Dynamic 3D Avatar Creation from Hand-held Video Input

Alexandru Eugen Ichim[*]
EPFL

Sofien Bouaziz[†]
EPFL

Mark Pauly[‡]
EPFL

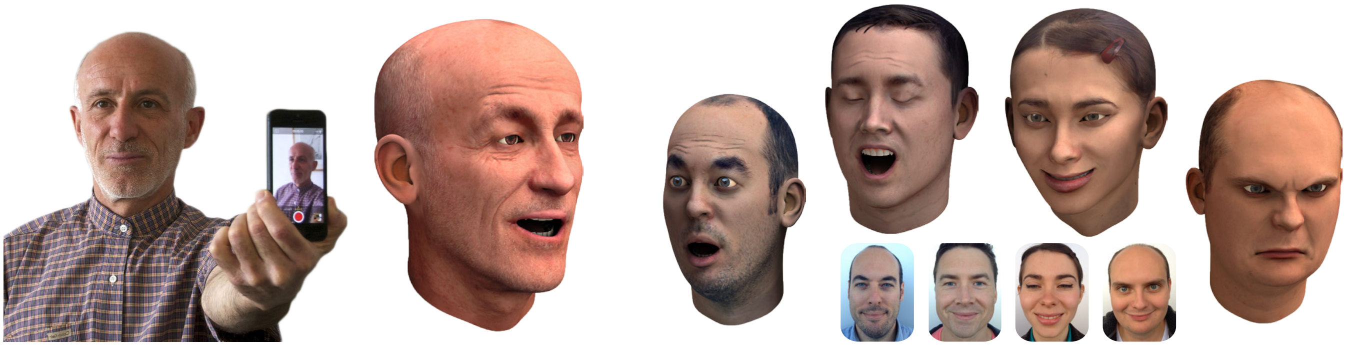


Figure 1: *Our system creates a fully rigged 3D avatar of the user from uncalibrated video input acquired with a cell-phone camera. The blendshape models of the reconstructed avatars are augmented with textures and dynamic detail maps, and can be animated in realtime.*

Abstract

We present a complete pipeline for creating fully rigged, personalized 3D facial avatars from hand-held video. Our system faithfully recovers facial expression dynamics of the user by adapting a blendshape template to an image sequence of recorded expressions using an optimization that integrates feature tracking, optical flow, and shape from shading. Fine-scale details such as wrinkles are captured separately in normal maps and ambient occlusion maps. From this user- and expression-specific data, we learn a regressor for on-the-fly detail synthesis during animation to enhance the perceptual realism of the avatars. Our system demonstrates that the use of appropriate reconstruction priors yields compelling face rigs even with a minimalistic acquisition system and limited user assistance. This facilitates a range of new applications in computer animation and consumer-level online communication based on personalized avatars. We present realtime application demos to validate our method.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: 3D avatar creation, face animation, blendshapes, rigging

1 Introduction

Recent advances in realtime face tracking enable fascinating new applications in performance-based facial animation for entertainment and human communication. Current realtime systems typically use the extracted tracking parameters to animate a set of pre-defined characters [Weise et al. 2009; Weise et al. 2011; Li et al. 2013; Cao et al. 2013; Bouaziz et al. 2013; Cao et al. 2014a]. While this allows the user to enact virtual avatars in realtime, personalized interaction requires a custom rig that matches the facial geometry, texture, and expression dynamics of the user. With accurate tracking solutions in place, creating compelling user-specific face rigs is currently a major challenge for new interactive applications in online communication. In this paper we propose a software pipeline for building fully rigged 3D avatars from hand-held video recordings of the user.

Avatar-based interactions offer a number of distinct advantages for online communication compared to video streaming. An important benefit for mobile applications is the significantly lower demand on bandwidth. Once the avatar has been transferred to the target device, only animation parameters need to be transmitted during live interaction. Bandwidth can thus be reduced by several orders of magnitude compared to video streaming, which is particularly relevant for multi-person interactions such as conference calls.
A second main advantage is the increased content flexibility. A 3D avatar can be more easily integrated into different scenes, such as games or virtual meeting rooms, with changing geometry, illumination, or viewpoint. This facilitates a range of new applications, in particular on mobile platforms and for VR devices such as the Occulus Rift.

Our goal is to enable users to create fully rigged and textured 3D avatars of themselves at home. These avatars should be as realistic as possible, yet lightweight, so that they can be readily integrated into realtime applications for online communication. Achieving this goal implies meeting a number of constraints: the acquisition hardware and process need to be *simple and robust*, precluding any custom-build setups that are not easily deployable. Manual assistance needs to be *minimal* and restricted to operations that can be easily performed by untrained users. The created rigs need to be *efficient* to support realtime animation, yet accurate and detailed to enable engaging virtual interactions.

[*] alexandru.ichim@epfl.ch
[†] sofien.bouaziz@epfl.ch
[‡] mark.pauly@epfl.ch

ACM Reference Format
Ichim, A., Bouaziz, S., Pauly, M. 2015. Dynamic 3D Avatar Creation from Hand-Held Video Input. ACM Trans. Graph. 34, 4, Article 45 (August 2015), 14 pages. DOI = 10.1145/2766974 http://doi.acm.org/10.1145/2766974.

SIGGRAPH '15 Technical Paper, August 09 – 13, 2015, Los Angeles, CA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3331-3/15/08 ... $15.00.
DOI: http://dx.doi.org/10.1145/2766974

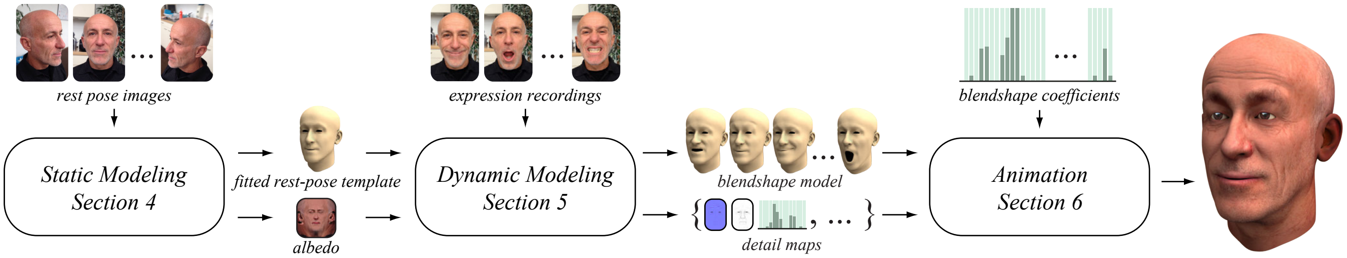


Figure 2: *The main stages of our processing pipeline.* Static Modeling *reconstructs the geometry and albedo of the neutral pose.* Dynamic Modeling *adapts a generic blendshape model to the recorded user and reconstructs detail maps for each video frame.* Animation *drives the reconstructed rig using blendshape coefficients and synthesizes new pose-specific detail maps on the fly.*

These requirements pose significant technical challenges. We maximize the potential user base of our system by only relying on simple photo and video recording using a hand-held cell-phone camera to acquire user-specific data.
The core processing algorithms of our reconstruction pipeline run automatically. To improve reconstruction quality, we integrate a simple UI to enable the user to communicate tracking errors with simple point clicking. User assistance is minimal, however, and required less than 15 minutes of interaction for all our examples.
Realtime applications are enabled by representing the facial rig as a set of blendshape meshes with low polygon count. Blendshapes allow for efficient animation and are compatible with all major animation tools. We increase perceptual realism by adding fine-scale facial features such as dynamic wrinkles that are synthesized on the fly during animation based on precomputed normal and ambient occlusion maps.

We aim for the best possible quality of the facial rigs in terms of geometry, texture, and expression dynamics. To achieve this goal, we formulate dynamic avatar creation as a geometry and texture reconstruction problem that is regularized through the use of carefully designed facial priors. These priors enforce consistency and guarantee a complete output for a fundamentally ill-posed reconstruction problem.

Contributions. We present a comprehensive pipeline for video-based reconstruction of fully-rigged, user-specific 3D avatars for consumer applications in uncontrolled environments. Our core technical contributions are:

- a two-scale representation of a dynamic 3D face rig that enables realtime facial animation by integrating a medium-resolution blendshape model with a high-resolution albedo map and dynamic detail maps;
- a novel optimization method for reconstructing a consistent albedo texture from a set of input images that factors out the incident illumination;
- a new algorithm to build the dynamic blendshape rig from video input using a joint optimization that combines feature-based registration, optical flow, and shape-from-shading;
- an offline reconstruction and online synthesis method for fine-scale detail stored in pose-specific normal and ambient occlusion maps.

We demonstrate the application potential of our approach by driving our reconstructed rigs both in a realtime animation demo and using a commodity performance capture system. With our minimalistic acquisition setup using only a single cellphone camera, our system has the potential to be used by millions of users worldwide.

2 Related Work

We provide an overview of relevant techniques for 3D facial avatar creation. We start by covering techniques for high quality *static* modeling of human faces. We then discuss approaches that attempt to capture fine-scale information associated with *dynamic* facial deformations, like expression lines and wrinkles. Finally, as our target is the creation of an animatable avatar, we will also discuss methods that attempt to map the acquired dynamic details onto given input animation data.

Static modeling. Due to the high complexity of facial morphology and heterogeneous skin materials, the most common approaches in facial modeling are data-driven. The seminal work of [Blanz and Vetter 1999] builds a statistical (PCA) model of facial geometry by registering a template model to a collection of laser scans. Such a model can be employed to create static avatars from a single image [Blanz and Vetter 1999] or from multiple images [Amberg et al. 2007], or for the creation of personalized real-time tracking profiles [Weise et al. 2011; Li et al. 2013; Bouaziz et al. 2013]. However, as a compact PCA model only captures the coarse-scale characteristics of the dataset, the generated avatars are typically rather smooth, lacking the ability to represent fine-scale features like wrinkles and expression lines.

Fine-scale detail for facial modeling has been recovered in a controlled environment with multiple calibrated DSLR cameras in the work of Beeler et al. [2010]. This setup allows capturing wrinkles, skin pores, facial hair [Beeler et al. 2012], and eyes [Bérard et al. 2014]. The more involved system of [Ghosh et al. 2011] uses fixed linear polarizers in front of the cameras and enables accurate acquisition of diffuse, specular, and normal maps. While effective for high-end productions, such systems require a complex calibration within a lab environment and are thus unsuitable for personalized avatar creation at home. In contrast, our approach uses only a cell-phone camera, requires neither calibration nor a controlled environment, and only relies on minimal user assistance.

Dynamic modeling. A static reconstruction only recovers the geometry and texture for a single facial expression. To build compelling avatars, we also need to reconstruct a dynamic expression model that faithfully captures the user's specific facial movements. One approach to create such a model is to simulate facial muscle activation and model the resulting bone movements and viscoelastic skin deformations [Venkataraman et al. 2005; Wu et al. 1996]. However, the large computational cost and complex parameter estimation make such an approach less suitable for facial animation.

