Face2Face:
Real-Time Facial Reenactment

(Face2Face: Übertragung von Gesichtsausdrücken in Echtzeit)

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg
zur
Erlangung des Grades Dr.-Ing.

vorgelegt von

Justus Philipp-Andrei Thies

aus Buchen (Odw.)

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung: 16. Oktober 2017
Vorsitzende des Promotionsorgans: Prof. Dr.-Ing. Reinhard Lerch
Gutachter: Prof. Dr. Günther Greiner
Prof. Dr. Christian Theobalt

Revision 1.00
©2017, Copyright Justus Philipp-Andrei Thies
justus.thies@fau.de
All Rights Reserved
Alle Rechte vorbehalten

In Memoriam Patris Mei
Dipl.-Ing. und Facharzt für Allgemeinmedizin
Heinz-Wolfgang Ernst Johann Thies (1950-2013)

Abstract

In this dissertation we show our advances in the field of 3D reconstruction of human faces using commodity hardware. Beside the reconstruction of the facial geometry and texture, real-time face tracking is demonstrated. The developed algorithms are based on the principle of analysis-by-synthesis. To apply this principle, a mathematical model that represents a face virtually is defined. In addition to the face, the sensor observation process of the used camera is modeled. Utilizing this model to synthesize facial imagery, the model parameters are adjusted, such that the synthesized image fits the input image as good as possible. Thus, in reverse, this process transfers the input image to a virtual representation of the face. The achieved quality allows many new applications that require a good reconstruction of the face. One of these applications is the so-called ”Facial Reenactment”. Our developed methods show that such an application does not need any special hardware. The generated results are nearly photo-realistic videos that show the transfer of the mimic of one person to another person. These techniques can for example be used to bring movie dubbing to a new level. Instead of adapting the audio to the video, which might also include changes of the text, the video can be post-processed to match the mouth movements of the dubber. Since the approaches that we show in this dissertation run in real-time, one can also think of a live dubber in a video teleconferencing system that simultaneously translates the speech of a person to another language.

The published videos of our projects in this dissertation led to a broad discussion in the media. On the one hand this is due to the fact that our methods are designed such that they run in real-time and on the other hand that we reduced the hardware requirements to a minimum. In fact, after some preprocessing, we are able to edit ordinary videos from the Internet in real-time. Amongst others, we impose a different mimic to faces of prominent persons like former presidents of the United States of America. This led inevitably to a discussion about trustworthiness of video material, especially from unknown source. Most people did not expect that such manipula-

tions are possible, neglecting existing methods that are already able to edit videos (e.g. special effects in movie productions). Thus, beside the advances in real-time face tracking, our projects raised the awareness of video manipulation.

Acknowledgments

This dissertation recaps the last three years of my work at the Computer Graphics Group of the University Erlangen-Nuremberg. I want to thank my two supervising professors Günther Greiner and Marc Stamminger for their support; they gave me the freedom to follow my own research interests. I also want to stress our great and successful cooperation with the Graphics, Vision & Video Group at the Max Planck Institute Informatik in Saarbrücken. Prof. Christian Theobalt gave me the feeling of being a part of his group; I want to thank him for the internships at his group and the innumerous Skype meetings - full of insightful discussions. In particular, I want to thank Michael Zollhöfer, whom I would like to call my mentor; I learned a lot from him. In years of collaboration, he still impresses me with his ideas and his knowledge about optimization and GPU-programming. I would also like to thank Matthias Nießner, the perfectionist and propulsive power in our team, who caused me sometimes quite a headache. The discussions with him, Michael, Christian and Marc always pushed the results of our projects to a remarkably higher level. I had a lot of fun working with them and I'm looking forward to continue it.

The time at the computer graphics chair was extraordinary; Günther Greiner and Marc Stamminger did a perfect job in ensuring an excellent and creative environment. To work and to party with my colleagues was marvelous. I want to thank Frank Bauer, Matteo Colaianni, Roberto Grosso, Matthias Innmann, Benjamin Keinert, Franziska Kranz, Vanessa Lange, Alexander Lier, Magdalena Martinek, Michael Martinek, Jana Martschinke, Falko Matern, Dominik Penk, Jürgen Pröll, Kai Selgrad, Christian Siegl, Lucas Thies and Christoph Weber for the wonderful time that I had here in Erlangen and for volunteering as demo actors in my projects.

Last but not least, I am thankful for all the support that I got from my family. I dedicate this dissertation to my father who aroused my interest and passion in science.

Contents

0 Introduction and Basics 1

1 Motivation 3

2 Analysis-by-Synthesis 7
  2.1 Parametric Face Model 8
    2.1.1 Statistical Model - Morphable Model 8
    2.1.2 Expression Augmentation - Blendshapes 12
    2.1.3 Illumination - Spherical Harmonics 15
  2.2 Sensors 18
    2.2.1 Commodity RGB Cameras 18
    2.2.2 Depth Cameras 21
  2.3 Optimization 24
    2.3.1 Non-Linear Optimization 24
    2.3.2 GPU-based Analysis-by-Synthesis Optimization Framework 26

3 Contribution and Outline 29

I Real-time Expression Transfer for Facial Reenactment 33

4 Introduction 35

5 Related Work 39
    5.0.1 Facial Performance Capture 39
    5.0.2 Face Re-targeting and Facial Animation 39
    5.0.3 Face Replacement in Video 40

6 Overview 43

7 Synthesis of Facial Imagery 45
7.0.1 Parametric Face Model 45
7.0.2 Illumination Model 46
7.0.3 Image Formation Model 47

8 Parametric Model Fitting 49
8.0.1 Input Data 49
8.0.2 Implementation of the Image Formation Model . . 50
8.0.3 Energy Formulation 50

9 Parallel Energy Minimization 55
9.0.1 Core Solver 55
9.0.2 Memory Efficient Solution Strategy on the GPU . . 56
9.0.3 Initialization of Identity and Albedo 58

10 Facial Reenactment and Applications 59
10.0.1 Live Reenactment Setup 59
10.0.2 Expression Transfer 60
10.0.3 Wrinkel-Level Detail Transfer 60
10.0.4 Final Compositing 61
10.0.5 Re-Texturing and Re-Lighting Applications 63

11 Results 65
11.0.1 Real-time Facial Performance Capture 65
11.0.2 Facial Reenactment 72
11.0.3 Virtual Mirror 72

12 Conclusion and Discussion 75

II Face2Face: Real-time Face Capture and Reenactment of RGB Videos 77

13 Introduction 79

14 Related Work 83

15 Synthesis of Facial Imagery **87**

16 Energy Formulation **89**

17 Data-parallel Optimization Strategy **91**

18 Non-Rigid Model-Based Bundling **93**

19 Expression Transfer **95**
19.1 Mouth Retrieval 96

20 Results **99**
20.1 Limitations 102
20.2 Conclusion 105

21 Appendix: ”Face2Face: Real-time Face Capture and Reenactment of RGB Videos” **107**
21.1 Optimization Framework 107
21.2 Non-rigid Bundling 108
21.3 Reenactment Evaluation 110

III FaceVR: Real-Time Facial Reenactment and Eye Gaze Control in Virtual Reality **113**

22 Introduction **115**

23 Related Work **119**

24 Hardware Setup **123**
24.0.1 Head-Mounted Display for the Source Actor 124
24.0.2 3D Stereo Rig for Target Actor Tracking 125

25 Synthesis of Facial Imagery **127**

26 An Image-based Eye and Eyelid Model **129**
26.0.1 Training Data Generation 129

26.0.2 Random Ferns for Eye-gaze Classification 131
26.0.3 Hierarchical Eye-gaze Classification 132
26.0.4 Temporal Stabilization of Classification Results . . 132

27 Parametric Model Fitting 135
27.0.1 Target Actor Energy Formulation 135
27.0.2 Source Actor Tracking Objective 137
27.0.3 Data-Parallel Optimization 140

28 Face Rig and Compositing 141

29 Results 145
29.0.1 Gaze Correction for Video Conferencing 145
29.0.2 Gaze-aware Facial Reenactment 146
29.0.3 Self-Reenactment for VR Video Conferencing . . . 148
29.0.4 Evaluation of Eye Tracking Accuracy 149
29.0.5 Evaluation of Face Identity Estimation 151
29.0.6 Evaluation of Face Tracking Accuracy 152

30 Conclusion and Discussion 157

31 Summary and Outlook 159

Bibliography 163

List of Figures

1.1 Virtual Augustus bust 5

2.1 Standard deviation of the shape dimension 9
2.2 Standard deviation of the albedo dimension 9
2.3 Statistical face model 10
2.4 Global support of the principle components 11
2.5 Blendshapes 12
2.6 Non-rigid registration 13
2.8 Alignment mask 13
2.7 Deformation Transfer 14
2.9 Local support of the delta blendshapes 15
2.10 Spherical harmonics 17
2.11 Pinhole camera model 19
2.12 3D-2D Correspondence 20
2.13 Stereo Webcam 21
2.14 Xtion - RGBD camera 22
2.15 RGBD camera data 22

4.1 Our live facial reenactment setup 35

6.1 Our live facial reenactment pipeline 43

8.1 Overview of our real-time fitting pipeline 49

9.1 Structure of the Jacobian 57
9.2 Convergence of the Gauss-Newton solver 58

10.1 Detail transfer 60
10.2 Final compositing 62
10.3 Re-texturing and re-lighting 63

11.1 Tracking accuracy 67

11.2 Stability under lighting changes 68
11.3 Stability under head motion 68
11.4 Analysis of the different data terms 70
11.5 Comparison to *FaceShift* 71
11.6 State-of-the-art comparison 72
11.7 State-of-the-art comparison 73
11.8 Re-texturing and re-lighting a facial performance 73
11.9 Results of our reenactment system 74

13.1 Proposed online reenactment setup 79
13.2 Method overview 81

19.1 Mouth database 96

20.1 Comparison to Cao et al. [CHZ14] and Thies et al. [TZN*15] 100
20.2 Comparison to Shi et al. [SWTC14] 101
20.3 Comparison to *FaceShift* 102
20.4 Comparison to Garrido et al. [GVS*15] 103
20.5 Comparison to Thies et al. [TZN*15] 103
20.6 Results of our reenactment system 104

21.1 Jacobian Matrix 108
21.2 Bundling hierarchy 110
21.3 Comparison to Garrido et al. 111
21.4 Comparison to Dale et al. and Garrido et al. 111
21.5 Cross-Validation 112

22.1 Setup of *FaceVR* 115

24.1 Hardware setup 123

26.1 Eye calibration pattern 130
26.2 Comparison of fern classifiers 133

27.1 Comparison ArUco marker 137
27.2 Building a personalized stereo avatar 139

29.1 Gaze correction 146
29.2 Gaze-aware facial reenactment 147
29.3 Self-Reenactment 148
29.4 Gaze-aware stereo facial reenactment 150
29.5 Comparison to Tobii EyeX 151
29.6 Accuracy of reconstructed identity 151
29.7 Stereo alignment 152
29.8 Ground truth comparison 154

List of Tables

2.1 Spherical harmonics basis functions 16

11.1 Run times for three of the sequences of Fig. 9.2 66

20.1 Avg. run times for the three sequences of Fig. 20.6 99

29.1 Comparison of the tracking accuracy 153

PART 0

Introduction and Basics

CHAPTER 1

Motivation

Nowadays, computing devices are omnipresent. To interact with the real world these devices are equipped with a bunch of sensors like cameras, motion sensors, fingerprint sensors and much more. A goal of these sensors in devices like Smart-phones is to improve the man-machine interaction. E.g. instead of typing a password the fingerprint of a person can be used as an identification characteristic. Or a camera can be used to track the eyes of a person, which can be used to analyze the user's behavior and focus. As a result of this analysis, specific content can be generated for the user, like hints with additional information. It can also be used to render the focused point on the screen with a higher resolution, while the other regions in the visual periphery are rendered with a reduced resolution. This technique is called foveated rendering [GFD*12], since it considers the structure and the visual acuity of the human eye.

Similar to a fingerprint, a camera can be used to identify the face of a person. But beside the identification task, the face gives much more information of the state of a person [Ekm82]. E.g. expressions give insides of the current emotions (e.g. surprise, anger or happiness). Thus, based on the emotional state of a user, a computer is able to make other decisions and can adapt its content accordingly, e.g. adjust a playlist of music such that it fits to the mood of the user.

Beside the analysis of the user, cameras are also used to reconstruct the surrounding, especially reconstructing a three dimensional representation of the surfaces of objects in the scene. In particular, depth cameras are used for this purpose, e.g. in ”KinectFusion” [NIH*11] or in ”Real-time 3D Reconstruction at Scale Using Voxel Hashing” [NZIS13]. The three dimensional representation allows a variety of new applications in the field of Augmented Reality (AR) and Virtual Reality (VR). Objects are measured without a rule and modifications of the scene can be simulated virtually.

For example a virtual mirror that simulates different make-up or make-up suggestions [SRH*11]. One can also think of tailored fashion based on a three dimensional reconstruction of an individual, e.g. a customized glasses frame that exactly fits the head.

These applications have in common that they need a good (and probably dynamic) 3D reconstruction. This dissertation tackles the problem of reconstructing and tracking faces in 3D. To demonstrate the effectiveness of the developed algorithms, we do not only show tracking results, but also synthesize photo-realistic facial images. This allows us not only to demonstrate a virtual mirror, but also facial reenactment. Facial reenactment is a synonym of puppeteering another face. To this end, we reconstruct and track the faces of two actors, a source and a target actor. Using the reconstruction we transfer the expressions of the source actor to the target actor and re-render the manipulated face on top of the original video stream, resulting in a photo-realistic video.

The proposed facial tracking and reenactment has several use-cases. In movie productions it can be used as a video editing tool to change for example the expression of an actor in a scene. It can also be used to modify the appearance of a face in a post-process, e.g. changing the illumination situation. Another field in post-processing is the synchronization of speech to the video. If a movie is translated to another language, the movements of the mouth do not match the audio of the dubber. Nowadays, to match the video, the audio including the spoken text is adapted, which might result in loss of information. Using facial reenactment instead, the expressions of the dubber are transferred to the actor in the movie and thus, synchronizing audio and video. Since our reenactment approaches run in real-time it is also possible to setup a teleconferencing system with a live dubber that simultaneously translates the speech of a person to another language.

In contrast to state-of-the-art production setups that work with markers and complex camera setups, our systems presented in this dissertation only need commodity hardware without the need of markers. Our tracking results can also be used to animate virtual characters like the Augustus bust in Fig. 1.1. These virtual characters can be part of animation movies, but

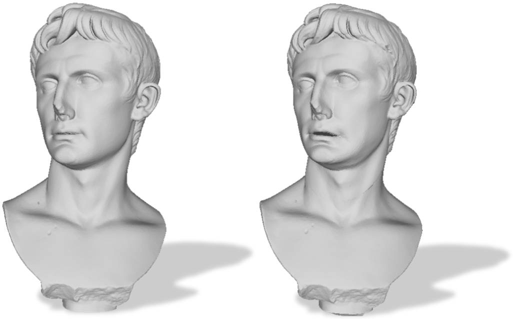


Figure 1.1: Reenacted virtual Augustus bust. Left the neutral pose of the Augustus bust, right the modified bust.

also in computer games. With the introduction of virtual reality glasses, also called head mounted displays (HMD's), the realistic animation of such virtual avatars, becomes more and more important for an immersive gameplay. We demonstrate in our FaceVR project (see Part III) that facial tracking is also possible if the face is mostly occluded by such an HMD. The project also paves the way to new applications like teleconferencing in VR including HMD removal.

Beside these consumer applications, social psychological researchers are interested in the reenactment system. For example, they want to analyze how the visual impression of a person biases the trustworthiness in a conversation. Thus, in such an experiment the voice and the message would stays the same, but the face would differ. You can also think of a training system that helps patients to train expressions after a stroke. To help surgeons, reconstruction and tracking methods are also very important in modern medicine. The reconstruction of a head can be used to plan a surgery virtually. Then, during the real surgery, the tracking will be used to guide a surgeon, considering the previously planned surgery or additional data like CT scans that are attached to the 3D reconstruction.

Our reconstruction and photo-realistic re-rendering allows to manipulate videos in real-time. In addition with a voice imitator or a person specific voice synthesis, it allows to generate facial videos to defame people or to spread so-called ”fake-news”. The generation of such videos is already possible, but it is time consuming (c.f. movie production) and specialists are needed. The striking demonstrations of our reenactment systems teach the people to rethink the value of videos without proof of origin. Beside this effect, our reconstruction methods can be used to analyze the physical plausibility of an image (→ digital forensic / fraud detection). An important indicator whether an image is manipulated or not, is the consistency of the illumination. The techniques presented in this dissertation compute the illumination of the face region. To detect manipulations, this estimate can be compared to illumination estimations in other parts of the scene.

To summarize, our primary goal is to create a mathematical representation of a real world scenario. These models enable computers to reconstruct, understand, and interact with it. New technologies like Virtual Reality or Augmented Reality rely on such data. Better reconstructions lead to a more immersive experience. Our projects concentrate on the reconstruction of non-static faces, even in uncontrolled environments. Most existing real-time face trackers are based on sparse features and thus capture only a coarse face model. Our approach tries to use all available information of a captured image of a face, i.e. every pixel of the face. That’s why we call it a dense face tracker, which is the heart of all listed use-cases above. The presented face trackers follow the principle of analysis-by-synthesis which is described in the next chapter. It also shows the underlying models and assumptions. These fundamentals build the basis of the selected projects presented in Part I, Part II and Part III.

CHAPTER 2

Analysis-by-Synthesis

To reconstruct and track a face we tackle the inverse rendering problem. Inverse rendering is a field of computer vision that tries to invert the image formation process. A commonly applied scheme is the principle of analysis-by-synthesis. The idea of analysis-by-synthesis is to synthesize something (e.g. an image) such that it matches the observation as close as possible in an iterative manner [Koc93]. Thus, the result describes the synthesis of an observation. In our case the rendering of a face (including the geometry, albedo and the illumination of the face).

To run an analysis-by-synthesis approach, a model has to be defined to synthesize new data. In this dissertation we concentrate on facial imagery. The synthesis of facial images is described in Section 2.1 and Section 2.2. Depending on the observed information an analysis-by-synthesis approach analyzes the difference between the synthetic and the observed data. Based on this difference, a parameter update of the model is computed. Using the new parameters, a new synthetic image can be generated. These steps of synthesizing and analyzing are repeated until convergence. Convergence can be either measured in parameter space or in the residual of the fitting error (difference of the images). Often, you have to find a compromise between convergence and runtime, especially in the case of real-time applications where you only have a time-frame of $33ms$.

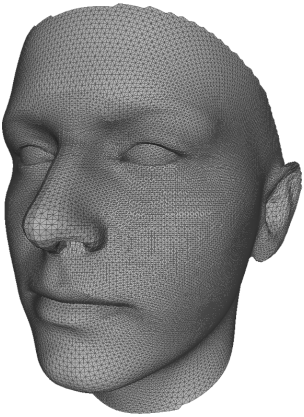
The analysis and the computation of a parameter update in each iteration results in an optimization problem. As most (non-linear) optimization problems, an analysis-by-synthesis approach heavily depends on the initial guess of parameters. Otherwise it might converge to a wrong solution (local minimum) or it might diverge. The used optimization strategies are described in Section 2.3 and more detailed information can be found in the chapters of the single projects of this dissertation.

2.1 Parametric Face Model

This section is about the parametric face model that is used in this dissertation. It manly consists of a statistical model that describes the shape and albedo of a human's face (see Sec. 2.1.1). On top of this so-called identity, we model expressions using a blendshape model (see Sec. 2.1.2). To render the face model we also need to incorporate illumination. Therefore, we apply a commonly used approximation of environment maps - spherical harmonics (see Sec. 2.1.3). The following sections show the details of these components.

2.1.1 Statistical Model - Morphable Model

To allow a reconstruction of a face based on incomplete or noisy data, we use a prior that models faces in a low dimensional space. This prior is based on the work of Blanz and Vetter [BV99]. Blanz and Vetter built a database of 200 scanned human faces using a laser scanner (for details see [BV99]).



Beside geometry they also captured the illumination corrected textures of the faces. Based on non-rigid template-fitting these scans are registered and aligned in a common coordinate system. The resulting faces share the same topology, but differ in geometry and albedo. The average mesh of the scanned faces is depicted in the figure on the right.The template mesh is a simple triangle mesh and consists of 53490 vertices and 106466 triangles. Beside the position, every vertex also stores an albedo value.

To reduce the dimensionality of the dataset a principle component analysis (PCA) is independently applied to geometry and albedo. The PCA computes the principle components of a dataset and the corresponding standard deviations. As one can see in Fig. 2.1 the standard deviation of the shape

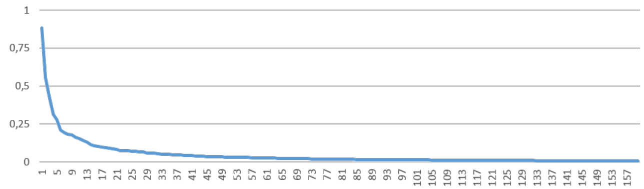


Figure 2.1: Standard deviation σ^{shape} of the first 160 principle components of the shape. The horizontal axis shows the index of the principle component, the vertical axis the standard deviation.

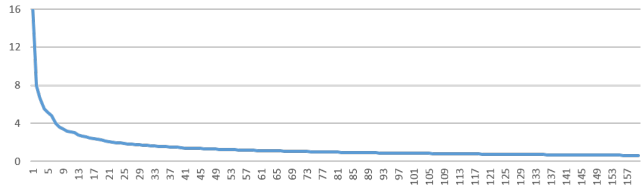


Figure 2.2: Standard deviation σ^{albedo} of the first 160 principle components of the albedo. The horizontal axis shows the index of the principle component, the vertical axis the standard deviation.

dimension σ^{shape} drops very quickly. The standard deviation of the albedo dimension σ^{albedo} (see Fig. 2.2) has a similar shape. We exploit this behavior to reduce the number of dimensionality, i.e., instead of using all 199 principle components of the dataset, we use a lower number of the principle components (e.g. 80 in Face2Face [TZS*16b]). Using this PCA model, new faces are synthesized via a linear combination of n principle components $\mathbf{S} \in \mathbb{R}^{3 \cdot 53490 \times n}$ plus the average face $\bar{S} \in \mathbb{R}^{3 \cdot 53490}$. Resulting in the mathematical description of face geometry:

$$Shape(\alpha) = \bar{S} + \mathbf{S}\alpha$$

The albedo of a face is described in the same way using the principle components $\mathbf{A} \in \mathbb{R}^{3 \cdot 53490 \times n}$ plus the average face albedo $\bar{A} \in \mathbb{R}^{3 \cdot 53490}$:

$$Albedo(\beta) = \bar{A} + \mathbf{A}\beta$$

The shape parameter vector α and the albedo parameter vector β describe the identity of a person, and, thus, are called identity parameters. Fig. 2.3 shows some synthetic faces that were generated using this model.

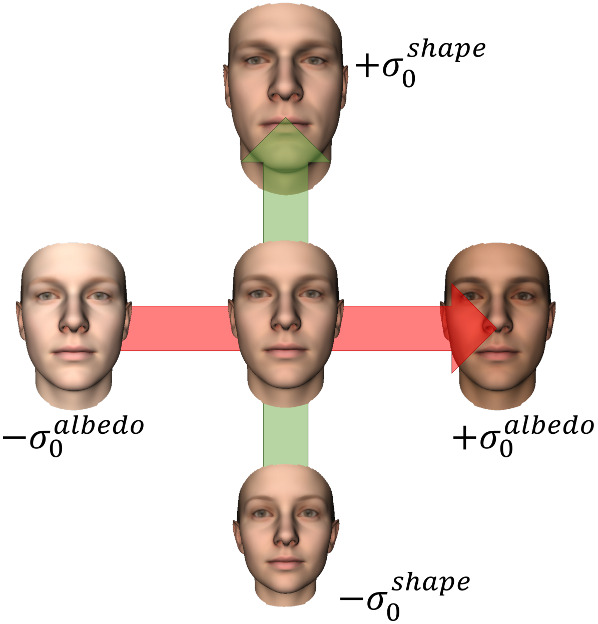


Figure 2.3: Statistical face model: the face in the middle shows the average face geometry and albedo. The red arrow shows how the albedo is changed if the first principle component of the albedo times the std. dev. is added (right) and subtracted (left). The green arrow illustrates the shape dimension. The face on the top is the result of adding the first component of the shape times the std. dev. to the average face, the bottom face shows the result when this principle component is subtracted from the average face.

The standard deviations allow us to estimate how likely a face with certain shape and albedo parameter is. We use a measurement $R(\alpha, \beta)$ that sets the parameters in relation to their std. dev.:

$$R(\alpha, \beta) = \sum_{i=1}^{n} \left\| \frac{\alpha_i}{\sigma_i^{shape}} \right\|^2 + \left\| \frac{\beta_i}{\sigma_i^{albedo}} \right\|^2$$

This measurement is used as a regularizer in this dissertation to prevent degeneration of faces during reconstruction. An important property of the PCA model is the global support of the principle components. The global support of the principle components allows us to estimate regions that are unobserved, based on the regions that are visible. As can be seen in Fig. 2.4 the single principle components influence the whole model. Another inside that we can read out of this figure is that the first principle component is the smoothest principle component. The smoothness is reduces with increasing index of the principle component (i.e. the last principle components mainly consist of higher frequencies that stem from noise).

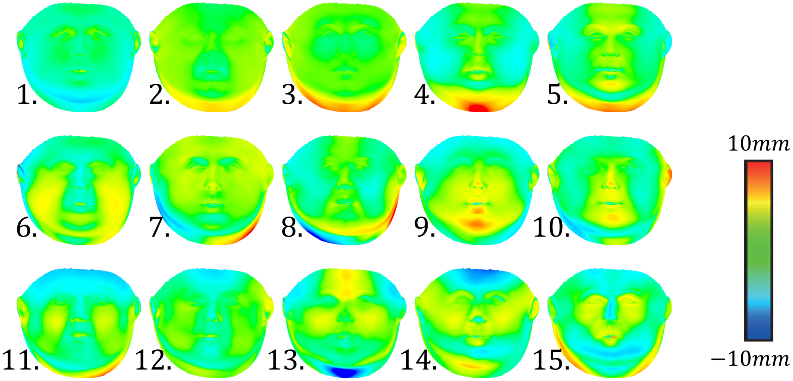


Figure 2.4: The principle components have global support. Here we show the first 15 principle components of the shape projected onto the normal of the average face. These distances are visualized in the texture space of the face model.

2.1.2 Expression Augmentation - Blendshapes

To bring the neutral pose of a reconstructed face to life, we use so called blendshapes. Blendshapes are meshes that share the same topology but have a different geometry, i.e., another pose or in our case expression (see Fig. 2.5).

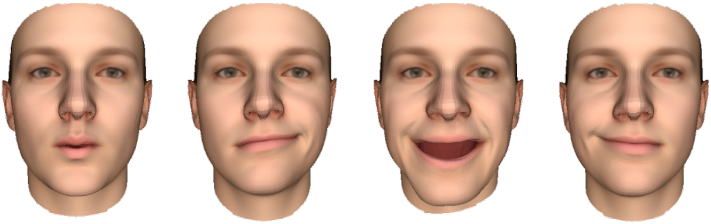


Figure 2.5: Blendshapes: set of example poses.

These blendshape meshes are blended together to form a new expression. Since the expressions of a face and the resulting deformations are mostly linear, we use a linear combination of these blendshape meshes. The linear coefficients are called blendshape weights. The statistical model of Blanz and Vetter does not provide such blendshapes. Thus, we built our own expression blendshapes, which is described in the following. There is a couple of possibilities to generate blendshapes. In the film industry blendshapes of characters are typically created by artists who deform the neutral mesh manually. This task is very time consuming and needs skilled artists. An advantage is that this allows to animate characters that do not exist in reality. In contrast, if the character is an existing person, blendshapes can be reconstructed from real data. Alexander et al. [ARL*09] used a light stage in their "Digital Emily" project to reconstruct 33 facial expressions of an actress. These 33 expressions are based on the Facial Action Coding System (FACS) [EF78]. Similar Cao et al. [CWZ*14] built a database called FaceWarehouse where faces including different expressions where scanned with a Kinect depth sensor. Using the deformation transfer technique of Sumner et al. [SP04] we transfer the expressions of both datasets (Digital Emily and FaceWarehouse) to the average face of the statistical model. In

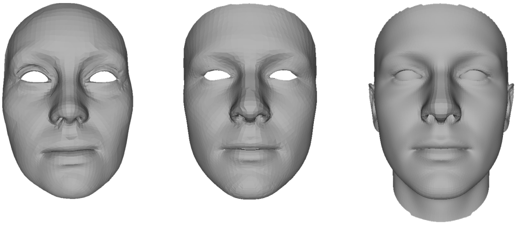


Figure 2.6: Non-rigid registration of the digital Emily mesh [ARL*09] against the average face of the statistical model. From left to right: original Emily mesh, non-rigid deformed Emily mesh, target mesh (average mesh of the statistical model).

the following the face from these two datasets are called source meshes and the average of the statistical model target mesh. A face in rest pose (i.e., with no expression) is called neutral face pose. In a first step we register the neutral face meshes non-rigidly to the statistical face model (see middle of Fig. 2.6) to establish a correspondence between the source mesh and the target mesh. The correspondences between the deformed source mesh and the target mesh are established based on the distance. Since there is a 1:1 correspondence between the source and the deformed source mesh, we also have a correspondence between the original source mesh and the target mesh. Using this correspondence, we transfer the deformations of every source blendshape model to the target mesh, solving a linear system of equations (for details see [SP04]). Fig. 2.7 shows some transfer results. In total we use 76 transferred expressions of both datasets.

