Real-time Expression Transfer for Facial Reenactment

Justus Thies[1] Michael Zollhöfer[2] Matthias Nießner[3] Levi Valgaerts[2] Marc Stamminger[1] Christian Theobalt[2]
[1]University of Erlangen-Nuremberg [2]Max-Planck-Institute for Informatics [3]Stanford University

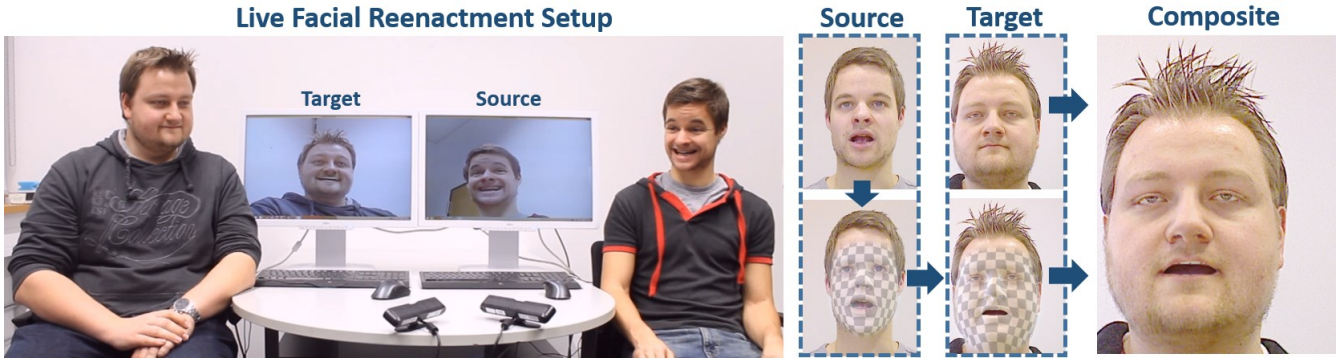

Figure 1: *Our live facial reenactment technique tracks the expression of a source actor and transfers it to a target actor at real-time rates. The synthetic result is photo-realisticly re-rendered on top of the original input stream maintaining the target's identity, pose and illumination.*

Abstract

We present a method for the real-time transfer of facial expressions from an actor in a source video to an actor in a target video, thus enabling the ad-hoc control of the facial expressions of the target actor. The novelty of our approach lies in the transfer and photo-realistic re-rendering of facial deformations and detail into the target video in a way that the newly-synthesized expressions are virtually indistinguishable from a real video. To achieve this, we accurately capture the facial performances of the source and target subjects in real-time using a commodity RGB-D sensor. For each frame, we jointly fit a parametric model for identity, expression, and skin reflectance to the input color and depth data, and also reconstruct the scene lighting. For expression transfer, we compute the difference between the source and target expressions in parameter space, and modify the target parameters to match the source expressions. A major challenge is the convincing re-rendering of the synthesized target face into the corresponding video stream. This requires a careful consideration of the lighting and shading design, which both must correspond to the real-world environment. We demonstrate our method in a live setup, where we modify a video conference feed such that the facial expressions of a different person (e.g., translator) are matched in real-time.

CR Categories: I.3.7 [Computer Graphics]: Digitization and Image Capture—Applications I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Range Data

Keywords: faces, real-time, depth camera, expression transfer

1 Introduction

In recent years, several approaches have been proposed for facial expression re-targeting, aimed at transferring facial expressions captured from a real subject to a virtual CG avatar [Weise et al. 2011; Li et al. 2013; Cao et al. 2014a]. Facial *reenactment* goes one step further by transferring the captured source expressions to a different, real actor, such that the new video shows the target actor reenacting the source expressions photo-realistically. Reenactment is a far more challenging task than expression re-targeting as even the slightest errors in transferred expressions and appearance and slight inconsistencies with the surrounding video will be noticed by a human user. Most methods for facial reenactment proposed so far work offline and only few of those produce results that are close to photo-realistic [Dale et al. 2011; Garrido et al. 2014].

In this paper, we propose an end-to-end approach for real-time facial reenactment at previously unseen visual realism. We believe that in particular the real-time capability paves the way for a variety of new applications that were previously impossible. Imagine a multilingual video-conferencing setup in which the video of one participant could be altered in real time to photo-realistically reenact the facial expression and mouth motion of a real-time translator. Or imagine another setting in which you could reenact a professionally captured video of somebody in business attire with a new real-time face capture of yourself sitting in casual clothing on your sofa. Application scenarios reach even further as photo-realistic reenactment enables the real-time manipulation of facial expression and motion in videos while making it challenging to detect that the video input is spoofed.

In order to achieve this goal, we need to solve a variety of challenging algorithmic problems under real-time constraints. We start by capturing the identity of the actor in terms of geometry and skin albedo maps; i.e., we obtain a personalized model of the actor. We then capture facial expressions of a *source actor* and a *target actor* using a commodity RGB-D camera (Asus Xtion Pro) for each subject. The ultimate goal is to map expressions from the source to the target actor, in real time, and in a photo-realistic fashion. Note that our focus is on the modification of the target face; however, we want to keep non-face regions in the target video unchanged.

Real-time Face Tracking and Reconstruction Our first contribution is a new real-time algorithm to reconstruct high-quality facial performance of each actor in real time from an RGB-D stream captured in a general environment with largely Lambertian surfaces and smoothly varying illumination. Our method uses a parametric face model that spans a PCA space of facial identities, face poses, and corresponding skin albedo. This model, which is learned from real face scans, serves us as a statistical prior and an intermediate representation to later enable photo-realistic re-rendering of the entire face. At runtime, we fit this representation to the RGB-D video in real time using a new analysis-through-synthesis approach, thus minimizing the difference between model and RGB-D video. To this end, we introduce a new objective function which is jointly optimized in the unknown head pose, face identity parameters, facial expression parameters, and face albedo values, as well as the incident illumination in the scene. Our energy function comprises several data terms that measure the alignment of the model to captured depth, the alignment to sparsely-tracked face features, as well as the similarity of rendered and captured surface appearance under the estimated lighting. Note that we fundamentally differ from other RGB and RGB-D tracking techniques [Weise et al. 2011; Li et al. 2013; Cao et al. 2014a], as we aim to manipulate real-world video (rather than virtual avatars) and as we optimize for (dense) photo-consistency between the RGB video and the synthesized output stream. In order to enable the minimization of our objective in real time, we tailor a new GPU-based data-parallel Gauss-Newton optimizer. The challenge in our setup is the efficient data-parallel optimization of a non-linear energy with a highly-dense Jacobian. To this end, we reformulate the optimization by Zollhöfer et al. [2014] in order to minimize the amount of global memory access required to apply the Jacobian matrix.

In practice, our system has two distinct stages. Immediately after recording commences, identity, head pose, initial coarse skin albedo, and incident illumination are jointly estimated in an interactive calibration stage that is only a few seconds long. Once our system is initialized, a personalized identity and fine-grained albedo map is available. In the second stage, we fix the identity and albedo, and continuously estimate the head pose, facial expression, and incident lighting for all subsequent frames at real-time rates.

Expression Transfer and Photo-realistic Re-rendering Our second contribution is a new technique to map facial expressions from source to target actors, and a method to photo-realistically render the modified target. The core idea behind the facial expression transfer is an efficient mapping between pose spaces under the consideration of transfer biases due to person-specific idiosyncrasies. For the final visualization of the target, we require face rendering to be photo-realistic under the estimated target illumination, and we need to seamlessly overlay face regions of the original target video with the synthesized face. To this end, we use a data-parallel blending strategy based on Laplacian pyramids. In addition, we propose an efficient way to synthesize the appearance of the mouth cavity and teeth in real time. To achieve this, we augment the face with a parametric teeth model and a cavity texture which is deformed along with the underlying shape template.

In our results, we demonstrate our reenactment approach in a live setup, where facial expressions are transferred from a source to a target actor in real time, with each subject captured by a separate RGB-D sensor (see Fig. 1). We show a variety of sequences with different subjects, challenging head motions, and expressions that are realistically reenacted on target facial performances in real time. In addition, we provide a quantitative evaluation of our face tracking method, showing how we achieve photo-realism by using dense RGB-D tracking to fit the shape identity (in contrast to sparse RGB feature tracking). Beyond facial reenactment, we also demonstrate the benefits of photo-realistic face capture and re-rendering, as we can easily modify facial appearances in real-time. For instance, we show how one would look like under different lighting, with different face albedo to simulate make-up, or after simply transferring facial characteristics from another person (e.g., growing a beard).

2 Related Work

2.1 Facial Performance Capture

Traditional facial performance capture for film and game productions achieves high-quality results using controlled studio conditions [Borshukov et al. 2003; Pighin and Lewis 2006]. A typical strategy to obtain robust features is the use of invisible makeup [Williams 1990] or facial makers [Guenter et al. 1998; Bickel et al. 2007; Huang et al. 2011]. Another option is to capture high-quality multi-view data from calibrated camera arrays [Bradley et al. 2010; Beeler et al. 2011; Valgaerts et al. 2012; Fyffe et al. 2014]. Dynamic active 3D scanners, for instance based on structured light projectors, also provide high-quality data which has been used to capture facial performances [Zhang et al. 2004; Wang et al. 2004; Weise et al. 2009]. Under controlled lighting conditions and the consideration of photometric cues, it is even possible to reconstruct fine-scale detail at the level of skin pores [Alexander et al. 2009; Wilson et al. 2010].

Monocular fitting to RGB-D data from a depth camera by non-rigid mesh deformation was shown in [Chen et al. 2013], but neither photo-realistic nor extremely detailed reconstruction is feasible. Recently, monocular off-line methods were proposed that fit a parametric blend shape [Garrido et al. 2013] or multi-linear face model [Shi et al. 2014] to RGB video; both approaches extract fine-scale detail via lighting and albedo estimation from video, followed by shading-based shape refinement.

While these methods provide impressive results, they are unsuited for consumer-level applications, such as facial reenactment in video telephony, which is the main motivating scenario of our work.

2.2 Face Re-targeting and Facial Animation

Many lightweight face tracking methods obtain 2D landmarks from RGB video and fit a parametric face model to match the tracked positions. A prominent example is active appearance models (AAM) [Cootes et al. 2001] which are used to determine the parameters of a 3D PCA model while only using 2D features [Xiao et al. 2004]. Another popular representation is the blend shape model [Pighin et al. 1998; Lewis and Anjyo 2010] which embeds pose variation in a low-dimensional PCA space; blend shapes can be constrained by image feature points [Chuang and Bregler 2002; Chai et al. 2003]. The key advantage of these approaches is that they work on unconstrained RGB input. Unfortunately, retrieving accurate shape identities is either challenging or computationally expensive. An alternative research direction is based on regressing parameters of statistical facial models, enabling face tracking using only RGB [Cao et al. 2013; Cao et al. 2014a] input. As these methods run at high real-time rates, even on mobile hardware, they focus on animating virtual avatars rather than photo-realistic rendering or detailed shape acquisition.