Consequently, parametric models are typically employed to represent dynamic skin behavior [Oat 2007; Jimenez et al. 2011]. Unfortu-

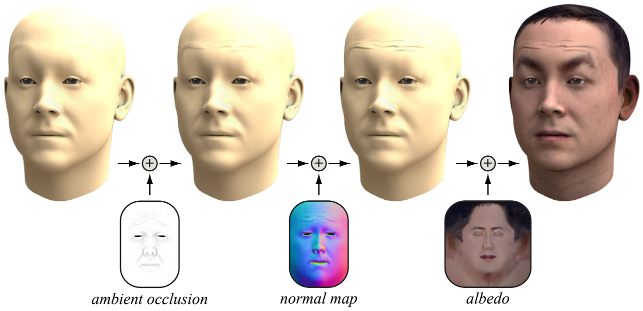


Figure 3: *Our dynamic face rig augments a low-resolution blendshape pose (left) with dynamic per-texel ambient occlusion coefficients and normals, and a high-resolution albedo texture.*

nately, such models are not only difficult to design, but are typically also custom-tuned to a particular animation rig. This makes it difficult to infer generic models for facial dynamics that can easily be adapted to specific subjects. For these reasons, data-driven techniques are again the most common way to approach the reconstruction of facial dynamics.

The multi-linear models introduced by [Vlasic et al. 2005] and then further explored in [Cao et al. 2014b] offer a way of capturing a joint space of pose and identity. Alternatively, rather than assuming an offline prior on pose and identity, dynamic geometry variations can be linearly modeled in realtime while tracking videos [Cao et al. 2014a] or RGB-D data [Bouaziz et al. 2013; Li et al. 2013]. These compact linear models are tailored towards estimating a small set of tracking parameters to enable realtime performance, and consequently are not suitable to recover detailed avatars. Our approach builds upon this prior work, but utilizes detail information in the acquired images to recover a significantly richer set of facial features for avatar creation.

The use of custom hardware has been the most successful way of estimating dynamic avatars for high-end productions. For example, the Digital Emily project [Alexander et al. 2009] demonstrates how the Light Stage system enables photorealistic dynamic avatars. The work of Alexander et al. [2013] recently extended this approach to enable real-time rendering of highly detailed facial rigs. Structured light and laser scanners have also been used to acquire facial geometry at the wrinkle scale [Zhang et al. 2004; Ma et al. 2008; Li et al. 2009; Huang et al. 2011]. Similarly, the setup of [Beeler et al. 2010; Beeler et al. 2011] is capable of reconstructing fine-scale detail using multiple calibrated/synchronized DSLR cameras. More recent work attempts to further reduce the setup complexity by only considering a *binocular* [Valgaerts et al. 2012] or a hybrid *binocular/monocular* setup [Garrido et al. 2013]. We push this trend to its limit by only requiring hand-held video recording in an uncontrolled environment.

Animation. While the methods above are able to infer detailed geometry we aim for the creation of an avatar of the recorded user, that can be animated programmatically or using other sources of tracking parameters. The systems of [Garrido et al. 2013], and [Shi et al. 2014] essentially recover detailed facial geometry by providing one mesh per frame deformed to match the input data. The former uses a pre-built user-specific blendshape model for the face alignment by employing automatically corrected feature points [Saragih et al. 2011]. A dense optical flow field is computed in order to smoothly deform the tracked mesh at each frame, after which a shape-from-shading stage adds high frequency details. Although our tracking approach and detail enhancement is based on similar principles, the aim of our approach is to integrate all these shape corrections directly into our proposed two-scale representation of dynamic 3D faces. Shi et al. [2014] use their own feature detector along with a non-rigid structure-from-motion algorithm to track and model the identity and per-frame expressions of the face by employing a bilinear face model. Additionally, a keyframe-based iterative approach using shape from shading is employed in order to further refine the bilinear model parameters, as well as the albedo texture of the face, and per-frame normal maps exhibiting high frequency details such as wrinkles Neither method aims at creating an animation-ready avatar that incorporates all of the extracted details.

Of the methods presented above, only Alexander and colleagues [2009; 2013] produce a blendshape model that can be directly embedded in animation software, but as mentioned, the complexity of the setup makes it unsuitable for consumer applications. The recent techniques of [Bermano et al. 2014], and [Li et al. 2015] can re-introduce high frequency details in a coarse input animation, if a high-resolution performance database is provided. Conversely, our technique generates an animatable blendshape model augmented with dynamic detail maps using only consumer camera data. Our rigged avatars can thus be directly driven by tracking software, e.g. [Weise et al. 2011; Saragih et al. 2011], or posed in a keyframe animation system.

3 Overview

We first introduce our two-scale representation for 3D facial expression rigs. Then we discuss the acquisition setup and provide a high-level description of the main processing steps for the reconstruction and animation of our rigs (Figure 2). The subsequent sections explain the core technical contributions, present results, and provide an evaluation of our method. The paper concludes with a discussion of limitations and an outline of potential future work. Implementation details are provided in the Appendix.

Dynamic Face Rig. Our method primarily aims at the reconstruction of 3D facial rigs for realtime applications. We therefore propose a two-scale representation that strikes a balance between a faithful representation of the dynamic expression space of the recorded user and the efficient animation of the reconstructed avatar. This balance can be achieved using a coarse blendshape mesh model of approximatively 10k vertices that is personalized to the specific user and augmented with texture and detail information as shown in Figure 3.

A specific facial expression is represented by a linear combination of a set of blendshapes [Lewis et al. 2014]. At low resolution, the blendshape representation can be efficiently evaluated and rendered, but lacks fine-scale detail. We therefore augment the mesh with a static high-resolution albedo map to capture color variations across the face. In addition, we build dynamic high-resolution maps with per-pixel normals and ambient occlusion coefficients to represent fine-scale geometric features. We refer to these latter maps as *detail maps* in the subsequent text. Previous methods such as [Bickel et al. 2008] use a similar two-scale decomposition, but operate on high-resolution meshes and can thus represent details as displacements. To avoid the complexities of realtime displacement mapping we opted for normal and ambient occlusion maps that can be synthesized and rendered more efficiently during animation.

Acquisition. In order to build a dynamic rig of the user we need to capture enough information to reconstruct the blendshapes, the albedo texture, and the detail maps. At the same time, keeping our consumer application scenario in mind, we want to restrict the acquisition to simple hardware and a minimalistic process that can be robustly performed by anyone. We therefore opted for a simple hand-held cell-phone camera. The user first records her- or

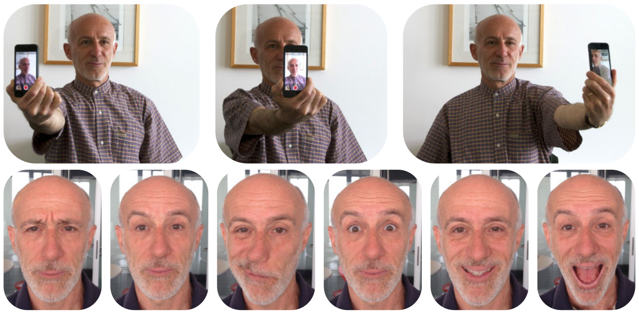


Figure 4: *All acquisition is performed with a hand-held cell phone camera. A semi-circular sweep is performed for the static reconstruction (top row), a frontal video is recorded for the dynamic modeling (bottom row).*

himself in neutral expression by sweeping the camera around the face capturing images in burst mode. We then ask the user to record a video in a frontal view while performing different expressions to capture user-specific dynamic face features (see Figure 4). For all our acquisitions we use an Apple iPhone 5 at 8 megapixel resolution for static photo capture and 1080p for dynamic video recordings (see accompanying video).

The key advantage of our acquisition setup is that we do not require any calibration, synchronization, or controlled lighting. All acquisitions can be done by an inexperienced user in approximatively 10 minutes. However, this simplistic acquisition process poses significant challenges for our reconstruction algorithms as the quality of the input data is significantly lower than for existing calibrated studio setups.

Processing Pipeline. Figure 2 provides an overview of our processing pipeline. We split the reconstruction into a static and a dynamic modeling stage. In the static stage (Section 4) we fist reconstruct a 3D point cloud from the photos taken in neutral pose using a multi-view stereo algorithm. We then apply non-rigid registration to align a template mesh to this point cloud to model the user's face geometry. A static albedo texture is extracted by integrating the color images into a consistent texture map.

The dynamic modeling stage (Section 5) reconstructs expression-specific information. Given the neutral pose, we first transfer the deformations of a generic blendshape model to obtain an initial blendshape representation for the user. We further refine this user-specific blendshape model using an optimization that integrates texture-based tracking and shading cues to best match the geometric features of the recorded user. The reconstructed blendshape model then faithfully recovers the low- and medium frequency dynamic geometry of the user's face. However, high frequency details such as wrinkles are still missing from the rig. In order to capture these details we automatically extract a set of dynamic detail maps from the recorded video frames.

Finally, in the animation stage (Section 6), the reconstructed rig can be driven by a temporal sequence of blendshape coefficients. These animation parameters can either be provided manually through interactive controllers, or transferred from a face tracking software. The specific detail map for each animated pose of the avatar is synthesized on the fly from the captured detail maps using a trained regressor driven by surface strain.

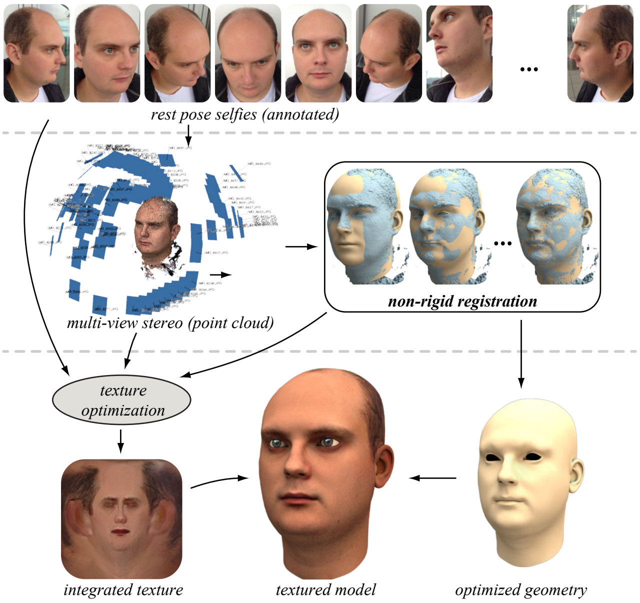


Figure 5: *Static Modeling recovers the neutral pose. A deformable template model is registered to a 3D point cloud computed from a set of images using multi-view stereo. A static albedo texture integrates color information of all recorded images while factoring out the illumination.*

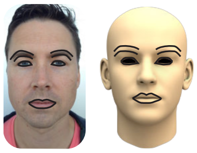
4 Static Modeling

This section describes the static modeling stage of the reconstruction pipeline (see Figure 5). The first part of the acquisition provides us with a set of images of the user in neutral expression from different viewpoints. From these uncalibrated images we extract a point cloud using a state-of-the-art structure from motion (SFM) software [Furukawa and Ponce 2010; Wu 2013]. We then use a geometric morphable model [Blanz and Vetter 1999], representing the variations of different human faces in neutral expression, as a prior for reconstruction.