When generating blendshapes, a consistent global alignment is important. Thus, after deformation transfer we rigidly align all blendshapes using an ICP (Iterative closest point) method. Instead of using the whole face we restrict the method to a certain region of the face (see Fig. 2.8). This mask represents the ”anchor” of the blendshape model, and contains the region that stays relatively rigid during all expressions.

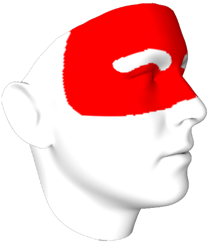


Figure 2.8: ”Anchor” mask.

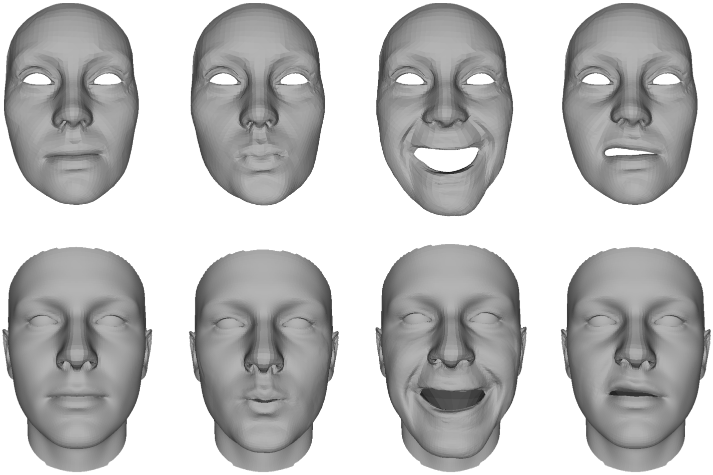


Figure 2.7: Deformation Transfer: the first row shows the source meshes from the Digital Emily project [ARL*09], the second row shows the resulting blendshapes of our face model.

As described above we only transfer the expressions to the average mesh of the statistical model. To allow the deformation of other faces generated with the statistical model, we use delta blendshapes. A delta blendshape describes a mesh relative to its neutral pose. Thus, delta blendshapes are displacement vectors for every vertex of a mesh, similar to the principle components of the statistical model. But in contrast to the principle components of the statistical model the delta blendshapes are relatively sparse (see Fig. 2.9). The sparsity of the delta blendshapes is not exploited in this dissertation, but can be used in future projects to reduce run-time during reconstruction. Another advantage of the blendshapes is that they have a semantic meaning. There is for example a blendshape that opens the mouth and one that lifts the right eye brow. In contrast, the principle components of the statistical face model do not have such a semantic meaning. This allows us to directly transfer expression blendshape weights from one model to another, if the corresponding blendshapes exist.

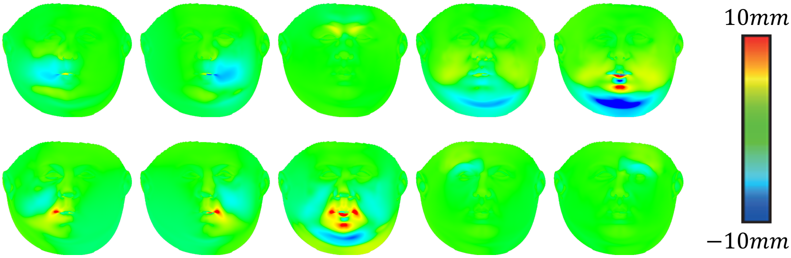


Figure 2.9: The blendshapes have relatively local support. Here we show 10 of our 76 delta blendshapes projected onto the normal of the average face. The distances are visualized in the texture space of the face model.

2.1.3 Illumination - Spherical Harmonics

Illumination plays an important role in our parametric face model. Since we use a analysis-by-synthesis approach in our reconstruction / tracking methods, we have to match the virtual model with the real face. The real world illumination situation can be modeled by a multitude of illumination models which differ in complexity and in their assumptions. An important property of the light model in our methods is that it is differentiable and easy/fast to evaluate. A simple representation would be a point light illumination model, where only a single point light emits light into the scene. This might be suitable for a specific setup, but has problems with multiple light sources and indirect illumination. Most real scenes have such a complex lighting situation. A commonly used representation to model the illumination of a certain scene is the usage of environment maps. The idea of environment maps is to store the light that the surrounding emits to the object. The surrounding is assumed to be distant, such that the same map can be used for arbitrary points on the surface of the model. A cube map can be used to store such information. Depending on the resolution k, there are $6 \cdot k^2$ variables per color channel for all 6 faces. To compute the outgoing light of a point x in the scene (irradiance), the incoming light (radiance) has to be integrate over the hemisphere. This integration acts like a smoothing filter on the environment map if the material of the object is Lambertian [RH01a]. Most of the surface of a face fulfills such an assumption from a macroscopic view. Effects like subsurface scattering or specular skin regions are ignored. Following [RH01a] the so-called irradiance environment map can be represented by spherical harmonics. They state that because of the smoothness only three bands are required to achieve an average error of 1% which results in nine variables (coefficients) per color channel. Spherical harmonics are basis functions defined on the unit sphere. They are organized in bands with increasing frequency. Spherical harmonics can be written in polar angle representation or in Euclidean coordinates. To avoid conversions we use Euclidean coordinates. The basis functions $Y_l^m((x,y,z)^T)$ of the first three bands are listed in Table 2.1. As can be seen,

Band	Index within a band (m)				
(l)	-2	-1	0	1	2
0			$\frac{1}{2\sqrt{\pi}}$		
1		$\frac{\sqrt{3}}{2\sqrt{\pi}}y$	$\frac{\sqrt{3}}{2\sqrt{\pi}}z$	$\frac{\sqrt{3}}{2\sqrt{\pi}}x$	
2	$\frac{\sqrt{15}}{4\sqrt{\pi}}(x^2-y^2)$	$\frac{\sqrt{15}}{2\sqrt{\pi}}xz$	$\frac{\sqrt{5}}{4\sqrt{\pi}}(3z^2-1)$	$\frac{\sqrt{15}}{2\sqrt{\pi}}yz$	$\frac{\sqrt{15}}{2\sqrt{\pi}}xy$

Table 2.1: Spherical harmonics basis functions $Y_l^m((x,y,z)^T)$ [Jar08].

the first band represents the average irradiance (basis function is constant), the second band the average light direction (basis functions are linear). The third band contains the quadratic basis functions. Using these basis functions the color of a surface point x with surface normal n and albedo a is evaluated with:

$$L(n,a) = a \circ \sum_{l=0}^{2} \sum_{m=-l}^{l} c_l^m \cdot Y_l^m(n) \tag{2.1}$$

Here $\circ$ is used for the component-wise product of two vectors. $c_l^m \in \mathbb{R}^3$ is the coefficient of the corresponding basis function. Thus, the representation of the irradiance environment map with spherical harmonics needs $3 \cdot 9 = 27$ parameters. The nine basis functions of the first three bands and their partial derivatives are fast to evaluate without the need of any trigonometric functions. An example for a light situation is shown in Fig. 2.10.

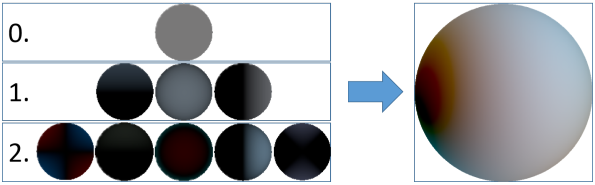


Figure 2.10: Example of spherical harmonics illuminating a sphere. Left: the three bands of the spherical harmonics; Right: composition of all three bands.

2.2 Sensors

To capture the reality and especially the surrounding a computer needs sensors. These sensors allow for an interaction of a human with a computer. In this dissertation we use optical sensors, i.e., commodity RGB cameras and depth cameras. These sensors have a variety of advantages. They are easy to use and do not need a special setup. RGB cameras are passive, thus, they do not influence the scene that is captured.

A major advantage is that nearly every Laptop and Smart-phone is equipped with such a RGB camera (also known as webcam). Beside a RGB camera, new devices also have depth cameras or multiple RGB cameras included. This allows new applications that need depth information like segmentation, refocusing or measuring tools. We use these devices (both RGB or depth cameras) to reconstruct human faces and to track their facial expressions. Since we are using an analysis-by-synthesis approach, we have to model these cameras. Details on both camera types are given in the following sections.

2.2.1 Commodity RGB Cameras

RGB Cameras are wildly spread and can be found in Laptops and Smart-Phones. Thus, they are ideal to develop algorithms that can be used by (basically) everyone. To model a RGB camera we use the Pinhole Camera Model (see Fig. 2.11). An overview of different camera models is given in [HZ03] (Chapter 6 Camera Models). The pinhole camera model can be described by a perspective projection. In this dissertation, we assume that the viewing frustum is not skewed. Thus, the perspective projection can be written as:

$$\Pi((x,y,z)^T) = \begin{pmatrix} \frac{fov_x \cdot x}{z} + c_x \\ \frac{fov_y \cdot y}{z} + c_y \end{pmatrix} \tag{2.2}$$

fov_x and fov_y are the field of view in x and y-direction in pixels. $(c_x, c_y)^T$ is the center of the image (also known as principle point).

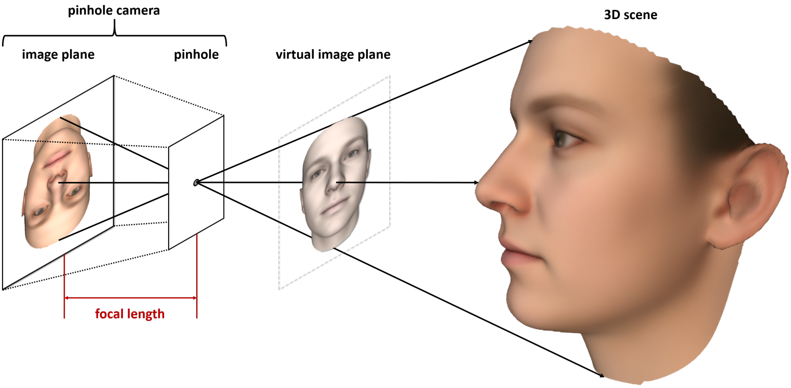


Figure 2.11: Pinhole camera model. On the left you see a pinhole camera that captures the 3D scene on the right.

In a controlled setup, these camera parameters can be estimated in a calibration step. Typically, a calibration board is captured by the camera. A calibration board has features that are easy to detect in a 2D image (e.g. corners, circles). The features have a known alignment in 3D space. If the feature alignment is non-symmetric, there is a unique correspondence between the observed 2D image of the camera and the 3D model of the calibration board.

In an uncontrolled setup, which is the case for videos from the Internet, we do not have observations of a calibration pattern. Instead, we estimate the intrinsics of the camera using automatic detected facial landmarks [SLC11a] (see Fig. 2.12). Every landmark point corresponds to a point of the morphable model. At this initialization stage we do not have a reconstruction of the face, thus, we are using the average mesh as an approximation. Leveraging the 3D-2D correspondences in both scenarios (controlled and uncontrolled setup) also known as world to image correspondences, we have to solve the classical resectioning problem [HZ03].

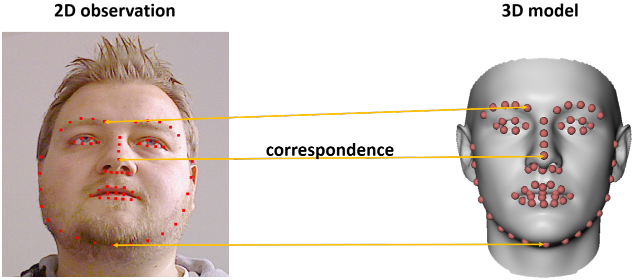


Figure 2.12: 3D-2D correspondence used for re-sectioning. The landmarks in the 2D observation are computed by the method of Saragih et al. [SLC11a].

For every corresponding pair $x_i \in \mathbb{R}^3$ and $y_i \in \mathbb{R}^2$ the following equation has to be solved:

$$\Pi(R \cdot x_i + t) = y_i \tag{2.3}$$

Where $R \in \mathbb{R}^{3\times 3}$ and $t \in \mathbb{R}^3$ define the unknown extrinsic transformation of the camera. R is a rotation matrix and t a translation vector. In homogeneous coordinates the equation 2.3 is:

$$\underbrace{\begin{pmatrix} f_x & 0 & 0 & c_x \\ 0 & f_y & 0 & c_y \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{K} \cdot \underbrace{\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}}_{E} \cdot x_i' = P \cdot x_i' = y_i' \tag{2.4}$$

$P \in \mathbb{R}^{3\times 4}$ is the unknown camera matrix including the projection matrix K (camera intrinsics) and the extrinsic transformation E. Using the Gold Standard algorithm [HZ03] (Algorithm 7.1, page 181) we can solve for the unknown camera matrix P. To decompose the camera matrix into intrinsics and extrinsics we apply a RQ decomposition [HZ03] (A4.1.1 Givens rotation and RQ decomposition, page 579).

Note, in case of an uncontrolled setup this estimation of the camera parameters is only an initial guess and is refined afterwards in a joint optimization problem with the other model parameters.

2.2.2 Depth Cameras

In contrast to RGB cameras, RGBD cameras have an additional depth channel. To compute the depth of a scene, the cameras typically utilize the epipolar geometry or they measure the round-trip time that light needs from the camera to the scene and back to the sensor. Both variants are used in today's consumer hardware.

Cameras that are based on the epipolar geometry need at least two views of the scene. There are two kinds of such stereo cameras - active and passive. A passive stereo camera consists of two cameras that observe the scene (see Fig. 2.13).

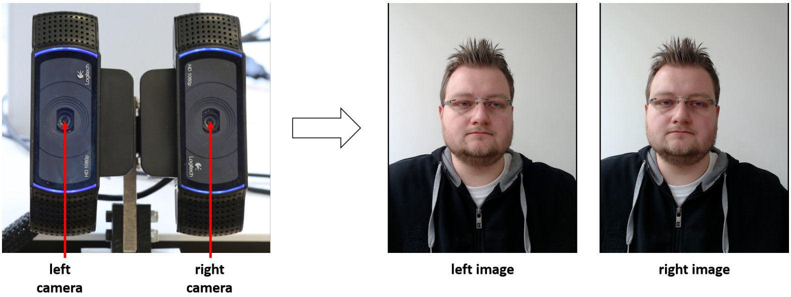


Figure 2.13: Custom stereo setup consisting of two commodity webcams (left). On the right you can see the output of the camera setup.

If a point is found in both views of a calibrated stereo setup, the 3D point can be reconstructed by triangulation. Finding a corresponding point for one pixel of the first image in the second view, is a hard problem. Utilizing the epipolar geometry of calibrated cameras, the search problem can be reduced to a 1D search. The search is based on features (e.g. color, color gradients, edges), but if there are no unique features the search fails. This problem occurs for example if you want to reconstruct the depth of a white wall. To solve this problem, one of the cameras is replaced with a projector resulting in an active stereo camera setup (see Fig. 2.14). The projector is assumed to act like a "inverse camera". Thus, the projector projects known

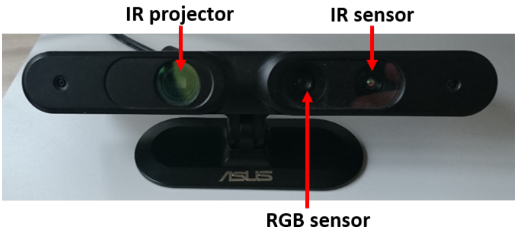


Figure 2.14: Asus Xtion Pro: An active stereo camera.

feature patterns into the scene. These patterns are then searched in the observed camera image using their structure (→ structured light). Since the projector changes the scene the cameras are called active. To avoid that humans can see this change, active stereo cameras typically work with IR light and IR sensors. For example Fig. 2.15 shows the observation of an Asus Xtion Pro. This RGBD camera is an active stereo camera like the Microsoft Kinect and the Primesense Carmine cameras. The named active stereo cameras have problems with sun light, since the sun outshines the IR projector. This problem is tackled with the new Intel Realsense R200 cameras that combine active and passive stereo using two IR cameras and one projector. The projector emits a random pattern, thus, generating features in the scene. These augmented features enable the passive stereo setup to reconstruct depth in otherwise homogeneous regions. In an outdoor environment the projector has no effect over the IR light of the sun and the system works as a classical passive stereo setup.

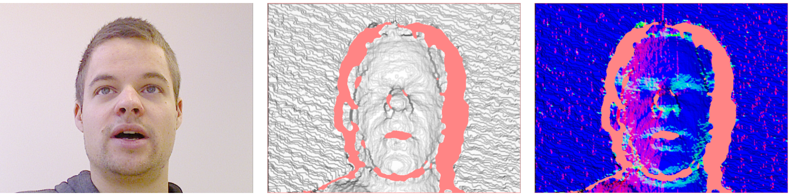


Figure 2.15: RGBD camera (Asus Xtion Pro) output: Left the RGB image, in the middle the Phong-shaded depth map and right the corresponding normal map (the red areas indicate that there is no observation).

In contrast to stereo cameras, time-of-flight cameras (ToF cameras) like the Micosoft Kinect One compute the depth by measuring the round-trip time of the light from the camera to the scene and back. Because of the speed of light, measuring the time is a big challenge. These cameras have in contrast to the active stereo cameras a poor depth resolution and higher noise in the near range. But this is the important range for our face tracking/reconstruction scenarios. Thus, we concentrate our work on stereo cameras.

As can be seen in Fig. 2.15 the depth information of an active stereo depth camera typically has noise and regions with no data. The regions with no data stem from missing correspondences, especially regions that are only seen from one perspective or reflections. To reduce the noise of the depth data, a Gaussian filter can be used. Similar to the RGB camera model in the previous chapter we model the depth camera as a pinhole camera (see Equation 2.2). Thus, given the depth z at a certain pixel position $(u, v)^T$, we re-project it to a 3D point p in the camera coordinate system:

$$p = \begin{pmatrix} z \cdot \frac{(u - c_x^{depth})}{fov_x^{depth}} \\ z \cdot \frac{(v - c_y^{depth})}{fov_y^{depth}} \\ z \end{pmatrix} \tag{2.5}$$

Using this equation with the calibrated depth camera parameters fov_x^{depth}, fov_y^{depth} and c_x^{depth}, c_y^{depth} a position map is generated. For simplicity we project the positions into the RGB camera space, and thus, align the position with the color information of the camera. To compute a normal map of the position map, central differences of the position values of the surrounding pixel are applied (see Fig. 2.15).

2.3 Optimization

One of the main components of an analysis-by-synthesis approach is the estimation of parameters that reduce the difference between the input and the synthesized data. In general, to get a new estimation of parameters an optimization problem has to be solved. Optimization problems can have different levels of difficulty. It depends on the number of unknowns, the error measurement and whether additional constraints have to be fulfilled.

In our algorithms we are typically confronted with unconstrained non-linear optimization problems. In the following it is briefly discussed, how we tackle these optimization problems.

2.3.1 Non-Linear Optimization

An analysis-by-synthesis approach iteratively computes a new set of parameters x^{i+1} based on the synthesis generated with the old parameters x^i.

The parameters x^{i+1} are chosen such that the energy that measures the difference between the observation and the synthetic data is minimized. An energy function $E(x)$ of an analysis-by-synthesis approach has in general the following form:

$$E(x) = D(I - M(x))$$

Here, $D(r) : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ is a function that maps the difference between the observation $I \in \mathbb{R}^n$ and the model $M(x) \in \mathbb{R}^n$ to a scalar. n is the dimensionality of the observation, e.g. the number of pixels. Note, the dimensionality of the observation n does not have to be static, i.e., it might changes during the analysis-by-synthesis iterations.

We are using different error metrics in our projects. The most common metric is to use the ℓ_2-norm ($D(r) = ||r||^2$). This results in a least-squares problem. If the model $M(x)$ is linear, it collapses to a linear least-squares problem ($||A \cdot x + b||^2 \rightarrow min$) that can be solved by solving the corresponding normal equation which is linear ($A^T \cdot (A \cdot x + b) = 0$).

In general, to solve non-linear least-squares problems, iterative methods like gradient descent, *Gauss-Newton* or *Levenberg-Marquardt* are applied. The *Gauss-Newton* algorithm is an approximation of the *Newton* method. The Newton method can be applied to non-linear problems and calculates iteratively a new solution x^{i+1} using the formula:

$$x^{i+1} = x^i - H_E(x^i)^{-1} \cdot \nabla E(x^i)$$

$H_E(x)$ is the Hessian of the energy function, thus, involving second order derivatives of the residuum vector $r(x) = I - M(x)$. The *Gauss-Newton* method is limited to non-linear least-squares problems and approximates the Hessian using only first order derivatives of the residuum vector $r(x)$:

$$H_E(x) \approx 2 \cdot J_r^T(x) \cdot J_r(x)$$

Here, $J_r(x)$ is the Jacobian of the residual function $r(x)$. Using this approximation, results in the following update rule:

$$x^{i+1} = x^i - \underbrace{(J_r^T(x^i) \cdot J_r(x^i))^{-1} \cdot J_r^T(x^i) \cdot r(x^i)}_{\Delta^i} \tag{2.6}$$

Thus, to compute the parameter update Δ^i the following linear equation has to be solved:

$$(J_r^T(x^i) \cdot J_r(x^i)) \cdot \Delta^i = J_r^T(x^i) \cdot r(x^i) \tag{2.7}$$

In our experiments we solve these kinds of linear equations using a preconditioned conjugate gradient (PCG) solver. Using a Jacobi preconditioner, the PCG converges fast (see convergence of the *Gauss-Newton* solver in Fig. 9.2). Considering a limited time budget, an iterative method has the advantage that it can be stopped after a certain amount of time / iterations.

The *Gauss-Newton* method is an iterative algorithm which is highly dependent on the initial guess. We use sparse detected landmarks to initialize our face tracking and then propagate the solution from one frame to the following frame. To allow fast motions we are using a hierarchical optimization strategy, i.e., we down-sample our observations (half the image resolution) and optimize starting from a coarse level and propagate the solution to the next finer level. A pixel of a down-sampled image covers a larger part of the scene. Thus, if a motion in the original image is for example four pixels in x-direction it is a motion of two pixels in the once and a motion of only one pixel in the twice down-sampled image. Optimizing first on a down-sampled image, the numerical derivatives of the observed image have a more global footprint and the residual function is smoother than the residual function of the original image. This reduces local minimas and leads to a good initial solution for the next finer level which optimizes a less smoothed residual function.

To allow for real-time tracking we built our own GPU-based *Gauss-Newton* optimization framework. Details are given in the next section.

2.3.2 GPU-based Analysis-by-Synthesis Optimization Framework

Modern graphic processing units (GPU's) are equipped with thousands of small processors. We utilize this computing power to enable a real-time tracking and optimization of the analysis-by-synthesis energy. As described in the previous chapters about the face model and sensors, the synthetic image is generated using a pinhole camera model. The graphics pipeline is optimized to render triangle meshes with such a camera model. Thus, we synthesize facial images using the graphics card of a computer. Beside the synthesis, we also utilize the graphics card to analyze the difference between the synthetic and the original image. We therefore make use of so-called compute shaders. A compute shader enables the usage of graphics cards for general purpose computing (GPGPU). Since a GPU consists of many small processors, the workload of a computing step has to be distributed and parallelized. For example the synthesis of the 3D face model can be done for every single vertex in parallel. Each vertex of the face model is a weighted sum of the principle components and independent of other vertices. Thus, we can launch a single thread per vertex, to compute the current 3D mesh.

The *Gauss-Newton* algorithm solves a linear system of equations in every iteration (see Equation 2.7). To compute the right hand side of this system of equations (the gradient of $E(x)$), we have to evaluate the Jacobian $J_r(x)$. Thus, while computing the gradient of $E(x)$, we store the Jacobian of the residuum function $r(x)$. To evaluate the gradient of $E(x)$ we first have to determine the dimensionality of the residuum, i.e., we only compare pixels that are visible in both the synthetic and the observed image. This is done by using a GPU-based scan, which is based on a prefix sum. Knowing the number of pixels that have to be compared, we start threads for every pixel i and unknown variable x_j, to compute the per pixel partial derivative $\frac{\delta r_i(x)}{\delta x_j}$ and the local gradient $\frac{\delta r_i(x)}{\delta x_j} \cdot r_i(x)$ ($r_i(x) \in \mathbb{R}$ is the component of the residuum function $r(x)$ corresponding to pixel i, for simplicity we assume here a scalar per pixel residuum). To compute the per pixel partial derivative $\frac{\delta r_i(x)}{\delta x_j}$ we need to know which vertex contributes to that pixel. Therefore, we use a deferred renderer that stores all information that are used to generate a single pixel. Especially it has to store the vertex indices per pixel and the corresponding barycentric coordinates. Storing this additional data results in a differentiable renderer. Using the reduction schema of Harris [Har07], the local gradients are summed-up to the global gradients $\nabla E(x)$. As described in the previous section, we employ a PCG. We use an adapted form of the classical PCG that does not need to explicitly compute $J^T \cdot J$. Computing $J^T \cdot J$ is in $O(m^2 \cdot n)$, where n is the number of residuals and m the number of unknowns. In our scenarios, n is typically magnitudes larger than m (e.g. 80000 pixels versus 269 unknowns). To evaluate a conjugated update step, the PCG method has to compute $J^T \cdot J \cdot p$ amongst other steps. Thus, instead of computing $J^T \cdot J$ and then a matrix-vector product, we compute two matrix-vector multiplications in succession: $(J^T \cdot J) \cdot p = (J^T \cdot (J \cdot p))$. Which results in a complexity of only $O(m \cdot n)$.

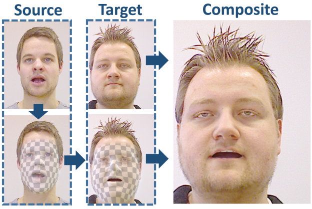
To avoid unnecessary staging and remapping overhead, the whole optimization framework is written in DirectX11, since it allows rendering and compute shaders in one context. Especially copying data from CPU memory to GPU memory and vice versa is reduced to a minimum, since the GPU and the CPU has to be synchronized to execute such an operation, wasting compute power.

CHAPTER 3

Contribution and Outline

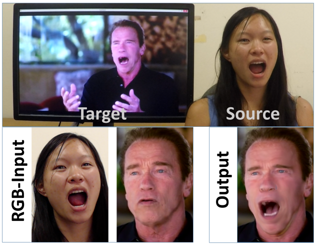
In this dissertation, we show the advances in face tracking and facial reenactment. We focus our work on real-time algorithms that are based on consumer-grade hardware. The first part presents a live reenactment system that is based on the input of an active stereo RGBD camera (see Section 2.2.2). This setup is using a calibrated camera and depth information, thus, it is not suitable to be applied to videos from the internet, where these information are missing. To reduce the hardware requirements we concentrated our work on an RGB-only tracking and reenactment system. Part 2 demonstrates the enhancements, which allow us to modify ordinary monocular videos. For the upcoming VR-devices like Oculus Rift or HTC Vive, a monocular video is not sufficient. Stereo videos are needed. Another problem that arises, is the strong occlusion of the face if such a head mounted display is worn. In Part 3 we show an adapted tracking algorithm that can handle the strong occlusions. It also enables gaze-aware reenactments of stereo videos. In the following a more detailed abstract of the single projects is given.

Part 1: Real-time Expression Transfer for Facial Reenactment



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. This work has been published and presented at Siggraph Asia 2015 [TZN*15].

Part 2: Face2Face: Real-time Face Capture and Reenactment of RGB Videos



We present a novel approach for real-time facial reenactment of a monocular target video sequence (e.g., Youtube video). The source sequence is also a monocular video stream, captured live with a commodity webcam. Our goal is to animate the facial expressions of the target video by a source actor and re-render the manipulated output video in a photo-realistic fashion. To this end, we first address the under-constrained problem of facial identity recovery from monocular video by non-rigid model-based bundling. At run time, we track facial expressions of both source and target video using a dense photometric consistency measure. Reenactment is then achieved by fast and efficient deformation transfer between source and target. The mouth interior that best matches the re-targeted expression is retrieved from the target sequence and warped to produce an accurate fit. Finally, we convincingly re-render the synthesized target face on top of the corresponding video stream such that it seamlessly blends with the real-world illumination. We demonstrate our method in a live setup, where Youtube videos are reenacted in real time. Face2Face has been published and presented at CVPR 2016 [TZS*16b] and a demonstration has been given at Siggraph Emerging Technologies 2016 [TZS*16a].

Part 3: FaceVR: Real-Time Facial Reenactment and Eye Gaze Control in Virtual Reality

We introduce *FaceVR*, a novel method for gaze-aware facial reenactment in the Virtual Reality (VR) context. The key component of *FaceVR* is a robust algorithm to perform real-time facial motion capture of an actor who is wearing a head-mounted display (HMD), as well as a new data-driven approach for eye tracking from monocular videos. In addition to these face reconstruction components, *FaceVR* incorporates photo-realistic re-rendering in real time, thus allowing artificial modifications of face and eye appearances. For instance, we can alter facial expressions, change gaze directions, or remove the VR goggles in realistic re-renderings. In a live setup with a source and a target actor, we apply these newly-introduced algorithmic components. We assume that the source actor is wearing a VR device, and we capture his facial expressions and eye movement in real-time. For the target video, we use a stereo camera rig that enables us to reconstruct a stereoscopic avatar. To capture a face in a stereo video, we propose a novel tracking approach, leveraging the information of both cameras. Finally, we map the expressions of the source input to the stereo target including gaze-aware eye animations. In the end, *FaceVR* produces compelling results for a variety of applications, such as gaze-aware facial reenactment, reenactment in virtual reality, removal of VR goggles, and re-targeting of somebody's gaze direction in a video conferencing call. FaceVR is currently unpublished work that is available as a technical report on ArXiv [TZS*16c].