Fitting face templates directly to multi-view or dense RGB-D input enables facial reconstructions to reflect more skin detail [Valgaerts et al. 2012; Suwajanakorn et al. 2014]; however, these methods are relatively slow and limited to offline applications. Real-time performance on dense RGB-D input has recently been achieved by tracking a personalized blend shape model [Weise et al. 2011; Li et al. 2013; Bouaziz et al. 2013; Hsieh et al. 2015], or by the deformation of a face template mesh in an as-rigid-as-possible framework [Zollhöfer et al. 2014]. The results of these methods are quite impressive, as

they typically have ways to augment the low-dimensional face template with fine-scale detail; however, they only show re-targeting results for hand-modeled or cartoon-like characters. In this paper, we focus on the photo-realistic capture and re-rendering of facial templates, as our goal is the expression transfer between real actors. The main difference in our tracking pipeline is a new analysis-through-synthesis approach whose objective is the minimization of the photometric re-rendering error.

2.3 Face Replacement in Video

One type of face replacement techniques uses a morphable 3D model as an underlying face representation that parameterizes identity, facial expressions, and other properties, such as visemes or face texture [Blanz and Vetter 1999; Blanz et al. 2003; Blanz et al. 2004; Vlasic et al. 2005]. These systems can produce accurate 3D textured meshes and can establish a one-to-one expression mapping between source and target actor, thereby simplifying and speeding up expression transfer. Morphable models are either generated by learning a detailed 3D multi-linear model from example data spanning a large variety of identities and expressions [Vlasic et al. 2005], or by purposely building a person-specific blend shape model from scans of an actor using specialized hardware [Eisert and Girod 1998; Alexander et al. 2009; Weise et al. 2011]. The morphable model based face replacement technique of Dale et al. [2011] could be used for similar purposes as ours to replace the face region of a target video with a new performance. However, their approach is neither automatic, nor real-time, and only works if the source and target actor are the same person, and have comparable head poses in the source and target recordings. Our method, on the other hand, is fully automatic and tracks, transfers and renders facial expressions in real-time between different individuals for a large variety of head poses and facial performances.

Another line of research for synthesizing novel facial expressions finds similarities in head pose and facial expression between two videos solely based on image information. These image-based methods track the face using optical flow [Li et al. 2012] or a sparse set of 2D facial features [Saragih et al. 2011b], and often include an image matching step to look up similar expressions in a database of facial images [Kemelmacher-Shlizerman et al. 2010], or a short sequence of arbitrary source performances [Garrido et al. 2014]. Many image-based face replacement systems do not allow much head motion and are limited in their ability to rendering facial dynamics, especially of the mouth region. Moreover, most approaches cannot handle lighting changes, such that substantial differences in pose and appearance may produce unrealistic composites or blending artifacts. In this paper, we demonstrate stable tracking and face replacement results for substantial head motion and because we model environment lighting explicitly we also succeed under changing illumination. If the task is to create a new facial animation, additional temporal coherence constraints must be embedded in the objective to minimize possible in-between jumps along the sequence [Kemelmacher-Shlizerman et al. 2011]. Expression mapping [Liu et al. 2001] transfers a target expression to a neutral source face, but does not preserve the target head motion and illumination, and has problems inside the mouth region, where teeth are not visible. In this paper, we generate a convincingly rendered inner mouth region by using a textured 3D tooth proxy that is rigged to the tracked blend shape model and warping an image of the mouth cavity according to tracked mouth features.

Our approach is related to the recent virtual dubbing method by Garrido et al. [2015] who re-render the face of an actor in video such that it matches a new audio track. The method uses a combination of model-based monocular tracking, inverse rendering for reflectance, lighting and detail estimation, and audio-visual expression mapping between a target and a dubbing actor. This yields highly realistic results, but processing times are far from real-time.

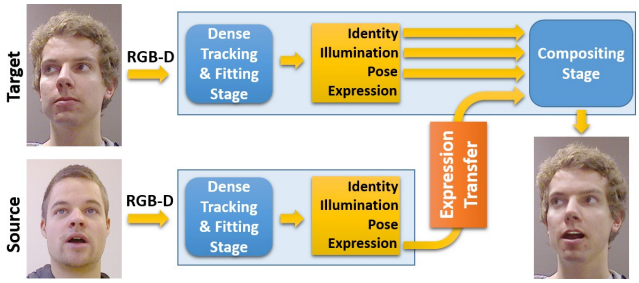

Figure 2: *Our live facial reenactment pipeline.*

3 Overview

The key idea of our approach is to use a linear parametric model for facial identity, expression, and albedo as an intermediate representation for tracking, transferring and photo-realistically re-rendering facial expressions in a live video sequence.

Tracking We use a commodity RGB-D sensor to estimate the parameters of the statistical face model, the head pose, and the unknown incident illumination in the scene from the input depth and video data. Our face model is custom built by combining a morphable model for identity and skin albedo [Blanz and Vetter 1999] with the expression space of a blend shape model [Alexander et al. 2009; Cao et al. 2014b] (see Sec. 4). The face model is linear in these three attributes, with a separate set of parameters encoding identity, expression, and reflectance. In addition to this parametric prior, we use a lighting model with a Lambertian surface reflectance assumption to jointly estimate the environment lighting. This is necessary for robustly matching the face model to the video stream and for the convincing rendering of the final composite. We determine the model and lighting parameters by minimizing a non-linear least squares energy that measures the discrepancy between the RGB-D input data and the estimated face shape, pose, and albedo (see Sec. 5). We solve for all unknowns simultaneously using a data parallel Gauss-Newton solver which is implemented on the GPU for real-time performance and specifically designed for our objective energy (see Sec. 6). The tracking stage is summarized in Fig. 3

Reenactment Once we have estimated the model parameters and the head pose, we can re-render the face back into the underlying input video stream (see Sec. 5.2) in photo-realistic quality. By modifying the different model parameters on-the-fly, a variety of video modification applications become feasible, such as re-lighting the captured subject as if he would appear in a different environment and augmenting the face reflectance with virtual textures or make-up (see Sec. 7.5). Yet, the key application of our approach is the transfer of expressions from one actor to another without changing other parameters. To this end, we simultaneously capture the performance of a source and target actor and map the corresponding expression parameters from the source to the target (see Sec. 7). While the identity of the target actor is preserved, we can composite the synthesized image on top of the target video stream. An illustration of this pipeline based on our real-time tracking and fitting stage is shown in Fig. 2.

4 Synthesis of Facial Imagery

To synthesize and render new human facial imagery, we use a parametric 3D face model as an intermediary representation of facial

identity, expression, and reflectance. This model also acts as a prior for facial performance capture, rendering it more robust with respect to noisy and incomplete data. In addition, we model the environment lighting to estimate the illumination conditions in the video. Both of these models together allow for a photo-realistic re-rendering of a person's face with different expressions under general unknown illumination.

4.1 Parametric Face Model

As a face prior, we use a linear parametric face model $\mathcal{M}_{\text{geo}}(\boldsymbol{\alpha}, \boldsymbol{\delta})$ which embeds the vertices $\boldsymbol{v}_i \in \mathbb{R}^3, i \in \{1, \dots, n\}$ of a generic face template mesh in a lower-dimensional subspace. The template is a manifold mesh defined by the set of vertex positions $V = [\boldsymbol{v}_i]$ and corresponding vertex normals $N = [\boldsymbol{n}_i]$, with $|V| = |N| = n$. The model $\mathcal{M}_{\text{geo}}(\boldsymbol{\alpha}, \boldsymbol{\delta})$ parameterizes the face geometry by means of a set of dimensions encoding the identity with weights $\boldsymbol{\alpha}$ and a set of dimensions encoding the facial expression with weights $\boldsymbol{\delta}$. In addition to the geometric prior, we also use a prior for the skin albedo $\mathcal{M}_{\text{alb}}(\boldsymbol{\beta})$, which reduces the set of vertex albedos of the template mesh $C = [\boldsymbol{c}_i]$, with $\boldsymbol{c}_i \in \mathbb{R}^3$ and $|C| = n$, to a linear subspace with weights $\boldsymbol{\beta}$. More specifically, our parametric face model is defined by the following linear combinations

$$\mathcal{M}_{\text{geo}}(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \boldsymbol{a}_{\text{id}} + E_{\text{id}}\,\boldsymbol{\alpha} + E_{\text{exp}}\,\boldsymbol{\delta} \;, \tag{1}$$

$$\mathcal{M}_{\text{alb}}(\boldsymbol{\beta}) = \boldsymbol{a}_{\text{alb}} + E_{\text{alb}}\,\boldsymbol{\beta} \;. \tag{2}$$

Here, $\mathcal{M}_{\text{geo}} \in \mathbb{R}^{3n}$ and $\mathcal{M}_{\text{alb}} \in \mathbb{R}^{3n}$ contain the n vertex positions and vertex albedos, respectively, while the columns of the matrices E_{id}, E_{exp}, and E_{alb} contain the basis vectors of the linear subspaces. The vectors $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$ and $\boldsymbol{\beta}$ control the identity, the expression and the skin albedo of the resulting face, and $\boldsymbol{a}_{\text{id}}$ and $\boldsymbol{a}_{\text{alb}}$ represent the mean identity shape in rest and the mean skin albedo. While $\boldsymbol{v}_i$ and $\boldsymbol{c}_i$ are defined by a linear combination of basis vectors, the normals $\boldsymbol{n}_i$ can be derived as the cross product of the partial derivatives of the shape with respect to a (u, v)-parameterization.

Our face model is built once in a pre-computation step. For the identity and albedo dimensions, we make use of the morphable model of Blanz and Vetter [1999]. This model has been generated by non-rigidly deforming a face template to 200 high-quality scans of different subjects using optical flow and a cylindrical parameterization. We assume that the distribution of scanned faces is Gaussian, with a mean shape $\boldsymbol{a}_{\text{id}}$, a mean albedo $\boldsymbol{a}_{\text{alb}}$, and standard deviations $\boldsymbol{\sigma}_{\text{id}}$ and $\boldsymbol{\sigma}_{\text{alb}}$. We use the first 160 principal directions to span the space of plausible facial shapes with respect to the geometric embedding and skin reflectance. Facial expressions are added to the identity model by transferring the displacement fields of two existing blend shape rigs by means of deformation transfer [Sumner and Popović 2004]. The used blend shapes have been created manually [Alexander et al. 2009] [1] or by non-rigid registration to captured scans [Cao et al. 2014b] [2]. We parameterize the space of plausible expressions by 76 blendshapes, which turned out to be a good trade-off between computational complexity and expressibility. Note that the identity is parameterized in PCA space with linearly independent components, while the expressions are represented by blend shapes that may be overcomplete.

4.2 Illumination Model

To model the illumination, we assume that the lighting is distant and that the surfaces in the scene are predominantly Lambertian. This suggests the use of a Sherical Harmonics (SH) basis [Müller 1966] for a low dimensional representation of the incident illumination. Following Ramamoorthi and Hanrahan [2001], the *irradiance* in a vertex with normal $\boldsymbol{n}$ and scalar albedo c is represented using $b = 3$ bands of SHs for the incident illumination:

$$\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{n}, c) = c \cdot \sum_{k=1}^{b^2} \gamma_k \, y_k(\boldsymbol{n}) \;, \tag{3}$$

with y_k being the k-th SH basis function and $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_{b^2})$ the SH coefficients. Since we only assume distant light sources and ignore self-shadowing or indirect lighting, the irradiance is independent of the vertex position and only depends on the vertex normal and albedo. In our application, we consider the three RGB channels separately, thus irradiance and albedo are RGB triples. The above equation then gives rise to 27 SH coefficients ($b^2 = 9$ basis functions per channel).