4.1 Geometry Registration

We register the morphable model towards the point cloud to obtain a template mesh that roughly matches the geometry of the user's face. We improve the registration accuracy using non-rigid registration based on thin-shell deformation [Botsch et al. 2010; Bouaziz et al. 2014].

The registration is initialized by using 2D-3D correspondences of automatically detected 2D facial features [Saragih et al. 2011] in each input frame. For the precise non-rigid alignment of the mouth, eye and eyebrow regions, the user is asked to mark a few contours in one of the frontal images as illustrated on the right.



To improve the realism of the reconstructed avatars, we add eyes and inner mouth components, i.e., teeth, tongue, and gums. These parts are transferred from the template model and deformed to match the reconstructed head geometry by optimizing for the rotation, translation and anisotropic scaling using a set of predefined feature points around the mouth and eye regions. We also adapt the texture for the eye meshes to the recorded user. The iris is found by detecting

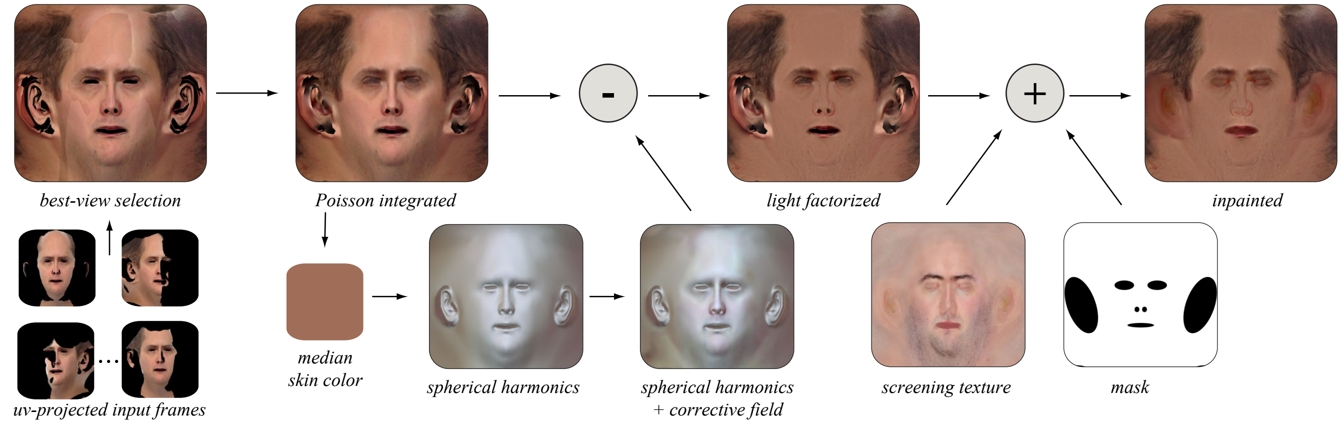
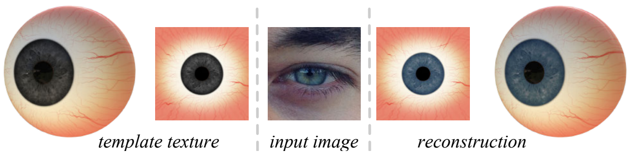


Figure 6: *Reconstructing the albedo map. Poisson integration combines the pixel colors of different input images using a gradient optimization. The illumination is factored out based on a lighting model that combines a 2nd order spherical harmonics approximation with a per-pixel corrective field. The screening texture provides a reconstruction prior to complete missing parts in the albedo map.*

the largest ellipse inside the projection of the eye region to the most frontal input image using Hough transform [Duda and Hart 1972]. Histogram matching is performed between a template eye texture and the image patch corresponding to the iris [Gonzalez and Woods 2006]. The images below illustrate the eye texture adaptation for one example subject.



We currently do not recover the specific geometry or appearance of the user's teeth or tongue, which is an interesting topic for future work.

4.2 Texture Reconstruction

Given the registered template mesh, the next step in the pipeline is the reconstruction of a high-resolution albedo texture (see Figure 6). We use the UV parameterization of the template to seamlessly combine all images using Poisson integration [Pérez et al. 2003]. This is achieved by selecting the color gradients of the pixels with the most parallel view rays to the surface normals.

Factorizing Illumination. After integration, the texture map not only contains the RGB reflectance but also the specific illumination of the recording environment. This may be problematic as the final mesh will be used in a virtual environment where the lighting may not match the one baked into the texture. To factor out the illumination from the skin albedo, we define the color of a skin pixel $\{i, j\}$ as $\mathbf{c}_{ij} = \mathbf{r}_{ij} \circ \mathbf{s}_{ij}$, where $\mathbf{r}_{ij}$ is the skin reflectance, $\mathbf{s}_{ij}$ accounts for the illumination, and $\circ$ denotes the entry-wise product. We assume a smooth illumination and we represent it using spherical harmonics. Low-dimensional lighting representations using spherical harmonics are effective in numerous lighting situations with a variety of object geometries [Frolova et al. 2004]. However, they are not expressive enough to account for complex conditions involving self-shadowing or complex specularities. This is due to the fact that spherical harmonics have the limitation of being only expressed as a function of the surface normals, i.e., points with similar normals will have a similar illumination. To compensate for the inaccuracy of this illumination model, we augment the spherical harmonics with corrective fields in uv-space $\mathbf{d}_{ij} = [d^r_{ij} \;\; d^g_{ij} \;\; d^b_{ij}]$ for the R, G and B color channel, respectively. This leads to

$$\mathbf{s}_{ij} = \mathbf{y}^T \phi(\mathbf{n}_{ij}) + \mathbf{d}^T_{ij}, \tag{1}$$

where $\mathbf{n}_{ij}$ is the mesh normal at the pixel p and

$$\phi(\mathbf{n}) = [1, n_x, n_y, n_z, n_x n_y, n_x n_z, n_y n_z, n_x^2 - n_y^2, 3n_z^2 - 1]^T \tag{2}$$

is a vector of second order spherical harmonics with corresponding weight vectors $\mathbf{y} = [\mathbf{y}^r \;\; \mathbf{y}^g \;\; \mathbf{y}^b]$. As the illumination is assumed to be of low frequency, we require the corrective fields to be smooth. In addition, we assume that the required corrections are small. This leads to a minimization over the spherical harmonics weight vectors and the corrective fields expressed as

$$\min_{\mathbf{y},\mathbf{d}} \sum_{i,j} \|\mathbf{r} \circ \mathbf{s}_{ij} - \mathbf{c}_{ij}\|_2^2 + \lambda_1 \|\mathbf{d}\|_F^2 + \lambda_2 \|\mathbf{G}\mathbf{d}\|_F^2 + \lambda_3 \|\mathbf{L}\mathbf{d}\|_F^2, \tag{3}$$

where $\|\cdot\|_F$ is the Frobenius norm, $\mathbf{d}$ stacks all the $\mathbf{d}_{ij}$, $\mathbf{G}$ is the gradient matrix, and $\mathbf{L}$ is the graph Laplacian matrix. Both the gradient and Laplacian are computed with periodic boundary condition. The non-negative weights λ_1, λ_2, and λ_3 control the magnitude and the smoothness of the corrective fields. To optimize Equation 3, we employ a two-stage process, where the skin reflectance is set to a constant $\mathbf{r}$ using the median color of the face pixels. We first compute the spherical harmonics weight vectors by initializing the corrective fields to zero and only optimizing over $\mathbf{y}$. This only requires solving a 9×9 linear system. We then solve for the corrective fields keeping the weight vectors fixed. This minimization can be performed efficiently using a Fast Fourier Transform (FFT) as the system matrix is circulant [Gray 2006].

We use the extracted illumination $\mathbf{s}_{ij}$ to reconstruct the illumination-free texture. Finally, to generate a complete texture map, we reintegrate the illumination-free texture into the template texture map using Poisson integration. Because the extracted illumination is smooth, i.e., of low frequency, high frequency details are preserved in the final albedo texture (see Figure 6).

5 Dynamic Modeling

The goal of the dynamic modeling phase is to complete the face rig by reconstructing user-specific blendshapes as well as dynamic

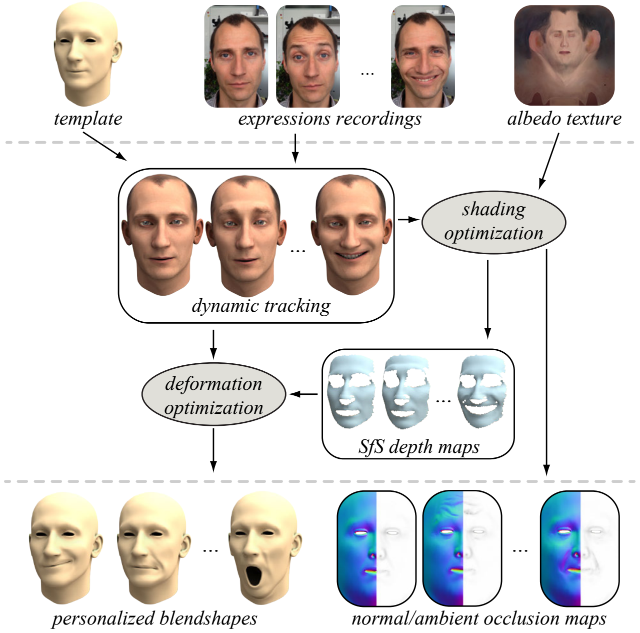


Figure 7: *Dynamic Modeling adapts a generic blendshape model to the facial characteristics of the user and recovers expression-specific detail maps from the recorded video sequence.*

normal and ambient occlusion maps (see Figure 7). We focus here on the general formulation of our optimization algorithm and refer to the appendix for more details on the implementation.

5.1 Reconstructing the Blendshape Model

The blendshape model is represented as a set of meshes $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_n]$, where $\mathbf{b}_0$ is the neutral pose and the $\mathbf{b}_i, i > 0$ are a set of predefined facial expressions. A novel facial expression is generated as $\mathbf{F}(\mathbf{B}, \mathbf{w}) = \mathbf{b}_0 + \Delta\mathbf{B}\mathbf{w}$, where $\Delta\mathbf{B} = [\mathbf{b}_1 - \mathbf{b}_0, \ldots, \mathbf{b}_n - \mathbf{b}_0]$, and $\mathbf{w} = [w_1, \ldots, w_n]^T$ are blendshape weights. The reconstruction prior at this stage is a generic blendshape model consisting of 48 blendshapes (see also accompanying material). We denote with $\mathbf{F_T}$ the facial expression $\mathbf{F}$ transformed by the rigid motion $\mathbf{T} = (\mathbf{R}, \mathbf{t})$ with rotation $\mathbf{R}$ and translation $\mathbf{t}$.