PART I

Real-time Expression Transfer for Facial Reenactment

CHAPTER 4

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 [WBLP11, LYYB13, CHZ14]. 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 [DSJ*11, GVR*14].

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

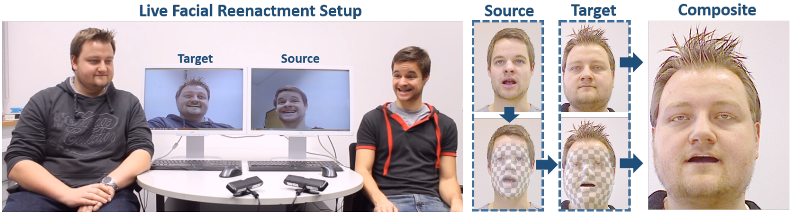


Figure 4.1: Our live facial reenactment technique tracks the expression of a source actor and transfers it to a target actor at real-time rates. The synthetic result is photo-realisticly re-rendered on top of the original input stream maintaining the target's identity, pose and illumination.

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 [WBLP11, LYYB13, CHZ14], 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. [ZNI*14] 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. 4.1). 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).

CHAPTER 5

Related Work

5.0.1 Facial Performance Capture

Traditional facial performance capture for film and game productions achieves high-quality results using controlled studio conditions [BPL*03, PL06]. A typical strategy to obtain robust features is the use of invisible makeup [Wil90] or facial markers [GGW*98, BBA*07, HCTW11]. Another option is to capture high-quality multi-view data from calibrated camera arrays [BHPS10, BHB*11, VWB*12, FJA*14]. Dynamic active 3D scanners, for instance based on structured light projectors, also provide high-quality data which has been used to capture facial performances [ZSCS04, WHSL*04, WLGP09]. 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 [ARL*09, WGP*10].

Monocular fitting to RGB-D data from a depth camera by non-rigid mesh deformation was shown in [CWS*13], but neither photo-realistic nor extremely detailed reconstruction is feasible. Recently, monocular off-line methods were proposed that fit a parametric blend shape [GVWT13] or multi-linear face model [SWTC14] 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.

5.0.2 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) [CET01] which are used to determine the parameters of a 3D PCA model while only using 2D features [XBMK04]. Another popular representation is the blend shape model [PHL*98, LA10] which embeds pose variation in a low-dimensional PCA space; blend shapes can be constrained by image feature points [CB02, CXH03]. 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 [CWLZ13, CHZ14] 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 [VWB*12, SKS14]; 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 [WBLP11, LYYB13, BWP13, HMYL15], or by the deformation of a face template mesh in an as-rigid-as-possible framework [ZNI*14]. 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.

5.0.3 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 [BV99, BBPV03, BSVS04, VBPP05]. 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 [VBPP05], or by purposely building a person-specific blend shape model from scans of an actor using specialized hardware [EG98, ARL*09, WBLP11]. The morphable model based face replacement technique of Dale et al. [DSJ*11] 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 [LXW*12] or a sparse set of 2D facial features [SLC11b], and often include an image matching step to look up similar expressions in a database of facial images [KSSS10], or a short sequence of arbitrary source performances [GVR*14]. 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 [KSSGS11]. Expression mapping [LSZ01] 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. [GVS*15] 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.

CHAPTER 6

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 [BV99] with the expression space of a blend shape model [ARL*09, CWZ*14] (see Sec. 7). 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

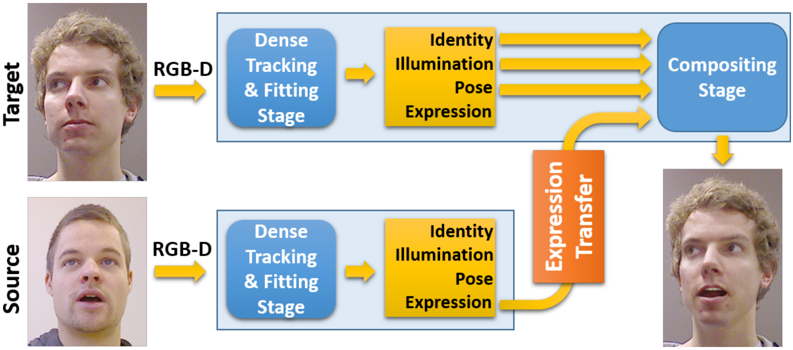


Figure 6.1: Our live facial reenactment pipeline.

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. 8). 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. 9). The tracking stage is summarized in Fig. 8.1

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. 8.0.2) 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. 10.0.5). 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. 10). 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. 6.1.

CHAPTER 7

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.

7.0.1 Parametric Face Model

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

$$\mathcal{M}_{\text{geo}}(\alpha, \delta) = a_{\text{id}} + E_{\text{id}}\,\alpha + E_{\text{exp}}\,\delta \ , \tag{7.1}$$

$$\mathcal{M}_{\text{alb}}(\beta) = a_{\text{alb}} + E_{\text{alb}}\,\beta \ . \tag{7.2}$$

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 α, δ and β control the identity, the expression and the skin albedo of the resulting face, and a_{id} and a_{alb} represent the mean identity shape in rest and the mean skin albedo. While v_i and c_i are defined by a linear combination of basis vectors, the normals 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 [BV99]. 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 a_{id}, a mean albedo a_{alb}, and standard deviations σ_{id} and σ_{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 [SP04]. The used blend shapes have been created manually [ARL*09] [1] or by non-rigid registration to captured scans [CWZ*14] [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.

7.0.2 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 Spherical Harmonics (SH) basis [Mül66] for a low dimensional representation of the incident illumination. Following Ramamoorthi and Hanrahan [RH01b], the *irradiance* in a vertex with normal n and scalar albedo

---

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

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

c is represented using $b=3$ bands of SHs for the incident illumination:

$$\mathcal{L}(\gamma, n, c) = c \cdot \sum_{k=1}^{b^2} \gamma_k y_k(n) \quad , \tag{7.3}$$

with y_k being the k-th SH basis function and $\gamma = (\gamma_1, \ldots, \gamma_{b^2})$ 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).

7.0.3 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 $\Phi(v) = R\,v + t$, where R is a 3×3 rotation matrix and $t \in \mathbb{R}^3$ a translation vector. R is parameterized using Euler angles and, together with 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:

$$\boldsymbol{\mathcal{P}} = (\alpha, \beta, \delta, \gamma, R, t) \quad , \tag{7.4}$$

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 $\mathcal{P}$ that faithfully reproduce the observed face in each RGB-D input frame. In the next section, we will describe how we can optimize for $\mathcal{P}$ in real-time. The use of the estimated parameters for video manipulation will be described in Sec. 10.

CHAPTER 8

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 $\mathcal{S}(\mathcal{P})$ to the input of a commodity RGB-D camera recording an actor's performance. Our goal is to obtain the best fitting parameters $\mathcal{P}$ 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 $\mathcal{P}$ further by comparing the rendered image to the captured RGB-D input. This is a hard inverse rendering problem in the unknowns $\mathcal{P}$ 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. 8.1.

8.0.1 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 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 $p = (i, j)$ is given

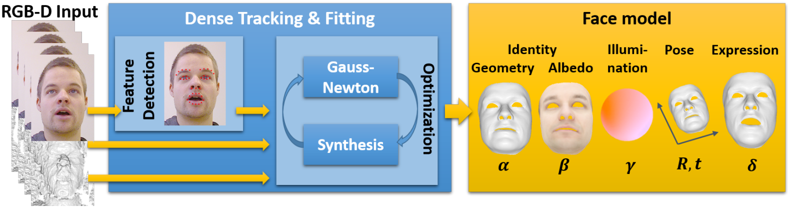


Figure 8.1: Overview of our real-time fitting pipeline.

by $C_{\mathcal{I}}(p) \in \mathbb{R}^3$ and $X_{\mathcal{I}}(p) \in \mathbb{R}^3$, respectively. The range sensor implicitly provides us with a normal field $N_{\mathcal{I}}$, where $N_{\mathcal{I}}(p) \in \mathbb{R}^3$ is obtained as the cross product of the partial derivatives of $X_{\mathcal{I}}$ with respect to the continuous image coordinates.

8.0.2 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 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 p. Note that $C_{\mathcal{S}}(p)$, $X_{\mathcal{S}}(p)$, and $N_{\mathcal{S}}(p)$ 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. An input pixel is valid if color and depth of the pixel is valid.

8.0.3 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:

$$E(\boldsymbol{\mathcal{P}}) = E_{\text{emb}}(\boldsymbol{\mathcal{P}}) + w_{\text{col}} E_{\text{col}}(\boldsymbol{\mathcal{P}}) + w_{\text{lan}} E_{\text{lan}}(\boldsymbol{\mathcal{P}}) + w_{\text{reg}} E_{\text{reg}}(\boldsymbol{\mathcal{P}}) \ . \tag{8.1}$$

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:

$$E_{\mathrm{emb}}(\boldsymbol{\mathcal{P}}) = w_{\mathrm{point}} E_{\mathrm{point}}(\boldsymbol{\mathcal{P}}) + w_{\mathrm{plane}} E_{\mathrm{plane}}(\boldsymbol{\mathcal{P}}) \ . \tag{8.2}$$

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

$$E_{\mathrm{point}}(\boldsymbol{\mathcal{P}}) = \sum_{p \in \mathcal{V}} \left\| d_{\mathrm{point}}(p) \right\|_2^2 \ , \tag{8.3}$$

with $d_{\mathrm{point}}(p) = X_{\mathcal{S}}(p) - X_{\mathcal{I}}(p)$ 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 [CM92]. 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:

$$E_{\mathrm{plane}}(\boldsymbol{\mathcal{P}}) = \sum_{p \in \mathcal{V}} \left[d_{\mathrm{plane}}^2 \left(N_{\mathcal{S}}(p), p \right) + d_{\mathrm{plane}}^2 \left(N_{\mathcal{I}}(p), p \right) \right], \tag{8.4}$$

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

$$E_{\text{col}}(\boldsymbol{\mathcal{P}}) = \sum_{p \in \mathcal{V}} \| C_{\mathcal{S}}(p) - C_{\mathcal{I}}(p) \|_2^2 \ , \tag{8.5}$$

where $C_{\mathcal{S}}(p)$ 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 [SLC11a]. Each detected feature $f_j = (u_j, v_j)$ is a 2D location in the image domain that corresponds to a consistent 3D vertex 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:

$$E_{\text{lan}}(\boldsymbol{\mathcal{P}}) = \sum_{f_j \in \mathcal{F}} w_{\text{conf},j} \left\| f_j - \Pi(\Phi(v_j)) \right\|_2^2 \ . \tag{8.6}$$

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 [SLC11a]. This allows us to effectively prune wrongly classified features, which are common under large head rotations ($> 30°$).

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 α, β, and δ to be statistically small compared to their standard deviation:

$$E_{\text{reg}}(\boldsymbol{\mathcal{P}}) = \sum_{i=1}^{160} \left[\left(\frac{\alpha_i}{\sigma_{\text{id},i}} \right)^2 + \left(\frac{\beta_i}{\sigma_{\text{alb},i}} \right)^2 \right] + \sum_{i=1}^{76} \left(\frac{\delta_i}{\sigma_{\text{exp},i}} \right)^2 . \tag{8.7}$$

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. 7.0.1). 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.

CHAPTER 9

Parallel Energy Minimization

The proposed energy $E(\mathcal{P}) : \mathbb{R}^p \rightarrow \mathbb{R}$ of Eq. 8.1 is non-linear in the parameters $\mathcal{P}$, and finding the best set of parameters $\mathcal{P}^*$ amounts to solving a non-linear least squares problem in the p unknowns:

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} E(\mathcal{P}) \ . \tag{9.1}$$

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 $p \in \mathcal{V}$ contributes with 8 residuals (3 from the point-to-point term of Eq. 8.2, 2 from the point-to-plane term of Eq. 8.4 and 3 from the color term of Eq. 8.5), while the feature term of Eq. 8.6 contributes with $2 \cdot 38$ residuals and the regularizer of Eq. 8.7 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.

9.0.1 Core Solver

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

$$E(\mathcal{P}^k + \Delta\mathcal{P}) \approx \left\| r(\mathcal{P}^k) + J(\mathcal{P}^k)\Delta\mathcal{P} \right\|_2^2 , \tag{9.2}$$

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

$$J^T(\mathcal{P}^k)J(\mathcal{P}^k)\Delta\mathcal{P} = -J^T(\mathcal{P}^k)r(\mathcal{P}^k) \ , \tag{9.3}$$

and the parameters are updated as $\mathcal{P}^{k+1} = \mathcal{P}^k + \Delta\mathcal{P}$. 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^TJ are computed in the initialization stage of the PCG.

Close in spirit to [ZNI*14], 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.

9.0.2 Memory Efficient Solution Strategy on the GPU

The normal equations 9.3 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^TJ with the previous descent direction. Pre-computing J^TJ 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 [ZNI*14], 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 β and γ, influence each residual (see Fig. 9.1). In addition, we optimize for all unknowns si-

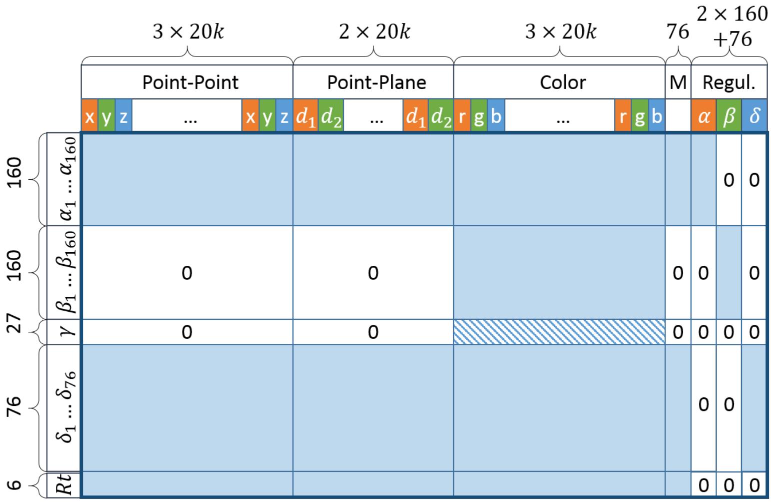


Figure 9.1: Non-zero structure of J^T for 20k visible pixels.

multaneously and our energy has a larger number of residuals compared to [ZNI*14]. 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. 9.2. 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.

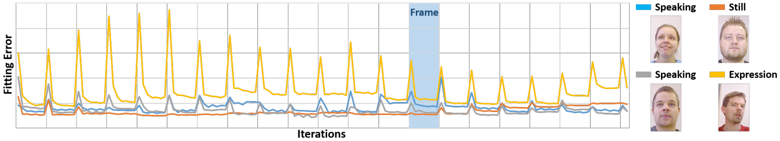


Figure 9.2: Convergence of the Gauss-Newton solver for four different facial performances. The horizontal axis breaks up convergence for each captured frame (at 30 fps); the vertical axis shows the fitting error. Even for expressive motion, we converge well within a single frame.

9.0.3 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. 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 [AAB*84]. The final high-resolution albedo map is used for rendering.

CHAPTER 10

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.

10.0.1 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. 4.1). After estimating the identity, reflectance, and lighting in a calibration step (see Sec. 9.0.3), 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.

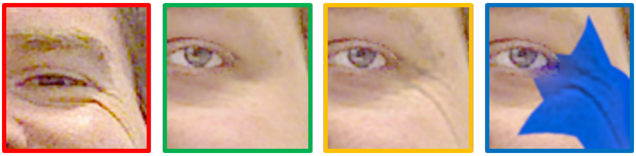


Figure 10.1: Wrinkel-level detail transfer. From left to right: (a) the input source frame, (b) the rendered target geometry using only the target albedo map, (c) our transfer result, (d) a re-texturing result.

10.0.2 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

$$\mathcal{M}_{\mathrm{geo}}\left(\alpha_{\mathrm{t}}, \delta_{\mathrm{s}}\right) = a_{\mathrm{id}} + E_{\mathrm{id}}\, \alpha_{\mathrm{t}} + E_{\mathrm{exp}}\, \delta_{\mathrm{s}} \quad , \tag{10.1}$$

where α_{t} are the target identity parameters and δ_{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 [GVS*15] 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. 8.0.2.

10.0.3 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 [BA83] 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. 10.1 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. 10.0.5).

10.0.4 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. 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. 10.2).

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

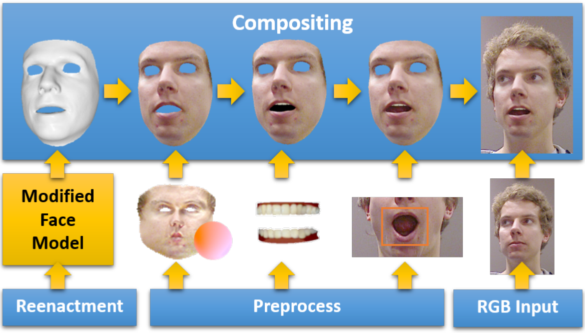


Figure 10.2: Final compositing: we render the modified target geometry with the target albedo under target lighting and transfer skin detail. After rendering a person-specific teeth proxy and warping a static mouth cavity image, all three layers are overlaid on top of the original target frame and blended using a frequency based strategy.

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. 9.0.3 and is illustrated in Fig. 10.2. Warping is based on tracked 2D landmarks around the mouth and implemented using generalized barycentric coordinates [MBLD02]. The brightness of the rendered teeth and warped mouth interior is adjusted to the degree of mouth opening for realistic shadowing effects.

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. 10.0.3) and performing blending on each frequency level separately. Computing and merging

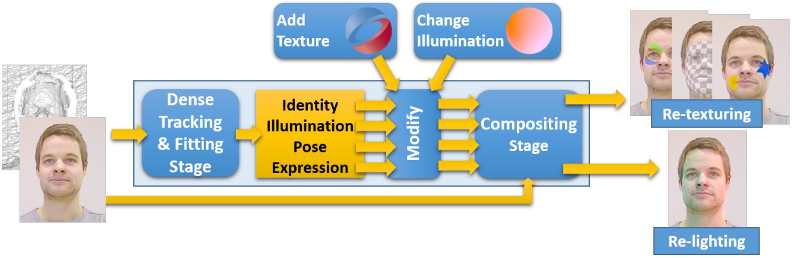


Figure 10.3: Re-texturing and re-lighting of a facial performance.

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.

10.0.5 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 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. 10.3. We show further examples in the experimental section.

CHAPTER 11

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: $w_{col} = 20$, $w_{lan} = 0.125$, $w_{reg} = 0.025$, $w_{point} = 2$, $w_{plane} = 10$.

11.0.1 Real-time Facial Performance Capture

We track several actors in different settings. Tracking results for facial reenactment (see Sec. 11.0.2) are also shown in Fig. 11.9. Our approach first performs a short calibration phase to obtain the model identity and albedo (see Sec. 9.0.3). 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.

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

	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 11.1: Run times for three of the sequences of Fig. 9.2 (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.

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 11.1 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. 9.2 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. [VWB*12] [1], 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. 11.1. 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

---

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

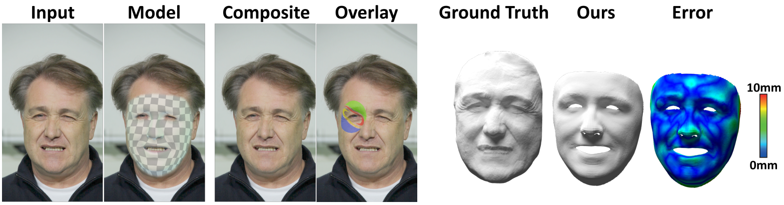


Figure 11.1: Tracking accuracy. Left: the input RGB frame, the tracked model overlay, the composite and the textured model overlay. Right: the reconstructed mesh of [VWB*12], our reconstructed shape, and the color coded distance between both reconstructions.

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. 11.2 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. 11.3 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 the 38 tracked landmark vertices from the feature similarity term of Eq. 8.6. The projections of these vertices can be compared to the feature locations of the 2D

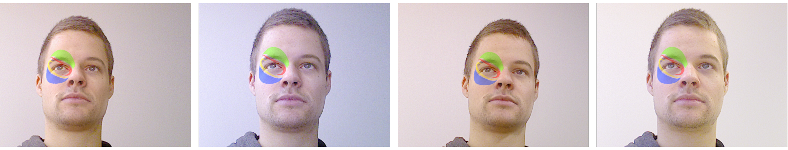


Figure 11.2: Stability under lighting changes.

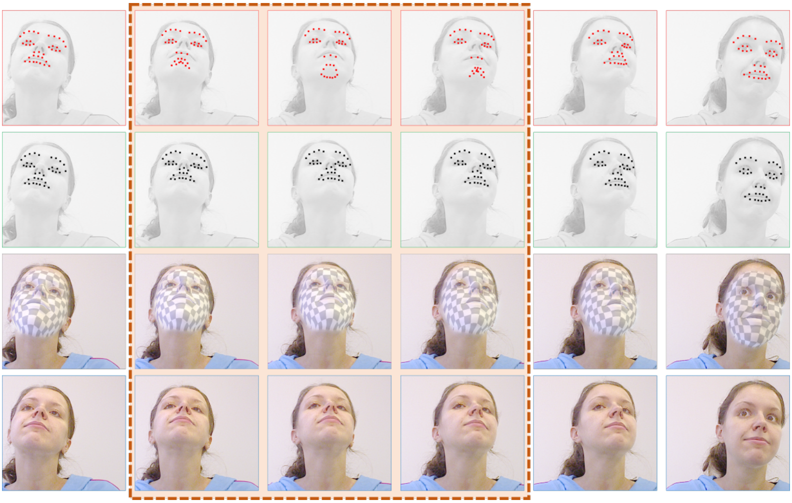


Figure 11.3: Stability under head motion. From top to bottom: (a) 2D features of [SLC11a], (b) our 3D landmark vertices, (c) overlaid face model, (d) textured and overlaid face model. Our method recovers the head motion, even when the 2D tracker fails.

tracker of Saragih et al. [SLC11a]; 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. 11.4. 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. [WBLP11]. 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. 11.5). 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. 11.5 (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. 11.9), accurate 2D alignment is crucial.

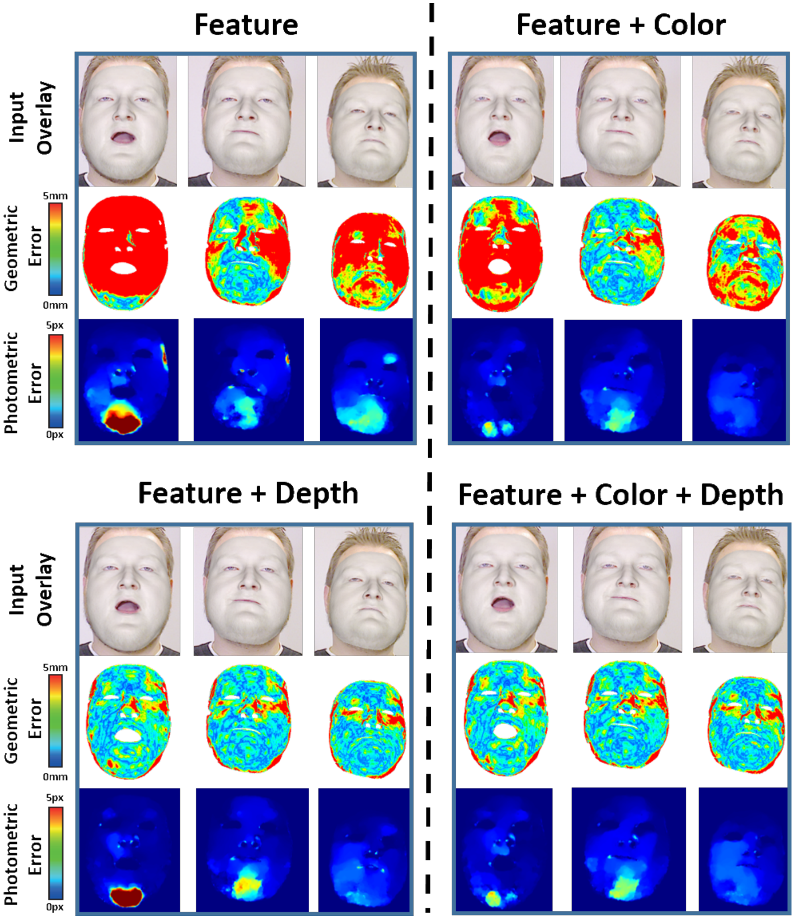


Figure 11.4: Importance of the different data terms in our objective function: tracking accuracy is evaluated in terms of geometric (middle) and photometric error (bottom). The final reconstructed pose is shown as an overlay on top of the input images (top). Mean and standard deviations of geometric 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.

Comparison to Cao et al. 2014 We also compare our method to the real-time face tracker of Cao et al. [CHZ14], 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 the shape identity. The results of both methods are shown in Fig. 11.6, along with the per-vertex Euclidean distance to the ground truth scan. The error color scale is the same as in Fig. 11.1. Overall, our method approximates the identity of the actor better; however, please note that Cao et al. [CHZ14] only use RGB video data as input.

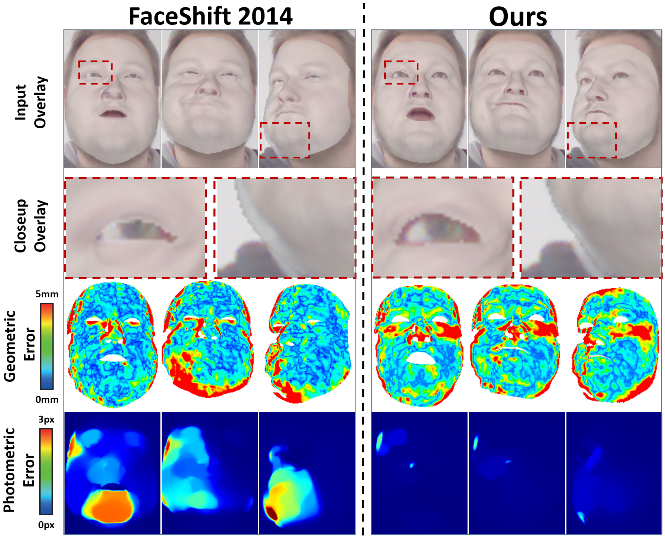


Figure 11.5: Comparison to *FaceShift*. From top to bottom: Reconstruction overlaid on top of the RGB input, closeups, geometric alignment error with respect to the input depth maps, and photometric re-rendering error. Note that while *FaceShift* [WBLP11] is able to obtain a comparable *model-to-depth* alignment error, our reconstructions exhibit significantly better 2D alignment.

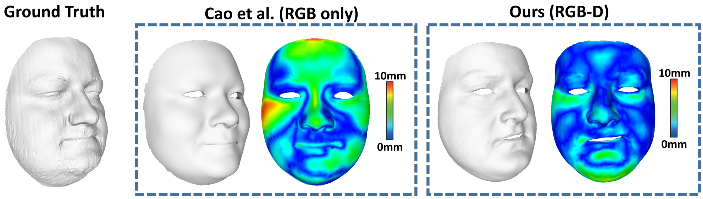


Figure 11.6: State-of-the-art comparison for fitting the shape identity on a neutral expression. From left to right: (a) structured light scan (ground truth), (b) result of [CHZ14], (c) our result. Using depth data allows us to achieve a better identity fit.

In Fig. 11.7, we compare our 3D tracking quality to Cao et al. [CHZ14] 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. [CHZ14] only track a sparse set of features. While less accurate, their method is significantly faster and runs in real-time even on mobile phones.

11.0.2 Facial Reenactment

The core of our approach is the live facial reenactment setup as shown in Fig. 4.1. Fig. 11.9 shows examples of three different actor pairs, with the tracked source and target shown at the top and the reenactment at the bottom. As can be seen, we are able to track various kinds of expressions resulting in a photo-realistic reenactment.

11.0.3 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.

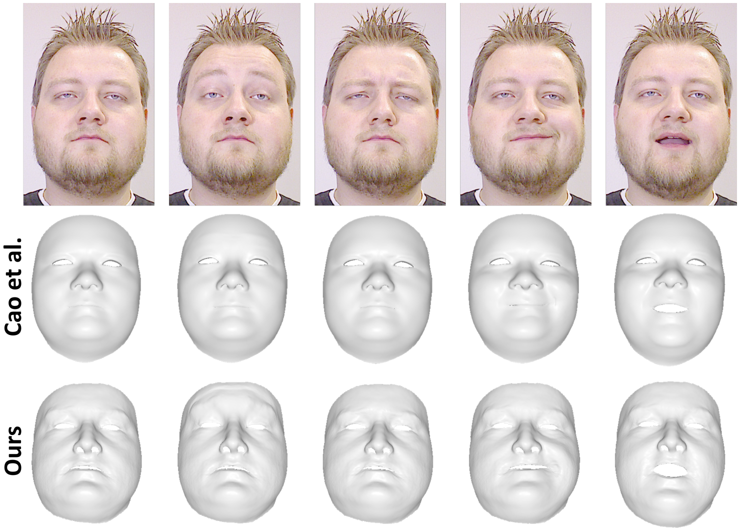


Figure 11.7: State-of-the-art comparison for fitting shape expressions (i.e., tracking) assuming a fixed shape identity (cf. Fig. 11.6). From top to bottom: (a) input color sequence, (b) result of [CHZ14] (RGB input), (c) our result (RGB-D input).