4.3 Image Formation Model

In addition to the face and illumination models, we need a representation for the head pose and the camera projection onto the virtual image plane. To this end, we anchor the origin and the axis of the world coordinate frame to the RGB-D sensor and assume the camera to be calibrated. The model-to-world transformation for the face is then given by $\Phi(\boldsymbol{v}) = \boldsymbol{R}\,\boldsymbol{v} + \boldsymbol{t}$, where $\boldsymbol{R}$ is a 3×3 rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ a translation vector. $\boldsymbol{R}$ is parameterized using Euler angles and, together with $\boldsymbol{t}$, represents the 6-DOF rigid transformation that maps the vertices of the face between the local coordinates of our parametric model and the world coordinates. The known intrinsic camera parameters define a full perspective projection Π that transforms the world coordinates to image coordinates. With this, we can define an image formation model $\mathcal{S}(\boldsymbol{\mathcal{P}})$, which allows us to generate synthetic views of virtual faces, given the parameters $\boldsymbol{\mathcal{P}}$ that govern the structure of the complete scene:

$$\boldsymbol{\mathcal{P}} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\gamma}, \boldsymbol{R}, \boldsymbol{t}) \;, \tag{4}$$

with $p = 160 + 160 + 76 + 27 + 3 + 3 = 429$ being the total amount of parameters. The image formation model enables the transfer of facial expressions between different persons, environments and viewpoints, but in order to manipulate a given video stream of a face, we first need to determine the parameters $\boldsymbol{\mathcal{P}}$ that faithfully reproduce the observed face in each RGB-D input frame. In the next section, we will describe how we can optimize for $\boldsymbol{\mathcal{P}}$ in real-time. The use of the estimated parameters for video manipulation will be described in Sec. 7.

5 Parametric Model Fitting

For the simultaneous estimation of the identity, facial expression, skin albedo, scene lighting, and head pose, we fit our image formation model $\mathcal{S}(\boldsymbol{\mathcal{P}})$ to the input of a commodity RGB-D camera recording an actor's performance. Our goal is to obtain the best fitting parameters $\boldsymbol{\mathcal{P}}$ that explain the input in real-time. We will do this using an *analysis-through-synthesis* approach, where we render the image formation model for the old set of (potentially non-optimal) parameters and optimize $\boldsymbol{\mathcal{P}}$ further by comparing the rendered image to the captured RGB-D input. This is a hard inverse rendering problem in the unknowns $\boldsymbol{\mathcal{P}}$ and in this section we will describe how to cast and solve it as a non-linear least squares problem. An overview of our fitting pipeline is shown in Fig. 3.

5.1 Input Data

The input for our facial performance capture system is provided by an RGB-D camera and consists of the measured input color sequence $C_{\mathcal{I}}$ and depth sequence $X_{\mathcal{I}}$. We assume that the depth and color

[1] Faceware Technologies `www.facewaretech.com`

[2] Facewarehouse `http://gaps-zju.org/facewarehouse/`

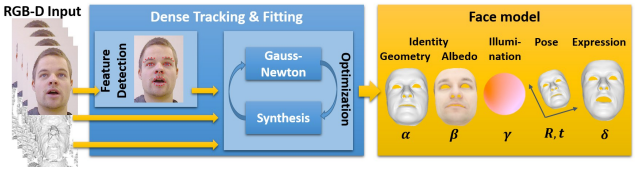

Figure 3: *Overview of our real-time fitting pipeline.*

data are aligned in image space and can be indexed by the same pixel coordinates; i.e., the color and back-projected 3D position in an integer pixel location $\boldsymbol{p} = (i, j)$ is given by $C_{\mathcal{I}}(\boldsymbol{p}) \in \mathbb{R}^3$ and $X_{\mathcal{I}}(\boldsymbol{p}) \in \mathbb{R}^3$, respectively. The range sensor implicitly provides us with a normal field $N_{\mathcal{I}}$, where $N_{\mathcal{I}}(\boldsymbol{p}) \in \mathbb{R}^3$ is obtained as the cross product of the partial derivatives of $X_{\mathcal{I}}$ with respect to the continuous image coordinates.

5.2 Implementation of the Image Formation Model

The image formation model $\mathcal{S}(\boldsymbol{\mathcal{P}})$, which generates a synthetic view of the virtual face, is implemented by means of the GPU rasterization pipeline. Apart from efficiency, this allows us to formulate the problem in terms of 2D image arrays, which is the native data structure for GPU programs. The rasterizer generates a fragment per pixel $\boldsymbol{p}$ if a triangle is visible at its location and barycentrically interpolates the vertex attributes of the underlying triangle. The output of the rasterizer is the synthetic color $C_{\mathcal{S}}$, the 3D position $X_{\mathcal{S}}$ and the normal $N_{\mathcal{S}}$ at each pixel $\boldsymbol{p}$. Note that $C_{\mathcal{S}}(\boldsymbol{p})$, $X_{\mathcal{S}}(\boldsymbol{p})$, and $N_{\mathcal{S}}(\boldsymbol{p})$ are functions of the unknown parameters $\boldsymbol{\mathcal{P}}$. The rasterizer also writes out the barycentric coordinates of the pixel and the indices of the vertices in the covering triangle, which is required to compute the analytical partial derivatives with respect to $\boldsymbol{\mathcal{P}}$.

From now on, we only consider pixels belonging to the set $\mathcal{V}$ of pixels for which both the input and the synthetic data is valid.

5.3 Energy Formulation

We cast the problem of finding the virtual scene that best explains the input RGB-D observations as an unconstrained energy minimization problem in the unknowns $\boldsymbol{\mathcal{P}}$. To this end, we formulate an energy that can be robustly and efficiently minimized:

$$E(\boldsymbol{\mathcal{P}}) = E_{\text{emb}}(\boldsymbol{\mathcal{P}}) + w_{\text{col}} E_{\text{col}}(\boldsymbol{\mathcal{P}}) + w_{\text{lan}} E_{\text{lan}}(\boldsymbol{\mathcal{P}}) + w_{\text{reg}} E_{\text{reg}}(\boldsymbol{\mathcal{P}}) \ . \quad (5)$$

The design of the objective takes the quality of the geometric embedding E_{emb}, the photo-consistency of the re-rendering E_{col}, the reproduction of a sparse set of facial feature points E_{lan}, and the geometric faithfulness of the synthesized virtual head E_{reg} into account. The weights w_{col}, w_{lan}, and w_{reg} compensate for different scaling of the objectives. They have been empirically determined and are fixed for all shown experiments. In the following, we detail on the different components of the objective function.

Geometry Consistency Metric The reconstructed geometry of the virtual face should match the observations captured by the input depth stream. To this end, we define a measure that quantifies the discrepancy between the rendered synthetic depth map and the input depth stream:

$$E_{\text{emb}}(\boldsymbol{\mathcal{P}}) = w_{\text{point}} E_{\text{point}}(\boldsymbol{\mathcal{P}}) + w_{\text{plane}} E_{\text{plane}}(\boldsymbol{\mathcal{P}}) \ . \quad (6)$$

The first term minimizes the sum of the projective Euclidean point-to-point distances for all pixels in the visible set: $\mathcal{V}$

$$E_{\text{point}}(\boldsymbol{\mathcal{P}}) = \sum_{\boldsymbol{p} \in \mathcal{V}} \| d_{\text{point}}(\boldsymbol{p}) \|_2^2 \ , \quad (7)$$

with $d_{\text{point}}(\boldsymbol{p}) = X_{\mathcal{S}}(\boldsymbol{p}) - X_{\mathcal{I}}(\boldsymbol{p})$ the difference between the measured 3D position and the 3D model point. To improve robustness and convergence, we also use a first-order approximation of the surface-to-surface distance [Chen and Medioni 1992]. This is particularly relevant for purely translational motion where a point-to-point metric alone would fail. To this end, we measure the symmetric point-to-plane distance from model to input and input to model at every visible pixel:

$$E_{\text{plane}}(\boldsymbol{\mathcal{P}}) = \sum_{\boldsymbol{p} \in \mathcal{V}} \left[d_{\text{plane}}^2 \left(N_{\mathcal{S}}(\boldsymbol{p}), \boldsymbol{p} \right) + d_{\text{plane}}^2 \left(N_{\mathcal{I}}(\boldsymbol{p}), \boldsymbol{p} \right) \right], \quad (8)$$

with $d_{\text{plane}}(\boldsymbol{n}, \boldsymbol{p}) = \boldsymbol{n}^T d_{\text{point}}(\boldsymbol{p})$ the distance between the 3D point $X_{\mathcal{S}}(\boldsymbol{p})$ or $X_{\mathcal{I}}(\boldsymbol{p})$ and the plane defined by the normal $\boldsymbol{n}$.

Color Consistency Metric In addition to our face model being metrically faithful, we require that the RGB images synthesized using our model are photo-consistent with the given input color images. Therefore, we minimize the difference between the input RGB image and the rendered view for every pixel $\boldsymbol{p} \in \mathcal{V}$:

$$E_{\text{col}}(\boldsymbol{\mathcal{P}}) = \sum_{\boldsymbol{p} \in \mathcal{V}} \| C_{\mathcal{S}}(\boldsymbol{p}) - C_{\mathcal{I}}(\boldsymbol{p}) \|_2^2 \ , \quad (9)$$

where $C_{\mathcal{S}}(\boldsymbol{p})$ is the illuminated (i.e., shaded) color of the synthesized model. The color consistency objective introduces a coupling between the geometry of our template model, the per vertex skin-reflectance map and the SH illumination coefficients. It is directly induced by the used illumination model $\mathcal{L}$.

Feature Similarity Metric The face contains many characteristic features, which can be tracked more reliably than other points. In addition to the dense color consistency metric, we therefore track a set of sparse facial landmarks in the RGB stream using a state-of-the-art facial feature tracker [Saragih et al. 2011a]. Each detected feature $\boldsymbol{f}_j = (u_j, v_j)$ is a 2D location in the image domain that corresponds to a consistent 3D vertex $\boldsymbol{v}_j$ in our geometric face model. If $\mathcal{F}$ is the set of detected features in each RGB input frame, we can define a metric that enforces facial features in the synthesized views to be close to the detected features:

$$E_{\text{lan}}(\boldsymbol{\mathcal{P}}) = \sum_{\boldsymbol{f}_j \in \mathcal{F}} w_{\text{conf},j} \left\| \boldsymbol{f}_j - \Pi(\Phi(\boldsymbol{v}_j)) \right\|_2^2 \ . \quad (10)$$

We use 38 manually selected landmark locations concentrated in the mouth, eye, and nose regions of the face. We prune features based on their visibility in the last frame and assign a confidence w_{conf} based on its trustworthiness [Saragih et al. 2011a]. This allows us to effectively prune wrongly classified features, which are common under large head rotations ($> 30^\circ$).