We initialize the user-specific blendshape model by applying deformation transfer [Sumner and Popović 2004] from the generic blendshape template to the reconstructed mesh of the user's neutral pose. Deformation transfer directly copies the deformation gradients of the template without accounting for the particular facial expression dynamics of the user. To personalize the blendshape model, we optimize for additional surface deformations of each blendshape to better match the facial expressions of the user in the recorded video sequence. Previous methods, such as [Bouaziz et al. 2013; Li et al. 2013], perform a similar optimization using 3D depth-camera input. However, these methods only aim at improving realtime tracking performance and do not recover detailed rigged avatars. Moreover, in our reconstruction setting we are not constrained to realtime performance and can thus afford a more sophisticated optimization specifically designed for our more challenging 2D video input data.

Our algorithm alternates between *tracking*, i.e., estimating the blendshape weights and rigid pose of the facial expressions in the image sequence, and *modeling*, i.e., optimizing the blendshapes to better fit the user's expression.

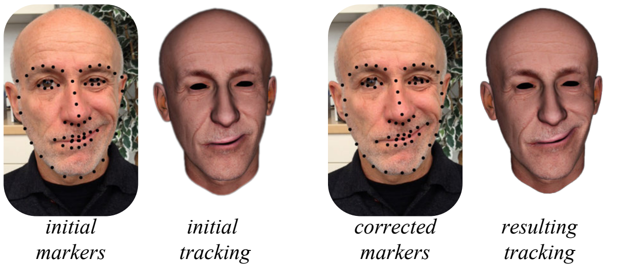


Figure 8: *The initial marker locations extracted using feature detection can be corrected by the user to improve the tracking. Here the markers at the mouth corners and jaw line have been manually displaced. Such user annotations are propagated through the entire sequence, so that only a small number of frames need to be corrected.*

Tracking. We propose a tracking algorithm using 2D image-based registration based on a combination of feature alignment and optical flow. This results in a per-frame optimization over the blendshape weights $\mathbf{w}$ and the rigid motion $\mathbf{T}$ expressed as

$$\arg\min_{\mathbf{w},\mathbf{T}} E_{\text{feature}} + E_{\text{flow}} + E_{\text{sparse}}. \tag{4}$$

We formulate the facial feature energy as

$$E_{\text{feature}} = \gamma_1 \sum_{v \in \mathcal{M}} \|\mathbf{m}_v - P(\mathbf{F_T}(\mathbf{B}_v, \mathbf{w}))\|_2^2, \tag{5}$$

where $\mathcal{M}$ is the set of points representing the facial feature locations on the mesh surface, $\mathbf{m}_v$ is the 2D image location of the feature point v extracted using the method of [Saragih et al. 2011], $P(\cdot)$ projects a 3D point to 2D, and $\mathbf{B}_v = \mathbf{c}_v^T\mathbf{B}$, where the vector $\mathbf{c}_v$ contains the barycentric coordinates corresponding to v.

The feature extraction algorithm of [Saragih et al. 2011] is fairly accurate, but does not always find the correct marker locations. To improve the quality of the tracking, we ask the user to correct marker locations in a small set of frames (see Figure 8). Following [Garrido et al. 2013], these edits are then propagated through the image sequence using frame-to-frame optical flow [Zach et al. 2007]. For a sequence of 1500 frames, we typically require 25 frames to be manually corrected. With more sophisticated feature extraction algorithms such as [Cao et al. 2014a], this manual assistance can potentially be dispensed with completely.

To complement the feature energy, we use a texture-to-frame optical flow using a gradient-based approach. This formulation increases the robustness to lighting variations between the static and dynamic acquisition. This energy is defined as

$$E_{\text{flow}} = \gamma_2 \sum_{v \in \mathcal{O}} \left\| \begin{bmatrix} \rho_{v+\Delta v_x} - \rho_v \\ \rho_{v+\Delta v_y} - \rho_v \end{bmatrix} - \begin{bmatrix} I(\mathbf{u}_{v+\Delta v_x}) - I(\mathbf{u}_v) \\ I(\mathbf{u}_{v+\Delta v_y}) - I(\mathbf{u}_v) \end{bmatrix} \right\|_2^2, \tag{6}$$

where $\mathcal{O}$ is the set of visible points located on the mesh surface involved in the optical flow constraint, and $\mathbf{u}_v = P(\mathbf{F_T}(\mathbf{B}_v, \mathbf{w}))$. Δv_x is a 3D displacement along the surface such that the surface point $v + \Delta v_x$ maps to the texture pixel immediately above the one corresponding to point v; analogously, $v + \Delta v_y$ maps to the texture pixel on the right. ρ_v is the grayscale value for the point v extracted from the albedo texture, and $I(\mathbf{x})$ is the grayscale color extracted from the image at location $\mathbf{x}$.

We apply an ℓ_1-norm regularization on the blendshape coefficients using

$$E_{\text{sparse}} = \gamma_3 \|\mathbf{w}\|_1. \tag{7}$$

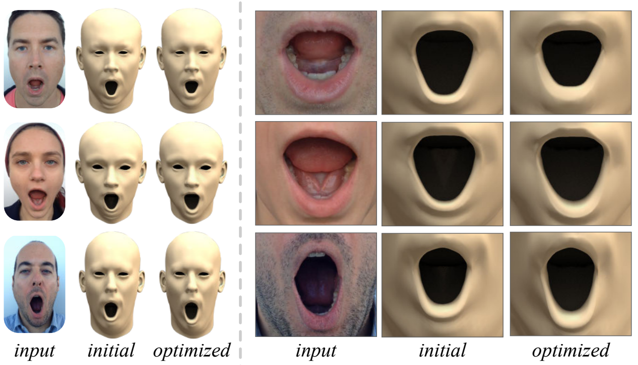


Figure 9: *Optimizing blendshapes is essential to accurately represent user-specific expressions. The initial blendshapes computed with deformation transfer (middle column) are registered towards the input images (left column). The resulting optimized blendshapes (right column) faithfully capture expression asymmetries.*

This sparsity-inducing term stabilizes the tracking and avoids too many blendshapes being activated with a small weight. Compared to the more common ℓ_2 regularization, this better retains the expression semantics of the blendshape model and thus simplifies tracking and retargeting as shown in [Bouaziz et al. 2013]. Similar to [Weise et al. 2011], we alternate between optimizing for the rigid transformation $\mathbf{T}$ and the blendshape weights $\mathbf{w}$.

Modeling. After solving for the tracking parameters, we keep these fixed and optimize for the vertex positions of the blendshapes. We again use facial features and optical flow leading to

$$\arg\min_{\mathbf{B}} E_{\text{feature}} + E_{\text{flow}} + E_{\text{close}} + E_{\text{smooth}}. \tag{8}$$

The closeness term penalizes the magnitude of the deformation from the initial blendshapes $\mathbf{B}^*$ created using deformation transfer:

$$E_{\text{close}} = \gamma_4 \|\mathbf{B} - \mathbf{B}^*\|_F^2. \tag{9}$$

The smoothness term regularizes the blendshapes by penalizing the stretching and the bending of the deformation:

$$E_{\text{smooth}} = \gamma_5 \|\mathbf{G}(\mathbf{B} - \mathbf{B}^*)\|_F^2 + \gamma_6 \|\mathbf{L}(\mathbf{B} - \mathbf{B}^*)\|_F^2. \tag{10}$$

In contrast to the tracking optimization of Equation 4 that is performed separately for each frame, the blendshape modeling optimization is performed jointly over the whole sequence. Tracking and modeling are iterated 20 times for all our examples.

Geometric Refinement. The blendshape modeling optimization from 2D images is effective for recovering the overall shape of the user-specific facial expressions (see Figure 9). We further improve the accuracy of the blendshapes using a 3D refinement step. For this purpose we extract one depth map per frame using a photometric approach [Kemelmacher-Shlizerman and Basri 2011; Wu et al. 2014]. The input video is downsampled by a factor of 8 to a resolution of 150 × 240 pixels in order to capture only the medium-scale details corresponding to the mesh resolution of the blendshape model (see Figure 10). Fine-scale detail recovery will be discussed in Section 5.2.

For each video frame we rasterize the tracked face mesh recovered during the blendshape optimization to obtain the original 3D location $\bar{\mathbf{p}}_{ij}$ in camera space and the grayscale albedo value ρ_{ij} of each pixel $\{i, j\}$. We compute smooth interpolated positions using cubic Bézier triangles on the face mesh refined with two levels of Loop subdivision. To create the perspective displacement map, we apply the displacement along the view rays $\frac{\bar{\mathbf{p}}_{ij}}{\|\bar{\mathbf{p}}_{ij}\|_2}$. Therefore, the new 3D point location $\mathbf{p}_{ij}$ of the pixel $\{i, j\}$ can be expressed as

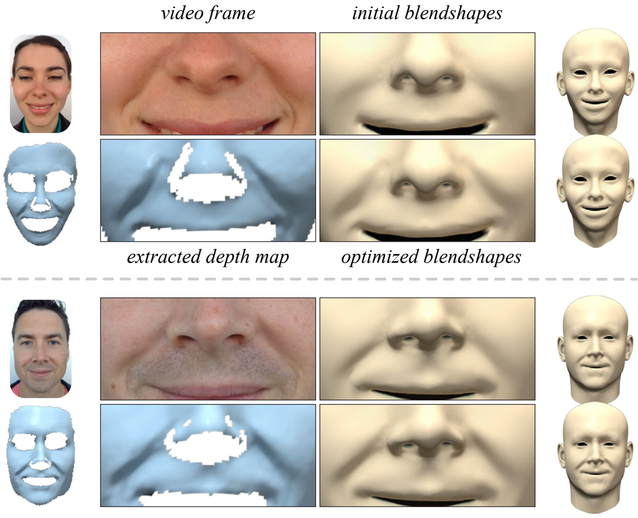


Figure 10: *Geometry refinement adds further nuances to the reconstructed blendshapes. For each frame of the video sequence we extract a depth map using shape-from-shading that serves as a constraint for the refinement optimization.*

$$\mathbf{p}_{ij} = \bar{\mathbf{p}}_{ij} + d_{ij} \frac{\bar{\mathbf{p}}_{ij}}{\|\bar{\mathbf{p}}_{ij}\|_2}, \tag{11}$$

where d_{ij} is the displacement value for this pixel. The normal at that pixel can then be estimated as

$$\mathbf{n}_{ij} = \frac{1}{N_{ij}} (\mathbf{p}_{i+1,j} - \mathbf{p}_{ij}) \times (\mathbf{p}_{i,j+1} - \mathbf{p}_{ij}), \tag{12}$$