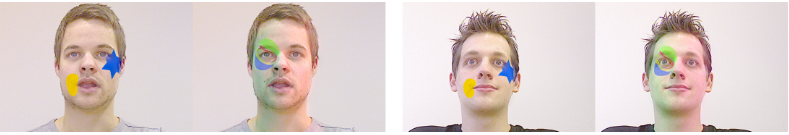


Figure 11.8: Re-texturing and re-lighting a facial performance.

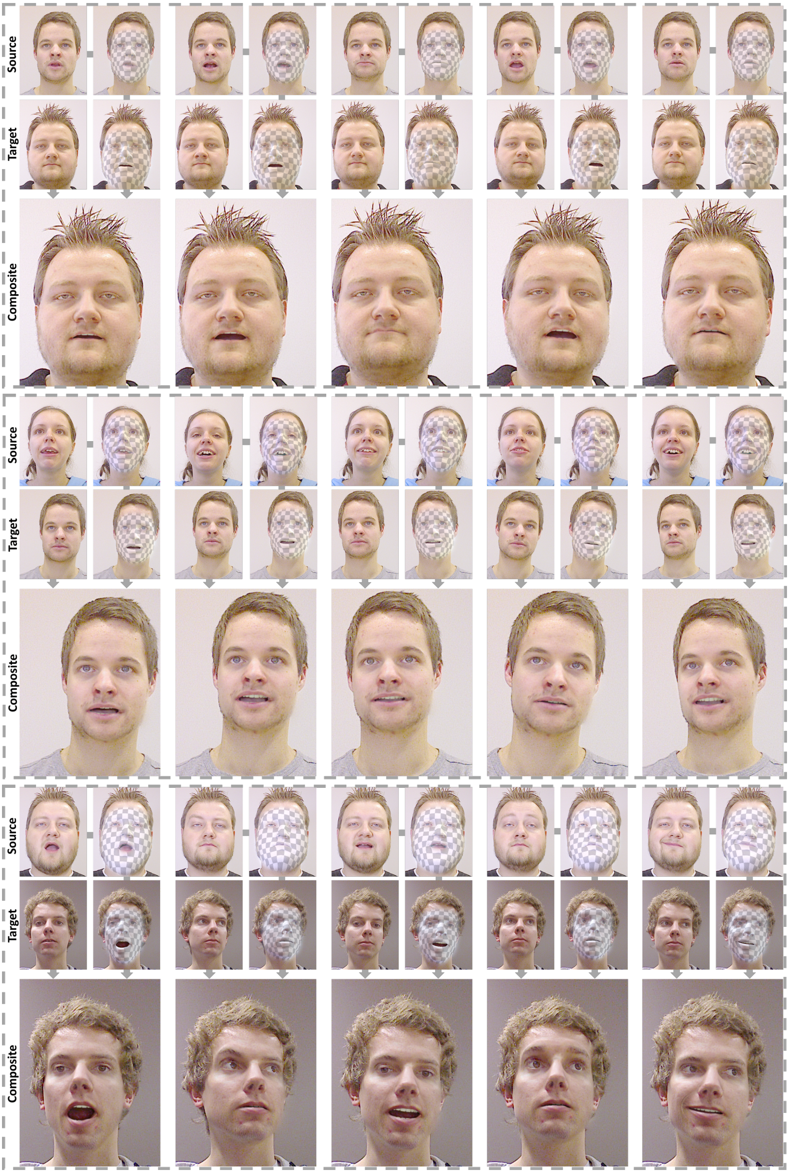


Figure 11.9: Results of our reenactment system. The gray arrows show the workflow of our method.

CHAPTER 12

Conclusion and Discussion

Our method employs a new GPU-based face tracking algorithm using a single RGB-D camera. It fits a parametric face model of shape identity, face expression, and detailed albedo to RGB and depth cues, and continuously re-estimates the scene illumination. 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. [CHZ14], 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 [LYYB13, BWP13] 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).

To conclude, 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 teleconferencing. 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.

PART II

Face2Face: Real-time Face Capture and Reenactment of RGB Videos

CHAPTER 13

Introduction

In recent years, real-time markerless facial performance capture based on commodity sensors has been demonstrated. Impressive results have been achieved, both based on RGB [CWLZ13, CBZB15] as well as RGB-D data [WBLP11, CWS*13, LYYB13, BWP13, HMYL15]. These techniques have become increasingly popular for the animation of virtual CG avatars in video games and movies. It is now feasible to run these face capture and tracking algorithms from home, which is the foundation for many VR and AR applications, such as teleconferencing.

In this paper, we employ a new dense markerless facial performance capture method based on monocular RGB data, similar to state-of-the-art methods. However, instead of transferring facial expressions to virtual CG characters, our main contribution is monocular *facial reenactment* in real-time. In contrast to previous reenactment approaches that run offline [BCS97, DSJ*11, GVR*14], our goal is the *online* transfer of facial expressions of a source actor captured by an RGB sensor to a target actor. The target sequence can be any monocular video; e.g., legacy video footage downloaded from Youtube with a facial performance. We aim to modify the tar-

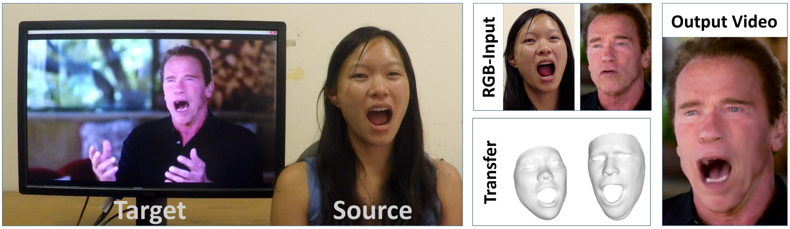


Figure 13.1: Proposed online reenactment setup: a monocular target video sequence (e.g., from Youtube) is reenacted based on the expressions of a source actor who is recorded live with a commodity webcam.

get video in a photo-realistic fashion, such that it is virtually impossible to notice the manipulations. Faithful photo-realistic facial reenactment is the foundation for a variety of applications; for instance, in video conferencing, the video feed can be adapted to match the face motion of a translator, or face videos can be convincingly dubbed to a foreign language.

In our method, we first reconstruct the shape identity of the target actor using a new global non-rigid model-based bundling approach based on a prerecorded training sequence. As this preprocess is performed globally on a set of training frames, we can resolve geometric ambiguities common to monocular reconstruction. At runtime, we track both the expressions of the source and target actor's video by a dense analysis-by-synthesis approach based on a statistical facial prior. We demonstrate that our RGB tracking accuracy is on par with the state of the art, even with online tracking methods relying on depth data. In order to transfer expressions from the source to the target actor in real-time, we propose a novel transfer functions that efficiently applies deformation transfer [SP04] directly in the used low-dimensional expression space. For final image synthesis, we re-render the target's face with transferred expression coefficients and composite it with the target video's background under consideration of the estimated environment lighting. Finally, we introduce a new image-based mouth synthesis approach that generates a realistic mouth interior by retrieving and warping best matching mouth shapes from the offline sample sequence. It is important to note that we maintain the appearance of the target mouth shape; in contrast, existing methods either copy the source mouth region onto the target [VBPP05, DSJ*11] or a generic teeth proxy is rendered [GVS*15, TZN*15], both of which leads to inconsistent results. Fig. 13.2 shows an overview of our method.

We demonstrate highly-convincing transfer of facial expressions from a source to a target video in real time. We show results with a live setup where a source video stream, which is captured by a webcam, is used to manipulate a target Youtube video. In addition, we compare against state-of-the-art reenactment methods, which we outperform both in terms of resulting video quality and runtime (we are the first real-time RGB reenactment method).

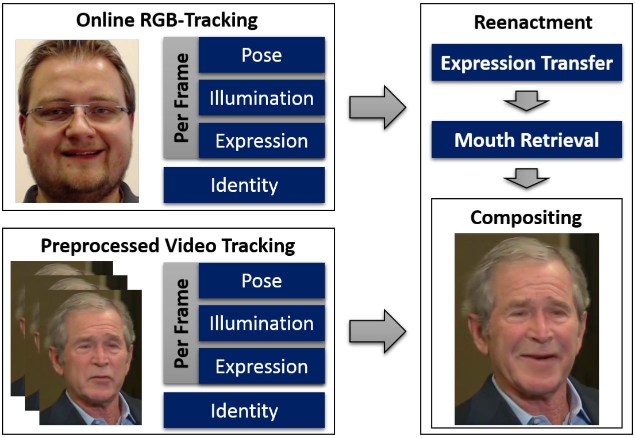


Figure 13.2: Method overview.

In summary, our key contributions are:

- dense, global non-rigid model-based bundling,
- accurate tracking, appearance, and lighting estimation in unconstrained live RGB video,
- person-dependent expression transfer using subspace deformations,
- and a novel mouth synthesis approach.

CHAPTER 14

Related Work

Offline RGB Performance Capture Recent offline performance capture techniques approach the hard monocular reconstruction problem by fitting a blendshape [GVWT13] or a multi-linear face [SWTC14] model to the input video sequence. Even geometric fine-scale surface detail is extracted via inverse shading-based surface refinement. Ichim et al. [IBP15] build a personalized face rig from just monocular input. They perform a structure-from-motion reconstruction of the static head from a specifically captured video, to which they fit an identity and expression model. Person-specific expressions are learned from a training sequence. Suwajanakorn et al. [SKS14] learn an identity model from a collection of images and track the facial animation based on a model-to-image flow field. Shi et al. [SWTC14] achieve impressive results based on global energy optimization of a set of selected keyframes. Our model-based bundling formulation to recover actor identities is similar to their approach; however, we use robust and dense global photometric alignment, which we enforce with an efficient data-parallel optimization strategy on the GPU.

Online RGB-D Performance Capture Weise et al. [WLGP09] capture facial performances in real-time by fitting a parametric blendshape model to RGB-D data, but they require a professional, custom capture setup. The first real-time facial performance capture system based on a commodity depth sensor has been demonstrated by Weise et al. [WBLP11]. Follow up work [LYYB13, BWP13, CWS*13, HMYL15] focused on corrective shapes [BWP13], dynamically adapting the blendshape basis [LYYB13], non-rigid mesh deformation [CWS*13], and robustness against occlusions [HMYL15]. These works achieve impressive results, but rely on depth data which is typically unavailable in most video footage.

Online RGB Performance Capture While many sparse real-time face trackers exist, e.g., [SLC11b], real-time dense monocular tracking is the basis of realistic online facial reenactment. Cao et al. [CWLZ13] propose a real-time regression-based approach to infer 3D positions of facial landmarks which constrain a user-specific blendshape model. Follow-up work [CBZB15] also regresses fine-scale face wrinkles. These methods achieve impressive results, but can not directly be used as a component in facial reenactment, since they do not facilitate dense, pixel-accurate tracking.

Offline Reenactment Vlasic et al. [VBPP05] perform facial reenactment by tracking a face template, which is re-rendered under different expression parameters on top of the target; the mouth interior is directly copied from the source video. Dale et al. [DSJ*11] achieve impressive results using a parametric model, but they target face replacement and compose the source face over the target. Image-based offline mouth re-animation was shown in [BCS97]. Garrido et al. [GVR*14] propose an automatic purely image-based approach to replace the entire face. These approaches merely enable self-reenactment; i.e., when source and target are the same person; in contrast, we perform reenactment of a different target actor. Recent work presents virtual dubbing [GVS*15], a problem similar to ours; however, the method runs at slow offline rates and relies on a generic teeth proxy for the mouth interior. Kemelmacher et al. [KSSGS11] generate face animations from large image collections, but the obtained results lack temporal coherence. Li et al. [LXW*12] retrieve frames from a database based on a similarity metric. They use optical flow as appearance and velocity measure and search for the k-nearest neighbors based on time stamps and flow distance. Saragih et al. [SLC11b] present a real-time avatar animation system from a single image. Their approach is based on sparse landmark tracking, and the mouth of the source is copied to the target using texture warping. Berthouzoz et al. [BLA12] find a flexible number of in-between frames for a video sequence using shortest path search on a graph that encodes frame similarity. Kawai et al. [KIM*14] re-synthesize the inner mouth for a given frontal 2D animation using a tooth and tongue image database; they are limited to frontal poses, and do not produce as realistic renderings as ours under general head motion.

Online Reenactment Recently, first online facial reenactment approaches based on RGB-(D) data have been proposed. Kemelmacher-Shlizerman et al. [KSSS10] enable image-based puppetry by querying similar images from a database. They employ an appearance cost metric and consider rotation angular distance, which is similar to Kemelmacher et al. [KSSGS11]. While they achieve impressive results, the retrieved stream of faces is not temporally coherent. Thies et al. [TZN*15] show the first online reenactment system; however, they rely on depth data and use a generic teeth proxy for the mouth region. In this paper, we address both shortcomings: 1) our method is the first real-time RGB-only reenactment technique; 2) we synthesize the mouth regions exclusively from the target sequence (no need for a teeth proxy or direct source-to-target copy).

CHAPTER 15

Synthesis of Facial Imagery

We use a multi-linear PCA model based on [BV99, ARL*09, CWZ*14]. The first two dimensions represent facial identity – i.e., geometric shape and skin reflectance – and the third dimension controls the facial expression. Hence, we parametrize a face as:

$$\begin{aligned} \mathcal{M}_{\text{geo}}(\alpha, \delta) &= a_{\text{id}} + E_{\text{id}} \cdot \alpha + E_{\text{exp}} \cdot \delta \,, && (15.1) \\ \mathcal{M}_{\text{alb}}(\beta) &= a_{\text{alb}} + E_{\text{alb}} \cdot \beta \;. && (15.2) \end{aligned}$$

This prior assumes a multivariate normal probability distribution of shape and reflectance around the average shape $a_{\text{id}} \in \mathbb{R}^{3n}$ and reflectance $a_{\text{alb}} \in \mathbb{R}^{3n}$. The shape $E_{\text{id}} \in \mathbb{R}^{3n \times 80}$, reflectance $E_{\text{alb}} \in \mathbb{R}^{3n \times 80}$, and expression $E_{\text{exp}} \in \mathbb{R}^{3n \times 76}$ basis and the corresponding standard deviations $\sigma_{\text{id}} \in \mathbb{R}^{80}$, $\sigma_{\text{alb}} \in \mathbb{R}^{80}$, and $\sigma_{\text{exp}} \in \mathbb{R}^{76}$ are given. The model has 53K vertices and 106K faces. A synthesized image $C_{\mathcal{S}}$ is generated through rasterization of the model under a rigid model transformation $\Phi(v)$ and the full perspective transformation $\Pi(v)$. Illumination is approximated by the first three bands of Spherical Harmonics (SH) [RH01b] basis functions, assuming Lambertian surfaces and smooth distant illumination, neglecting self-shadowing.

Synthesis is dependent on the face model parameters α, β, δ, the illumination parameters γ, the rigid transformation $\mathbf{R}$, $\mathbf{t}$, and the camera parameters κ defining the perspective transformation Π. The vector of unknowns $\boldsymbol{\mathcal{P}}$ is the union of these parameters.

CHAPTER 16

Energy Formulation

Given a monocular input sequence, we reconstruct all unknown parameters $\mathcal{P}$ jointly with a robust variational optimization. The proposed objective function is highly non-linear in the unknowns and has the following components:

$$E(\boldsymbol{\mathcal{P}}) = \underbrace{w_{col} E_{col}(\boldsymbol{\mathcal{P}}) + w_{lan} E_{lan}(\boldsymbol{\mathcal{P}})}_{data} + \underbrace{w_{reg} E_{reg}(\boldsymbol{\mathcal{P}})}_{prior} \quad . \tag{16.1}$$

The data term measures the similarity between the synthesized imagery and the input data in terms of photo-consistency E_{col} and facial feature alignment E_{lan}. The likelihood of a given parameter vector $\boldsymbol{\mathcal{P}}$ is taken into account by the statistical regularizer E_{reg}. The weights w_{col}, w_{lan}, and w_{reg} balance the three different sub-objectives. In all of our experiments, we use the empirically chosen parameters $w_{col} = 1$, $w_{lan} = 10$, and $w_{reg} = 2.5 \cdot 10^{-5}$. In the following, we introduce the different sub-objectives.

Photo-Consistency We seek to reproduce the appearance of the person in the input image as close as possible

In order to quantify how well the input data is explained by a synthesized image, we measure the photo-metric alignment error on pixel level:

$$E_{\mathrm{col}}(\boldsymbol{\mathcal{P}}) = \frac{1}{|\mathcal{V}|} \sum_{p \in \mathcal{V}} \|C_{\mathcal{S}}(p) - C_{\mathcal{I}}(p)\|_2 \quad , \tag{16.2}$$

where $C_{\mathcal{S}}$ is the synthesized image, $C_{\mathcal{I}}$ is the input RGB image, and $p \in \mathcal{V}$ denote all visible pixel positions in $C_{\mathcal{S}}$. We use the $\ell_{2,1}$-norm [DZHZ06] instead of a least-squares formulation to be robust against outliers. In our scenario, distance in color space is based on ℓ_2, while in the summation over all pixels an ℓ_1-norm is used to enforce sparsity.

Feature Alignment In addition to dense photo-metric alignment, we enforce feature similarity between a set of salient facial feature point pairs detected in the RGB stream:

$$E_{\text{lan}}(\boldsymbol{\mathcal{P}}) = \frac{1}{|\mathcal{F}|} \sum_{f_j \in \mathcal{F}} w_{\text{conf},j} \left\| f_j - \Pi(\Phi(v_j)) \right\|_2^2 . \tag{16.3}$$

To this end, we employ a state-of-the-art facial landmark tracking algorithm by [SLC11a]. Each feature point $f_j \in \mathcal{F} \subset \mathbb{R}^2$ comes with a detection confidence $w_{\text{conf},j}$ and corresponds to a unique vertex $v_j = \mathcal{M}_{geo}(\alpha, \delta) \in \mathbb{R}^3$ of our face prior. This helps avoiding local minima in the highly-complex energy landscape of $E_{\text{col}}(\boldsymbol{\mathcal{P}})$.

Statistical Regularization We enforce plausibility of the synthesized faces based on the assumption of a normal distributed population. To this end, we enforce the parameters to stay statistically close to the mean:

$$E_{\text{reg}}(\boldsymbol{\mathcal{P}}) = \sum_{i=1}^{80} \left[\left(\frac{\alpha_i}{\sigma_{\text{id},i}} \right)^2 + \left(\frac{\beta_i}{\sigma_{\text{alb},i}} \right)^2 \right] + \sum_{i=1}^{76} \left(\frac{\delta_i}{\sigma_{\text{exp},i}} \right)^2 . \tag{16.4}$$

This commonly-used regularization strategy prevents degenerations of the facial geometry and reflectance, and guides the optimization strategy out of local minima [BV99].

CHAPTER 17

Data-parallel Optimization Strategy

The proposed robust tracking objective is a general unconstrained non-linear optimization problem. We minimize this objective in real-time using a novel data-parallel GPU-based *Iteratively Reweighted Least Squares* (IRLS) solver. The key idea of IRLS is to transform the problem, in each iteration, to a non-linear least-squares problem by splitting the norm in two components:

$$||r(\mathcal{P})||_2 = \underbrace{(||r(\mathcal{P}_{old})||_2)^{-1}}_{constant} \cdot ||r(\mathcal{P})||_2^2 \, .$$

Here, $r(\cdot)$ is a general residual and $\mathcal{P}_{old}$ is the solution computed in the last iteration. Thus, the first part is kept constant during one iteration and updated afterwards. Close in spirit to [TZN*15], each single iteration step is implemented using the Gauss-Newton approach. We take a single GN step in every IRLS iteration and solve the corresponding system of normal equations $\mathbf{J}^T\mathbf{J}\delta^* = -\mathbf{J}^T\mathbf{F}$ based on PCG to obtain an optimal linear parameter update δ^*. The Jacobian $\mathbf{J}$ and the systems' right hand side $-\mathbf{J}^T\mathbf{F}$ are precomputed and stored in device memory for later processing as proposed by Thies et al. [TZN*15]. As suggested by [ZNI*14, TZN*15], we split up the multiplication of the old descent direction d with the system matrix $\mathbf{J}^T\mathbf{J}$ in the PCG solver into two successive matrix-vector products. Additional details regarding the optimization framework are provided in the Chapter 21.

CHAPTER 18

Non-Rigid Model-Based Bundling

To estimate the identity of the actors in the heavily underconstrained scenario of monocular reconstruction, we introduce a non-rigid model-based bundling approach. Based on the proposed objective function, we jointly estimate all parameters over k key-frames of the input video sequence. The estimated unknowns are the global identity $\{\alpha, \beta\}$ and intrinsics κ as well as the unknown per-frame pose $\{\delta^k, \mathbf{R}^k, \mathbf{t}^k\}_k$ and illumination parameters $\{\gamma^k\}_k$. We use a similar data-parallel optimization strategy as proposed for model-to-frame tracking, but jointly solve the normal equations for the entire keyframe set. For our non-rigid model-based bundling problem, the non-zero structure of the corresponding Jacobian is block dense. Our PCG solver exploits the non-zero structure for increased performance (see Chapter 21). Since all keyframes observe the same face identity under potentially varying illumination, expression, and viewing angle, we can robustly separate identity from all other problem dimensions. Note that we also solve for the intrinsic camera parameters of Π, thus being able to process uncalibrated video footage.

CHAPTER 19

Expression Transfer

To transfer the expression changes from the source to the target actor while preserving person-specificness in each actor's expressions, we propose a sub-space deformation transfer technique. We are inspired by the deformation transfer energy of Sumner et al. [SP04], but operate directly in the space spanned by the expression blendshapes. This not only allows for the precomputation of the pseudo-inverse of the system matrix, but also drastically reduces the dimensionality of the optimization problem allowing for fast real-time transfer rates. Assuming source identity α^S and target identity α^T are fixed, transfer takes as input the neutral δ_N^S, deformed source δ^S, and the neutral target δ_N^T expression. Output is the transferred facial expression δ^T directly in the reduced sub-space of the parametric prior.

As proposed by [SP04], we first compute the source deformation gradients $\mathbf{A}_i \in \mathbb{R}^{3\times3}$ that transform the source triangles from neutral to deformed. The deformed target $\hat{v}_i = M_i(\alpha^T, \delta^T)$ is then found based on the undeformed state $v_i = M_i(\alpha^T, \delta_N^T)$ by solving a linear least-squares problem. Let (i_0, i_1, i_2) be the vertex indices of the i-th triangle, $\mathbf{V} = [v_{i_1} - v_{i_0}, v_{i_2} - v_{i_0}]$ and $\hat{\mathbf{V}} = [\hat{v}_{i_1} - \hat{v}_{i_0}, \hat{v}_{i_2} - \hat{v}_{i_0}]$, then the optimal unknown target deformation δ^T is the minimizer of:

$$E(\delta^T) = \sum_{i=1}^{|F|} \left|\left|\mathbf{A}_i\mathbf{V} - \hat{\mathbf{V}}\right|\right|_F^2 . \tag{19.1}$$

This problem can be rewritten in the canonical least-squares form by substitution:

$$E(\delta^T) = \left|\left|\mathbf{A}\delta^T - b\right|\right|_2^2 . \tag{19.2}$$

The matrix $\mathbf{A} \in \mathbb{R}^{6|F|\times76}$ is constant and contains the edge information of the template mesh projected to the expression sub-space. Edge information of the target in neutral expression is included in the right-hand side $b \in \mathbb{R}^{6|F|}$. b varies with δ^S and is computed on the GPU for each new input frame. The minimizer of the quadratic energy can be computed by solving

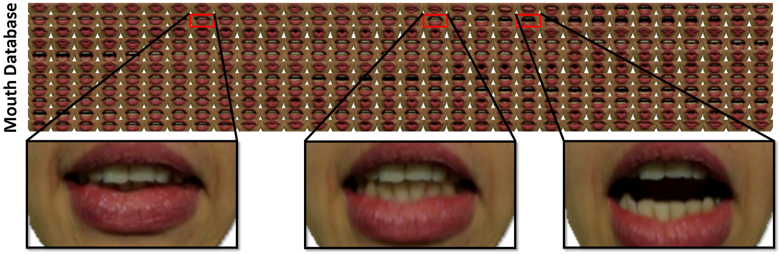


Figure 19.1: Mouth Retrieval: we use an appearance graph to retrieve new mouth frames. In order to select a frame, we enforce similarity to the previously-retrieved frame while minimizing the distance to the target expression.

the corresponding normal equations. Since the system matrix is constant, we can precompute its *Pseudo Inverse* using a Singular Value Decomposition (SVD). Later, the small 76×76 linear system is solved in real-time. No additional smoothness term as in [SP04, BWP13] is needed, since the blendshape model implicitly restricts the result to plausible shapes and guarantees smoothness.

19.1 Mouth Retrieval

For a given transferred facial expression, we need to synthesize a realistic target mouth region. To this end, we retrieve and warp the best matching mouth image from the target actor sequence. We assume that sufficient mouth variation is available in the target video. It is also important to note that we maintain the appearance of the target mouth. This leads to much more realistic results than either copying the source mouth region [VBPP05, DSJ*11] or using a generic 3D teeth proxy [GVS*15, TZN*15].

Our approach first finds the best fitting target mouth frame based on a frame-to-cluster matching strategy with a novel feature similarity metric. To enforce temporal coherence, we use a dense appearance graph to find a compromise between the last retrieved mouth frame and the target mouth frame (cf. Fig. 19.1). We detail all steps in the following.

Similarity Metric Our similarity metric is based on geometric and photometric features. The used descriptor $\mathcal{K} = \{\mathbf{R}, \delta, \mathcal{F}, \mathcal{L}\}$ of a frame is composed of the rotation $\mathbf{R}$, expression parameters δ, landmarks $\mathcal{F}$, and a Local Binary Pattern (LBP) $\mathcal{L}$. We compute these descriptors $\mathcal{K}^S$ for every frame in the training sequence. The target descriptor $\mathcal{K}^T$ consists of the result of the expression transfer and the LBP of the frame of the driving actor. We measure the distance between a source and a target descriptor as follows:

$$D(\mathcal{K}^T, \mathcal{K}^S_t, t) = D_p(\mathcal{K}^T, \mathcal{K}^S_t) + D_m(\mathcal{K}^T, \mathcal{K}^S_t) + D_a(\mathcal{K}^T, \mathcal{K}^S_t, t) \,.$$

The first term D_p measures the distance in parameter space:

$$D_p(\mathcal{K}^T, \mathcal{K}^S_t) = \|\delta^T - \delta^S_t\|_2^2 + \|\mathbf{R}^T - \mathbf{R}^S_t\|_F^2 \,.$$

The second term D_m measures the differential compatibility of the sparse facial landmarks:

$$D_m(\mathcal{K}^T, \mathcal{K}^S_t) = \sum_{(i,j)\in\Omega} \left(\|\mathcal{F}^T_i - \mathcal{F}^T_j\|_2 - \|\mathcal{F}^S_{t,i} - \mathcal{F}^S_{t,j}\|_2 \right)^2 \,.$$

Here, Ω is a set of predefined landmark pairs, defining distances such as between the upper and lower lip or between the left and right corner of the mouth. The last term D_a is an appearance measurement term composed of two parts:

$$D_a(\mathcal{K}^T, \mathcal{K}^S_t, t) = D_l(\mathcal{K}^T, \mathcal{K}^S_t) + w_c(\mathcal{K}^T, \mathcal{K}^S_t) D_c(\tau, t) \,.$$

τ is the last retrieved frame index used for the reenactment in the previous frame. $D_l(\mathcal{K}^T, \mathcal{K}^S_t)$ measures the similarity based on LBPs that are compared via a *Chi Squared Distance* (for details see [GVR*14]). $D_c(\tau, t)$ measures the similarity between the last retrieved frame τ and the video frame t based on RGB cross-correlation of the normalized mouth frames. Note that the mouth frames are normalized based on the models texture parameterization (cf. Fig. 19.1). To facilitate fast frame jumps for expression changes, we incorporate the weight $w_c(\mathcal{K}^T, \mathcal{K}_t^S) = e^{-(D_m(\mathcal{K}^T, \mathcal{K}_t^S))^2}$. We apply this frame-to-frame distance measure in a frame-to-cluster matching strategy, which enables real-time rates and mitigates high-frequency jumps between mouth frames.

Frame-to-Cluster Matching Utilizing the proposed similarity metric, we cluster the target actor sequence into $k = 10$ clusters using a modified k-means algorithm that is based on the pairwise distance function D. For every cluster, we select the frame with the minimal distance to all other frames within that cluster as a representative. During runtime, we measure the distances between the target descriptor $\mathcal{K}^T$ and the descriptors of cluster representatives, and choose the cluster whose representative frame has the minimal distance as the new target frame.