Regularization Constraints The final component of our objective is a statistical regularization term that expresses the likelihood of observing the reconstructed face, and keeps the estimated parameters within a plausible range. Under the assumption of Gaussian distributed parameters, the interval $[-3\sigma_{\bullet,i}, +3\sigma_{\bullet,i}]$ contains $\approx 99\%$ of the variation in human faces that can be reproduced by our model. To this end, we constrain the model parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\delta}$ to be statistically small compared to their standard deviation:

$$E_{\text{reg}}(\boldsymbol{\mathcal{P}}) = \sum_{i=1}^{160} \left[\left(\frac{\boldsymbol{\alpha}_i}{\sigma_{\text{id},i}} \right)^2 + \left(\frac{\boldsymbol{\beta}_i}{\sigma_{\text{alb},i}} \right)^2 \right] + \sum_{i=1}^{76} \left(\frac{\boldsymbol{\delta}_i}{\sigma_{\text{exp},i}} \right)^2 \ . \quad (11)$$

For the shape and reflectance parameters, $\sigma_{\text{id},i}$ and $\sigma_{\text{alb},i}$ are computed from the 200 high-quality scans (see Sec. 4.1). For the blend shape parameters, $\sigma_{\text{exp},i}$ is fixed to 1 in our experiments.

Analytical Partial Derivatives In order to minimize the proposed energy, we need to compute the analytical derivatives of the synthetic images with respect to the parameters $\boldsymbol{\mathcal{P}}$. This is non-trivial, since a derivation of the complete transformation chain in the image formation model is required. To this end, we also emit the barycentric coordinates during rasterizeration at every pixel in addition to the indices of the vertices of the underlying triangle. Differentiation of $\mathcal{S}(\boldsymbol{\mathcal{P}})$ starts with the evaluation of the face model ($\mathcal{M}_{geo}$ and $\mathcal{M}_{alb}$), the transformation to world space via Φ, the illumination of the model with the lighting model $\mathcal{L}$, and finally the projection to image space via Π. The high number of involved rendering stages leads to many applications of the chain rule and results in high computational costs.

6 Parallel Energy Minimization

The proposed energy $E(\boldsymbol{\mathcal{P}}) : \mathbb{R}^p \rightarrow \mathbb{R}$ of Eq. (5) is non-linear in the parameters $\boldsymbol{\mathcal{P}}$, and finding the best set of parameters $\boldsymbol{\mathcal{P}}^*$ amounts to solving a non-linear least squares problem in the p unknowns:

$$\boldsymbol{\mathcal{P}}^* = \underset{\boldsymbol{\mathcal{P}}}{\operatorname{argmin}}\, E(\boldsymbol{\mathcal{P}}) \quad . \tag{12}$$

Even at the moderate image resolutions used in this paper (640×480), our energy gives rise to a considerable amount of residuals: each visible pixel $\boldsymbol{p} \in \mathcal{V}$ contributes with 8 residuals (3 from the point-to-point term of Eq. (6), 2 from the point-to-plane term of Eq. (8) and 3 from the color term of Eq. (9)), while the feature term of Eq. (10) contributes with $2 \cdot 38$ residuals and the regularizer of Eq. (11) with $p - 33$ residuals. The total number of residuals is thus $m = 8|\mathcal{V}| + 76 + p - 33$, which can equal up to 180K equations for a close-up frame of the face. To minimize a non-linear objective with such a high number of residuals in real-time, we propose a data parallel GPU-based Gauss-Newton solver that leverages the high computational throughput of modern graphic cards and exploits smart caching to minimize the number of global memory accesses.

6.1 Core Solver

We minimize the non-linear least-squares energy $E(\boldsymbol{\mathcal{P}})$ in a Gauss-Newton framework by reformulating it in terms of its residual $r : \mathbb{R}^p \rightarrow \mathbb{R}^m$, with $r(\boldsymbol{\mathcal{P}}) = (r_1(\boldsymbol{\mathcal{P}}), \ldots, r_m(\boldsymbol{\mathcal{P}}))^T$. If we assume that we already have an approximate solution $\boldsymbol{\mathcal{P}}^k$, we seek for an parameter increment $\Delta\boldsymbol{\mathcal{P}}$ that minimizes the first-order Taylor expansion of $r(\boldsymbol{\mathcal{P}})$ around $\boldsymbol{\mathcal{P}}^k$. So we approximate

$$E(\boldsymbol{\mathcal{P}}^k + \Delta\boldsymbol{\mathcal{P}}) \approx \left\| r(\boldsymbol{\mathcal{P}}^k) + J(\boldsymbol{\mathcal{P}}^k)\Delta\boldsymbol{\mathcal{P}} \right\|_2^2, \tag{13}$$

for the update $\Delta\boldsymbol{\mathcal{P}}$, with $J(\boldsymbol{\mathcal{P}}^k)$ the $m \times p$ Jacobian of $r(\boldsymbol{\mathcal{P}}^k)$ in the current solution. The corresponding normal equations are

$$J^T(\boldsymbol{\mathcal{P}}^k)J(\boldsymbol{\mathcal{P}}^k)\Delta\boldsymbol{\mathcal{P}} = -J^T(\boldsymbol{\mathcal{P}}^k)r(\boldsymbol{\mathcal{P}}^k) \quad , \tag{14}$$

and the parameters are updated as $\boldsymbol{\mathcal{P}}^{k+1} = \boldsymbol{\mathcal{P}}^k + \Delta\boldsymbol{\mathcal{P}}$. We solve the normal equations iteratively using a preconditioned conjugate gradient (PCG) method, thus allowing for efficient parallelization on the GPU (in contrast to a direct solve). Moreover, the normal equations need not to be solved until convergence since the PCG step only appears as the inner loop (*analysis*) of a Gauss-Newton iteration. In the outer loop (*synthesis*), the face is re-rendered and the Jacobian is recomputed using the updated barycentric coordinates. We use Jacobi preconditioning, where the inverse of the diagonal elements of $J^T J$ are computed in the initialization stage of the PCG.

Close in spirit to [Zollhöfer et al. 2014], we speed up convergence by embedding the energy minimization in a multi-resolution coarse-to-fine framework. To this end, we successively blur and resample the input RGB-D sequence using a Gaussian pyramid with 3 levels and apply the image formation model on the same reduced resolutions. After finding the optimal set of parameters on the current resolution level, a prolongation step transfers the solution to the next finer level to be used as an initialization there.

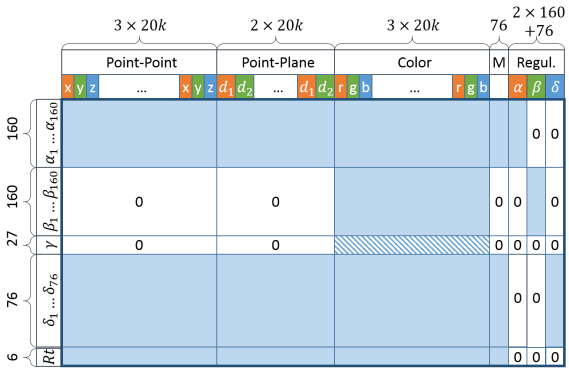

Figure 4: *Non-zero structure of J^T for 20k visible pixels.*

6.2 Memory Efficient Solution Strategy on the GPU

The normal equations (14) are solved using a novel data-parallel PCG solver that exploits smart caching to speed up the computation. The most expensive task in each PCG step is the multiplication of the system matrix $J^T J$ with the previous descent direction. Pre-computing $J^T J$ would take $\mathcal{O}(n^3)$ time in the number of Jacobian entries and would be too costly for real-time performance, so instead we apply J and J^T in succession. In previous work [Zollhöfer et al. 2014], the PCG solver is optimized for a sparse Jacobian and the entries of J are computed on-the-fly in each iteration. For our problem, on the other hand, J is block-dense because all parameters, except for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, influence each residual (see Fig. 4). In addition, we optimize for all unknowns simultaneously and our energy has a larger number of residuals compared to [Zollhöfer et al. 2014]. Hence, repeatedly recomputing the Jacobian would require significant read access from global memory, thus significantly affecting run time performance.

The key idea to adapting the parallel PCG solver to deal with a dense Jacobian is to write the derivatives of each residual in global memory, while pre-computing the right-hand side of the system. Since all derivatives have to be evaluated at least once in this step, this incurs no computational overhead. We write J, as well as J^T, to global memory to allow for coalesced memory access later on when multiplying the Jacobian and its transpose in succession. This strategy allows us to better leverage texture caches and burst load of data on modern GPUs. Once the derivatives have been stored in global memory, the cached data can be reused in each PCG iteration by a single read operation.

The convergence rate of our data-parallel Gauss-Newton solver for different types of facial performances is visualized in Fig. 5. These timings are obtained for an input frame rate of 30 fps with 7 Gauss-Newton outer iterations and 4 PCG inner iterations. Even for expressive motion, we converge well within a single time step.

6.3 Initialization of Identity and Albedo

As we assume that facial identity and reflectance for an individual remain constant during facial performance capture, we do not optimize for the corresponding parameters on-the-fly. Both are estimated in an initialization step by running our optimizer on a short control sequence of the actor turning his head under constant illumination.

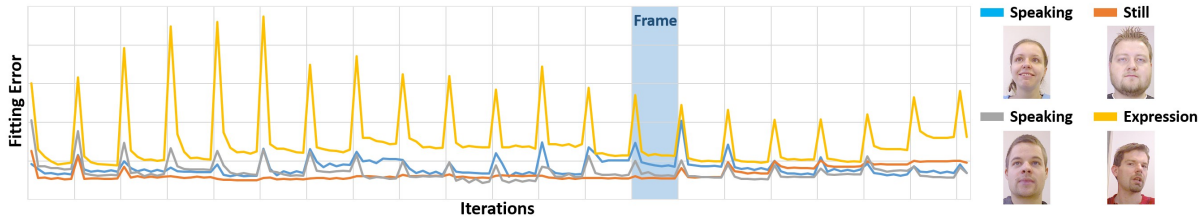

Figure 5: *Convergence of the Gauss-Newton solver for different facial performances. The horizontal axis breaks up convergence for each captured frame (at 30 fps); the vertical axis shows the fitting error. Even for expressive motion, we converge well within a single frame.*

In this step, all parameters are optimized and the estimated identity and reflectance are fixed for subsequent capture. The face does not need to be in rest for the initialization phase and convergence is usually achieved between 5 and 10 frames.

For the fixed reflectance, we do not use the values given by the linear face model, but compute a more accurate skin albedo by building a skin texture for the face and dividing it by the estimated lighting to correct for the shading effects. The resolution of this texture is much higher than the vertex density for improved detail (2048×2048 in our experiments) and is generated by combining three camera views (front, 20° left and 20° right) using pyramid blending [Adelson et al. 1984]. The final high-resolution albedo map is used for rendering.

7 Facial Reenactment and Applications

The real-time capture of identity, reflectance, facial expression, and scene lighting, opens the door for a variety of new applications. In particular, it enables on-the-fly control of an actor in a target video by transferring the facial expressions from a source actor, while preserving the target identity, head pose, and scene lighting. Such face reenactment can, for instance, be used for video-conferencing, where the facial expression and mouth motion of a participant are altered photo-realistically and instantly by a real-time translator or puppeteer behind the scenes. In this section, we will simulate such a scenario and describe the hardware setup and algorithmic components. We will also touch on two special cases of this setup, namely face re-texturing and re-lighting in a virtual mirror application.