where $N_{ij} = \|(\mathbf{p}_{i+1,j} - \mathbf{p}_{ij}) \times (\mathbf{p}_{i,j+1} - \mathbf{p}_{ij})\|_2$. Let $\mathbf{d}$ be a vector that stacks all the displacements d_{ij} and $\mathbf{y}$ be the vector of spherical harmonics coefficients. To reconstruct the displacement map we optimize

$$\min_{\mathbf{d},\mathbf{y}} \sum_{ij} \left\| \begin{bmatrix} \rho_{i+1,j}\mathbf{s}_{i+1,j} - \rho_{ij}\mathbf{s}_{ij} \\ \rho_{i,j+1}\mathbf{s}_{i,j+1} - \rho_{ij}\mathbf{s}_{ij} \end{bmatrix} - \begin{bmatrix} c_{i+1,j} - c_{ij} \\ c_{i,j+1} - c_{ij} \end{bmatrix} \right\|_2^2 + \\ + \mu_1 \|\mathbf{d}\|_2^2 + \mu_2 \|\mathbf{G}\mathbf{d}\|_2^2 + \mu_3 \|\mathbf{L}\mathbf{d}\|_2^2 \tag{13}$$

over $\mathbf{d}$ and $\mathbf{y}$, where c_{ij} is the grayscale value at pixel $\{i, j\}$ of the input frame and $\mathbf{s}_{ij} = \mathbf{y}^T \phi(\mathbf{n}_{ij})$. Similar to Equation 3, we regularize the displacements to be smooth and of low magnitude. To solve this optimization we alternately minimize Equation 13 over $\mathbf{y}$ by solving a linear system with fixed normals initialized from the original mesh, and over $\mathbf{d}$ with fixed weights $\mathbf{y}$ using a Gauss-Newton method. The depth and normal maps are then computed from the displacement maps using Equation 11 and Equation 12, respectively.

After extracting the depth and normal maps, we use a non-rigid registration approach to refine the blendshapes. We formulate a registration energy

$$E_{\text{reg}} = \sum_{v \in \mathcal{V}} \|\mathbf{n}_v^T (\mathbf{F}_{\mathbf{T}}(\mathbf{B}_v, \mathbf{w}) - \mathbf{p}_v)\|_2^2, \tag{14}$$

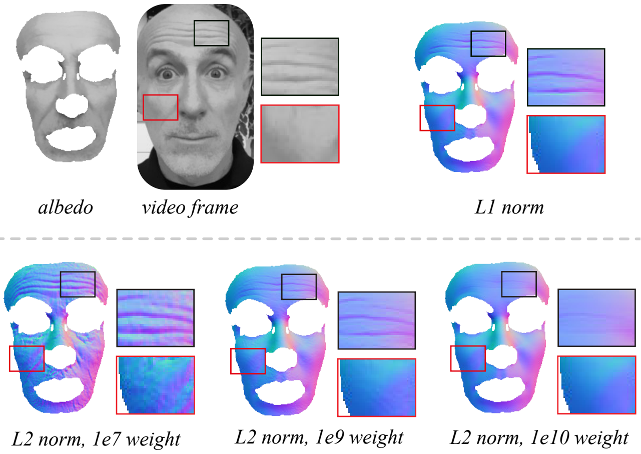


Figure 11: *Our reconstruction of detail maps uses ℓ_1-norm optimization to separate salient features such as the wrinkles on the forehead from noise. The lower row shows that ℓ_2 optimization with a low smoothness weight retains too much noise (left), while increasing the smoothness weights blurs out salient features (right).*

where $\mathcal{V}$ is the set of blendshape vertices, $\mathbf{p}_v$ is the closest point of $\mathbf{F}_\mathbf{T}(\mathbf{B}_v, \mathbf{w})$ on the depth map, and $\mathbf{n}_v$ is the normal at that point. This energy is optimized over the blendshapes $\mathbf{B}$ jointly over the whole sequence combined with a closeness and a smoothness energy (Equation 9 and Equation 10, respectively).

5.2 Reconstructing Detail Maps

In high-end studio systems, fine-scale details such as wrinkles are commonly directly encoded into the mesh geometry [Beeler et al. 2010; Garrido et al. 2013]. However, this requires a very fine discretization of the mesh which may not be suitable for realtime animation and display. Instead, we create a set of detail maps in an offline optimization to enable realtime detail synthesis and rendering at animation runtime.

Similar to the geometric refinement step, we extract one depth map per frame. This time the input video is downsampled 4 times to a resolution of 270 × 480 in order to keep small-scale details while reducing noise. To reconstruct sharp features we modify Equation 13 by replacing the ℓ_2 norm in the smoothness energies by an ℓ_1 norm. The ℓ_1 norm has been widely employed for image processing tasks such as denoising [Chambolle et al. 2010] as it allows preserving sharp discontinuities in images while removing noise. To solve the ℓ_1 optimization, Gauss-Newton is adapted using an iterative reweighing approach [Chartrand and Yin 2008]. The normal maps are then computed from the displacement maps using Equation 12. Figure 11 shows a visualization of the effect of the ℓ_1-norm in the extraction of the detail maps.

After extracting normals, we compute ambient occlusion maps by adapting the disk based approach proposed in [Bunnell 2005] to texture space, where we directly estimate ambient occlusion coefficients from the extracted normal and displacement maps. For each pixel p we calculate the ambient occlusion value $\mathrm{ao}(p)$ by sampling a set $\mathcal{S}_p$ of nearby pixels such that

$$\mathrm{ao}(p) = 1 - \sum_{k \in \mathcal{S}_p} \left(1 - \frac{1}{\sqrt{\frac{1}{\|\mathbf{v}_{pk}\|_2} + 1}}\right) \frac{\sigma(\mathbf{v}_{pk}^T \mathbf{n}_p)\sigma(\mathbf{v}_{pk}^T \mathbf{n}_k)}{|\mathcal{S}_p|}, \tag{15}$$

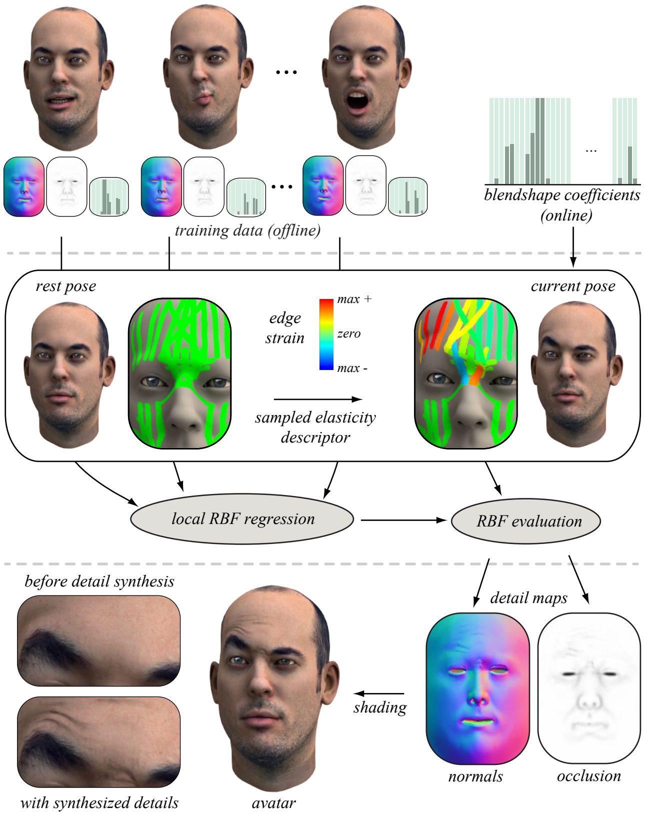


Figure 12: *On-the-fly detail synthesis. Blendshape coefficients drive the reconstructed face rig during realtime animation. For each animated expression, pose-specific details are synthesized using an RBF regressor that is trained with the detail maps reconstructed offline during dynamic modeling. The RBF function is evaluated based on deformation strains measured on a sparse set of edges (colored).*

where $\sigma(x)$ clamps x between 0 and 1, $\mathbf{n}_k$ is the normal at pixel k of the normal map, and $\mathbf{v}_{pk}$ is the vector between the 3D locations of pixels p and k reconstructed using the displacement maps.

6 Animation

The dynamic reconstruction stage provides us with a user-adapted blendshape model and a set of high-resolution detail maps containing normal and ambient occlusion maps that correspond to the recorded expressions of the video sequence. The blendshape representation allows for simple and intuitive animation. Blendshape coefficients can be directly mapped to animation controllers for keyframe animation or retargeted from face tracking systems (see also Figure 19). To augment the blendshape rig, we synthesize dynamic details on the fly by blending the reconstructed detail maps of the dynamic modeling stage using a local strain measure evaluated on the posed blendshape meshes (see Figure 12).

Detail Map Regression. Our detail synthesis method is inspired by the approach of [Bickel et al. 2008] that links edge strain to a displacement function. In contrast, we learn a mapping between edge strain and normal and ambient occlusion maps which facilitates more efficient detail synthesis using GPU shaders.
In a preprocessing stage, we train a radial basis function (RBF) regressor using the detail maps extracted for each frame of the

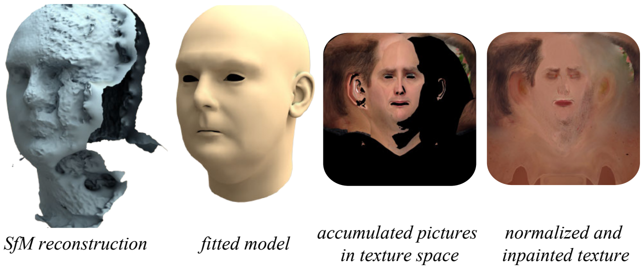

SfM reconstruction
fitted model
accumulated pictures in texture space
normalized and inpainted texture

Figure 13: *Degrading input data, in this case missing images on the cheek of the captured user, can lead to lower accuracy in the reconstructed static pose and texture artifacts produced by the inpainting algorithm (c.f. with Figure 5).*

tracked sequences and a strain measure computed on a sparse set of feature edges $\mathcal{E}$ defined on the mesh (see Figure 12). We compute the strain value of an edge $e \in \mathcal{E}$ as $f_e = (\|\mathbf{e}_1 - \mathbf{e}_2\|_2 - l_e)/l_e$, where $\mathbf{e}_1$ and $\mathbf{e}_2$ are the positions of the edge endpoints and l_e is the edge rest length. We then learn the coefficients w of an RBF regressor independently for each layer of the detail map. The regression for each pixel $\{i, j\}$ of a particular layer is formulated as

$$\mathbf{l}_{ij}(\mathbf{f}) = \sum_{k\in\mathcal{K}} \eta_k \varphi\Big(\|\mathbf{D}_{ij,k}^{\frac{1}{2}}(\mathbf{f} - \mathbf{f}_k)\|_2\Big), \tag{16}$$

where $\mathcal{K}$ is a set of selected keyframes, $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_k]$ are the RBF weights, and $\mathbf{f} = [f_1, \ldots, f_{|\mathcal{E}|}]^T$ is a vector stacking the strain f of all feature edges. We employ the biharmonic RBF kernel $\varphi(x) = x$ in our implementation. To localize the strain measure, we integrate for each keyframe a per-pixel diagonal matrix $\mathbf{D}_{ij,k} = \text{diag}(\alpha_{ij,k,1}, \ldots, \alpha_{ij,k,|\mathcal{E}|})$. Dropping the index $_{ij,k}$ for notational brevity, we define the weight α_e for each edge $e \in \mathcal{E}$ based on the distance of the pixel $\{i, j\}$ with 3D position $\mathbf{p}_{ij}$ to the edge e as

$$\alpha_e = \frac{\bar{\alpha}_e}{\sum_{l\in\mathcal{E}} \bar{\alpha}_l} \quad \text{with} \quad \bar{\alpha}_e = \exp^{-\beta(\|\mathbf{p}_{ij}-\mathbf{e}_1\|_2 + \|\mathbf{p}_{ij}-\mathbf{e}_2\|_2 - l_e)}. \tag{17}$$

The parameter β controls the drop-off. The localization spatially decouples the keyframes to avoid global dependencies and facilitates independent detail synthesis for different regions of the face. The RBF weights $\mathbf{w}$ are trained by minimizing the reconstruction error to the frames of the tracked sequences. The keyframes are selected greedily by sequentially adding the frame with maximum reconstruction error.