Appearance Graph We improve temporal coherence by building a fully-connected appearance graph of all video frames. The edge weights are based on the RGB cross-correlation between the normalized mouth frames, the distance in parameter space D_p, and the distance of the landmarks D_m. The graph enables us to find an inbetween frame that is both similar to the last retrieved frame and the retrieved target frame (see Fig. 19.1). We compute this perfect match by finding the frame of the training sequence that minimizes the sum of the edge weights to the last retrieved and current target frame. We blend between the previously-retrieved frame and the newly-retrieved frame in texture space on a pixel level after optic flow alignment. Before blending, we apply an illumination correction that considers the estimated Spherical Harmonic illumination parameters of the retrieved frames and the current video frame. Finally, we composite the new output frame by alpha blending between the original video frame, the illumination-corrected, projected mouth frame, and the rendered face model.

CHAPTER 20

Results

Live Reenactment Setup Our live reenactment setup consists of standard consumer-level hardware. We capture a live video with a commodity webcam (source), and download monocular video clips from Youtube (target). In our experiments, we use a *Logitech HD Pro C920* camera running at 30Hz in a resolution of 640 × 480; although our approach is applicable to any consumer RGB camera. Overall, we show highly-realistic reenactment examples of our algorithm on a variety of target Youtube videos at a resolution of 1280 × 720. The videos show different subjects in different scenes filmed from varying camera angles; each video is reenacted by several volunteers as source actors. Reenactment results are generated at a resolution of 1280 × 720. We show real-time reenactment results in Fig. 20.6.

Runtime For all experiments, we use three hierarchy levels for tracking (source and target). In pose optimization, we only consider the second and third level, where we run one and seven Gauss-Newton steps, respectively. Within a Gauss-Newton step, we always run four PCG steps. In addition to tracking, our reenactment pipeline has additional stages whose timings are listed in Table 20.1. Our method runs in real-time on a commodity desktop computer with an NVIDIA Titan X and an Intel Core i7-4770.

CPU		GPU			FPS
SparseFT	MouthRT	DenseFT	DeformTF	Synth	
5.97ms	1.90ms	22.06ms	3.98ms	10.19ms	**27.6Hz**
4.85ms	1.50ms	21.27ms	4.01ms	10.31ms	**28.1Hz**
5.57ms	1.78ms	20.97ms	3.95ms	10.32ms	**28.4Hz**

Table 20.1: Avg. run times for the three sequences of Fig. 20.6, from top to bottom. Standard deviations w.r.t. the final frame rate are 0.51, 0.56, and 0.59 fps, respectively. Note that CPU and GPU stages run in parallel.

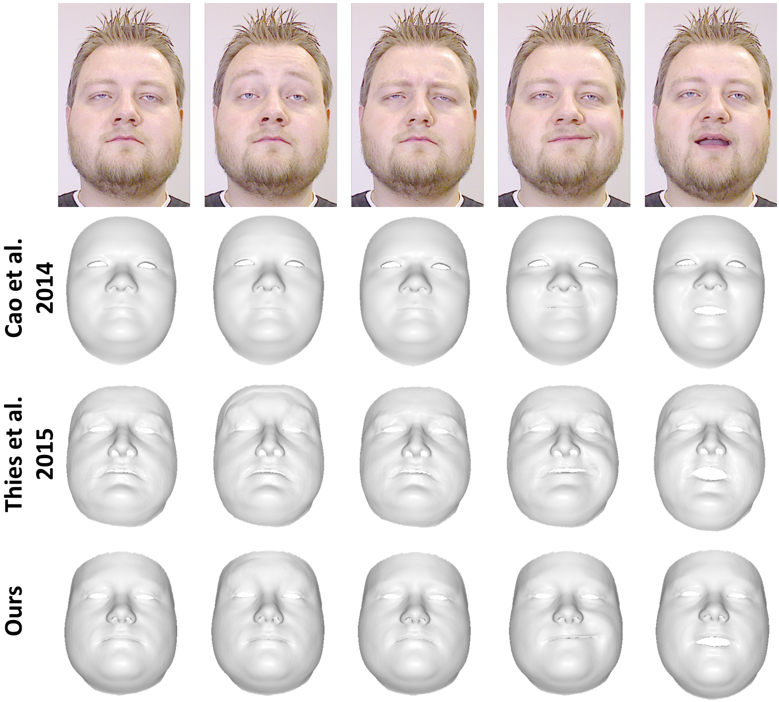


Figure 20.1: Comparison of our RGB tracking to Cao et al. [CHZ14], and to RGB-D tracking by Thies et al. [TZN*15].

Tracking Comparison to Previous Work Face tracking alone is not the main focus of our work, but the following comparisons show that our tracking is on par with or exceeds the state of the art.

Shi et al. 2014 [SWTC14]: They capture face performances offline from monocular unconstrained RGB video. The close-ups in Fig. 20.2 show that our online approach yields a closer face fit, particularly visible at the silhouette of the input face. We believe that our new dense non-rigid bundle adjustment leads to a better shape identity estimate than their sparse approach.

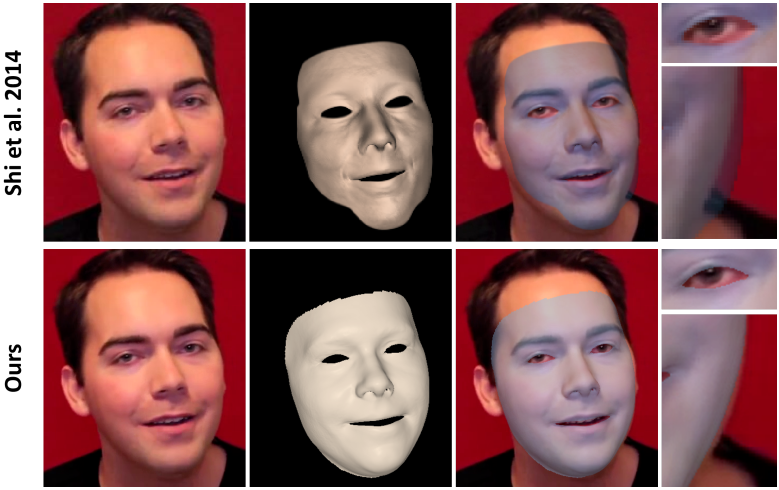


Figure 20.2: Comparison of our tracking to Shi et al. [SWTC14]. From left to right: RGB input, reconstructed model, overlay with input, close-ups on eye and cheek. Note that Shi et al. perform shape-from-shading in a post process.

Cao et al. 2014 [CHZ14]: They capture face performance from monocular RGB in real-time. In most cases, our and their method produce similar high-quality results (see Fig. 20.1); our identity and expression estimates are slightly more accurate though.

*Thies et al. 2015 [TZN*15]:* Their approach captures face performance in real-time from RGB-D, Fig. 20.1. Results of both approaches are similarly accurate; but our approach does not require depth data.

FaceShift 2014: We compare our tracker to the commercial real-time RGB-D tracker from *FaceShift*, which is based on the work of Weise et al. [WBLP11]. Fig. 20.3 shows that we obtain similar results from RGB only.

Reenactment Evaluation In Fig. 20.4, we compare our approach against state-of-the art reenactment by Garrido et al. [GVS*15]. Both methods provide highly-realistic reenactment results; however, their method is fun-

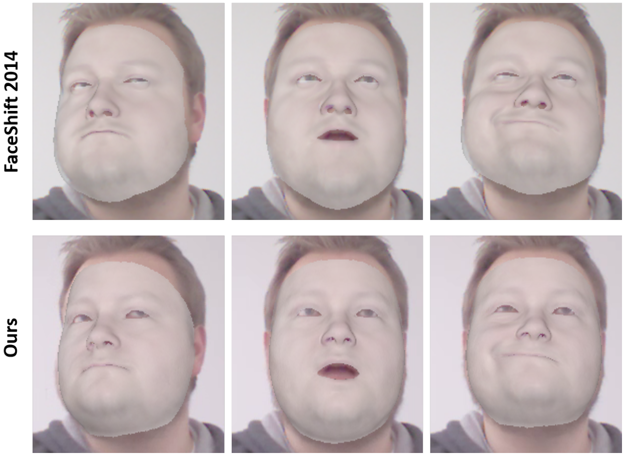


Figure 20.3: Comparison against *FaceShift* RGB-D tracking.

damentally offline, as they require all frames of a sequence to be present at any time. In addition, they rely on a generic geometric teeth proxy which in some frames makes reenactment less convincing. In Fig. 20.5, we compare against the work by Thies et al. [TZN*15]. Runtime and visual quality are similar for both approaches; however, their geometric teeth proxy leads to undesired appearance changes in the reenacted mouth. Moreover, Thies et al. use an RGB-D camera, which limits the application range; they cannot reenact Youtube videos. We show additional comparisons in Chapter 21 against Dale et al. [DSJ*11] and Garrido et al. [GVR*14].

20.1 Limitations

The assumption of Lambertian surfaces and smooth illumination is limiting, and may lead to artifacts in the presence of hard shadows or specular highlights; a limitation shared by most state-of-the-art methods. Scenes with face occlusions by long hair and a beard are challenging. Furthermore, we only reconstruct and track a low-dimensional blendshape model (76 ex-

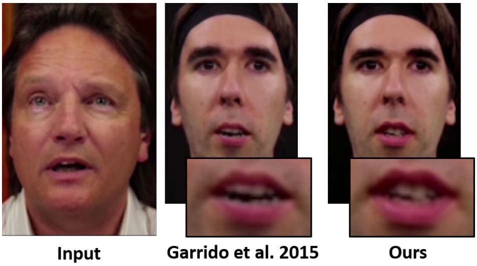


Figure 20.4: Dubbing: Comparison to Garrido et al. [GVS*15].

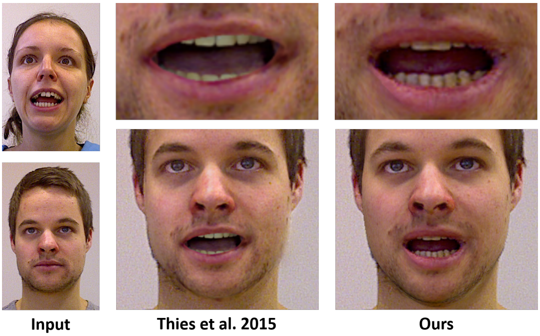


Figure 20.5: Comparison of the proposed RGB reenactment to the RGB-D reenactment of Thies et al. [TZN*15].

pression coefficients), which omits fine-scale static and transient surface details. Our retrieval-based mouth synthesis assumes sufficient visible expression variation in the target sequence. On a too short sequence, or when the target remains static, we cannot learn the person-specific mouth behavior. In this case, temporal aliasing can be observed, as the target space of the retrieved mouth samples is too sparse. Another limitation is caused by our hardware setup (webcam, USB, and PCI), which introduces a small delay of ≈ 3 frames. Specialized hardware could resolve this, but our aim is a setup with commodity hardware.

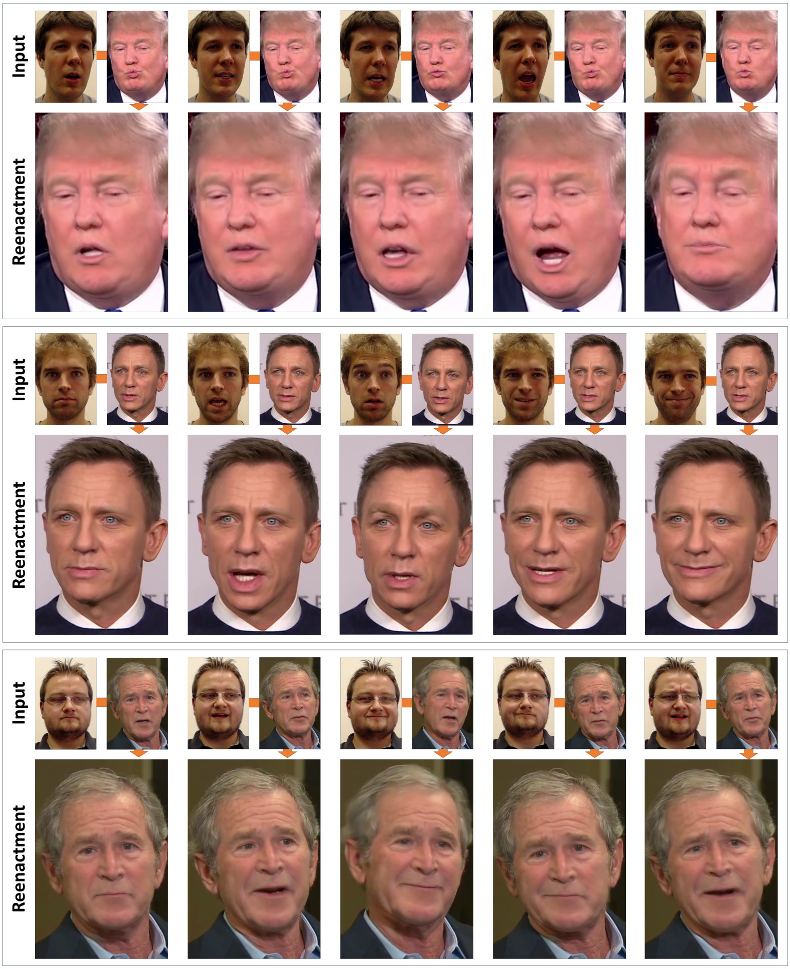


Figure 20.6: Results of our reenactment system. Corresponding run times are listed in Table 20.1. The length of the source and resulting output sequences is 965, 1436, and 1791 frames, respectively; the length of the input target sequences is 431, 286, and 392 frames, respectively.

20.2 Conclusion

The presented approach is the first real-time facial reenactment system that requires just monocular RGB input. Our live setup enables the animation of legacy video footage – e.g., from Youtube – in real time. Overall, we believe our system will pave the way for many new and exciting applications in the fields of VR/AR, teleconferencing, or on-the-fly dubbing of videos with translated audio.

CHAPTER 21

Appendix: ”Face2Face: Real-time Face Capture and Reenactment of RGB Videos”

In this chapter, we provide additional information to the Face2Face method [TZS*16b]. More specifically, we include additional detail about our optimization framework (see Section 21.1 and 21.2), and we show further comparisons to other methods (see Section 21.3). We also evaluate the reconstruction error in a self-reenactment scenario.

21.1 Optimization Framework

Our Gauss-Newton optimization framework is based on the work of Thies et al. [TZN*15]. Our aim is to include every visible pixel $p \in \mathcal{V}$ in $C_{\mathcal{S}}$ in the optimization process. To this end, we gather all visible pixels in the synthesized image using a parallel prefix scan. The computation of the Jacobian J of the residual vector F and the gradient $J^T F$ of the energy function are then parallelized across all GPU processors. This parallelization is feasible since all partial derivatives and gradient entries with respect to a variable can be computed independently. During evaluation of the gradient, all components of the Jacobian are computed and stored in global memory. In order to evaluate the gradient, we use a two-stage reduction to sum-up all local per pixel gradients. Finally, we add the regularizer and the sparse feature term to the Jacobian and the gradient.

Using the computed Jacobian J and the gradient $J^T F$, we solve the corresponding normal equation $J^T J \Delta x = -J^T F$ for the parameter update Δx using a preconditioned conjugate gradient (PCG) method. We apply a Jacobi preconditioner that is precomputed during the evaluation of the gradient.

To avoid the high computational cost of J^TJ, our GPU-based PCG method splits up the computation of J^TJp into two successive matrix-vector products.

Our complete framework is implemented using DirectX for rendering and DirectCompute for optimization. The joint graphics and compute capability of DirectX11 enables the processing of rendered images by the graphics pipeline without resource mapping overhead. In the case of an *analysis-by-synthesis approach* like ours, this is essential to runtime performance, since many rendering-to-compute switches are required.

21.2 Non-rigid Bundling

For our non-rigid model-based bundling problem, the non-zero structure of the corresponding Jacobian is block dense. We visualize its non-zero structure, which we exploit during optimization, in Fig. 21.1. In order to leverage the sparse structure of the Jacobian, we adopt the Gauss-Newton framework as follows: we modify the computation of the gradient $J^T(\mathcal{P}) \cdot F(\mathcal{P})$ and the matrix vector product $J^T(\mathcal{P}) \cdot J(\mathcal{P}) \cdot x$ that is used in the PCG method. To this end, we define a promoter function $\Psi_f : \mathbb{R}^{|\mathcal{P}_{global}|+|\mathcal{P}_{local}|} \rightarrow \mathbb{R}^{|\mathcal{P}_{global}|+k\cdot|\mathcal{P}_{local}|}$ that lifts a per frame parameter vector to the parameter vector space of all frames (Ψ_f^{-1} is the inverse of this

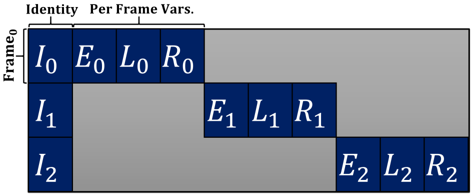


Figure 21.1: Non-zero structure of the Jacobian matrix of our non-rigid model-based bundling approach for three key-frames. Where I_i, E_i, L_i, R_i are the i-th per frame Jacobian matrices of the identity, expression, illumination, and rigid pose parameters.

promoter function). $\mathcal{P}_{global}$ are the global parameters that are shared over all frames, such as the identity parameters of the face model and the camera parameters. $\mathcal{P}_{local}$ are the local parameters that are only valid for one specific frame (i.e., facial expression, rigid pose and illumination parameters). Using the promoter function Ψ_f the gradient is given as

$$J^T(\boldsymbol{\mathcal{P}}) \cdot F(\boldsymbol{\mathcal{P}}) = \sum_{f=1}^{k} \Psi_f(J_f^T(\Psi_f^{-1}(\boldsymbol{\mathcal{P}})) \cdot F_f(\Psi_f^{-1}(\boldsymbol{\mathcal{P}}))),$$

where J_f is the per-frame Jacobian matrix and F_f the corresponding residual vector.

As for the parameter space, we introduce another promoter function $\hat{\Psi}_f$ that lifts a local residual vector to the global residual vector. In contrast to the parameter promoter function, this function varies in every Gauss-Newton iteration since the number of residuals might change. As proposed in [ZNI*14, TZN*15], we split up the computation of $J^T(\boldsymbol{\mathcal{P}}) \cdot J(\boldsymbol{\mathcal{P}}) \cdot x$ into two successive matrix vector products, where the second multiplication is analogue to the computation of the gradient. The first multiplication is as follows:

$$J(\boldsymbol{\mathcal{P}}) \cdot x = \sum_{f=1}^{k} \hat{\Psi}_f \left(J_f(\Psi_f^{-1}(\boldsymbol{\mathcal{P}})) \cdot \Psi_f^{-1}(x) \right)$$

Using this scheme, we are able to efficiently solve the normal equations.

The Gauss-Newton framework is embedded in a hierarchical solution strategy (see Fig. 21.2). This hierarchy allows to prevent convergence to local minima. We start optimizing on a coarse level and propagate the solution to the next finer level using the parametric face model. In our experiments we used three levels with 25, 5, and 1 Gauss-Newton iterations for the coarsest, the medium and the finest level respectively, each with 4 PCG steps. Our implementation is not restricted to the number k of used keyframes. The processing time is linear in the number of keyframes. In our experiments we used $k = 6$ keyframes to estimate the identity parameters resulting in a processing time of a few seconds ($\sim 20s$).

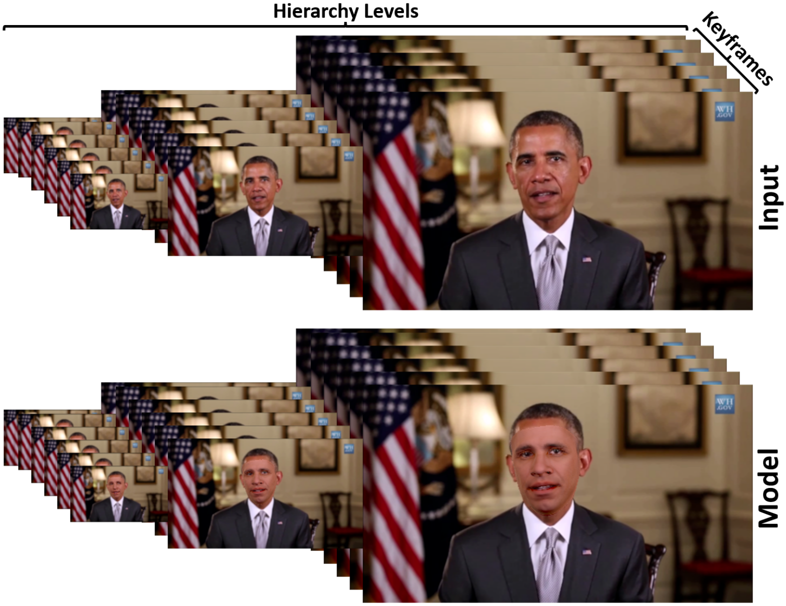


Figure 21.2: Non-rigid model-based bundling hierarchy: the top row shows the hierarchy of the input video and the second row the overlaid face model.

21.3 Reenactment Evaluation

In addition to the results in the main paper [TZS*16b], we compare our method to other existing reenactment pipelines. Fig. 21.3 shows a self-reenactment scenario (i.e., the source and the target actor is the same person) in comparison to Garrido et al. [GVR*14]. Our online approach is able to achieve similar or better quality as the offline approach of Garrido et al. [GVR*14]. In Fig. 21.4, we show a comparisons to Dale et al. [DSJ*11] and Garrido et al. [GVR*14]. Note that both methods do not preserve the identity of the target actor outside of the self-reenactment scenario. In contrast, our method preserves the identity and alters the expression with respect to the source actor, which enables more plausible results.

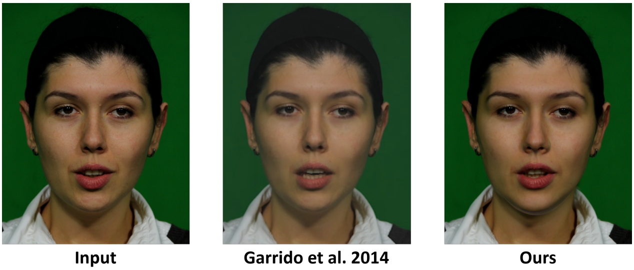


Figure 21.3: Self-Reenactment comparison to Garrido et al. [GVR*14]. The expression of the actress is transferred to a recorded video of herself.

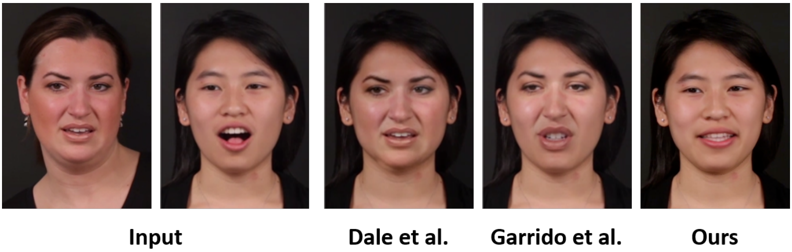


Figure 21.4: Comparison to Dale et al. [DSJ*11] and Garrido et al. [GVR*14]. The expression of the left input actor is transferred to the right input actor without changing the person's identity.

We evaluate the presented reenactment method by measuring the photometric error between the input sequence and the self-reenactment of an actor using cross-validation (see Fig. 21.5). The first 1093 frames of the video are used to retrieve mouth interiors (training data). Thus, self-reenactment of the first half results in a small mean photometric error of 0.33 pixels (0.157px std.Dev.) measured via optical flow. In the second half (frames 1093-2186) of the video, the photometric error increases to a mean value of 0.42 pixels (0.17px std.Dev.).

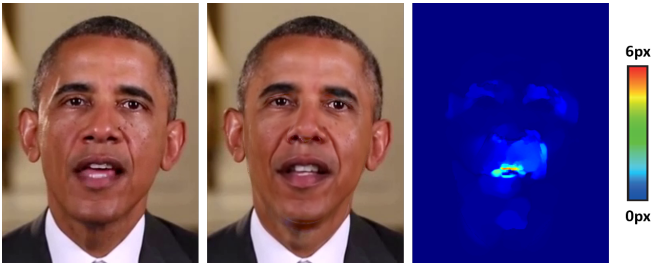


Figure 21.5: Self-Reenactment / Cross-Validation; from left to right: input frame (ground truth), resulting self-reenactment, and the photometric error.

PART III

FaceVR: Real-Time Facial Reenactment and Eye Gaze Control in Virtual Reality

CHAPTER 22

Introduction

Modern head-mounted virtual reality displays, such as the Oculus Rift™ or the HTC Vive™, are able to provide very believable and highly immersive stereo renderings of virtual environments to a user. In particular, for teleconferencing scenarios, where two or more people at distant locations meet (virtually) face-to-face in a virtual meeting room, VR displays can provide a far more immersive and connected atmosphere than today's teleconferencing systems. These teleconferencing systems usually employ one or several video cameras at each end to film the participants, whose video(s) are then shown on one or several standard displays at the other end. Imagine one could take this to the next level, and two people in a VR teleconference would each see a photo-realistic 3D rendering of their actual conversational partner, not simply an avatar, but in their own HMD. The biggest obstacle

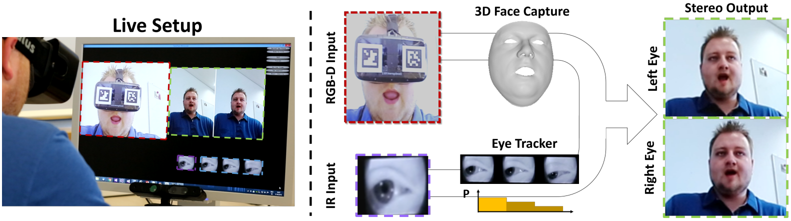


Figure 22.1: We present *FaceVR*, a novel method to perform real-time gaze-aware facial reenactment with a virtual reality device (left). In order to capture a face, we use a commodity RGB-D sensor with a frontal view; the eye region is tracked using a new data-driven approach based on data from an IR camera located inside the head-mounted display. Using the 3D reconstructed face as an intermediate, we can modify and edit the face, as well as re-render it in a photo-realistic fashion, allowing for a variety of applications; e.g., removal of VR goggles or gaze re-targeting. In addition, we render our output in stereo (right), which enables display on stereo devices such as other VR headsets.

in making this a reality is that while the HMD allows for very immersive rendering, it is a large physical device which occludes the majority of the face. In other words, even if each participant of a teleconference was recorded with a 3D video rig, whose feed is streamed to the other end's HMD, natural conversation is not possible due to the display occluding most of the face.

Recent advancements in VR displays are flanked by great progress in face performance capture methods. State-of-the-art approaches enable dense reconstruction of dynamic face geometry in real-time, from RGB-D [WBLP11, BWP13, LYYB13, ZNI*14, HMYL15] or even RGB cameras [CHZ14, CBZB15, TZS*16b]. A further step has been taken by recent RGB-D [TZN*15] or RGB-only [TZS*16b] real-time facial reenactment methods. These methods estimate dense face geometry along with scene illumination of a source and target actor, transfer the source expression to the target, and re-render and composite the modified face and the target video in a photo-realistic fashion. In order to render the modified mouth in the target view, 3D shape proxies [TZN*15] or image-based methods [TZS*16b] are used.

In the aforementioned VR teleconferencing setting, a (self-)facial reenactment approach could be used to remove the display from the face of each participant by rendering the unoccluded view of the face on top of the VR display at the other end. Unfortunately, the stability of many real-time face capture methods suffers if the tracked person wears an HMD. Furthermore, existing reenactment approaches cannot transfer the appearance of eyes, including blinking and eye gaze - yet exact reproduction of the entire face expression, including the eye region, is crucial for conversations in VR.

In our work, we therefore propose *FaceVR*, a new real-time facial reenactment approach that can transfer facial expressions and realistic eye appearance between a source and a target actor video. Eye movements are tracked using an infrared camera inside the HMD, in addition to outside-in cameras tracking the unoccluded face regions (see Fig. 22.1). It is also suited for self-reenactment with HMDs, thus enabling VR teleconferencing as described above. In order to achieve this goal, we make several algorithmic key contributions:

- Robust real-time facial performance capture of a person wearing an HMD, using an RGB-D camera stream, with rigid and non-rigid degrees of freedom, and an HMD-internal camera.
- Real-time eye-gaze tracking with a novel classification approach based on random ferns, for video streams of an HMD-internal camera or a regular webcam.
- Facial reenactment with photo-realistic re-rendering of the face region including the mouth and the eyes, using model-based face, appearance, and lighting capture.
- Capture of target actors using a lightweight stereo rig which significantly improves tracking accuracy over monocular setups and enables photo-realistic re-rendering of stereo content.
- An end-to-end system for facial reenactment in VR, where the source actor is wearing an HMD and the target actor is recorded in stereo. This facilitates VR goggle removal from a video stream, allows for gaze-aware VR conversations, and enables many other reenactment applications such as eye-gaze correction in video chats.

CHAPTER 23

Related Work

A variety of methods exist to capture detailed static and dynamic face geometry with specialized controlled acquisition setups [KRP*15]. Some methods use passive multi-view reconstruction in a studio setup [BPL*03, PL06, BHB*11, FJA*14], optionally with the support of face markers [Wil90, HCTW11]. Methods using active scanners for capture were also developed [ZSCS04, WLGP09].