7.1 Live Reenactment Setup

To perform live face reenactment, we built a setup consisting of two RGB-D cameras, each connected to a computer with a modern graphics card (see Fig. 1). After estimating the identity, reflectance, and lighting in a calibration step (see Sec. 6.3), the facial performance of the source and target actor is captured on separate machines. During tracking, we obtain the rigid motion parameters and the corresponding non-rigid blend shape coefficients for both actors. The blend shape parameters are transferred from the source to the target machine over an Ethernet network and applied to the target face model, while preserving the target head pose and lighting. The modified face is then rendered and blended into the original target sequence, and displayed in real-time on the target machine.

7.2 Expression Transfer

We synthesize a new performance for the target actor by applying the 76 captured blend shape parameters of the source actor to the personalized target model for each frame of target video. Since the source and target actor are tracked using the same parametric face model, the new target shapes can be easily expressed as

$$\mathcal{M}_{\text{geo}}\left(\boldsymbol{\alpha}_{\text{t}}, \boldsymbol{\delta}_{\text{s}}\right) = \boldsymbol{a}_{\text{id}} + E_{\text{id}}\ \boldsymbol{\alpha}_{\text{t}} + E_{\text{exp}}\ \boldsymbol{\delta}_{\text{s}}\ , \qquad (15)$$

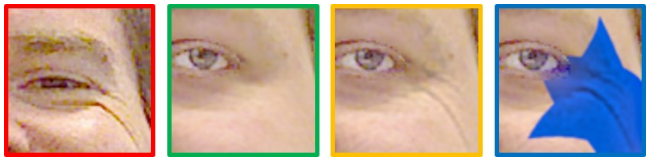

Figure 6: *Wrinkel-level detail transfer. From left to right: (a) the input source frame, (b) the rendered target geometry using only the target albedo map, (c) our transfer result, (d) a re-texturing result.*

where $\boldsymbol{\alpha}_{\text{t}}$ are the target identity parameters and $\boldsymbol{\delta}_{\text{s}}$ the source expressions. This transfer does not influence the target identity, nor the rigid head motion and scene lighting, which are preserved. Since identity and expression are optimized separately for each actor, the blend shape activation might be different across individuals. In order to account for person-specific offsets, we subtract the blendshape response for the neutral expression [Garrido et al. 2015] prior to transfer.

After transferring the blend shape parameters, the synthetic target geometry is rendered back into the original sequence using the target albedo and estimated target lighting as explained in Sec. 5.2.

7.3 Wrinkel-Level Detail Transfer

Fine-scale transient skin detail, such as wrinkles and folds that appear and disappear with changing expression, are not part of our face model, but are important for a realistic re-rendering of the synthesized face. To include dynamic skin detail in our reenactment pipeline, we model wrinkles in the image domain and transfer them from the source to the target actor. We extract the wrinkle pattern of the source actor by building a Laplacian pyramid [Burt and Adelson 1983] of the input source frame. Since the Laplacian pyramid acts as a band-pass filter on the image, the finest pyramid level will contain most of the high-frequency skin detail. We perform the same decomposition for the rendered target image and copy the source detail level to the target pyramid using the texture parametrization of the model. In a final step, the rendered target image is recomposed using the transferred source detail.

Fig. 6 illustrates our detail transfer strategy, with the source input frame shown on the left. The second image shows the rendered target face without detail transfer, while the third image shows the result obtained using our pyramid scheme. The last image shows a re-texturing result with transferred detail obtained by editing the albedo map (see Sec. 7.5). We also refer to the supplementary video for an illustration of the synthesized dynamic detail.

7.4 Final Compositing

Our face model only represents the skin surface and does not include the eyes, teeth, and mouth cavity. While we preserve the eye motion of the underlying video, we need to re-generate the teeth and inner mouth region photo-realistically to match the new target expressions.

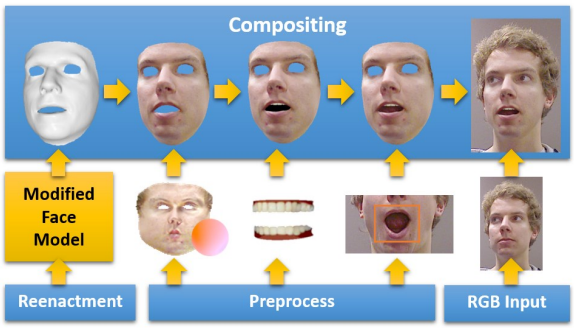

Figure 7: *Final compositing: we render the modified target geometry with the target albedo under target lighting and transfer skin detail. After rendering a person-specific teeth proxy and warping a static mouth cavity image, all three layers are overlaid on top of the original target frame and blended using a frequency based strategy.*

This is done in a compositing step, where we combine the rendered face with a teeth and inner mouth layer before blending the results in the final reenactment video (see Fig. 7).

7.4.1 Teeth Proxy and Mouth Interior

To render the teeth, we use two textured 3D proxies (billboards) for the upper and lower teeth that are rigged relative to the blend shapes of our face model and move in accordance with the blend shape parameters. Their shape is adapted automatically to the identity by means of anisotropic scaling with respect to a small, fixed number of vertices. The texture is obtained from a static image of an open mouth with visible teeth and is kept constant for all actors.

A realistic inner mouth is created by warping a static frame of an open mouth in image space. The static frame is recorded in the calibration step of Sec. 6.3 and is illustrated in Fig. 7. Warping is based on tracked 2D landmarks around the mouth and implemented using generalized barycentric coordinates [Meyer et al. 2002]. The brightness of the rendered teeth and warped mouth interior is adjusted to the degree of mouth opening for realistic shadowing effects.

7.4.2 Image Compositing

The three image layers, produced by rendering the face and teeth and warping the inner mouth, need to be combined with the original background layer and blended into the target video. Compositing is done by building a Laplacian pyramid of all the image layers (see also Sec. 7.3) and performing blending on each frequency level separately. Computing and merging the Laplacian pyramid levels can be implemented efficiently using mipmaps on the graphics hardware. To specify the blending regions, we use binary masks that indicate where the face or teeth geometry is. These masks are smoothed on successive pyramid levels to avoid aliasing at layer boundaries, e.g., at the transition between the lips, teeth, and inner mouth.

7.5 Re-Texturing and Re-Lighting Applications

Face reenactment exploits the full potential of our real-time system to instantly change model parameters and produce a realistic live rendering. The same algorithmic ingredients can also be applied in lighter variants of this scenario where we do not transfer model parameters between video streams, but modify the face and scene attributes for a single actor captured with a single camera. Examples of such an application are face re-texturing and re-lighting in a *virtual mirror* setting, where a user can apply virtual make-up

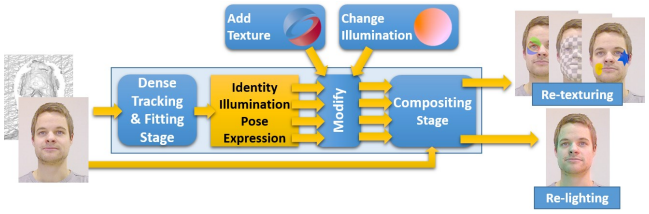

Figure 8: *Re-texturing and re-lighting of a facial performance.*

	1st Level			2nd Level			total
	#res	Syn	Ana	#res	Syn	Ana	
S1	33k	10.7ms	13.2ms	132k	1.6ms	7.1ms	32.6ms
S2	18k	11.5ms	8.2ms	72k	1.7ms	4.3ms	25.7ms
S3	22k	11.5ms	9.5ms	85k	1.7ms	5.2ms	27.9ms

Table 1: *Run times for three of the sequences of Fig. 5 (S1: Still, S2: Speaking, S3: Expression). Run time scales with the number of visible pixels in the face (distance from actor to camera), which is largest for S1, but all are real-time. '#res' is the number of residuals on that coarse-to-fine level, 'Syn' the time needed for the synthesis step and 'Ana' the time needed for the analysis step. All timings are average per-frame values computed over approx. 1000 frames.*

or tattoos and readily find out how they look like under different lighting conditions. This requires to adapt the reflectance map and illumination parameters on the spot, which can be achieved with the rendering and compositing components described before. Since we only modify the skin appearance, the virtual mirror does not require the synthesis of a new mouth cavity and teeth. An overview of this application is shown in Fig. 8. We show further examples in the experimental section and the supplementary video.

8 Results

We evaluate the performance of our tracking and reconstruction algorithm, and show visual results for facial reenactment and virtual mirror applications. For all our experiments, we use a setup consisting of an Nvidia GTX980, an Intel Core i7 Processor, and an Asus Xtion Pro RGB-D sensor that captures RGB-D frames at 30 fps. In order to obtain high-resolution textures, we record color at a resolution of 1280×1024, and upsample and register depth images accordingly. Since a face only covers the center of an image, we can safely crop the input to 640×480. During the evaluation, it turned out that our approach is insensitive to the choice of parameters. Therefore, we use the following values in all our experiments: $w_{col} = 20$, $w_{lan} = 0.125$, $w_{reg} = 0.025$, $w_{point} = 2$, $w_{plane} = 10$.

8.1 Real-time Facial Performance Capture

We track several actors in different settings, and show live demonstrations in the supplementary video. Tracking results for facial reenactment (see Sec. 8.2) are also shown in Fig. 15. Our approach first performs a short calibration phase to obtain the model identity and albedo (see Sec. 6.3). This optimization requires only a few seconds, after which the tracker continues to optimize expression and lighting in real time. Visually, the estimated identity resembles the actor well and the tracked expressions are very close to the input performance. In the following, we will provide a quantitative analysis and compare our method to state-of-the-art tracking approaches. Note however, that facial tracking is only a subcomponent of our algorithm.

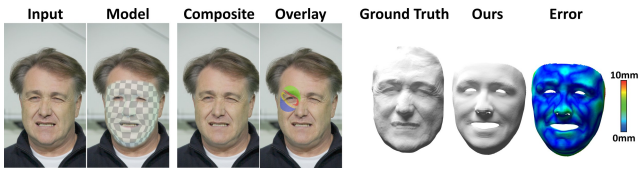

Figure 9: *Tracking accuracy. Left: the input RGB frame, the tracked model overlay, the composite and the textured model overlay. Right: the reconstructed mesh of [Valgaerts et al. 2012], our reconstructed shape, and the color coded distance between both reconstructions.*

Run Time Performance capture runs in real-time, leveraging a 3-level coarse-to-fine hierarchy to speed up convergence. In our experiments, we found that the finest level does not contribute to stability and convergence due to the noise in the consumer-level RGB-D input and the lack of information in the already upsampled depth stream. Hence, we only run our Gauss-Newton solver on the 1st and 2nd coarsest levels. Per-frame timings are presented in Table 1 for different sequences. Major pose and expression changes are captured on the 1st (coarsest) level using 7 Gauss-Newton iterations and 4 PCG steps, while parameters are refined on the 2nd level using a single Gauss-Newton iteration with 4 PCG steps. We also refer to Fig. 5 for a convergence plot. Preprocessing, including the 2D feature tracker, takes about 6ms, and blending the face with the background 3.8ms. Detail transfer between two actors for face reenactment takes about 3ms.