Detail Synthesis. The trained RBF regressor can now be used for detail synthesis during animation. The face rig is driven by blendshape coefficients. For the posed mesh, we compute the strain vector of the feature edges and evaluate Equation 16 to create new detail maps. The synthesized normal and ambient occlusion maps are then applied in the pixel shader.

7 Evaluation

We applied our dynamic 3D avatar reconstruction pipeline on a variety of subjects as shown in Figures 1, 16, and 18. For all subjects, we use around 80 images for the static reconstruction and less than 90 seconds of video for the dynamic modeling. These examples illustrate that our approach faithfully recovers the main geometric and texture features of the scanned subjects. We also show the effect of on-the-fly detail synthesis. The combination of per-pixel normals and ambient occlusion coefficients, which can both be integrated efficiently into per-pixel shading models, leads to further improvements on the appearance of the animated face rig (see also accompanying video).

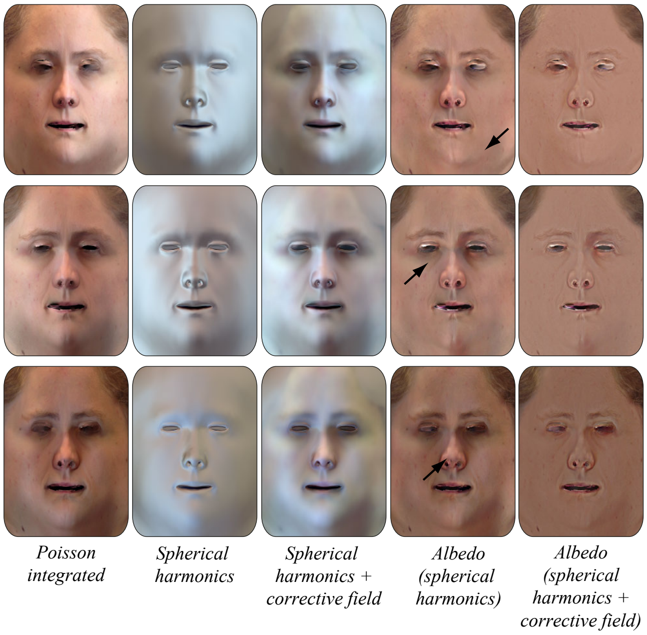

Poisson integrated
Spherical harmonics
Spherical harmonics + corrective field
Albedo (spherical harmonics)
Albedo (spherical harmonics + corrective field)

Figure 14: *Our lighting factorization approach successfully normalizes the light in three datasets captured under different illuminations. Notice how the corrective field aids in capturing shadows and specularities better than using spherical harmonics alone.*

Data Quality. We investigated how the output of our algorithm is affected by degrading input data quality. In particular, insufficient coverage of the face for the acquisition of the static model can lead to artifacts in the reconstruction. This lack of coverage can either result from the user failing to capture sufficiently many images, or from images being automatically discarded by the MVS algorithm due to, for example, excessive motion blur. Figure 13 illustrates how artifacts in the reconstructed point cloud can to a certain extent be compensated by the PCA regularization at the cost of diminished resemblance to the recorded user. Similarly, texture inpainting can fill missing texture information, but leads to visible artifacts in the reconstruction. While more sophisticated inpainting methods could alleviate these artifacts, we found that the simplest solution is to give visual feedback to the user to ensure adequate data capture. In all our experiments, users were able to record images of sufficient quality after being instructed about potential pitfalls in the acquisition, such as fast camera motion, changing illumination during capture, or insufficient coverage of the face.

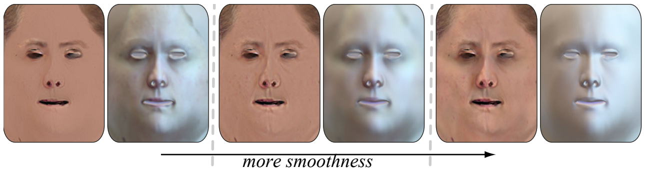


Figure 15: *The influence of the parameters in the albedo extraction. By increasing the smoothness of the corrective field more details are captured in the albedo.*

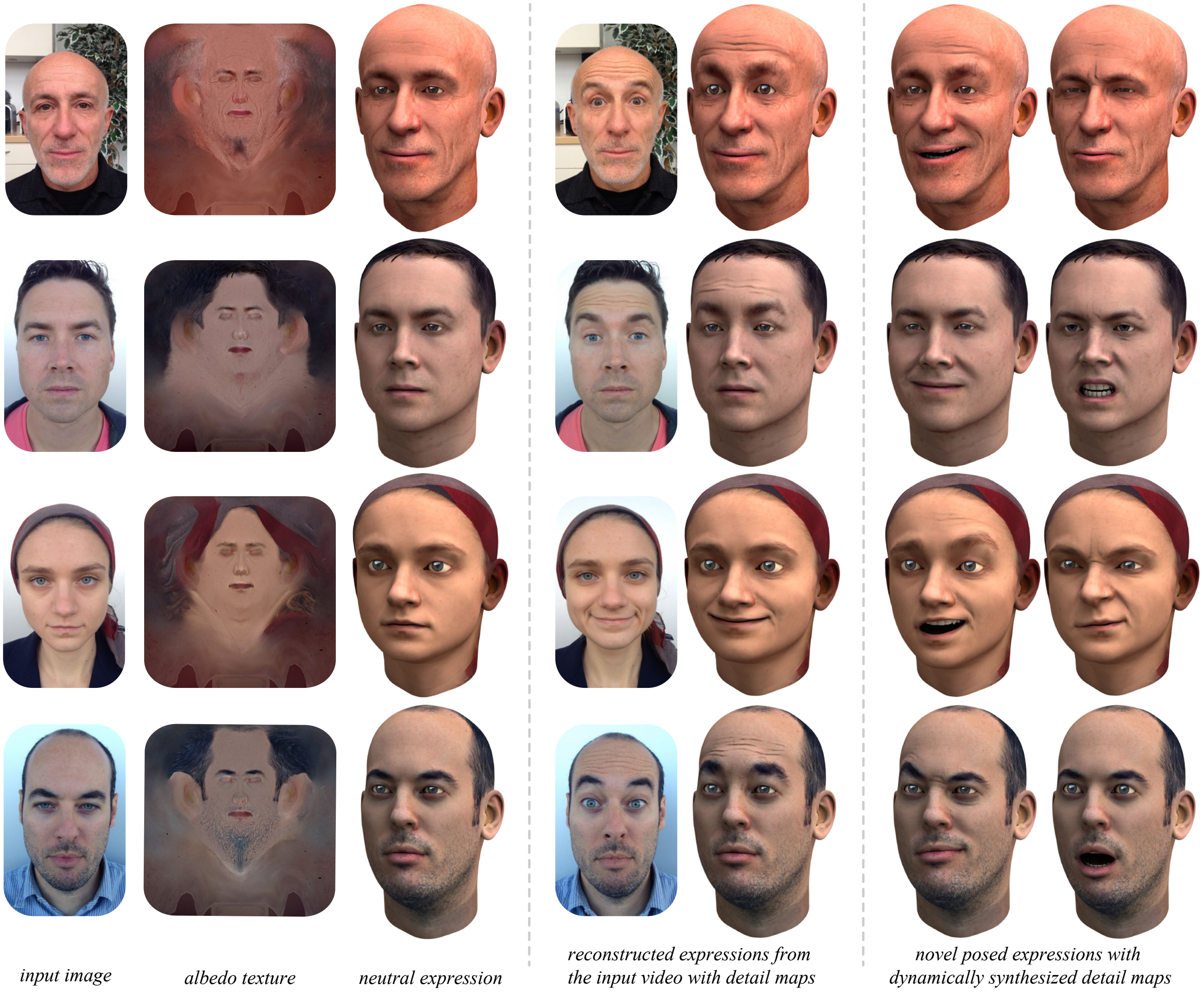

input image
albedo texture
neutral expression
reconstructed expressions from the input video with detail maps
novel posed expressions with dynamically synthesized detail maps

Figure 16: *Fully rigged 3D facial avatars of different subjects reconstructed with our method.*

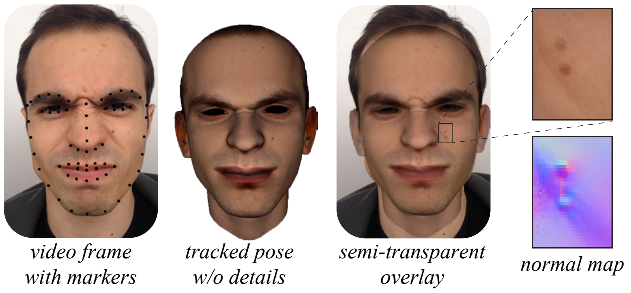


Figure 17: *Misalignments between video frames and the textured model can cause inconsistencies in the detail maps, here most visible at the mole on the cheek.*

Light Extraction. We ran experiments to verify the robustness of our albedo extraction given different lighting conditions, using the same set of parameters. Figure 14 displays the results for one of our test subjects, as well as the intermediate lighting estimations. Due to ambiguity introduced by the dependency of the captured skin color to the real skin color and the environment lighting, considering the skin color as the median color inside the face mask outputs slightly different results.

Furthermore, Figure 15 shows the behaviour of the albedo extraction under different parameters. We vary the smoothness of the corrective field in equation 3, regularizing the level of detail included into the extracted lighting.