Many approaches employ a parametric model of face identity [BV99, BBPV03], and face expression [TDlTM11]. Blend shape models are widely used for representing the expression space [PHL*98, LAR*14], and multi-linear models jointly represent the identity and expression space [VBPP05, SWTC14]. Newer methods enable dense face performance capture in more general scenes with more lightweight setups, such as a stereo camera [VWB*12], or even just a single RGB video at off-line frame rates [GVWT13, SKS14, SWTC14, FJA*14]. Garrido et al. [GZC*16] reconstruct a fully controllable parametric face rig including reflectance and fine scale detail, and [SSK15] build a modifiable mesh model of the face. [IBP15] reconstruct a game-type 3D face avatar from static multi-view images and a video sequence of face expressions. More recently, methods reconstructing dense dynamic face geometry in real-time from a single RGB-D camera [WBLP11, ZNI*14, BWP13, LYYB13, HMYL15] were proposed. Some of them estimate appearance and illumination along with geometry [TZN*15]. Using trained regressors [CHZ14, CBZB15], or parametric face models, dense dynamic face geometry can be reconstructed from monocular RGB video [TZS*16b]. Recently, Cao et al. [CWW*16] proposed an image-based representation for dynamic 3D avatars that supports various hairstyles and parts of the upper body.

The ability to reconstruct face models from monocular input data enables advanced image and video editing effects. Given a portrait of a person, a limitless number of appearances can be synthesized [KS16] based on face replacement and internet image search. Examples for video editing effects are re-arranging a database of video frames [LXW*12] such that mouth motions match a new audio stream [BCS97, TTM15], face puppetry by reshuffling a database of video frames [KSSS10], or re-rendering of an entire captured face model to make mouth motion match a dubbed audiotrack [GVS*15]. Other approaches replace the face identity in a target video [DSJ*11, GVR*14]. When face expressions are modified, it is often necessary to re-synthesize the mouth and its interior under new or unseen expressions, for which image-based [KIM*14, TZS*16b] or 3D template-based [TZN*15] methods were examined. Vlasic et al. [VBPP05] describe a model-based approach for expression mapping onto a target face video, enabling off-line reenactment of faces under controlled recording conditions. While Thies et al. [TZN*15] (see also Part I) enable real-time dense tracking and photo-realistic expression mapping between source and target RGB-D video, Face2Face [TZS*16b] (see also Part II) enables real-time face reenactment between captured RGB video of one actor and an arbitrary target face video. Under the hood, they use a real-time tracker capturing dense shape, appearance and lighting. Expression mapping and image-based mouth-re-rendering enables photo-realistic target appearance. None of the aforementioned capture and reenactment approaches succeeds under strong face occlusion by a VR headset, nor can combine data from several cameras – inside and outside the display – and thus cannot realistically re-render the eye region and appearance, including correct gaze direction.

Parts of our method are related to image-based eye-gaze estimation approaches. Commercial systems exist for eye gaze tracking of the unoccluded face using special externally placed cameras, e.g., from Tobii[1], or IR cameras placed inside a VR headset, e.g., from Pupil Labs[2]. Appearance-based methods for gaze-detection of the unoccluded face from standard externally placed cameras were also researched [SMS14, ZSFB15]. Wang et al. [WSXC16] simultaneously capture 3D eye gaze, head pose, and facial expressions using a single RGB camera at real-time rates. However, they consider a different problem; we need to reenact – i.e., photo-realistically

---

[1] www.tobii.com

[2] www.pupil-labs.com

synthesize – the entire eye region appearance in a target video of either a different actor, or the same actor in a different lighting, from input video of an in-display camera. Parts of our method are related to gaze correction algorithms for teleconferencing where the eyes are re-rendered such that they look into the web-cam, which is typically displaced from the video display [CSBT03, KPB*12, KL15]. Again, this setting is different from ours, as we need to realistically synthesize arbitrary eye region motions and gazes, and not only correct the gaze direction.

Related to our paper is the work by Li et al. [LTO*15] who capture moving facial geometry while wearing an HMD with a rigidly attached depth sensor. In addition, they measure strain signals with electronic sensors to estimate facial expressions of regions hidden by the display. As a result, they obtain the expression coefficients of the face model which are used to animate virtual avatars. Recently, Olszewski et al. [OLSL16] propose an approach for HMD users to control a digital avatar in real-time based on RGB data. The user's mouth is captured by a camera that is rigidly attached to the HMD and a convolutional neural network is used to regress from the images to the parameters that control a digital avatar. They also track eyebrow motion based on a camera that is integrated into the head mounted display. Both of these approaches only allow to control a virtual avatar – rather than a real video – and do not capture the eye motion. Our approach takes this a step further and captures facial performance as well as the eye motion of a person using an HMD. In addition, we allow to re-render and reenact the face, mouth, and eye motion of a target stereo stream photo-realistically and in real-time. Note that our face tracking is also different since our camera is not attached to the HMD (i.e., we solve for the rigid head pose).

CHAPTER 24

Hardware Setup

For our method, we consider a source and a target actor. The *source actor* is wearing a head-mounted display (HMD), and we use a lightweight hardware setup to reconstruct and track the source actor's face. To this end, we augment commodity VR goggles with a simple IR webcam on the inside for tracking one eye. For tracking the rigid pose and facial expressions, we use outside-in tracking with a real-time RGB-D sensor (Asus Xtion Pro), as well as ArUco AR markers on the front panel of the HMD.

The tracking and reconstruction pipeline for the *target actor* differs. Here, we use a lightweight stereo setup which is composed of two commodity webcams. This allows for robust face tracking and generation of 3D video content that we can display on the source actor's HMD. We typically pre-record the target actor's video stream, but we modify and replay it in real time. In addition, we assume that faces in the target video are mostly unoccluded.

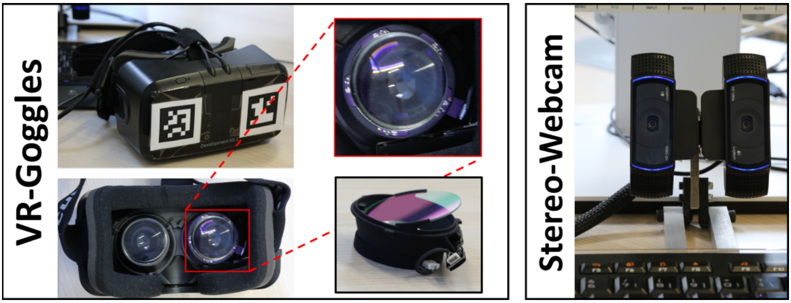


Figure 24.1: Hardware setups: a source actor experiences VR wearing an Oculus DK2 headset (left). We track the source actor using a commodity RGB-D sensor (front-facing), and augment the HMD with ArUco markers, as well as an IR webcam in the inside (mounted with Add-on Cups). The target actor footage is captured with a lightweight stereo rig, which is composed of two webcams (right).

In the following paragraphs, we detail the hardware configuration of our head-mounted display for source actor tracking, as well as the lightweight stereo rig for target actor tracking. Both setups are shown in Fig. 24.1.

24.0.1 Head-Mounted Display for the Source Actor

For visualizing 3D content to the source actor, we use an Oculus Rift DK2 head-mounted display, and we integrate a simple IR webcam to track the source actor's eyes. The camera is integrated inside the HMD with Oculus Rift DK2 Monocular Add-on Cups, which allows us to obtain a close-up camera stream of the right eye [Lab16]; see Fig. 24.1, left. Although we present results on this specific setup, our method is agnostic to the head-mounted display, and can be used in combination with any other VR device, such as the VR Box, Samsung Gear VR, or HTC Vive.

The monocular camera, which we integrate in the DK2, captures an IR stream of the eye region at a resolution of 640×480 pixels at 120Hz. IR LEDs are used as active light sources such that bright images can be obtained, and the camera latency is 5.7ms. The camera is mounted on the top of the VR device lens and an IR mirror is used to get a frontal view of the eye without interfering with the view on the display. The camera is located close to the lenses (see Fig. 24.1, left), and captures images $\mathbf{I}_{\mathcal{E}}$ of the eyes at real-time rates.

Note that our prototype has only one internal camera. Thus, we use the stream of the right eye to infer and reenact the motion of both the left and the right eye. This is feasible as long as we can assume that the focus distance is the same as during calibration, that is eye vergence (squinting) does not change. If this assumption does not hold, a second internal camera for the left eye can be easily integrated into our design.

In addition, we augment the DK2 by attaching two ArUco AR markers to the front of the HMD to robustly track the rigid pose. During face tracking, this allows us to decouple the rigid head pose from the facial expression parameters by introducing additional soft constraints obtained from the markers. The combination of marker tracking and joint optimization allows to further stabilize the estimates of the rigid head pose, leading to much higher tracking accuracy (see Fig. 27.1).

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

24.0.2 3D Stereo Rig for Target Actor Tracking

In order to obtain a 3D reconstruction of the target actor, we use the binocular image stream of a lightweight stereo rig. Our setup is composed of two commodity webcams (Logitech HD Pro Webcam C920), which are rigidly mounted side-by-side and facing the same direction on a stereo bar; see Fig. 24.1, right. The camera rig synchronously captures a stereo stream of two RGB pairs $\mathbf{I}_{\mathcal{I}}^{(c)},\ c \in \{1, 2\}$ at real-time rates. The two cameras are synchronized up to 33ms and capture images at the resolution of 800×600 pixels at 30Hz. The captured stereo content is used to capture the target 3D video content. We calibrate the stereo rig intrinsically and extrinsically using standard OpenCV routines.

Note that we have the option to track the target actor from a monocular stream (similar to Thies et al. [TZS*16b]); e.g., as shown in Fig. 29.2. However, our primary goal is the gaze-aware reenactment of a stereo stream which can be rendered on a 3D display (e.g., a VR device).

CHAPTER 25

Synthesis of Facial Imagery

We parameterize human heads under general uncontrolled illumination based on a multi-linear face and an analytic illumination model. A linear PCA basis is used for facial identity [BV99] (geometry and reflectance) and a blendshape basis for the expression variations [ARL*09, CWZ*14]. This results in the spatial embedding of the underlying mesh and the associated per-vertex color information parameterized by linear models, $\mathcal{F}(\mathbf{T}, \alpha, \beta, \delta)$ and $\mathcal{C}(\beta, \gamma)$, respectively. The mesh has 106K faces and 53K vertices. Here, $\mathbf{T} \in \mathbb{R}^{4\times4}$ models the rigid head pose, $\alpha \in \mathbb{R}^{80}$ the geometric identity, $\beta \in \mathbb{R}^{80}$ the surface reflectance properties, $\delta \in \mathbb{R}^{78}$ the facial expression, and $\gamma \in \mathbb{R}^{3\cdot9}$ the incident illumination situation. The $3 \cdot 9$ illumination coefficients encode the RGB illumination based on 9 Spherical Harmonics (SH) [RH01b] basis functions. For convenience, we stack all parameters of the model in a vector $\mathcal{X} = (\mathbf{T}, \alpha, \beta, \delta, \gamma) \in \mathbb{R}^{265}$. Synthetic monocular images $\mathbf{I}_\mathcal{S}$ and synthetic stereo pairs $(\mathbf{I}_\mathcal{S}^{(1)}, \mathbf{I}_\mathcal{S}^{(2)})$ of arbitrary virtual heads can be generated by varying the parameters $\mathcal{X}$ and using the GPU rasterization pipeline to simulate the image formation process. To this end, we use a standard pinhole camera model under a full perspective projection $\Pi(\bullet)$.

Mouth Interior The parametric head model does not contain rigged teeth, a tongue or a mouth interior, since these facial features are challenging to reconstruct and track from stereo input due to strong occlusions in the input sequence. Instead, we use an image-based retrieval strategy in the spirit of Thies et al. [TZS*16b] to find suitable mouth frames in a training sequence. In contrast to their approach, our retrieval clusters frames into static and dynamic motion segments leading to temporally more coherent results. The output of this step is then composited with the rendered model using alpha blending.

Eyeball and Eyelids We use a unified image-based strategy to synthesize plausible animated eyes (eyeball and eyelid) that can be used for photo-realistic facial reenactment in VR applications. This novel strategy is one of the main contributions of this work and is described in the next section.

CHAPTER 26

An Image-based Eye and Eyelid Model

We propose a novel image-based retrieval approach to track and synthesize the region of the eyes, including eyeballs and eyelids. This approach is later used in all presented applications, namely gaze correction for video calls (see Sec. 29.0.1), gaze-aware facial reenactment (see Sec. 29.0.2) and video conferencing in VR (see Sec. 29.0.3). We chose an image-based strategy, since it is specific to a person; it not only models the behavior of the eyeballs, but also captures idiosyncrasies of eyelid movement while enabling photo-realistic re-rendering. Our approach uses a hierarchical variant of random ferns [OCLF10] to robustly track the eye region. To this end, we propose a novel actor-specific and fully automatic training stage. In the following, we describe our fully automatic data generation process, the used classifier and the optimizations that are required to achieve fast, robust, and temporally stable gaze estimates.

26.0.1 Training Data Generation

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

During the training process, we display a small circle at different positions of a 7×5-tiled image grid on the screen in front of the user; see Fig. 26.1, left. This allows us to capture the space of all possible look-at points on the display. In addition, we capture an image of a closed eye for the synthesis of eye blinks. The captured image data $\mathcal{I}_n$ is divided into $36 = 7 \times 5 + 1$ unique classes l_n, where every class is associated with a view direction. The ground truth gaze directions are given by the current position of the *dot* on the screen in the training data. During training, the user focuses

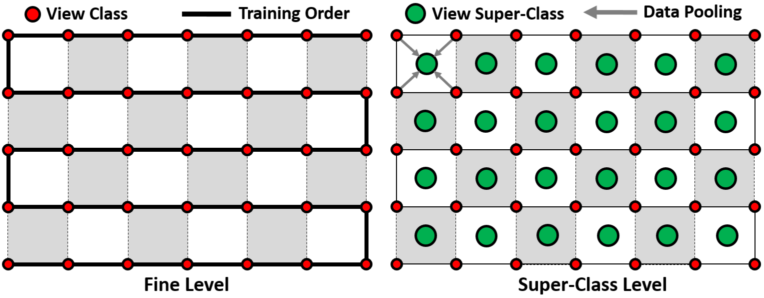


Figure 26.1: Left: the eye calibration pattern used to generate training data for learning our image-based eye-gaze retrieval. In the training phase, we progress row-by-row in a zig-zag order; each grid point is associated with an eye-gaze direction. Right: to obtain robust results, we perform a hierarchical classification where classes of the finer level are accumulated into a smaller set of super classes.

on the displayed dot with his eye gaze. We show every dot for 2 seconds for each location. The data captured in the first 0.4 seconds is rejected to allow the user a grace period to adjust his eye-gaze to new positions. In the remaining 1.6 seconds, we capture 50 frames which we use to populate the corresponding class. After that, we proceed to the next class, and move the dot to the next position. Note that the dot location for a given class is fixed, but we obtain multiple samples within each class (one for each frame) from the input data. This procedure progresses row-by-row in a zig-zag order; see Fig. 26.1, left. Finally, we augment the samples in each class by jittering each captured source image by ± 1 pixels, resulting in 9×50 training frames per class.

Each cluster is also associated with a representative image of the eye region obtained from the captured input data. The representative image of each class is given by the median of the corresponding video clip, which is later used for the synthesis of new eye movements. Finally, we add an additional class which represents eye blinks; this class is obtained by asking the user to close his eyes at the end of the training phase. This calibration sequence is performed for both the source and target actor. Since the calibration sequence is the same for both actors, we obtain one-to-one correspondences between matching classes across actors. As detailed in the following subsections, this allows us to train an eye-gaze classifier which predicts gaze directions for the source actor at runtime. Once trained, for a given source input frame, the classifier identifies cluster representatives from the target actor. The ability to robustly track the eye direction of the source actors forms the basis for real-time gaze-aware facial reenactment; i.e., we are able to photo-realistically animate/modify the eyes of a target actor based on a captured video stream of the source actor. In the following, we detail our eye tracking strategy.

26.0.2 Random Ferns for Eye-gaze Classification

The training data $\{\mathcal{I}_n, l_n\}_{n=1}^{N}$, which is obtained as described in the previous section, is a set of N input images $\mathcal{I}_n$ with associated class labels l_n. Each label $l_n \in \{c_l\}_{l=1}^{C}$ belongs to one of C classes c_l. In our case, the images of the eye region are clustered based on gaze direction. We tackle the associated supervised learning problem by an ensemble of M random ferns [OCLF10], where each fern is based on S features. To this end, we define a sequence of $K = MS$ binary intensity features $\mathbf{F} = \{f_k\}_{k=1}^{K}$, which is split into M independent subsets $\mathbf{F}_m$ of size S. Assuming statistical independence and applying *Bayes Rule*, the log-likelihood of the class label posterior can be written as:

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

The class likelihoods $P(\mathbf{F}_m|c_l)$ are learned using random ferns. Each fern performs S binary tests [OCLF10], which discretizes the per-class feature likelihood into $B = 2^S$ bins. At first, we initialize all bins with one to prevent taking the logarithm of zero. In all experiments, we use $M = 800$ ferns with $S = 5$ binary tests. Finally, the class with the highest posterior probability is chosen as the classification result. Training takes only around 4.9ms per labeled image, thus training runs in parallel to the calibration sequence. Once trained, the best class is obtained in less than 1.4ms.

26.0.3 Hierarchical Eye-gaze Classification

In order to efficiently handle classification outliers, we perform eye-gaze classification on a two-level hierarchy with a fine and a coarse level. The $35 + 1$ classes of the fine level are defined by the grid points of the zig-zag calibration pattern shown in Fig. 26.1, left. To create the coarse level, we merge neighboring classes of the fine level into superclasses. For a set of four adjacent classes (overlap of one), we obtain one superclass; see Fig. 26.1, right. This leads to a grid with $25 = 4 \times 6 + 1$ unique classes (rather than the $35 + 1$ classes; the class for eye blink is kept the same).

During training, we train the two hierarchy levels independently. The training data for the fine level is directly provided by the calibration pattern, and the data for the coarse level is inferred as described above. At test time, we first run the classifier of the coarse level which provides one of the superclasses. Then the classification on the fine level only considers the four classes of the best matching superclass.

The key insight of this coarse-to-fine classification is to break up the task into easier sub-problems. That is, the classification on the coarse level is more robust and less prone to outliers of the fern predictions since there are fewer classes to distinguish between. The fine level then complements the superclass prediction by increasing the accuracy of the inferred eye-gaze directions. In the end, this multi-level classifier leads to high accuracy results while minimizing the probability of noisy outliers. In Fig. 26.2, we show a comparison between a one and two level classifier. The two level approach obtains a lower error (mean 0.217973, std.dev. 0.168094) compared to the one level approach (mean 0.24036, std.dev. 0.18595).

26.0.4 Temporal Stabilization of Classification Results

In the previous sections, we introduced a classifier that infers the eye-gaze direction from a single RGB frame – or monochromatic IR frame in the case of the HMD camera – without the assumption of a temporal prior. Due to the probabilistic nature of random ferns, the classification results are some-

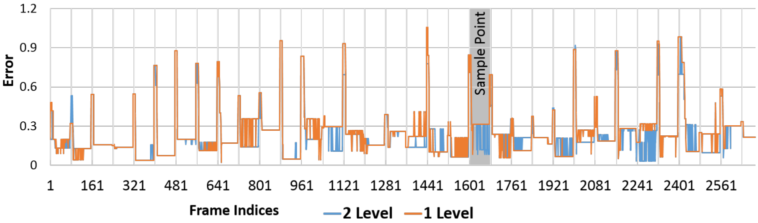


Figure 26.2: Comparison of a one (orange) and a two level (blue) classifier. Ground truth data is obtained by a test subject looking at a dot that appears every 80 frames (2.6 seconds) at random (Sample Point); error is measured in normalized screen space coordinates in $[0,1]^2$. As shown by the magnitude of the positional error, the multi-level classifier obtains higher accuracy.

times temporally unstable. In practice, this can lead to jitter even in the absence of eye motion.

In order to alleviate this problem, we introduce a temporal stabilizer that favors the previously-retrieved eye-gaze direction. This particularly helps in the case of small eye motions, where the switch to a new class would introduce unwanted jitter. To this end, we adjust the likelihood of a specific class $P(c_l)$ using a temporal prior such that the previously-predicted eye-gaze direction c_{old} is $1.05\times$ more likely than changing the state and predicting a different class.We integrate the temporal stabilization on both levels of the classification hierarchy. First, we *favor* the super class on the coarse level using the aforementioned temporal prior. If the current and previous prediction on the coarse level is the same, we apply a similar prior to the view within the superclass. Otherwise, we use no temporal bias on the fine level. This allows fast jumps of the eye direction, which is crucial for fast saccade motion that pushes the boundary of the 30Hz temporal resolution of the stereo setup.

CHAPTER 27

Parametric Model Fitting

Our approach uses two different tracking and reconstruction pipelines for each (source and target) actor, respectively. The source actor, who is wearing the HMD, is captured using an RGB-D camera; see Sec. 24.0.1. Here, we constrain the face model $\mathcal{F}$ by the visible pixels on the face that are not occluded by the HMD, as well as the attached ArUco AR markers. The target actor reconstruction – which becomes the corresponding VR target content that is animated at runtime – is obtained in a pre-process with the lightweight stereo setup described in Sec. 24.0.2. For both tracking pipelines, we use an *analysis-by-synthesis* approach to find the model parameters $\mathcal{X}$ that best explain the input observations. The underlying inverse rendering problem is tackled based on energy minimization. For simplicity, we first describe the energy formulation for tracking the target actor in Sec. 27.0.1. Then, we introduce the objective function for fitting the face model of the source actor in Sec. 27.0.2.

27.0.1 Target Actor Energy Formulation

In order to process the stereo video stream of the target actor, we introduce a model-based stereo reconstruction pipeline that constrains the face model according to both RGB views per frame. That is, we aim to find the optimal model parameters $\mathcal{X}$ constrained by the input stereo pair $\{\mathbf{I}_{\mathcal{I}}^{(c)}\}_{c=1}^{2}$.

Our model-based stereo reconstruction and tracking energy E_{target} is a weighted combination of alignment and regularization constraints:

$$E_{\text{target}}(\mathcal{X}) = \underbrace{\Big[w_{\text{ste}}E_{\text{ste}}(\mathcal{X}) + w_{\text{lan}}E_{\text{lan}}(\mathcal{X})\Big]}_{\text{alignment}} + \underbrace{\Big[w_{\text{reg}}E_{\text{reg}}(\mathcal{X})\Big]}_{\text{regularizer}} \quad . \tag{27.1}$$

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

Dense Photometric Stereo Alignment We enforce dense photometric alignment to both views in the stereo pair. For robustness against outliers, we use the $\ell_{2,1}$-norm [DZHZ06] instead of a traditional least-squares formulation:

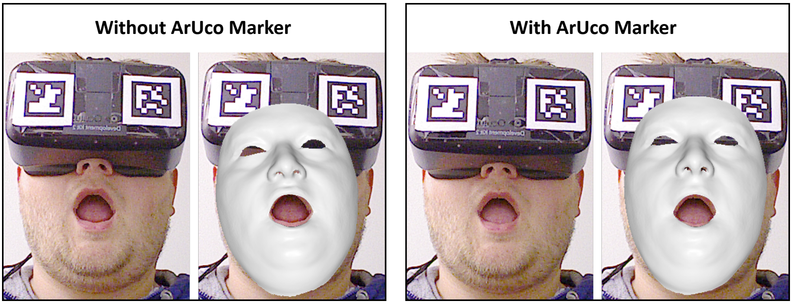
$$E_{\text{ste}}(\mathcal{X}) = \sum_{c=1}^{2} \frac{1}{|\mathcal{P}^{(c)}|} \sum_{\mathbf{p} \in \mathcal{P}^{(c)}} \left\| \mathbf{I}_{\mathcal{S}}^{(c)}(\mathbf{p}) - \mathbf{I}_{\mathcal{I}}^{(c)}(\mathbf{p}) \right\|_2 . \tag{27.2}$$

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

Sparse Stereo Landmark Alignment We use sparse point-to-point alignment constraints in 2D image space that are based on per-camera sets $\mathcal{L}^{(c)}$ of 66 automatically detected facial landmarks. The landmarks are obtained by the detector of Saragih et al. [SLC11a]:

$$E_{\text{lan}}(\mathcal{X}) = \sum_{c=1}^{2} \frac{1}{|\mathcal{L}^{(c)}|} \sum_{(\mathbf{l},k) \in \mathcal{L}^{(c)}} w_{\mathbf{l},k} \left\| \mathbf{l} - \Pi(\mathcal{F}_k(\mathbf{T}, \alpha, \beta, \delta)) \right\|_2^2 . \tag{27.3}$$

The projected vertices $\mathcal{F}_k(\mathbf{T}, \alpha, \beta, \delta)$ are enforced to be spatially close to the corresponding detected 2D feature $\mathbf{l}$. Constraints are weighted by the confidence measures $w_{\mathbf{l},k}$, which are provided by the sparse facial landmark detector.


Without ArUco Marker

With ArUco Marker

Figure 27.1: Tracking with and without ArUco Marker stabilization.

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

$$E_{\text{reg}}(\mathcal{X}) = \sum_{i=1}^{80} \left[\left(\frac{\alpha_i}{\sigma_{\text{id},i}} \right)^2 + \left(\frac{\beta_i}{\sigma_{\text{alb},i}} \right)^2 \right] + \sum_{i=1}^{76} \left(\frac{\delta_i}{\sigma_{\text{exp},i}} \right)^2 . \qquad (27.4)$$

27.0.2 Source Actor Tracking Objective

At runtime, we track the source actor who is wearing the HMD and is captured by the RGB-D sensor. The tracking objective for visible pixels that are not occluded by the HMD is similar to the symmetric point-to-plane tracking energy in Thies et al. [TZN*15]. In addition to this, we introduce rigid stabilization constraints which are given by the ArUco AR markers in front of the VR headset. These constraints are crucial to robustly separate the rigid head motion from the face identity and pose parameters (see Fig. 27.1). The total energy for tracking the source actor at runtime is given by the following linear combination of residual terms:

$$E_{\text{source}}(\mathcal{X}) = w_{\text{rgb}} E_{\text{rgb}}(\mathcal{X}) + w_{\text{geo}} E_{\text{geo}}(\mathcal{X}) + w_{\text{sta}} E_{\text{sta}}(\mathcal{X}) + w_{\text{reg}} E_{\text{reg}}(\mathcal{X}) . \qquad (27.5)$$

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

$$E_{\text{rgb}}(\mathcal{X}) = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \|\mathbf{I}_{\mathcal{S}}(\mathbf{p}) - \mathbf{I}_{\mathcal{I}}(\mathbf{p})\|_2 \ . \tag{27.6}$$

This color term is defined over all visible pixels $\mathcal{P}$ in the bottom half of the face that are not occluded by the HMD, and we use the same $\ell_{2,1}$-norm as in Eq. 27.2.

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

$$E_{\text{geo}}(\mathcal{X}) = w_{\text{point}} E_{\text{point}}(\mathcal{X}) + w_{\text{plane}} E_{\text{plane}}(\mathcal{X}) \ . \tag{27.7}$$

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

$$E_{\text{point}}(\mathcal{X}) = \sum_{\mathbf{p} \in \mathcal{P}} \|\mathbf{D}_{\mathcal{S}}(\mathbf{p}) - \mathbf{D}_{\mathcal{I}}(\mathbf{p})\|_2^2 \ , \tag{27.8}$$

as well as a symmetric point-to-plane term

$$E_{\text{plane}}(\mathcal{X}) = \sum_{\mathbf{p} \in \mathcal{P}} \left[d_{\text{plane}}^2(N_{\mathcal{S}}(\mathbf{p}), \mathbf{p}) + d_{\text{plane}}^2(N_{\mathcal{I}}(\mathbf{p}), \mathbf{p}) \right], \tag{27.9}$$

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

In addition to the constraints given by the raw RGB-D sensor data, the total energy of the source actor E_{source} incorporates rigid head pose stabilization. This is required, since in our VR scenario the upper part of the face is occluded by the HMD. Thus, only the lower part can be tracked and the constraints on the upper part of the face, which normally stabilize the head pose, are missing. To stabilize the rigid head pose, we use the two ArUco markers that are attached to the front of the HMD (see Fig. 24.1, left).

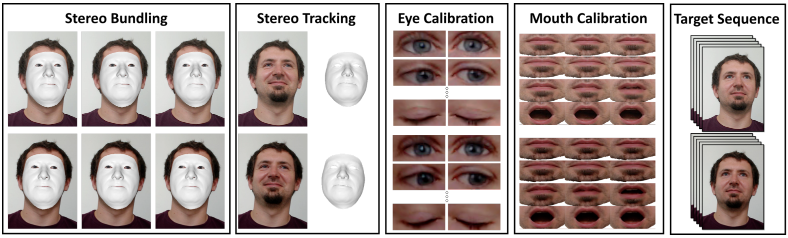


Figure 27.2: Building a personalized stereo avatar; from left to right: we first jointly optimize for all unknowns of our parametric face model using a non-rigid bundle adjustment formulation on the input of three stereo pairs. For tracking, we only optimize for expression, lighting, and rigid pose parameters constrained by synchronized stereo input; this optimization runs in real-time. Next, we train our data-driven eye tracker with data from an eye-calibration sequence. In addition to eye calibration, we build a database of mouth stereo pairs, which captures the variation of mouth motion. As a result, we obtain a tracked stereo target, which is used during live re-enactment (this is the target actor).

