyi Wang, Fuhao Shi, Shihong Xia, and Jinxiang Chai. 2016. Realtime 3d eye gaze animation using a single rgb camera. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 118.

Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime Performance-Based Facial Animation. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2011)* 30, 4 (July 2011).

Quan Wen, Feng Xu, and Jun-Hai Yong. 2016. Real-time 3D Eye Performance Reconstruction for RGBD Cameras. *IEEE Transactions on Visualization and Computer Graphics* (2016).

Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016a. A 3D morphable eye region model for gaze estimation. In *European Conference on Computer Vision*. Springer, 297–313.

Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016b. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. ACM, 131–138.

Erroll Wood, Tadas Baltrusaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2017. GazeDirector: Fully Articulated Eye Gaze Redirection in Video. *arXiv preprint arXiv:1704.08763* (2017).

Chenglei Wu, Derek Bradley, Pablo Garrido, Michael Zollhöfer, Christian Theobalt, Markus Gross, and Thabo Beeler. 2016. Model-based teeth reconstruction. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 220.

Saining Xie and Zhuowen Tu. 2015. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 1395–1403.

Xiangyu Zhu, Zhen Lei, Junjie Yan, Dong Yi, and Stan Z Li. 2015. High-fidelity pose and expression normalization for face recognition in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 787–796.

A EXPLANATION OF OUR SHAPE BASES

b_0^{id} is a pair of neutral eyelids. b_1^{id} controls the vertical location of the pair and b_2^{id} controls the distance between the two eyelids. $b_3^{id}\&b_4^{id}$ make the left and right eye round. $b_5^{id}\&b_6^{id}$ make the eyes flat for upper eyelids while $b_7^{id}\&b_8^{id}$ make them flat for lower eyelids. $b_9^{id}\&b_{10}^{id}$ generate upturned eyes, while $b_{11}^{id}\&b_{12}^{id}$ generate downturned eyes. $b_{13}^{id}\&b_{14}^{id}$ decide the horizontal length of the eyes. The following eight dimensions are for the characteristic of double-fold eyelids, in which b_{15}^{id} to b_{20}^{id} make the inner part, outer part and middle part of the folds higher and $b_{21}^{id}\&b_{22}^{id}$ have no fold on either of the two eyes. With these bases, we are able to generate various shape and strength of the double-folds. Similar to the double-folds, the bulges are represented by $b_{23}^{id}\&b_{24}^{id}$ generating bulges parallel to lower eye eyelids, $b_{25}^{id}\&b_{26}^{id}$ making the middle part of bulges even lowers and $b_{27}^{id}\&b_{28}^{id}$ making the outer part lower. As there is no bulges whose inner part is lower than the outer part, the aforementioned six bases are enough to represent bulges.

B EXPLANATION OF OUR POSE BASES

For eyelid poses, b_0^{exp} is the same as b_0^{id}. $b_1^{exp}\&b_2^{exp}$ controls the closing eye motion. $b_3^{exp}\&b_4^{exp}$ and $b_5^{exp}\&b_6^{exp}$ controls the downward motion at the inner and outer eye corner of the upper eyelids, while $b_7^{exp}\&_8^{exp}$ and $b_9^{exp}\&_1^{exp}0$ controls the upward motion of these parts of the lower eyelids. Thus b_1^{exp} to b_{10}^{exp} are able to fully define the eye contour for closed eyes or extremely opened eyes. Besides the eye contour, as mentioned before, the double-folds and bulges may also change with motions. Thus b_{15}^{id} to b_{28}^{id}, which control the folds and bulges in the shape rig, are again used for pose rig, as b_{11}^{exp} and b_{24}^{exp}.