Texture alignment. In general, the combination of feature energy and optical flow constraints in the tracking optimization of Equation 4 yields accurate alignments between the textured model and the video frames. However, in areas far away from the tracked features, such as the cheek, texture misalignments can occur that in turn can lead to reconstruction errors in the detail maps (see Figure 17). A possible solution to this problem is to adapt the optical flow energy of Equation 6 to incorporate additional texture features computed, for example, using SIFT descriptors.

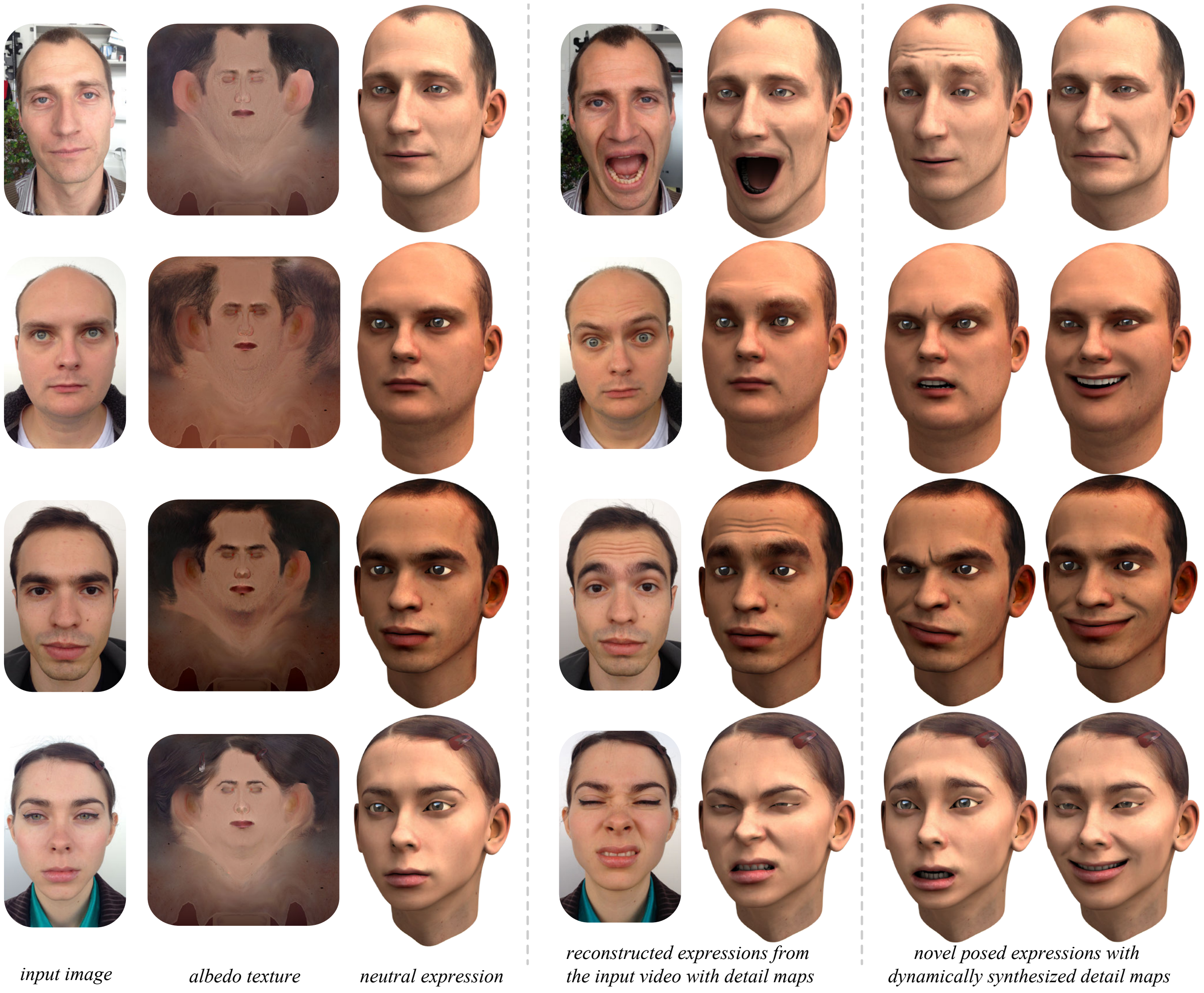

input image
albedo texture
neutral expression
reconstructed expressions from the input video with detail maps
novel posed expressions with dynamically synthesized detail maps

Figure 18: *Fully rigged 3D facial avatars of different subjects reconstructed with our method.*

Limitations. The simplicity of our acquisition setup implies a number of limitations in terms of scanning accuracy. As indicated above, limited spatial and temporal resolution of the camera, sensor noise, motion blur, or potentially insufficient illumination can adversely affect the reconstruction results.

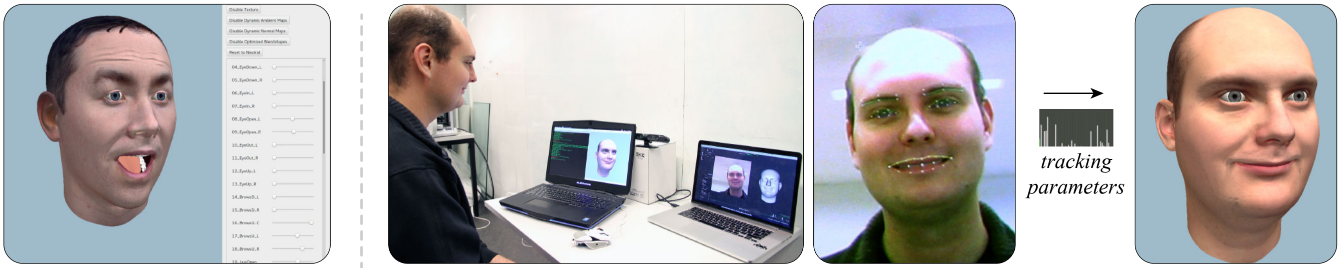
Our albedo factorization works well in casual lighting scenarios, but cannot fully handle high specularity or hard shadows in the acquired images. For such adverse lighting conditions, artifacts in the reconstructed albedo are likely to occur.

Blendshape models also have some inherent limitations. In particular, unnatural poses can be created for extreme expressions such as mouth very wide open, since a proper rotation of the lower lip is not represented in the linear model. Popular remedies, such as corrective shapes or a combination with joint-based rigging could potentially be integrated into our system, at the expense of a more complex tracking optimization.

A further limitation of our method is that we do not represent nor capture hair. This means that currently we can only reconstruct complete avatars for subjects with no hair or where the hair can be appropriately represented as a texture. More complex hair styles need to be treated separately outside our pipeline. Recent progress on hair capture [Hu et al. 2014] and realtime hair animation [Chai et al. 2014] offer promising starting points to further investigate this challenging problem. We also do not capture the teeth or tongue, but simply scale the template geometry appropriately.

Applications. Figure 19 shows reconstructed avatars in two application scenarios. Interactive character posing for keyframe animation is facilitated through direct control of the blendshape weights. Please see the additional material for a demo application that allows animating a reconstructed character in realtime. Alternatively, the character can be animated by transferring blendshape weights from a face tracking application. We use the commercial tool faceshift Studio that allows realtime streaming of blendshape coefficients. This demo illustrates the potential of our approach to bring personalized 3D avatars into consumer-level applications.

Future Work. Beyond addressing the limitations discussed above, we identify several interesting avenues for future work. Recent advances in RGB-D cameras show great promise of bringing active depth sensing into mobile devices such as tablets or phones. This



interactive character posing *Faceshift Studio live tracking* *realtime animation*

Figure 19: *Application demos utilizing our rigged avatars. Left: an interactive tool for posing the avatar by directly controlling blendshape weights. Right: The avatar is animated in realtime by streaming blendshape coefficients from a realtime tracking software.*

opens up interesting possibilities for new reconstruction algorithms that directly exploit the acquired depth maps.

Integrating sound seems a promising extension of our method, both on the reconstruction and the synthesis side. For example, an analysis of recorded speech sequences could guide the tracking and reconstruction of the blendshape model and detail maps. Avatars could also be driven by text-to-speech synthesis algorithms.

The possibility to transfer detail maps between subjects (see Figure 20) not only allows modifying the reconstructed avatars, but can potentially also simplify the acquisition process. Statistical priors for wrinkle formation could be learned from examples, given a sufficiently large database.

Further research is also required to answer important questions related to the perception of virtual avatars, such as: How well does an avatar resemble the user? How well does an animated avatar convey the true emotions of a tracked user? or What reactions does the virtual model evoke in online communication? We believe that these questions, along with the ultimate goal of creating complete, video-realistic 3D avatars with a consumer acquisition system lays out an exciting research agenda for years to come.

8 Conclusion

We have introduced a complete pipeline for reconstructing 3D face rigs from uncalibrated hand-held video input. While this minimalistic acquisition setup brings the creation of personalized 3D avatars into the realm of consumer applications, the limited input data quality also poses significant challenges for the generation of consistent and faithful avatars. Our solution combines carefully designed reconstruction priors, a two-scale dynamic blendshape representation, and advanced tracking and reconstruction algorithms to minimize the required user assistance while maximizing reconstruction quality. We believe that our solution provides an important first step towards realtime avatar-based interactions for the masses, which could have a significant impact on the way we communicate in virtual worlds.

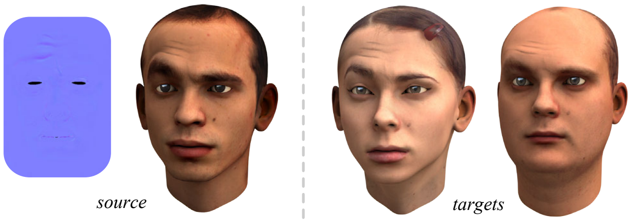


Figure 20: *Detail maps can easily be transferred between subjects thanks to the consistent parameterization of the blendshape meshes across all avatars.*

Acknowledgements

We thank Bailin Deng and Andrea Tagliasacchi for their valuable comments, Thibaut Weise, Nico Scapel and faceshift AG for their help and support. We are indebted to Andrea Tagliasacchi for helping with figures and with the video. We are grateful to Jean-Luc Benz, Emil Bryngelsson, Cristina Ţugui, Loïc Baboulaz, Remo Ziegler, Vlad Ureche, Arthur Giroux, and Anastasia Tkach for giving us the right to digitize them. This research is supported by grants from the Swiss Commision of Technology and Innovation 15105.1/580953 and the Swiss National Science Foundation 200021_153567/513497.

References

Alexander, O., Rogers, M., Lambeth, W., Chiang, M., and Debevec, P. 2009. Creating a photoreal digital actor: The digital emily project. In *Visual Media Production, 2009. CVMP'09. Conference for*.

Alexander, O., Fyffe, G., Busch, J., Yu, X., Ichikari, R., Jones, A., Debevec, P., Jimenez, J., Danvoye, E., Antionazzi, B., Eheler, M., Kysela, Z., and von der Pahlen, J. 2013. Digital ira: Creating a real-time photoreal digital actor. In *ACM SIGGRAPH 2013 Posters*.