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

$$E_{\text{sta}}(\mathcal{X}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{l},k)\in\mathcal{S}} \|\mathbf{l} - \Pi(\mathbf{T}A_k)\|_2^2 \ . \tag{27.10}$$

Here, $\mathcal{S}$ defines the correspondences between the detected 2D landmark positions $\mathbf{l}$ in the current frame and the reference positions A_k. In contrast to the other data terms, E_{sta} depends only on the rigid transformation $\mathbf{T}$ of the face and replaces the facial landmark term used by Thies et al. [TZN*15]. Note that the Saragih tracker is unable to robustly track landmarks in this scenario since only the lower part of the face is visible. The statistical regularization term E_{reg} is the same as for the target actor (see Eq. 27.4).

27.0.3 Data-Parallel Optimization

We find the optimum of both face tracking objectives $\mathcal{X}^*_{\text{source}} = \arg\min_{\mathcal{X}} E_{\text{source}}(\mathcal{X})$ and $\mathcal{X}^*_{\text{target}} = \arg\min_{\mathcal{X}} E_{\text{target}}(\mathcal{X})$ based on variational energy minimization, leading to an unconstrained non-linear optimization problem. Due to the robust $\ell_{2,1}$-norm used to enforce photo-metric alignment, we find the minimum based on a data-parallel *Iteratively Re-weighted Least Squares* (IRLS) solver [TZS*16b]. At the heart of the IRLS solver, a sequence of non-linear least squares problems are solved with a GPU-based *Gauss-Newton* approach [ZNI*14, WZN*14, ZDI*15, TZN*15, DMZ*16, TZS*16b] that builds on an iterative Preconditioned Conjugate Gradient (PCG) solver. The optimization is done in a coarse-to-fine fashion using a hierarchy with three levels. We only run tracking on the two coarser levels using seven IRLS steps on the coarsest and one on the medium level. For each IRLS iteration, we perform one *Gauss-Newton* step with four PCG steps. In order to exploit temporal coherence, we initialize the face model with the optimization results from the previous frame. First, this gives us a good estimate of the visible pixel count in the forward rendering pass, and second, it provides a good starting point for the *Gauss-Newton* optimization. Note that we never explicitly store J^TJ, but instead apply the multiplication of the Jacobian (and its transpose) on-the-fly within every PCG step. Thus, the compute cost for each PCG iteration becomes more expensive for multi-view optimization of E_{target}; although materialization is still less efficient, since we only need a small number of PCG iterations. In addition to optimizing for the parameters of the face model, we jointly optimize for the spherical harmonics coefficients. Since we use three bands, this involves 27 unknowns for the lighting (three per RGB channel); cf. Sec. 25.

CHAPTER 28

Face Rig and Compositing

Generation of a Personalized Face Rig At the beginning of each recording, both of the source and target actor, we compute a person-specific face rig in a short initialization stage. To this end, we capture three keyframes with slightly different head rotations in order to recover the user's facial geometry and skin reflectance. Given the constraints of these three keyframes, we jointly optimize for all unknowns of our face model $\mathcal{F}$ – facial geometry, skin reflectance, illumination, and expression parameters – using our tracking and reconstruction approach. Fig. 27.2 (left) shows this process for the stereo reconstruction step. This initialization requires a few seconds; once computed, we maintain a fixed estimate of the facial geometry and replace the reflectance estimate with person-specific illumination-corrected texture maps. In the stereo case, we compute one reflectance texture for each of the two cameras. This ensures that the two re-projections exactly match the input streams, even if the two used cameras have slightly different color response functions. In the following steps, we use this high-quality stereo albedo map for tracking, and we restrict the optimizer to only compute the per-frame expression and illumination parameters. All other unknowns (e.g., geometry) are person-specific and can remain fixed for a given user.

In order to enable high-quality reenactment of the mouth and the eye-/eyelids in the target video, we provide two additional short calibration sequences (each a few seconds); see Fig. 27.2 (right). First, the user slowly opens and closes his mouth in front of the sensor. This allows us to generate a mouth motion database. The database clusters frames into static and dynamic motion segments based on the space-time trajectory of the sparse landmark detections. At runtime, we use the mouth calibration to synthesize realistic textures for the mouth interior and the teeth. Since we record stereo mouth frames, we can retrieve coherent mouth frame pairs and visualize the result on a 3D display. Second, we capture the person-specific appearance and motion of the eyes and eyelids during the initialization stage.

The obtained eye-calibration data is then used for training our eye tracker (cf. Sec. 26).

Real-time Compositing and Facial Reenactment At run-time, we use the reconstructed face model along with its calibration data (eye and mouth) to photo-realistically re-render the face of the target actor. We first modify the facial expression parameters of the reconstructed face model of the target actor. The expressions are transfered from source to target in real-time using the subspace deformation transfer approach of Thies et al. [TZS*16b]. Based on the modified face parameters, the best fitting mouth is retrieved by finding the frame that has the most similar spatial distribution of 3D marker positions. In contrast to Thies et al. [TZS*16b], we prefer frames that belong to the same motion segment as the previously retrieved one. This leads to higher temporal coherence and hence less visual artifacts. The retrieved mouth frames do not exactly match the transfered facial expression. To account for this, Thies et al. [TZS*16b] stretch the texture based on the face parameterization leading to visual artifacts, i.e., unnaturally stretched teeth, which are temporally unstable. To alleviate this problem, we propose to match the retrieved texture to the outer mouth contour of the target expression using a saliency preserving image warp [WTSL08]. We use a modified as-rigid-as-possible regularizer that takes local saliency of image pixels into account. The idea is to deform the mouth texture predominantly in regions that will not lead to visual artifacts. Stretching is most noticeable for the bright teeth, since they are perfectly rigid in the physical world, while it is harder to detect in the darker regions that correspond to the mouth interior. Therefore, we use pixel intensity as a proxy to determine local rigidity weights (a high value for bright and low value for dark pixels) that control the amount of warping in different texture regions. Since our eye gaze estimator allows one-to-one correspondence between the source and the target actor, we also know the index of the gaze class in the eye database of the target actor. To synthesize temporal coherent and plausible eye motion, we temporally filter the eye motion by averaging the retrieved view direction of the gaze class in a small window of frames. Afterwards, we use the average view direction to perform the texture lookup. In the final compositing stage, we render the mouth texture, the eye textures, and the (potentially modified) 3D face model on top of the target video using alpha blending. Instead of a static face texture, we use a per-frame texture based on the current frame of the target video. This leads to higher resolution results, since slight miss-alignments during the generation of the personalized face rig have no influence on the final texture quality. Note that for the stereo rendering pipeline, we retrieve consistent mouth and eye frame pairs to facilitate visualization on a 3D display.

CHAPTER 29

Results

In this section, we evaluate our gaze-aware facial reenactment approach in detail and compare against state-of-the-art reenactment methods. All experiments run on a desktop computer with an Nvidia GTX980 and a 3.3GHz Intel Core i7-5820K processor. For tracking the source and target actor, we use our hardware setup as described in Sec. 24. However, note that our method is agnostic to the specific hardware components such as the VR headset. Our approach is robust to the specific choice of parameters, and we use a fixed parameter set in all experiments. For stereo tracking, we set the following weights in our energy formulation: $w_{\mathrm{ste}} = 100.0$, $w_{\mathrm{lan}} = 0.0005$, $w_{\mathrm{reg}} = 0.0025$. Our RGB-D tracking approach uses $w_{\mathrm{rgb}} = 100.0$, $w_{\mathrm{geo}} = 10000.0$, $w_{\mathrm{sta}} = 1.0$, $w_{\mathrm{reg}} = 0.0025$.

As our main results, we demonstrate three distinct applications for our real-time gaze-aware facial reenactment approach: gaze correction in live video footage, gaze-aware facial reenactment, and self-reenactment for VR goggle removal. All three applications share a common initialization stage that is required for the construction of a personalized face and eye/eyelid model of the users; see Sec. 28. The source video content is always captured using the Asus Xtion depth sensor. Depending on the application, we use our lightweight stereo rig (3D-stereo content; visualized as anaglyph) or the RGB-D sensor (standard video content).

29.0.1 Gaze Correction for Video Conferencing

Video conference calls, such as Skype chats, suffer from a lack of eye contact between participants due to the discrepancy between the physical location of the camera and the screen. To address this common problem, we apply our face tracking and reenactment approach to the task of online gaze correction for live video footage; see Fig. 29.1. Our goal is the photo-realistic

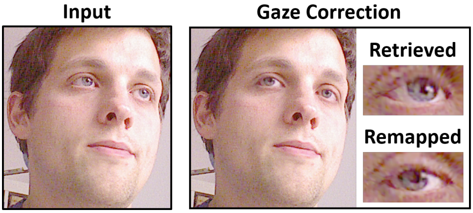


Figure 29.1: Gaze Correction: a common problem in video chats is the discrepancy between the physical location of the webcam and the screen, which leads to unnatural eye appearance (left). We use our eye tracking and retrieval strategy to correct the gaze direction in such a scenario, thus enabling realistic video conversations with natural eye contact (right).

modification of the eye motion in the input video stream using our image-based eye and eyelid model. To this end, we densely track the face of the user, and our eye-gaze classifier provides us with an estimate of the gaze direction; i.e., we determine the 2D screen position where the user is currently looking. Given the eye tracking result, we modify the look-at point by applying a delta offset to the gaze direction which corrects for the different positions of the camera and screen. Finally, we retrieve a suitable eye texture that matches the new look-at point and composite it with the input video stream to produce the final output. A gaze correction example is shown in Fig. 29.1.

29.0.2 Gaze-aware Facial Reenactment

Our approach enables real-time photo-realistic and gaze-aware facial reenactment of monocular RGB-D and 3D stereo videos; see Fig. 29.2 and 29.4. In both scenarios, we track the facial expressions of a source actor using an external Asus Xtion RGB-D sensor, and transfer the facial expressions – including eye motion – to the video stream of a target actor. The eye motion is tracked using our eye-gaze classifier based on the data captured by the external camera (monocular RGB-D reenactment) or the internal IR camera which is integrated into the HMD (stereo reenactment). We transfer the

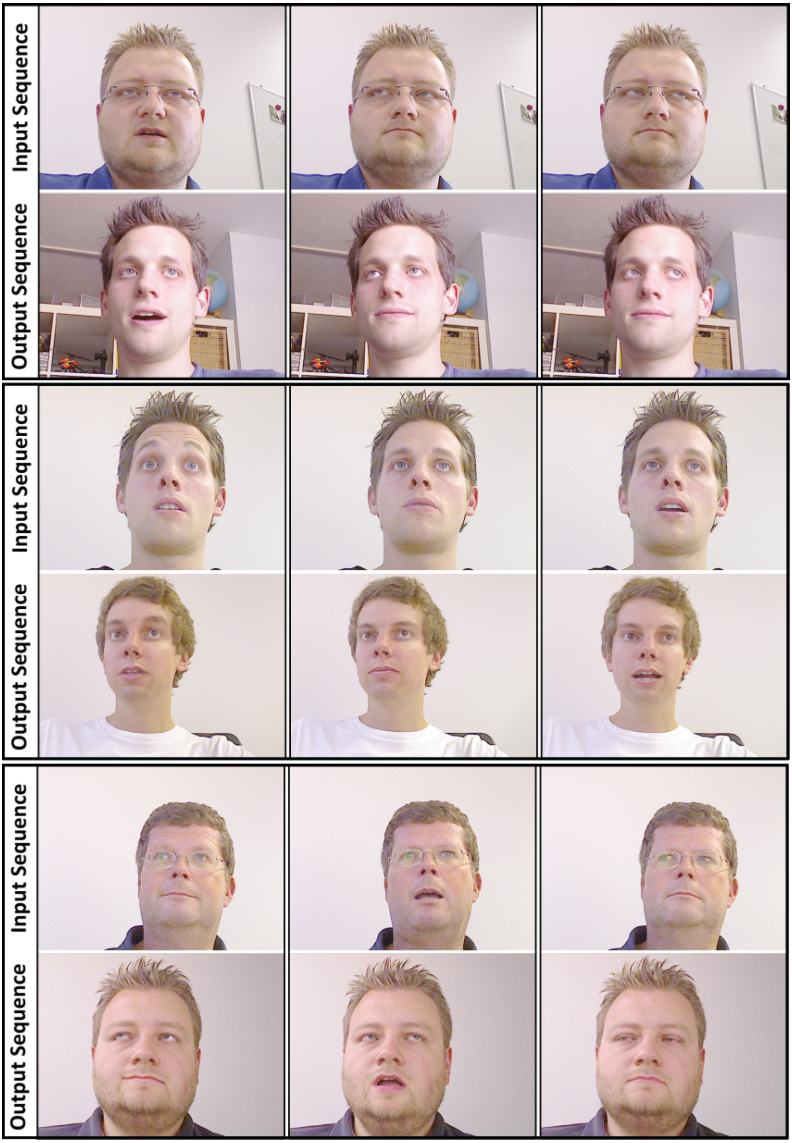


Figure 29.2: Gaze-aware facial reenactment of monocular RGB-D video streams: we employ our real-time performance capture and eye tracking approach in order to modify the facial expressions and eye motion of a target video. In each sequence, the source actor's performance (top) is used to drive the animation of the corresponding target video (bottom).

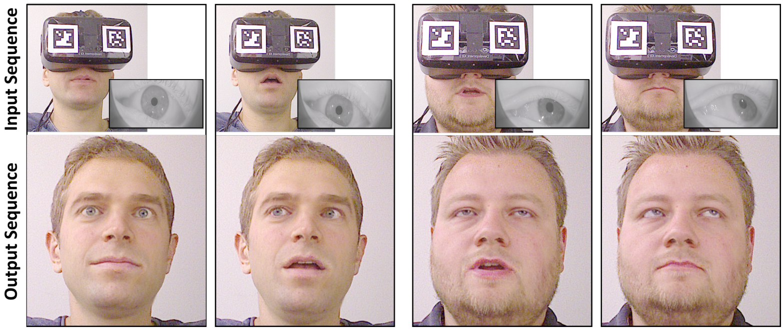


Figure 29.3: Self-Reenactment for VR Video Conferences: our real-time facial reenactment approach allows to virtually remove the HMD by driving a pre-recorded target video of the same person.

tracked facial motion to a target video stream using the presented (stereo) facial reenactment approach. The modified eye region is synthesized using our unified image-based eye and eyelid model. The re-rendering and compositing is detailed in Sec. 28. This allows the source actor to take full control of the face expression and eye gaze of the target video stream at real-time frame rates.

29.0.3 Self-Reenactment for VR Video Conferencing

Our real-time facial reenactment approach can be used to facilitate natural video chats in virtual reality. The major challenge for video conferencing in the VR context is that the majority of the face is occluded; thus, the other person in a VR conversation is unable to see the eye region. Rather than re-enacting different target actors with our facial reenactment method, we can self-reenact the source actor. Hence, our approach enables users to drive a pre-recorded target stereo video stream of themselves, which does not include the HMD. Using self-reenactment, the users can alter both the facial expression and the eye/eyelid motion of the pre-recorded video stream. This virtually removes the HMD from the face and allows users to appear as themselves in VR without suffering from occlusions due to the head mounted display; see Fig. 29.3. In addition, the output video stream mimics the eye motion, which is crucial since natural eye contact is essential in conversations. Note that the user can drive either a mono or a stereo video stream.

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

29.0.4 Evaluation of Eye Tracking Accuracy

We evaluate the accuracy of our monocular eye gaze classification strategy on ground truth data and compare to the commercial Tobii EyeX eye tracker[1]. To this end, a test subject looks at a video sequence of a dot that is displayed at random screen positions for 80 successive frames (2.6 seconds given 30Hz input) – this provides a ground truth dataset. During this test sequence, we capture the eye motion using both the Tobii EyeX tracker and our approach. We measure the per-frame magnitude of the positional 2D error of Tobii and our approach with respect to the known ground truth screen positions; see Fig. 29.5. Note that screen positions are normalized to $[0, 1]^2$ before comparison. As can be seen, we obtain consistently lower errors. On the complete test sequence (more than 74 seconds), our approach has a mean error of 0.206 (std. dev. 0.178). In contrast, the Tobii EyeX tracker has a higher error of 0.284 (std. dev. 0.245). The high accuracy of our approach is crucial for realistic and convincing eye reenactment results. Note, the outside-in tracking of Tobii EyeX does not generalize to the VR context, since both eyes are fully occluded by the HMD.

---

[1] `www.tobii.com/xperience/`

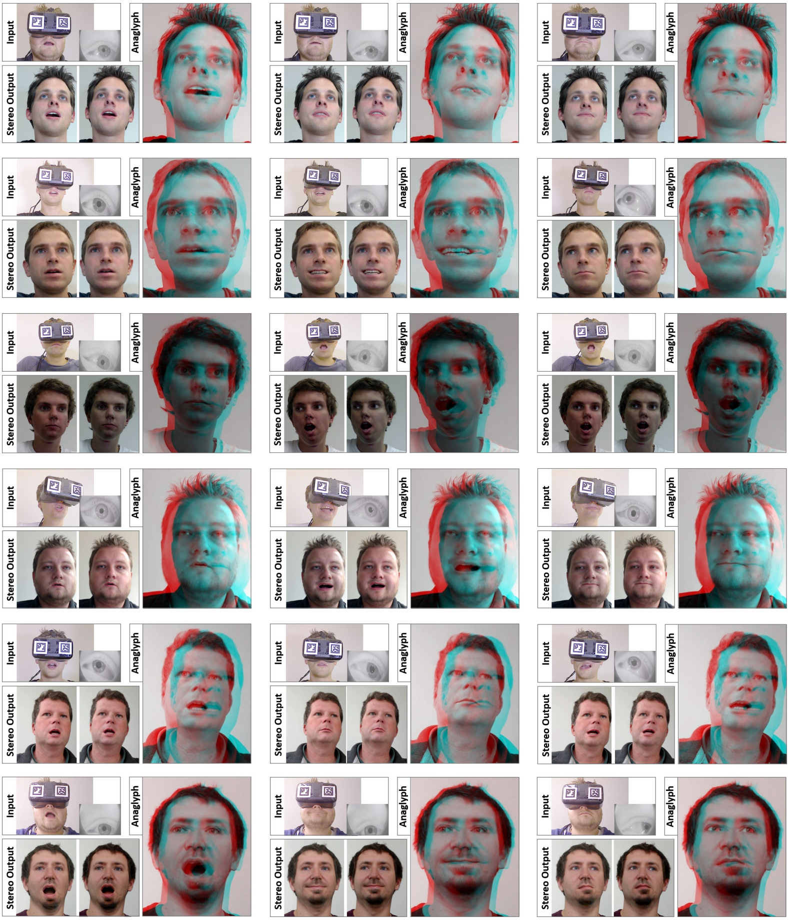


Figure 29.4: Gaze-aware facial reenactment of stereo target video content. We employ our real-time gaze-aware facial reenactment approach to modify the facial expressions and eye motion of stereo 3D content. The input (i.e., source actor) is captured with a frontal view and an internal IR camera. With our method, we can drive the facial animation of the stereo output videos shown below the input – the facial regions in these images are synthetically generated. The final results are visualized as anaglyph images.

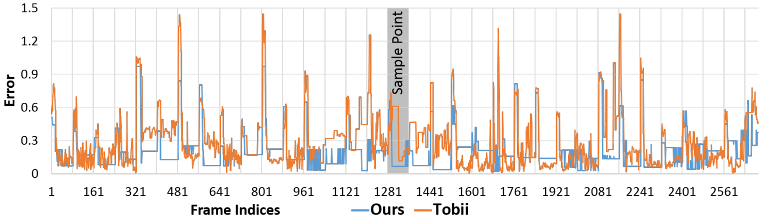


Figure 29.5: Comparison to the commercial Tobii EyeX eye tracking solution. The ground truth data is obtained by a test subject looking at a dot on the screen that appears every 80 frames (2.6 seconds) at random (Sample Point); error is measured in normalized screen space coordinates in $[0,1]^2$. We plot the magnitude of the positional error of Tobii EyeX (orange) and our approach (blue). Our approach obtains a consistently lower error.

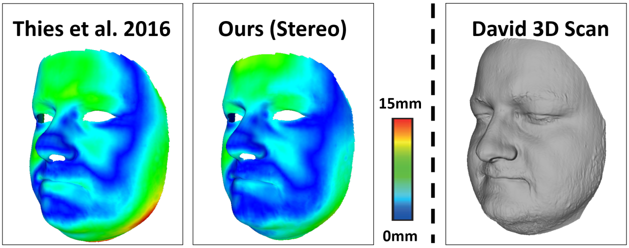


Figure 29.6: Accuracy of reconstructed identity: we compare our result against Face2Face [TZS*16b]. Note that our new approach obtains a better shape estimate of the chin, nose, and cheek regions. For reference, we use a structured light reconstruction from a David 3D scanner. The mean Hausdorff Distance of Face2Face is $3.751mm$ (RMSE $4.738mm$). Our approach has a mean distance of $2.672mm$ (RMSE $3.384mm$).

29.0.5 Evaluation of Face Identity Estimation

The identity of the target actor is obtained using our model-based stereo bundle adjustment strategy. We compare our identity estimate with the approach of Thies et al. [TZS*16b] (Face2Face); see Fig. 29.6. As a reference, we use a high-quality structured light scan of the same person taken with

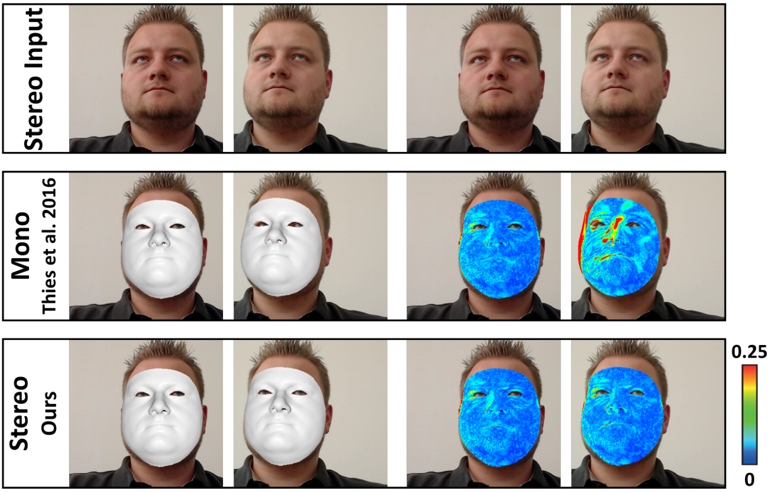


Figure 29.7: Stereo alignment: we compare the photometric alignment accuracy of our approach to Face2Face [TZS*16b]. Face2Face only obtains a good fit to the image captured by the left camera (average error of 0.011), but the reprojection to the right camera suffers from strong misalignments (average error of 0.019). In contrast, our stereo tracking method obtains consistently low errors for both views (average error of 0.011 left and 0.012 right).

a David 3D scanner. Our approach obtains a better reconstruction of the identity, especially the chin, nose, and cheek regions are of higher quality. Note that we estimate the identity by model-based bundle adjustment over three stereo pairs. Face2Face uses only the three images of one of the two RGB cameras.

29.0.6 Evaluation of Face Tracking Accuracy

In Fig. 29.7, we evaluate the stereo alignment accuracy of our approach and compare to the monocular face tracker of Face2Face [TZS*16b]. As input, we use the binocular image stream captured by our custom stereo setup; see Sec. 24. We measure the photometric error between the input frames

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

Table 29.1: Tracking accuracy of our approach (RGB Stereo) compared to Thies et al. [TZN*15] (RGB-D Mono) and Face2Face [TZS*16b] (RGB Mono). Our approach achieves low photometric and geometric errors for both views since we directly optimize for stereo alignment.

and the re-projection of the tracked face model. The tracking of Face2Face is based on the left camera stream, since this approach uses only monocular input data. Thus, Face2Face obtains a good fit with respect to the left camera (average error of 0.011), but the re-projection regarding the right camera suffers from strong misalignments (average error of 0.019). In contrast, our stereo tracking approach obtains consistently low errors for both views (average error of 0.011 left and 0.012 right), since we directly optimize for the best stereo overlap. For the aforementioned re-enactment applications in VR, it is crucial to obtain high-quality alignment with respect to both camera streams of the stereo setup.

We evaluate the accuracy of our approach on ground truth data; see Fig. 29.8. As ground truth, we use high-quality stereo reconstructions obtained by Valgaerts et al. [VWB*12]. To this end, we synthetically generate a high-quality binocular RGB-D stream from the reference data. Our approach achieves consistently low photometric and geometric errors. We also compare against the state-of-the-art face trackers of Thies et al. [TZN*15] (RGB-D Mono) and Face2Face [TZS*16b] (RGB Mono) on the same dataset. All three approaches are initialized using model-based RGB-(D) bundling of three (stereo) frames. The RGB and RGB-D trackers show consistently higher photometric errors for the right input stream, since they do not optimize for stereo alignment; see also Tab. 29.1. Given that Face2Face [TZS*16b] only uses monocular color input, it suffers from depth ambiguity, which results in high geometric errors. Due to

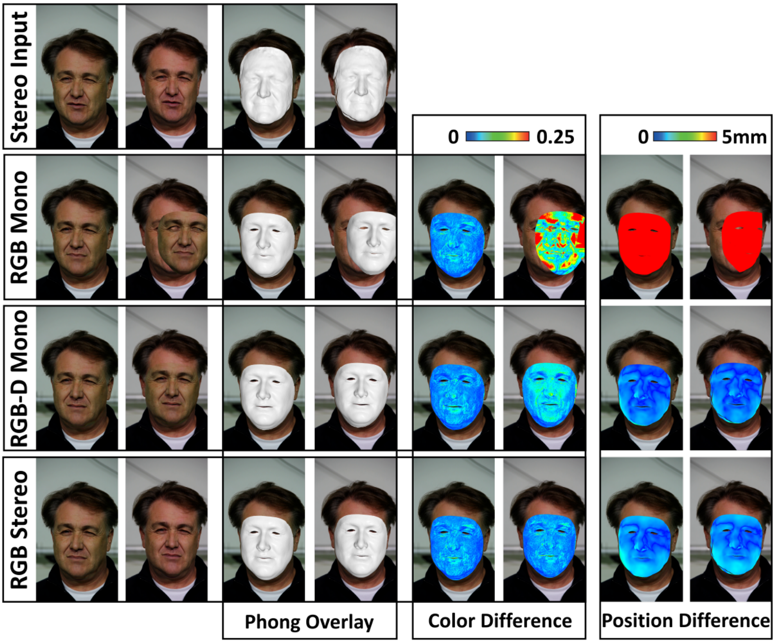


Figure 29.8: Ground truth comparison: we evaluate the photometric and geometric accuracy of our stereo tracking approach (RGB Stereo). As ground truth, we employ the high-quality stereo reconstructions of Valgaerts et al. [VWB*12]. Our approach achieves consistently low photometric and geometric error for both views. We also compare to Thies et al. [TZN*15] (RGB-D Mono) and Face2Face [TZS*16b] (RGB Mono). Both approaches show consistently higher photometric error, since they do not optimize for stereo alignment. Note that the RGB-D tracker uses the ground truth depth as input.

the wrong depth estimate, the re-projection to the right camera image does not correctly fit the input. The RGB-D based tracking approach of Thies et al. [TZN*15] resolves this ambiguity and therefore obtains the highest depth accuracy. Note, however, that this approach has access to the ground truth depth data for the sake of this evaluation. Since the two cameras have slightly different response functions, the reconstructed model colors do not match the right image, leading to high photometric error. Only our model-based stereo tracker is able to obtain high-accuracy geometric and photometric alignment in both views. This is crucial for the creation of 3D stereo output for VR applications, as demonstrated earlier. None of the two other approaches achieves this goal.

CHAPTER 30

Conclusion and Discussion

Although FaceVR is able to facilitate a wide range of face appearance manipulations in VR, it is one of the early methods in a new field. As such, it is a first stepping stone and thus constrained by several limitations. While our eye tracking solution provides great accuracy with little compute cost, it is specifically designed for the VR scenario. First, it is actor-specific, and we need to calibrate and train our classifier for each new person independently; this process takes only a few seconds, but it also makes the tracking solution less suitable compared to a general purpose eye tracker. Second, in the VR device, we track only one eye and infer the movement of the other eye. To correctly capture vergence and squinting one would need to add a second IR camera to the head mounted display. This is a straightforward modification; however, we wanted to keep our setup as simple as possible. Third, we assume that head rotation remains relatively constant with respect to its initialization. This always holds for tracking within an HMD, since the IR camera is rigidly attached; it also works for facial reenactment, since both transfer and eye tracking is in model space. However, for general eye tracking, one would want to include the rigid head pose into the classification framework. One important limitation of our approach is that we cannot modify the rigid head pose of the target videos. This would require a reconstruction of the background and the upper body of the actor, which we believe is an interesting research direction. Our VR face tracking is based on the rigid head pose estimates and the unoccluded face regions. Unfortunately, the field of view of the IR camera attached to the inside of the device is not large enough to cover the entire interior face region. Thus, we cannot track most of the upper face except the eyeballs. Here, our method is complementary to the approach of Li et al. [LTO*15]; they use additional sensor input from electronic strain measures to fill in this missing data. The resulting constraints could be easily included in our face tracking objective; note, however, that their current tracking setup is different, since the outside-in camera is rigidly attached to the HMD. In the context of facial reenactment, we have similar limitations to Thies et al. [TZN*15] and Face2Face [TZS*16b]; i.e., we cannot handle occlusions in the target video such as those caused by microphones or waving hands. We believe that this could be addressed by computing an explicit foreground-face segmentation; the work by Saito et al. [SLL16] already shows promising results to specifically detect such cases.