Tracking Accuracy To evaluate the accuracy of the reconstructed face shape, we capture the facial performance of synthetic input data with known ground truth geometry. This data was generated from a sequence of 200 high-quality facial meshes, obtained by the binocular performance capture method of Valgaerts et al. [2012] [3], by rendering successive depth maps from the viewpoint of one of the cameras. By construction, the synthetic depth sequence and the input RGB video have the same HD resolution and are aligned. Our results for a representative frame of synthetic input is shown in Fig. 9. We display the Euclidean distance between our reconstruction and the ground truth, as computed between the closest vertices on both meshes and color coded according to the accompanying scale. We see that our reconstruction closely matches the ground truth in identity and expression, with an average error of 1.5mm and a maximum error of 7.9mm over all frames. While we are able to achieve a high tracking accuracy, our face prior does not span the complete space of identities. Consequently, there will always be a residual error in shape for people who are not included in the training set.

Tracking Stability Fig. 10 demonstrates the tracking stability under rapid lighting changes. All shots are taken from the same sequence in which a light source was moved around the actor. Each shot shows the complete face model rendered back into the video stream using the albedo map with an inserted logo as well as the per-frame lighting coefficients. Note that the auto white balance of the sensor attempts to compensate for these lighting changes. In our experiments, we found that optimizing for the lighting parameters during tracking and re-rendering eliminates auto white balancing artifacts (i.e., the synthesized model will not fit the changed brightness in the input color).

Fig. 11 shows the robustness of our method under large and fast head motion. The third and fourth row depict the tracked and textured face model overlaid on the original sequence. The second row visualizes

[3] Available at `http://gvv.mpi-inf.mpg.de/projects/FaceCap/`

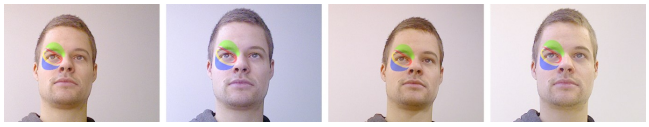

Figure 10: *Stability under lighting changes.*

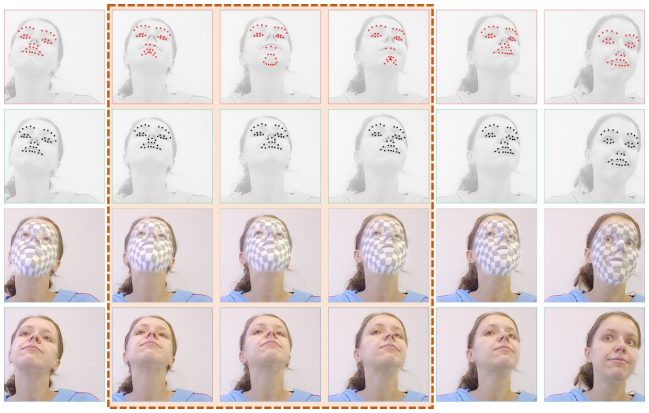

Figure 11: *Stability under head motion. From top to bottom: (a) 2D features of [Saragih et al. 2011a], (b) our 3D landmark vertices, (c) overlaid face model, (d) textured and overlaid face model. Our method recovers the head motion, even when the 2D tracker fails.*

the 38 tracked landmark vertices from the feature similarity term of Eq. (10). The projections of these vertices can be compared to the feature locations of the 2D tracker of Saragih et al. [2011a]; this difference is used in the energy term. Even when the sparse 2D tracker fails, our method can recover the head pose and expression due to the dense geometric and photo-consistency terms.

Tracking Energy We evaluate the importance of the data terms in our objective function; see Fig. 12. To this end, we measure the residual geometric (middle) and photometric error (bottom) of the reconstructed pose. Geometric error is computed with respect to the captured input depth values. Photometric error is measured as the magnitude of the residual flow field between the input and re-rendered RGB image. As we can see, relying only on the simple feature similarity measure (first column) leads to severe misalignments in the z-direction, as well as local photometric drift. While using a combination of feature similarity and photometric consistency (second column) deals with the drift in the re-rendering, the geometric error is still large due to the inherent depth ambiguity. In contrast, relying only on the geometric consistency measure (third column) removes the depth ambiguity, but is still prone to photometric drift. Only the combination of both strategies (fourth column) allows for the high geometric and photometric accuracy required in the presented real-time facial reenactment scenario.

Comparison to FaceShift We compare the tracking results of our approach to the official implementation of *FaceShift*, which is based on the work of Weise et al. [2011]. Note, this sequence has been captured with a Microsoft Kinect for Windows sensor. Our method is still able to produce high-quality results, despite the fact that the face covers a smaller 2D region in the image due to the camera's higher minimum range. In terms of the *model-to-depth* alignment error, our approach achieves comparable accuracy (see Fig. 13). For both approaches, the measured mean error is about 2mm (standard deviation of 0.4mm). Our approach achieves a much better photometric 2D alignment (measured as the magnitude of the residual flow field between the re-rendering and the RGB input); see Fig. 13

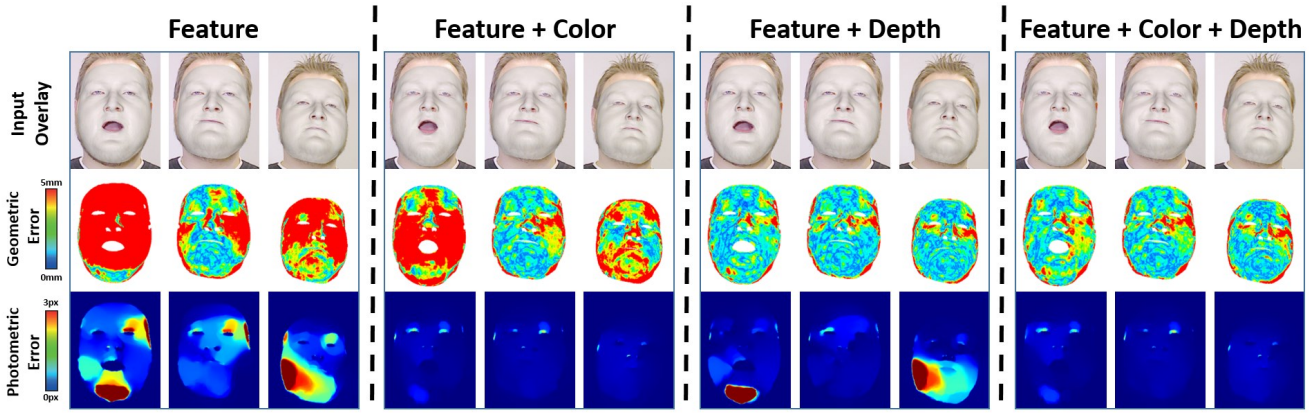

Figure 12: *Importance of the different data terms in our objective function: tracking accuracy is evaluated in terms of geometric (middle) and photometric error (bottom). The final reconstructed pose is shown as an overlay on top of the input images (top). Mean and standard deviations of geometric and photometric error are 6.48mm/4.00mm and 0.73px/0.23px for Feature, 3.26mm/1.16mm and 0.12px/0.03px for Features+Color, 2.08mm/0.16mm and 0.33px/0.19px for Feature+Depth, 2.26mm/0.27mm and 0.13px/0.03px for Feature+Color+Depth.*

(bottom). The photometric error for the *FaceShift* reconstruction is evaluated based on an illumination-corrected texture map generated based on the approach employed in our identity initialization stage. While the mean error for *FaceShift* is 0.32px (standard deviation of 0.31px), our approach has a mean error of only 0.07px (standard deviation of 0.05px). This significant improvement is a direct result of the proposed dense photometric alignment objective. Specifically in the context of photo-realistic facial reenactment (e.g., see Fig. 15), accurate 2D alignment is crucial.

Comparison to Cao et al. 2014 We also compare our method to the real-time face tracker of Cao et al. [2014a], which tracks 2D facial landmarks and infers the 3D face shape from a single RGB video stream. In a first comparison, we evaluate how well both approaches adapt to the shape identity of an actor. To this end, we use a high-quality structured light scanner to capture a static scan of the actor in rest (ground truth). We then capture a short sequence of the same rest pose with a commodity RGB-D camera for fitting

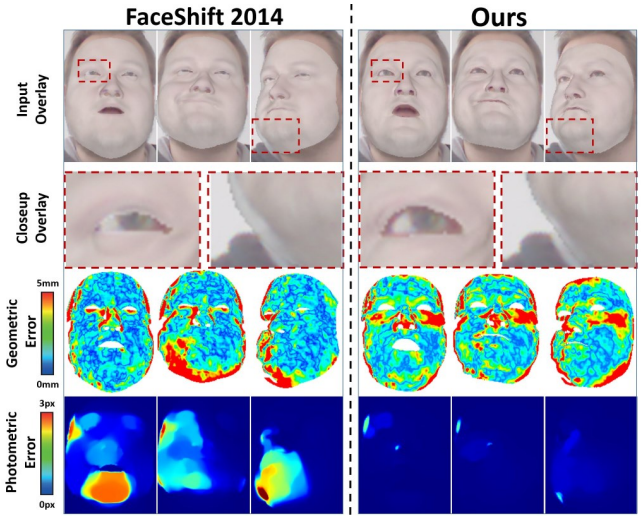

Figure 13: *Comparison to FaceShift. From top to bottom: Reconstruction overlaid on top of the RGB input, closeups, geometric alignment error with respect to the input depth maps, and photometric re-rendering error. Note that while FaceShift [Weise et al. 2011] is able to obtain a comparable model-to-depth alignment error, our reconstructions exhibit significantly better 2D alignment.*

the shape identity. The results of both methods are shown in Fig. 16, along with the per-vertex Euclidean distance to the ground truth scan. The error color scale is the same as in Fig. 9. Overall, our method approximates the identity of the actor better; however, please note that Cao et al. [2014a] only use RGB video data as input.

In Fig. 14, we compare our 3D tracking quality to Cao et al. [2014a] for the input sequence in the top row. Overall, we get more expressive results and a closer visual fit to the input expression. This is illustrated by the eyebrow raising in the second column and the cheek folding in the fourth column. A close visual fit to the input video is necessary for the applications that we aim for, namely a re-rendering of the geometry for believable video modification. Again, we would like to point out that Cao et al. [2014a] only track a sparse set of features. While less accurate, their method is significantly faster and runs in real-time even on mobile phones.