Amberg, B., Blake, A., Fitzgibbon, A. W., Romdhani, S., and Vetter, T. 2007. Reconstructing high quality face-surfaces using model based stereo. In *ICCV*.

Beeler, T., Bickel, B., Beardsley, P., Sumner, B., and Gross, M. 2010. High-quality single-shot capture of facial geometry. *ACM Transactions on Graphics (TOG)*.

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 Trans. Graph.*.

Beeler, T., Bickel, B., Noris, G., Beardsley, P., Marschner, S., Sumner, R. W., and Gross, M. 2012. Coupled 3d reconstruction of sparse facial hair and skin. *ACM Trans. Graph.*.

Bérard, P., Bradley, D., Nitti, M., Beeler, T., and Gross, M. 2014. High-quality capture of eyes. *ACM Trans. Graph. 33*, 6 (Nov.), 223:1–223:12.

Bermano, A. H., Bradley, D., Beeler, T., Zünd, F., Nowrouzezahrai, D., Baran, I., Sorkine, O., Pfister, H., Sumner, R. W., Bickel, B., and Gross, M. 2014. Facial performance enhancement using dynamic shape space analysis. *ACM Trans. Graph.*.

BICKEL, B., LANG, M., BOTSCH, M., OTADUY, M. A., AND GROSS, M. H. 2008. Pose-space animation and transfer of facial details. In *Symposium on Computer Animation*.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LEVY, B. 2010. *Polygon Mesh Processing*. AK Peters.

BOUAZIZ, S., WANG, Y., AND PAULY, M. 2013. Online modeling for realtime facial animation. *ACM Trans. Graph.*.

BOUAZIZ, S., TAGLIASACCHI, A., AND PAULY, M. 2014. Dynamic 2d/3d registration. *Eurographics Tutorial*.

BUNNELL, M. 2005. Dynamic ambient occlusion and indirect lighting. *Gpu gems*.

CAO, X., WEI, Y., WEN, F., AND SUN, J. 2012. Face alignment by explicit shape regression. In *CVPR*.

CAO, C., WENG, Y., LIN, S., AND ZHOU, K. 2013. 3d shape regression for real-time facial animation. *ACM Trans. Graph.*.

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

CAO, C., WENG, Y., ZHOU, S., TONG, Y., AND ZHOU, K. 2014. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*.

CHAI, M., ZHENG, C., AND ZHOU, K. 2014. A reduced model for interactive hairs. *ACM Transactions on Graphics* (July).

CHAMBOLLE, A., CASELLES, V., CREMERS, D., NOVAGA, M., AND POCK, T. 2010. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery 9*, 263–340.

CHARTRAND, R., AND YIN, W. 2008. Iteratively reweighted algorithms for compressive sensing. In *Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on*, IEEE, 3869–3872.

DUDA, R. O., AND HART, P. E. 1972. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*.

FROLOVA, D., SIMAKOV, D., AND BASRI, R. 2004. Accuracy of spherical harmonic approximations for images of lambertian objects under far and near lighting. In *Computer Vision-ECCV 2004*.

FU, W. J. 1998. Penalized Regressions: The Bridge versus the Lasso. *J. Comp. Graph. Stat.*.

FURUKAWA, Y., AND PONCE, J. 2010. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*.

GARRIDO, P., VALGAERTS, L., WU, C., AND THEOBALT, C. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM Transactions on Graphics*.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. In *Proc. of ACM SIGGRAPH Asia*.

GONZALEZ, R. C., AND WOODS, R. E. 2006. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc.

GRAY, R. M. 2006. *Toeplitz and circulant matrices: A review*. now publishers Inc.

HU, L., MA, C., LUO, L., AND LI, H. 2014. Robust hair capture using simulated examples. *ACM Transactions on Graphics*.

HUANG, H., CHAI, J., TONG, X., AND WU, H.-T. 2011. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph. (Proc. SIGGRAPH)*.

JIMENEZ, J., ECHEVARRIA, J. I., OAT, C., AND GUTIERREZ, D. 2011. *GPU Pro 2*. AK Peters Ltd., ch. Practical and Realistic Facial Wrinkles Animation.

KEMELMACHER-SHLIZERMAN, I., AND BASRI, R. 2011. 3d face reconstruction from a single image using a single reference face shape. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.

LEWIS, J. P., ANJYO, K., RHEE, T., ZHANG, M., PIGHIN, F., AND DENG, Z. 2014. Practice and Theory of Blendshape Facial Models. In *EG - STARs*.

LI, H., ADAMS, B., GUIBAS, L. J., AND PAULY, M. 2009. Robust single-view geometry and motion reconstruction. *ACM Trans. Graph.*.

LI, H., YU, J., YE, Y., AND BREGLER, C. 2013. Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics*.

LI, J., XU, W., CHENG, Z., XU, K., AND KLEIN, R. 2015. Lightweight wrinkle synthesis for 3d facial modeling and animation. *Computer-Aided Design 58*, 0, 117 – 122. Solid and Physical Modeling 2014.

MA, W.-C., JONES, A., CHIANG, J.-Y., HAWKINS, T., FREDERIKSEN, S., PEERS, P., VUKOVIC, M., OUHYOUNG, M., AND DEBEVEC, P. 2008. Facial performance synthesis using deformation-driven polynomial displacement maps. *Proc. of ACM SIGGRAPH Asia*.

OAT, C. 2007. Animated wrinkle maps. In *ACM SIGGRAPH 2007 courses*.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph.*.

SARAGIH, J. M., LUCEY, S., AND COHN, J. F. 2009. Face alignment through subspace constrained mean-shifts. In *Computer Vision, 2009 IEEE 12th International Conference on*.

SARAGIH, J. M., LUCEY, S., AND COHN, J. F. 2011. Deformable model fitting by regularized landmark mean-shift. *Int. J. Comput. Vision*.

SHI, F., WU, H.-T., TONG, X., AND CHAI, J. 2014. Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM Trans. Graph. 33*, 6 (Nov.), 222:1–222:13.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph.*.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *Proc. of ACM SIGGRAPH Asia*.

VENKATARAMAN, K., LODHA, S., AND RAGHAVAN, R. 2005. A kinematic-variational model for animating skin with wrinkles. *Computers & Graphics*.

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. 2005. Face transfer with multilinear models.

WEISE, T., LI, H., VAN GOOL, L., AND PAULY, M. 2009. Face/off: Live facial puppetry. *ACM Trans. Graph.*.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. In *ACM SIGGRAPH 2011 Papers*.

WU, Y., KALRA, P., AND THALMANN, N. M. 1996. Simulation of static and dynamic wrinkles of skin. In *Proc. of IEEE Computer Animation*.

WU, C., ZOLLHÖFER, M., NIESSNER, M., STAMMINGER, M., IZADI, S., AND THEOBALT, C. 2014. Real-time shading-based refinement for consumer depth cameras. *ACM Trans. Graph. 33*, 6 (Nov.), 200:1–200:10.

WU, C. 2013. Towards linear-time incremental structure from motion. In *3D Vision, 2013 International Conference on*.

ZACH, C., POCK, T., AND BISCHOF, H. 2007. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*. Springer, 214–223.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: High-resolution capture for modeling and animation. In *ACM Annual Conference on Computer Graphics*.

Appendix - Implementation Details

Our software is implemented in C++ and parallelized on the CPU using OpenMP. We use the Eigen library for fast linear algebra computations and OpenCV for all the image processing operations. Our implementation runs on a laptop with an Intel Core i7 2.7Ghz processor, 16 GBytes of main memory, and an NVIDIA GeForce GT 650M 1024MB graphics card.

Static Modeling. The dense point cloud reconstruction with about 500k points takes 30 to 40 minutes for approximatively 80 pictures. The static modeling is then performed using the identity PCA model of [Blanz and Vetter 1999]. We use 50 PCA basis vectors to approximate the neutral expression. The registration problem is optimized with Gauss-Newton using the supernodal Cholmod sparse Cholesky factorization. The non-rigid registration takes approximatively 10 seconds.

For the static model, we generate a high-resolution albedo texture of 4096 × 4096 pixels. To efficiently solve the Poisson integration [Pérez et al. 2003] and to minimize Equation 3 over the corrective fields we use the Matlab Engine FFT. The parameters of Equation 3 are set to $\lambda_1 = 10^2$, $\lambda_2 = 10^{-3}$, and $\lambda_3 = 10^3$ for all the examples. The static texture is created in approximatively 5 minutes.

Dynamic Modeling. In our current implementation we employ a blendshape model of 48 blendshapes (see also accompanying material). The input videos are recorded at 30Hz with an average length of 1 minute. The videos are temporally downsampled to 3Hz prior to processing. We then apply a multiresolution approach with a four-level image pyramid. The optimization is first solved on the coarsest level, the solution is then propagated as an initialization to the next finer level until reaching the original resolution. The combined tracking and modeling optimization takes approximatively 60 seconds per frame. We perform the tracking optimization using a warm started shooting method [Fu 1998], and the modeling using Gauss-Newton. The parameters are set to $\gamma_1 = 10^{-1}$, $\gamma_2 = 10^{-2}$, $\gamma_3 = 10^4$, $\gamma_4 = 10^4$, $\gamma_5 = 10^2$, and $\gamma_6 = 10^8$ for all our examples.

To solve the shape-from-shading optimization we use Gauss-Newton. Symbolic sparse Cholesky factorization is used to improve performance as the sparsity pattern of the system matrix remains constant. Computation time is around 5 seconds for extracting a 150 × 240 depth map for the geometric refinement. The detail map extraction takes 25 seconds for a 1024 × 1024 normal map and another 5 seconds for the corresponding ambient occlusion map. The optimization weights are set to $\mu_1 = 10^6$, $\mu_2 = 10^3$, and $\mu_3 = 10^7$ for the geometric refinement, and $\mu_1 = 10^6$, $\mu_2 = 10^4$, and $\mu_3 = 10^6$ for the detail map extraction. The non-rigid refinement of the blendshape model is performed in about 60 seconds. The parameters are set to $\gamma_4 = 10^5$, $\gamma_5 = 1$, and $\gamma_6 = 10$.

Animation. We implemented the RBF evaluation on the GPU using a GLSL fragment shader. For 6 keyframes of size 1024 × 1024, and 44 strain edges the animation can be performed at realtime frame rates, i.e., 100 fps. The training of the RBF regressor takes approximatively 5 seconds. The parameter β is set to 150 for our meshes with an average edge length of 4.1 cm.

Manual User Interaction. From our trials, we concluded that about 15 minutes are needed to perform the manual feature corrections necessary for the static ($\sim 1-2$ minutes) and dynamic reconstructions ($\sim 7-15$ minutes). The additional video shows a complete example. The decision of using [Saragih et al. 2009] was based on code and documentation availability, and we believe that more recent and precise methods such as [Cao et al. 2013; Cao et al. 2012] could be used to reduce or eliminate the amount of manual corrections.