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

CHAPTER 31

Summary and Outlook

In this dissertation, we developed new face reconstruction and tracking approaches. These approaches differ in their requirements and use-cases.

Part I shows the first real-time facial reenactment system based on RGB-D cameras. In contrast to existing face tracking methods [WLGP09, WBLP11, LYYB13] that aim to animate virtual characters, our goal was to manipulate the RGB video stream photo-realistically. We therefore introduced an analysis-by-synthesis approach that minimizes the photometric and geometric error on a pixel level, resulting in a dense face tracker. To ensure real-time frame rates, we solved the underlying dense optimization problem using an optimized GPU-based *Gauss-Newton* solver. Using this dense face tracker, we showed real-time facial reenactment between a source and a target actor that are both captured with an RGB-D camera. The estimated expression parameters of the source actor are mapped to the target actor, exploiting the expression blendshape representation of the underlying face model. In combination with a generic teeth model, we then re-rendered the modified face of the target actor on top of the input video. The results of this work show that we are able to achieve nearly photo-realistic expression transfer in real-time, with consumer-grade hardware.

In Part II we developed a novel RGB-only face tracking and reenactment technique. The method shown in Part I is limited in its use-cases because of the required RGB-D camera. To this end, we developed a new approach that is only based on uncalibrated RGB videos. The reconstruction of a face in such a setting is way more challenging. Beside unknown camera parameters, there is no depth observation to rely on. To overcome the reduced information, we show a novel model-based bundling scheme, where we use several images of a person to reconstruct the shape, the albedo and the camera parameters. Using these estimations, we are able to track the face in a video stream in real-time. The parameter estimation during bundling and tracking is based on a dense photometric term that compares the synthesized facial imagery with the observations. The optimization strategy is based on an *Iterative Re-weighted Least-Squares* solver. Since the system only requires monocular RGB input, we are able to process videos from the Internet. Using a video sequence as input, we generate a person specific database of mouth interiors. Based on this database, we enable real-time facial reenactment of the person in the video. We track a source actor using an ordinary web-cam and apply the expressions to the target actor in the video. Using the mouth database to synthesize the mouth interior, allows us to increase the quality of our output to a level where the results are nearly indistinguishable from real videos. Our demonstrations of the project at several conferences and in the supplemental video on Youtube resulted in a controversial discussion of the developed technique. People, who watched our system for the first time, were thrilled by the results of our method and thought of funny thinks they could do with it. But soon they realized that it could also be used to manipulate videos for propaganda or other evil purposes. In general they are right with this rating, but it is important to sensitize people to such video manipulations. Existing methods that are used for movie production bring dead actors virtually to life, but nobody thinks of the abuse of such a technology. Demonstrating the simplicity of video manipulation teach the people to not blindly trust videos from unknown sources. Nevertheless, we believe that this system paves the way for many new and exciting applications in the fields of VR/AR, teleconferencing, or on-the-fly dubbing of videos with translated audio.

Part III presents FaceVR, a novel approach for real-time gaze-aware facial reenactment in the context of virtual reality (VR). In contrast to the methods shown in Part I and II, FaceVR is able to track the facial expressions of a person that is partially occluded by a head mounted display (HMD). Since eye motions and eye contact are very important during a conversation, we implemented a data-driven eye tracking method. Using this eye model we are not only able to estimate the gaze of a person, but also to synthesize photo-realistic images. To allow eye tracking of a person that is wearing an HMD, we use an IR-camera that is mounted inside the HMD. Besides tracking a face of a person that is wearing a VR goggle, the VR context requires stereoscopic results. Using a custom stereo camera, we reconstruct stereoscopic avatars that can be reenacted. As a result we show the virtual removal of the HMD that is based on gaze-aware self-reenactment. The shown results also demonstrate the stereoscopic reenactment where the source and target actor differ.

We believe that our work showcases some of the possibilities of the upcoming VR/AR technologies. And we are convinced that there will be even more exciting future work targeting photo-realistic video editing. Thus, in future projects, we plan to further extend and improve our shown methods. Currently, we require a good equipped PC or laptop, especially a modern graphics card. To bring our tracking methods to mobile devices, like Smartphones, the workload has to be reduced. Besides improving performance, our goal is to extend our reenactment to the whole body. Reenacting a body is way more ambitious than reenacting faces. Clothes and other occlusions are challenging and make it hard to reconstruct the body. In addition, during reenactment, the garment has to be animated to produce a believable result. The whole head including hairs has to be modeled and simulated.

To conclude, we enhanced existing face reconstruction and tracking approaches to a level, where the synthesized facial images are nearly indistinguishable from real ones. Our reenactment results led to a broad discussion in the media and opened up a new research field - real-time photo-realistic facial reenactment.

Bibliography

[AAB*84] Adelson E. H., Anderson C. H., Bergen J. R., Burt P. J., Ogden J. M.: *Pyramid methods in image processing.* RCA engineer 29, 6 (1984), 33–41.

[ARL*09] Alexander O., Rogers M., Lambeth W., Chiang M., Debevec P.: *The Digital Emily Project: photoreal facial modeling and animation.* In ACM SIGGRAPH Courses (2009), ACM, pp. 12:1–12:15.

[BA83] Burt P. J., Adelson E. H.: *The Laplacian pyramid as a compact image code. IEEE Trans. Communications 31* (1983), 532–540.

[BBA*07] Bickel B., Botsch M., Angst R., Matusik W., Otaduy M., Pfister H., Gross M.: *Multi-scale capture of facial geometry and motion.* ACM TOG 26, 3 (2007), 33.

[BBPV03] Blanz V., Basso C., Poggio T., Vetter T.: *Reanimating faces in images and video.* In Computer graphics forum (2003), Wiley Online Library, pp. 641–650.

[BCS97] Bregler C., Covell M., Slaney M.: *Video Rewrite: Driving Visual Speech with Audio.* In Proc. SIGGRAPH (1997), ACM Press/Addison-Wesley Publishing Co., pp. 353–360.

[BHB*11] Beeler T., Hahn F., Bradley D., Bickel B., Beardsley P., Gotsman C., Sumner R. W., Gross M.: *High-quality passive facial performance capture using anchor frames.* ACM TOG 30, 4 (2011), 75.

[BHPS10] Bradley D., Heidrich W., Popa T., Sheffer A.: *High resolution passive facial performance capture.* ACM TOG 29, 4 (2010), 41.

[BLA12] Berthouzoz F., Li W., Agrawala M.: *Tools for placing cuts* *and transitions in interview video*. ACM TOG 31, 4 (2012), 67.

[BPL*03] Borshukov G., Piponi D., Larsen O., Lewis J. P., Tempelaar-Lietz C.: *Universal capture: image-based facial animation for "The Matrix Reloaded"*. In SIGGRAPH Sketches (2003), ACM, pp. 16:1–16:1.

[BSVS04] Blanz V., Scherbaum K., Vetter T., Seidel H.-P.: *Exchanging faces in images*. In Computer Graphics Forum (2004), Wiley Online Library, pp. 669–676.

[BV99] Blanz V., Vetter T.: *A morphable model for the synthesis of 3D faces*. In Proc. SIGGRAPH (1999), ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

[BWP13] Bouaziz S., Wang Y., Pauly M.: *Online modeling for realtime facial animation*. ACM TOG 32, 4 (2013), 40.

[CB02] Chuang E., Bregler C.: *Performance-driven Facial Animation using Blend Shape Interpolation*. Tech. Rep. CS-TR-2002-02, Stanford University, 2002.

[CBZB15] Cao C., Bradley D., Zhou K., Beeler T.: *Real-time High-fidelity Facial Performance Capture*. ACM TOG 34, 4 (2015), 46:1–46:9.

[CET01] Cootes T. F., Edwards G. J., Taylor C. J.: *Active Appearance Models*. IEEE TPAMI 23, 6 (2001), 681–685.

[CHZ14] Cao C., Hou Q., Zhou K.: *Displaced Dynamic Expression Regression for Real-time Facial Tracking and Animation*. ACM TOG 33, 4 (2014), 43.

[CM92] Chen Y., Medioni G. G.: *Object modelling by registration of multiple range images*. Image and Vision Computing 10, 3 (1992), 145–155.

[CSBT03] Criminisi A., Shotton J., Blake A., Torr P. H.: *Gaze Manipulation for One-to-one Teleconferencing*. In Proc. ICCV

(2003).

[CWLZ13] Cao C., Weng Y., Lin S., Zhou K.: *3D shape regression for real-time facial animation.* In ACM TOG (2013), vol. 32, pp. 41:1–41:10.

[CWS*13] Chen Y.-L., Wu H.-T., Shi F., Tong X., Chai J.: *Accurate and Robust 3D Facial Capture Using a Single RGBD Camera. Proc. ICCV* (2013), 3615–3622.

[CWW*16] Cao C., Wu H., Weng Y., Shao T., Zhou K.: *Real-time Facial Animation with Image-based Dynamic Avatars.* ACM Trans. Graph. 35, 4 (July 2016).

[CWZ*14] Cao C., Weng Y., Zhou S., Tong Y., Zhou K.: *FaceWarehouse: A 3D Facial Expression Database for Visual Computing.* IEEE TVCG 20, 3 (2014), 413–425.

[CXH03] Chai J.-x., Xiao J., Hodgins J.: *Vision-based control of 3D facial animation.* In Proc. SCA (2003), Eurographics Association, pp. 193–206.

[DMZ*16] DeVito Z., Mara M., Zollhöfer M., Bernstein G., Ragan-Kelley J., Theobalt C., Hanrahan P., Fisher M., Niessner M.: *Opt: A Domain Specific Language for Non-linear Least Squares Optimization in Graphics and Imaging. arXiv preprint arXiv:1604.06525* (2016).

[DSJ*11] Dale K., Sunkavalli K., Johnson M. K., Vlasic D., Matusik W., Pfister H.: *Video face replacement.* ACM TOG 30, 6 (2011), 130.

[DZHZ06] Ding C. H. Q., Zhou D., He X., Zha H.: *R1-PCA: rotational invariant L1-norm principal component analysis for robust subspace factorization.* In ICML (2006), Cohen W. W., Moore A., (Eds.), vol. 148 of *ACM International Conference Proceeding Series*, ACM, pp. 281–288.

[EF78] Ekman P., Friesen W.: *Manual for Facial Action Coding Sys-*

tem. Consulting Psychologists Press Inc., 1978.

[EG98] EISERT P., GIROD B.: *Analyzing Facial Expressions for Virtual Conferencing.* CGAA 18, 5 (1998), 70–78.

[Ekm82] EKMAN P.: *Emotion in the Human Face*. Cambridge University Press, 1982.

[FJA*14] FYFFE G., JONES A., ALEXANDER O., ICHIKARI R., DEBEVEC P.: *Driving High-Resolution Facial Scans with Video Performance Capture*. ACM TOG 34, 1 (2014), 8.

[GFD*12] GUENTER B., FINCH M., DRUCKER S., TAN D., SNYDER J.: *Foveated 3D Graphics*. ACM SIGGRAPH Asia.

[GGW*98] GUENTER B., GRIMM C., WOOD D., MALVAR H., PIGHIN F.: *Making faces*. In Proc. SIGGRAPH (1998), ACM, pp. 55–66.

[GVR*14] GARRIDO P., VALGAERTS L., REHMSEN O., THORMAEHLEN T., PEREZ P., THEOBALT C.: *Automatic Face Reenactment*. In Proc. CVPR (2014).

[GVS*15] GARRIDO P., VALGAERTS L., SARMADI H., STEINER I., VARANASI K., PEREZ P., THEOBALT C.: *VDub: Modifying Face Video of Actors for Plausible Visual Alignment to a Dubbed Audio Track*. In Computer Graphics Forum (2015), Wiley-Blackwell.

[GVWT13] GARRIDO P., VALGAERTS L., WU C., THEOBALT C.: *Reconstructing detailed dynamic face geometry from monocular video.* ACM TOG 32, 6 (2013), 158.

[GZC*16] GARRIDO P., ZOLLHÖFER M., CASAS D., VALGAERTS L., VARANASI K., PÉREZ P., THEOBALT C.: *Reconstruction of Personalized 3D Face Rigs from Monocular Video*. ACM Transactions on Graphics (TOG) 35, 3 (2016), 28.

[Har07] HARRIS M.: *Optimizing cuda. SC07: High Performance Computing With CUDA* (2007).

[HCTW11] Huang H., Chai J., Tong X., Wu H.-T.: *Leveraging motion capture and 3D scanning for high-fidelity facial performance acquisition.* ACM TOG 30, 4 (2011), 74.

[HMYL15] Hsieh P.-L., Ma C., Yu J., Li H.: *Unconstrained realtime facial performance capture.* In Proc. CVPR (2015).

[HZ03] Hartley R., Zisserman A.: *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, New York, NY, USA, 2003.

[IBP15] Ichim A. E., Bouaziz S., Pauly M.: *Dynamic 3D Avatar Creation from Hand-held Video Input.* ACM TOG 34, 4 (2015), 45:1–45:14.

[Jar08] Jarosz W.: *Efficient Monte Carlo Methods for Light Transport in Scattering Media, Appendix B: Spherical Harmonics.* PhD thesis, UC San Diego, September 2008.

[KIM*14] Kawai M., Iwao T., Mima D., Maejima A., Morishima S.: *Data-Driven Speech Animation Synthesis Focusing on Realistic Inside of the Mouth.* Journal of Information Processing 22, 2 (2014), 401–409.

[KL15] Kononenko D., Lempitsky V.: *Learning to look up: Realtime monocular gaze correction using machine learning.* In Proc. CVPR (June 2015), pp. 4667–4675.

[Koc93] Koch R.: *Dynamic 3-D Scene Analysis Through Synthesis Feedback Control.* IEEE Trans. Pattern Anal. Mach. Intell. 15, 6 (June 1993), 556–568.

[KPB*12] Kuster C., Popa T., Bazin J.-C., Gotsman C., Gross M.: *Gaze Correction for Home Video Conferencing.* ACM Trans. Graph. (Proc. of ACM SIGGRAPH ASIA) 31, 6 (2012), to appear.

[KRP*15] Klehm O., Rousselle F., Papas M., Bradley D., Hery C., Bickel B., Jarosz W., Beeler T.: *Recent Advances in Facial*

Appearance Capture. CGF (EUROGRAPHICS STAR Reports) (2015).

[KS16] Kemelmacher-Shlizerman I.: *Transfiguring Portraits.* ACM Trans. Graph. 35, 4 (July 2016), 94:1–94:8.

[KSSGS11] Kemelmacher-Shlizerman I., Shechtman E., Garg R., Seitz S. M.: *Exploring Photobios.* ACM TOG 30, 4 (2011), 61.

[KSSS10] Kemelmacher-Shlizerman I., Sankar A., Shechtman E., Seitz S. M.: *Being John Malkovich.* In Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part I (2010), pp. 341–353.

[LA10] Lewis J., Anjyo K.-i.: *Direct manipulation blendshapes.* IEEE CGAA 30, 4 (2010), 42–50.

[Lab16] Labs P.: *Pupil Labs.* `https://pupil-labs.com/pupil/`, 2016. [Online; accessed 1-Sept-2016].

[LAR*14] Lewis J. P., Anjyo K., Rhee T., Zhang M., Pighin F., Deng Z.: *Practice and Theory of Blendshape Facial Models.* In Eurographics STAR reports (2014), pp. 199–218.

[LSZ01] Liu Z., Shan Y., Zhang Z.: *Expressive expression mapping with ratio images.* In Proc. SIGGRAPH (2001), ACM, pp. 271–276.

[LTO*15] Li H., Trutoiu L., Olszewski K., Wei L., Trutna T., Hsieh P.-L., Nicholls A., Ma C.: *Facial Performance Sensing Head-Mounted Display.* ACM Transactions on Graphics (Proceedings SIGGRAPH 2015) 34, 4 (July 2015).

[LXW*12] Li K., Xu F., Wang J., Dai Q., Liu Y.: *A data-driven approach for facial expression synthesis in video.* In Proc. CVPR (2012), pp. 57–64.

[LYYB13] Li H., Yu J., Ye Y., Bregler C.: *Realtime Facial Animation with On-the-fly Correctives.* ACM TOG 32, 4 (2013), 42.

[MBLD02] Meyer M., Barr A., Lee H., Desbrun M.: *Generalized barycentric coordinates on irregular polygons.* Journal of Graphics Tools 7, 1 (2002), 13–22.

[Mül66] Müller C.: *Spherical harmonics.* Springer, 1966.

[NIH*11] Newcombe R. A., Izadi S., Hilliges O., Molyneaux D., Kim D., Davison A. J., Kohli P., Shotton J., Hodges S., Fitzgibbon A.: *KinectFusion: Real-time Dense Surface Mapping and Tracking.* In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality (Washington, DC, USA, 2011), ISMAR '11, IEEE Computer Society, pp. 127–136.

[NZIS13] Niessner M., Zollhöfer M., Izadi S., Stamminger M.: *Real-time 3D Reconstruction at Scale Using Voxel Hashing.* ACM Trans. Graph. 32, 6 (Nov. 2013), 169:1–169:11.

[OCLF10] Ozuysal M., Calonder M., Lepetit V., Fua P.: *Fast Keypoint Recognition Using Random Ferns.* IEEE Trans. Pattern Anal. Mach. Intell. 32, 3 (Mar. 2010), 448–461.

[OLSL16] Olszewski K., Lim J. J., Saito S., Li H.: *High-Fidelity Facial and Speech Animation for VR HMDs.* ACM TOG 35, 6 (2016).

[PHL*98] Pighin F., Hecker J., Lischinski D., Szeliski R., Salesin D.: *Synthesizing realistic facial expressions from photographs.* In Proc. SIGGRAPH (1998), ACM Press/Addison-Wesley Publishing Co., pp. 75–84.

[PL06] Pighin F., Lewis J.: *Performance-Driven Facial Animation.* In ACM SIGGRAPH Courses (2006).

[RH01a] Ramamoorthi R., Hanrahan P.: *An Efficient Representation for Irradiance Environment Maps.* In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 497–500.

[RH01b] RAMAMOORTHI R., HANRAHAN P.: *A signal-processing framework for inverse rendering.* In Proc. SIGGRAPH (2001), ACM, pp. 117–128.

[SKS14] SUWAJANAKORN S., KEMELMACHER-SHLIZERMAN I., SEITZ S. M.: *Total Moving Face Reconstruction.* In Proc. ECCV (2014), pp. 796–812.

[SLC11a] SARAGIH J. M., LUCEY S., COHN J. F.: *Deformable Model Fitting by Regularized Landmark Mean-Shift.* IJCV 91, 2 (2011), 200–215.

[SLC11b] SARAGIH J. M., LUCEY S., COHN J. F.: *Real-Time Avatar Animation from a Single Image.* In Automatic Face and Gesture Recognition Workshops (2011), pp. 213–220.

[SLL16] SAITO S., LI T., LI H.: *Real-Time Facial Segmentation and Performance Capture from RGB Input.* In Proc. ECCV (2016).

[SMS14] SUGANO Y., MATSUSHITA Y., SATO Y.: *Learning-by-Synthesis for Appearance-Based 3D Gaze Estimation.* In Proc. CVPR (June 2014), pp. 1821–1828.

[SP04] SUMNER R. W., POPOVIĆ J.: *Deformation transfer for triangle meshes.* ACM TOG 23, 3 (2004), 399–405.

[SRH*11] SCHERBAUM K., RITSCHEL T., HULLIN M., THORMÄHLEN T., BLANZ V., SEIDEL H.-P.: *Computer-suggested Facial Makeup.* 485–492.

[SSK15] SUWAJANAKORN S., SEITZ S. M., KEMELMACHER-SHLIZERMAN I.: *What Makes Tom Hanks Look Like Tom Hanks.* In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015 (2015), pp. 3952–3960.

[SWTC14] SHI F., WU H.-T., TONG X., CHAI J.: *Automatic Acquisition of High-fidelity Facial Performances Using Monocular Videos.* ACM TOG 33, 6 (2014), 222.

[TDlTM11] TENA J. R., DE LA TORRE F., MATTHEWS I.: *Interactive Region-based Linear 3D Face Models.* ACM TOG 30, 4 (2011).

[TTM15] TAYLOR S. L., THEOBALD B., MATTHEWS I. A.: *A mouth full of words: Visually consistent acoustic redubbing.* In ICASSP (2015), IEEE, pp. 4904–4908.

[TZN*15] THIES J., ZOLLHÖFER M., NIESSNER M., VALGAERTS L., STAMMINGER M., THEOBALT C.: *Real-time Expression Transfer for Facial Reenactment.* ACM Transactions on Graphics (TOG) 34, 6 (2015).

[TZS*16a] THIES J., ZOLLHÖFER M., STAMMINGER M., THEOBALT C., NIESSNER M.: *Demo of Face2Face: Real-time Face Capture and Reenactment of RGB Videos.* In ACM SIGGRAPH 2016 Emerging Technologies (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 5:1–5:2.

[TZS*16b] THIES J., ZOLLHÖFER M., STAMMINGER M., THEOBALT C., NIESSNER M.: *Face2Face: Real-time Face Capture and Reenactment of RGB Videos.* In Proc. CVPR (2016), pp. 2387–2395.

[TZS*16c] THIES J., ZOLLHÖFER M., STAMMINGER M., THEOBALT C., NIESSNER M.: *FaceVR: Real-Time Facial Reenactment and Eye Gaze Control in Virtual Reality. ArXiv, non-peer-reviewed prepublication by the authors abs/1610.03151* (2016).

[VBPP05] VLASIC D., BRAND M., PFISTER H., POPOVIĆ J.: *Face transfer with multilinear models.* ACM TOG 24, 3 (2005), 426–433.

[VWB*12] VALGAERTS L., WU C., BRUHN A., SEIDEL H.-P., THEOBALT C.: *Lightweight Binocular Facial Performance Capture under Uncontrolled Lighting.* ACM Trans. Graph. 31, 6 (2012), 187.

[WBLP11] WEISE T., BOUAZIZ S., LI H., PAULY M.: *Realtime Performance-*

based Facial Animation. In ACM SIGGRAPH 2011 Papers (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 77:1–77:10.

[WGP*10] Wilson C. A., Ghosh A., Peers P., Chiang J.-Y., Busch J., Debevec P.: *Temporal upsampling of performance geometry using photometric alignment.* ACM TOG 29, 2 (2010), 17.

[WHSL*04] Wang Y., Huang X., Su Lee C., Zhang S., Li Z., Samaras D., Metaxas D., Elgammal A., Huang P.: *High resolution acquisition, learning and transfer of dynamic 3-D facial expressions. CGF 23* (2004), 677–686.

[Wil90] Williams L.: *Performance-driven facial animation.* In Proc. SIGGRAPH (1990), pp. 235–242.

[WLGP09] Weise T., Li H., Gool L. J. V., Pauly M.: *Face/Off: live facial puppetry.* In Proc. SCA (2009), pp. 7–16.

[WSXC16] Wang C., Shi F., Xia S., Chai J.: *Realtime 3D Eye Gaze Animation Using a Single RGB Camera.* ACM Trans. Graph. 35, 4 (July 2016).

[WTSL08] Wang Y.-S., Tai C.-L., Sorkine O., Lee T.-Y.: *Optimized Scale-and-stretch for Image Resizing.* ACM Trans. Graph. 27, 5 (Dec. 2008).

[WZN*14] Wu C., Zollhöfer M., Niessner M., Stamminger M., Izadi S., Theobalt C.: *Real-time Shading-based Refinement for Consumer Depth Cameras.* ACM Transactions on Graphics (TOG) 33, 6 (2014).

[XBMK04] Xiao J., Baker S., Matthews I., Kanade T.: *Real-Time Combined 2D+3D Active Appearance Models.* In Proc. CVPR (2004), pp. 535 – 542.

[ZDI*15] Zollhöfer M., Dai A., Innmann M., Wu C., Stamminger M., Theobalt C., Niessner M.: *Shading-based Refinement on Volumetric Signed Distance Functions.* ACM Transactions on Graphics (TOG) 34, 4 (2015).

[ZNI*14] Zollhöfer M., Niessner M., Izadi S., Rehmann C., Zach C., Fisher M., Wu C., Fitzgibbon A., Loop C., Theobalt C., Stamminger M.: *Real-time Non-rigid Reconstruction using an RGB-D Camera.* ACM TOG 33, 4 (2014), 156.

[ZSCS04] Zhang L., Snavely N., Curless B., Seitz S. M.: *Spacetime faces: high resolution capture for modeling and animation.* ACM TOG 23, 3 (2004), 548–558.

[ZSFB15] Zhang X., Sugano Y., Fritz M., Bulling A.: *Appearance-based gaze estimation in the wild.* In CVPR (2015).

Face2Face: Übertragung von Gesichtsausdrücken in Echtzeit

Kurzfassung

In dieser Dissertation werden die Fortschritte im Bereich der 3D-Rekonstruktion von menschlichen Gesichtern, basierend auf herkömmlicher Endverbraucher-Hardware gezeigt. Neben der Rekonstruktion der Geometrie und der Textur eines Gesichtes, wird auch die Verfolgung von Gesichtszügen in Echtzeit demonstriert. Die entwickelten Algorithmen basieren auf dem Prinzip der Analyse durch Synthese. Um dieses Prinzip anwenden zu können, muss zuerst ein mathematisches Modell definiert werden, welches es ermöglicht ein Gesicht virtuell darzustellen. Neben dem Gesichtsmodell wird auch der Aufnahmeprozess der verwendeten Kamera in einem Modell dargestellt werden. Durch die Möglichkeit ein Bild eines Gesichtes zu synthetisieren, können iterativ die Modellparameter so angepasst werden, dass das synthetisierte Bild bestmöglich das Eingabebild repräsentiert. Mit Hilfe dieses Verfahrens überführt man somit im Umkehrschluss das Eingabebild in eine virtuelle Darstellung eines Gesichtes. Die erreichte Qualität ermöglicht eine Vielzahl von neuen Anwendungen, die auf eine detailgetreue Rekonstruktion angewiesen sind. Dazu gehört auch das sogenannte "Facial Reenactment". Unsere entwickelten Methoden zeigen, dass eine solche Anwendung ohne spezielle Ausrüstung möglich ist. Die Resultate sind nahezu Photo-realistisch Videos, in denen die Mimik einer Person auf eine andere Person übertragen wird. Dadurch lässt sich zum Beispiel die Synchronisierung von Filmen, also das Übersetzen in eine andere Sprache verbessern. Anstatt die Audiospur an das Video anzupassen, was unter anderem auch zu Änderungen am Text führt, können die Mundbewegungen des Dolmetschers in einem Nachbearbeitungsschritt des Videomaterials auf den Schauspieler übertragen werden. Da die Techniken, die in dieser Dissertation gezeigt werden, in Echtzeit ablaufen, kann auch in einem Videotelekonferenzsystem die Mundbewegung eines Live-Dolmetschers virtuell auf eine andere Person übertragen werden.

Die Veröffentlichungen der Videos zu unseren hier gezeigten Projekten, führten zu einer breiten Diskussion in den Medien. Dies lag zum einen an der Tatsache, dass unsere Methoden so entwickelt wurden, dass sie in Echt-

zeit ablaufen können und zum anderen daran, dass wir die Anforderungen an Hardware auf ein Minimum reduziert konnten. So ist es uns möglich gewöhnliche Videos aus dem Internet zu bearbeiten und in Echtzeit zu editieren. Unter anderem haben wir somit bekannten Persönlichkeiten, wie zum Beispiel ehemaligen Präsidenten der USA, eine andere Mimik auferlegen. Dies führte unweigerlich zu einer Diskussion über die Glaubwürdigkeit von Videomaterial, vor allem aus unbekannten Quellen. Das eine solche Manipulation bereits vor unseren gezeigten Demonstrationen möglich war, wenn auch mit einem höheren Aufwand, war den meisten Menschen nicht bewusst. Damit konnten wir mit unseren Projekten, neben der Weiterentwicklung von Echtzeit Face Tracking, zu einer Sensibilisierung der Öffentlichkeit beitragen.

Curriculum Vitae

Justus Philipp-Andrei Thies

PERSONAL INFORMATION

Person:

Name	Justus Philipp-Andrei Thies
Gender	male
Nationality	German
Date of Birth	January 18, 1989
Place of Birth	Buchen (Odw.), Germany

Address:

Street	Hollergasse 11
City	74722 Buchen
Country	Germany

Contact:

Phone	+49 9131 85-29924
Fax	+49 9131 85-29931
Email	justus.thies@.fau.de
Web	http://lgdv.cs.fau.de/people/card/justus/thies/

SCHOOL AND ACADEMIC EDUCATION

Summer 2008	University-Entrance Qualification (German ”Abitur”)
Winter Term 2008/09 – Winter Term 2013/14	Studies in Computer Science at the Friedrich Alexander University (FAU) Erlangen-Nuremberg
November 2011	Bachelor of Science in Computer Science
February 2014	Master of Science in Computer Science
Since April 2014	PhD Student of Prof. Dr. Günther Greiner
September 2014 – October 2014	Internship at Max Planck Institute Informatik, in the Graphics, Vision & Video Group of Christian Theobalt
July 2015 – October 2015	Internship at the Stanford University, in the graphics group of Pat Hanrahan, supervised by Matthias Nießner
Februrary 2016 – March 2016	Internship at Max Planck Institute Informatik, in the Graphics, Vision & Video Group of Christian Theobalt