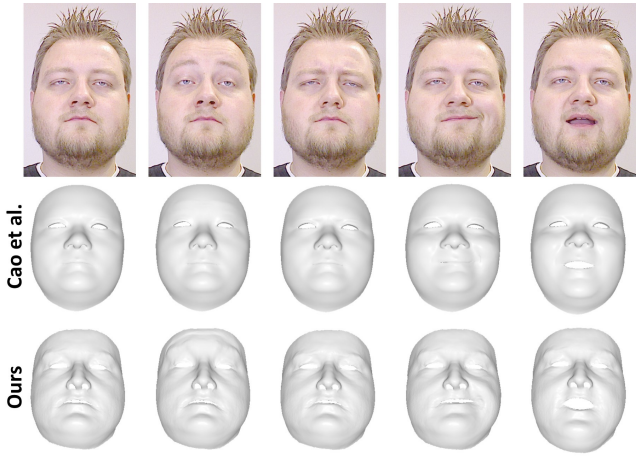

Figure 14: *State-of-the-art comparison for fitting shape expressions (i.e., tracking) assuming a fixed shape identity (cf. Fig. 16). From top to bottom: (a) input color sequence, (b) result of [Cao et al. 2014a] (RGB input), (c) our result (RGB-D input),*

8.2 Facial Reenactment

The core of our approach is the live facial reenactment setup as shown in Fig. 1. Fig. 15 shows examples of three different actor pairs, with the tracked source and target shown at the top and the

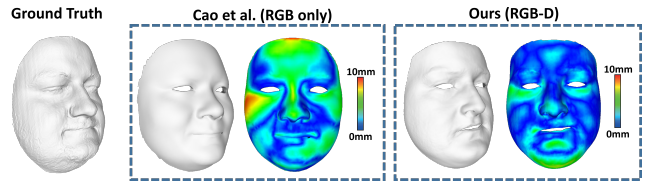

Figure 16: *State-of-the-art comparison for fitting the shape identity on a neutral expression. From left to right: (a) structured light scan (ground truth), (b) result of [Cao et al. 2014a], (c) our result. Using depth data allows us to achieve a better identity fit.*

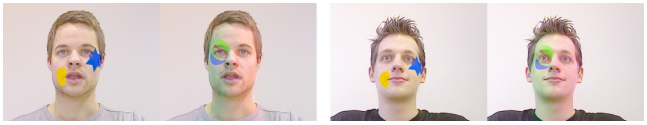

Figure 17: *Re-texturing and re-lighting a facial performance.*

reenactment at the bottom. As can be seen, we are able to track various kinds of expressions resulting in a photo-realistic reenactment. For the main results of this paper, we refer to the supplementary video, where we show many examples with a variety of actors.

8.3 Virtual Mirror

Our photo-realistic re-rendering can be also used to create a virtual mirror, where re-texturing and re-lighting can be applied to a single RGB-D input stream. For re-texturing, we apply a new texture to the albedo map, such as a logo, and render the face back into the video. To re-light the face, we replace the estimated illumination coefficients by new ones, and render the estimated face geometry under the new lighting. To avoid high-frequency changes of the illumination, we only re-light the foreground of the coarsest level of the Laplacian input pyramid that is used to composite the final output. Note that the coarsest level of the Laplacian pyramid contains only the low frequencies of the image.

9 Limitations

A limitation of our method is the assumption of Lambertian surface reflectance and smoothly varying illumination, which is parameterized by spherical harmonics. These may lead to artifacts in general environments (e.g., with strong subsurface scattering, high-frequency lighting changes, or self-shadowing). Note, however, that our method shares this limitation with related (even off-line) state-of-the-art approaches (e.g., general shape-from-shading methods or most monocular face capture methods).

In contrast to the method of Cao et al. [2014a], our real-time tracker uses dense depth and color information, which allows for tight fitting, but also leads to a high number of residuals. Currently, this makes it infeasible for our approach to run on a mobile platform, and requires a desktop computer to run in real time. Very fast head motion or extreme head poses, such as a lateral side view, may also lead to tracking failures. However, as the 2D sparse features can be robustly tracked without relying on temporal coherency, we can easily recover from tracking failures, even if previous frames were significantly misaligned. Unfortunately, darker environments introduce noise to the RGB stream of commodity depth sensors, such as the Kinect or PrimeSense, which reduces temporal tracking stability. While we are able to track extreme mouth expressions, the illusion of the mouth interior breaks at some point; i.e., if the mouth is opened too wide, the mouth interior warping and the teeth proxy lead to unnatural-looking results.

Our facial reenactment transfers expression characteristics from the source to the target actor. Thus, the reenacted performance may contain the unique style of the source actor, which is undesired in some situations. We transfer blend shape parameters one-to-one, but to account for personal differences in blend shape activation, a better mapping might be learned from the captured performances. We also assume that all actors share the same blend shapes, which might not be true in practice. An adaptation of the blend shapes to the actor [Li et al. 2013; Bouaziz et al. 2013] may improve tracking results.

Copying wrinkles from people with significantly different skin detail leads to implausible results. Predicting an actor- and expression-specific facial detail layer requires a custom-learned detail model. Unfortunately, this would involve a learning phase for each actor and expression. Nonetheless, our simple transfer strategy produces convincing results at real-time rates for a large variety of facial shapes, especially if the age of the actors is similar.

Maintaining a neutral expression for the target actor is not a hard constraint, as the non-rigid motion of the target is also tracked. However, if the synthesized face does not completely cover the input (i.e., due to strong expression changes), artifacts may appear. This could be solved using in-painting or by extending the face model (e.g., adding a neck).

10 Conclusion

We have presented the first real-time approach for photo-realistic transfer of a source actor's facial expressions to a target actor. In contrast to traditional face tracking methods, our aim is to manipulate an RGB video stream, rather the animation of a virtual character. To this end, we have introduced a novel analysis-through-synthesis approach for face tracking, which maximizes photometric consistency between the input and re-rendered output video. We are able to solve the underlying dense optimization problem with a new GPU solver in real time, thus obtaining the parameters of our face model. The parameters of the source actor are then mapped in real time to the target actor, and in combination with the newly-synthesized mouth interior, we are able to achieve photo-realistic expression transfer.

Overall, we believe that the real-time capability of our method paves the way for many new applications in the context of virtual reality and tele-conferencing. We also believe that our method opens up new possibilities for future research directions; for instance, instead of tracking a source actor with an RGB-D camera, the target video could be manipulated based on audio input.

Acknowledgements

We would like to thank Chen Cao and Kun Zhou for the blendshape models and comparison data, as well as Volker Blanz, Thomas Vetter, and Oleg Alexander for the provided face data. The facial landmark tracker was kindly provided by Jason Saraghi. We thank Angela Dai for the video voice over, as well Magdalena Prus and Matthias Innmann for being actors in our video. This research is funded by the German Research Foundation (DFG), grant GRK-1773 Heterogeneous Image Systems, the ERC Starting Grant 335545 CapReal, and the Max Planck Center for Visual Computing and Communications. In addition, we gratefully acknowledge the support from NVIDIA Corporation.

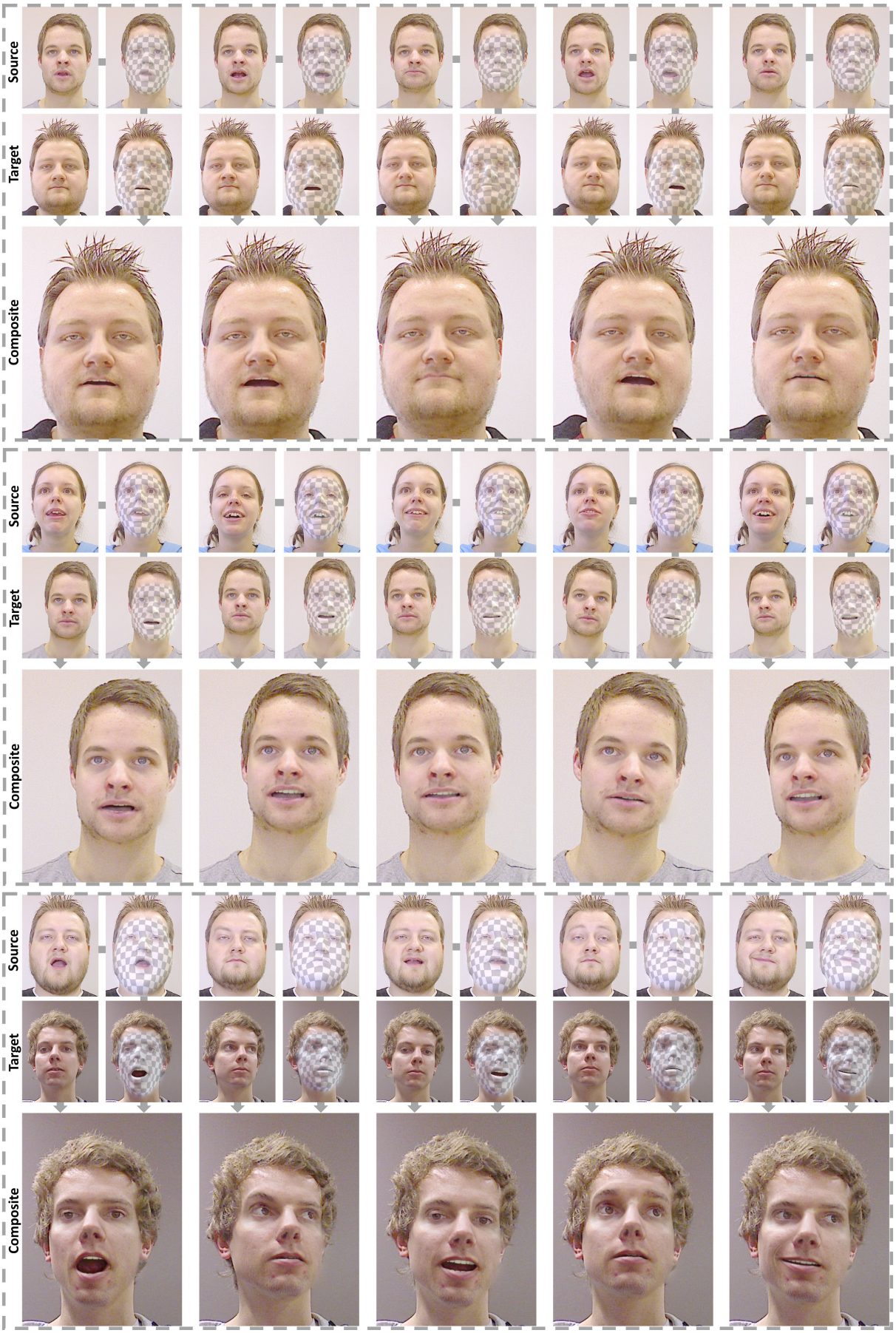

Figure 15: *Results of our reenactment system. The gray arrows show the workflow of our method.*

A List of Mathematical Symbols

Symbol	Description
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}$	shape, albedo, expression parameters
$\mathcal{M}_{\text{geo}}, \mathcal{M}_{\text{alb}}$	parametric face model
$\boldsymbol{a}_{\text{id}}, \boldsymbol{a}_{\text{alb}}$	average shape, albedo
$E_{\text{id}}, E_{\text{alb}}, E_{\text{exp}}$	shape, albedo, expression basis
$\boldsymbol{\sigma}_{\text{id}}, \boldsymbol{\sigma}_{\text{alb}}, \boldsymbol{\sigma}_{\text{exp}}$	std. dev. shape, albedo and expression
n	number of vertices
$\boldsymbol{V}, \boldsymbol{C}, \boldsymbol{N}$	set of vertices, albedos and normals
$\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{n}_i$	i-th vertex position, albedo and normal
$\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{n}, \boldsymbol{c})$	illumination model
$\boldsymbol{\gamma}$	illumination parameters
y_k	k-th SH basis function
b	number of SH bands
c	single channel albedo
$\Phi(\boldsymbol{v})$	model-to-world transformation
$\boldsymbol{R}$	rotation
$\boldsymbol{t}$	translation
$\boldsymbol{\mathcal{P}}$	vector of all parameters
p	number of all parameters
$\mathcal{S}(\boldsymbol{\mathcal{P}})$	image formation model
Π	full perspective projection
$\boldsymbol{p}$	integer pixel location
$C_{\mathcal{I}}, X_{\mathcal{I}}, N_{\mathcal{I}}$	input color, position and normal map
$C_{\mathcal{S}}, X_{\mathcal{S}}, N_{\mathcal{S}}$	synthesized color, position and normal map
$\mathcal{V}$	set of valid pixels
$E(\boldsymbol{\mathcal{P}})$	objective function
E_{emb}	geometric embedding term
E_{col}	photo-consistency term
E_{lan}	feature term
E_{reg}	regularization term
E_{point}	point-to-point term
E_{plane}	point-to-plane term
$d_{\text{point}}(\boldsymbol{p})$	point-to-point distance
$d_{\text{plane}}(\boldsymbol{p})$	point-to-plane distance
w_{col}	photo-consistency weight
w_{lan}	feature weight
w_{reg}	regularization weight
$w_{\text{point}}, w_{\text{plane}}$	geometric embedding weights
$\mathcal{F}$	set of detected features
$\boldsymbol{f}_j$	j-th feature point
$w_{\text{conf},j}$	confidence of j-th feature point
$r(\boldsymbol{\mathcal{P}})$	residual vector
$J(\boldsymbol{\mathcal{P}})$	jacobian matrix
m	number of residuals
$\boldsymbol{\mathcal{P}^k}$	parameters after the k-th iteration
$\Delta\boldsymbol{\mathcal{P}}$	parameter update

References

ADELSON, E. H., ANDERSON, C. H., BERGEN, J. R., BURT, P. J., AND OGDEN, J. M. 1984. Pyramid methods in image processing. *RCA engineer 29*, 6, 33–41.

ALEXANDER, O., ROGERS, M., LAMBETH, W., CHIANG, M., AND DEBEVEC, P. 2009. The Digital Emily Project: photoreal facial modeling and animation. In *ACM SIGGRAPH Courses*, ACM, 12:1–12:15.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM TOG 30*, 4, 75.

BICKEL, B., BOTSCH, M., ANGST, R., MATUSIK, W., OTADUY, M., PFISTER, H., AND GROSS, M. 2007. Multi-scale capture of facial geometry and motion. *ACM TOG 26*, 3, 33.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH*, ACM Press/Addison-Wesley Publishing Co., 187–194.

BLANZ, V., BASSO, C., POGGIO, T., AND VETTER, T. 2003. Reanimating faces in images and video. In *Computer graphics forum*, Wiley Online Library, 641–650.

BLANZ, V., SCHERBAUM, K., VETTER, T., AND SEIDEL, H.-P. 2004. Exchanging faces in images. In *Computer Graphics Forum*, Wiley Online Library, 669–676.

BORSHUKOV, G., PIPONI, D., LARSEN, O., LEWIS, J. P., AND TEMPELAAR-LIETZ, C. 2003. Universal capture: image-based facial animation for "The Matrix Reloaded". In *SIGGRAPH Sketches*, ACM, 16:1–16:1.

BOUAZIZ, S., WANG, Y., AND PAULY, M. 2013. Online modeling for realtime facial animation. *ACM TOG 32*, 4, 40.

BRADLEY, D., HEIDRICH, W., POPA, T., AND SHEFFER, A. 2010. High resolution passive facial performance capture. *ACM TOG 29*, 4, 41.

BURT, P. J., AND ADELSON, E. H. 1983. The Laplacian pyramid as a compact image code. *IEEE Trans. Communications 31*, 532–540.

CAO, C., WENG, Y., LIN, S., AND ZHOU, K. 2013. 3D shape regression for real-time facial animation. *ACM TOG 32*, 4, 41.

CAO, C., HOU, Q., AND ZHOU, K. 2014. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM TOG 33*, 4, 43.

CAO, C., WENG, Y., ZHOU, S., TONG, Y., AND ZHOU, K. 2014. Facewarehouse: A 3D facial expression database for visual computing. *IEEE TVCG 20*, 3, 413–425.

CHAI, J.-X., XIAO, J., AND HODGINS, J. 2003. Vision-based control of 3D facial animation. In *Proc. SCA*, Eurographics Association, 193–206.

CHEN, Y., AND MEDIONI, G. G. 1992. Object modelling by registration of multiple range images. *Image and Vision Computing 10*, 3, 145–155.

CHEN, Y.-L., WU, H.-T., SHI, F., TONG, X., AND CHAI, J. 2013. Accurate and robust 3d facial capture using a single rgbd camera. *Proc. ICCV*, 3615–3622.

CHUANG, E., AND BREGLER, C. 2002. Performance-driven facial animation using blend shape interpolation. Tech. Rep. CS-TR-2002-02, Stanford University.

COOTES, T. F., EDWARDS, G. J., AND TAYLOR, C. J. 2001. Active appearance models. *IEEE TPAMI 23*, 6, 681–685.

DALE, K., SUNKAVALLI, K., JOHNSON, M. K., VLASIC, D., MATUSIK, W., AND PFISTER, H. 2011. Video face replacement. *ACM TOG 30*, 6, 130.

EISERT, P., AND GIROD, B. 1998. Analyzing facial expressions for virtual conferencing. *CGAA 18*, 5, 70–78.

FYFFE, G., JONES, A., ALEXANDER, O., ICHIKARI, R., AND DEBEVEC, P. 2014. Driving high-resolution facial scans with video performance capture. *ACM TOG 34*, 1, 8.

GARRIDO, P., VALGAERTS, L., WU, C., AND THEOBALT, C. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM TOG 32*, 6, 158.

GARRIDO, P., VALGAERTS, L., REHMSEN, O., THORMAEHLEN, T., PEREZ, P., AND THEOBALT, C. 2014. Automatic face reenactment. In *Proc. CVPR*.

GARRIDO, P., VALGAERTS, L., SARMADI, H., STEINER, I., VARANASI, K., PEREZ, P., AND THEOBALT, C. 2015. Vdub: Modifying face video of actors for plausible visual alignment to a dubbed audio track. In *Computer Graphics Forum*, Wiley-Blackwell.

GUENTER, B., GRIMM, C., WOOD, D., MALVAR, H., AND PIGHIN, F. 1998. Making faces. In *Proc. SIGGRAPH*, ACM, 55–66.

HSIEH, P.-L., MA, C., YU, J., AND LI, H. 2015. Unconstrained realtime facial performance capture. In *Computer Vision and Pattern Recognition (CVPR)*.

HUANG, H., CHAI, J., TONG, X., AND WU, H.-T. 2011. Leveraging motion capture and 3D scanning for high-fidelity facial performance acquisition. *ACM TOG 30*, 4, 74.

KEMELMACHER-SHLIZERMAN, I., SANKAR, A., SHECHTMAN, E., AND SEITZ, S. M. 2010. Being John Malkovich. In *Proc. ECCV*, 341–353.

KEMELMACHER-SHLIZERMAN, I., SHECHTMAN, E., GARG, R., AND SEITZ, S. M. 2011. Exploring photobios. *ACM TOG 30*, 4, 61.

LEWIS, J., AND ANJYO, K.-I. 2010. Direct manipulation blendshapes. *IEEE CGAA 30*, 4, 42–50.

LI, K., XU, F., WANG, J., DAI, Q., AND LIU, Y. 2012. A data-driven approach for facial expression synthesis in video. In *Proc. CVPR*, 57–64.

LI, H., YU, J., YE, Y., AND BREGLER, C. 2013. Realtime facial animation with on-the-fly correctives. *ACM TOG 32*, 4, 42.

LIU, Z., SHAN, Y., AND ZHANG, Z. 2001. Expressive expression mapping with ratio images. In *Proc. SIGGRAPH*, ACM, 271–276.

MEYER, M., BARR, A., LEE, H., AND DESBRUN, M. 2002. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools 7*, 1, 13–22.

MÜLLER, C. 1966. *Spherical harmonics*. Springer.

PIGHIN, F., AND LEWIS, J. 2006. Performance-driven facial animation. In *ACM SIGGRAPH Courses*.

PIGHIN, F., HECKER, J., LISCHINSKI, D., SZELISKI, R., AND SALESIN, D. 1998. Synthesizing realistic facial expressions from photographs. In *Proc. SIGGRAPH*, ACM Press/Addison-Wesley Publishing Co., 75–84.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. A signal-processing framework for inverse rendering. In *Proc. SIGGRAPH*, ACM, 117–128.

SARAGIH, J. M., LUCEY, S., AND COHN, J. F. 2011. Deformable model fitting by regularized landmark mean-shift. *IJCV 91*, 2, 200–215.

SARAGIH, J. M., LUCEY, S., AND COHN, J. F. 2011. Real-time avatar animation from a single image. In *Automatic Face and Gesture Recognition Workshops*, 213–220.

SHI, F., WU, H.-T., TONG, X., AND CHAI, J. 2014. Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM TOG 33*, 6, 222.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM TOG 23*, 3, 399–405.

SUWAJANAKORN, S., KEMELMACHER-SHLIZERMAN, I., AND SEITZ, S. M. 2014. Total moving face reconstruction. In *Proc. ECCV*, 796–812.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph. 31*, 6, 187.

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. 2005. Face transfer with multilinear models. *ACM TOG 24*, 3, 426–433.

WANG, Y., HUANG, X., SU LEE, C., ZHANG, S., LI, Z., SAMARAS, D., METAXAS, D., ELGAMMAL, A., AND HUANG, P. 2004. High resolution acquisition, learning and transfer of dynamic 3-D facial expressions. *CGF 23*, 677–686.

WEISE, T., LI, H., GOOL, L. J. V., AND PAULY, M. 2009. Face/Off: live facial puppetry. In *Proc. SCA*, 7–16.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. 77.

WILLIAMS, L. 1990. Performance-driven facial animation. In *Proc. SIGGRAPH*, 235–242.

WILSON, C. A., GHOSH, A., PEERS, P., CHIANG, J.-Y., BUSCH, J., AND DEBEVEC, P. 2010. Temporal upsampling of performance geometry using photometric alignment. *ACM TOG 29*, 2, 17.

XIAO, J., BAKER, S., MATTHEWS, I., AND KANADE, T. 2004. Real-time combined 2D+3D active appearance models. In *Proc. CVPR*, 535 – 542.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: high resolution capture for modeling and animation. *ACM TOG 23*, 3, 548–558.

ZOLLHÖFER, M., NIESSNER, M., IZADI, S., REHMANN, C., ZACH, C., FISHER, M., WU, C., FITZGIBBON, A., LOOP, C., THEOBALT, C., AND STAMMINGER, M. 2014. Real-time Non-rigid Reconstruction using an RGB-D Camera. *ACM TOG 33*, 4, 156.