Realtime Face Tracking and Animation

THÈSE N^{O} 6666 (2015)

PRÉSENTÉE LE 31 AOÛT 2015

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE D'INFORMATIQUE GRAPHIQUE ET GÉOMÉTRIQUE
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Sofien BOUAZIZ

acceptée sur proposition du jury:

Prof. V. Cevher, président du jury
Prof. M. Pauly, directeur de thèse
Prof. C. Theobalt, rapporteur
Prof. S. Rusinkiewicz, rapporteur
Prof. P. Fua, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2015

The best and most beautiful things in the world cannot be seen or even touched -
they must be felt with the heart.
— Helen Keller

To my family…

Acknowledgements

I would like to thank my advisor Mark Pauly. I have spent five wonderful years conducting my research under his supervision. I am really thankful for the trust and the freedom he has given me. He always left his door open and more than a supervisor I consider Mark as a mentor and a friend.

I am grateful to Thibaut Weise, Brian Amberg and Mark Pauly for co-founding *faceshift* with me. It has been a unique experience on a professional and personal level. I am also thankful to the *faceshift* team for the exceptional work they are producing and for being amazing individuals.

During my Ph.D. I had the chance to meet numerous exceptional people. Among these people I would like to thank all my lab mates and friends at LGG: Mina Aleksandra Konaković, Duygu Ceylan, Minh Dang, Bailin Deng, Mario Deuss, Alexandru Ichim, Hao Li, Stefan Lienhard, Boris Neubert, Yuliy Schwartzburg, Andrea Tagliasacchi, Romain Testuz, Anastasia Tkach, Thibaut Weise. I am grateful to Jovan Popović for hosting me at Adobe Research and to all the great people I have met during this internship.

My passion for research started during my master thesis at Mitsubishi Electric Research Laboratories. This internship was a turning point in my life. I am thankful to Srikumar Ramalingam for giving me this opportunity and to all the fun people I have met at MERL.

I would not have published without my coauthors. I am especially thankful to Sebastian Martin, Ladislav Kavan and Tiantian Liu for the extraordinary collaboration we had during the *Projective Dynamics* project. I have also been fortunate to have the opportunity to discuss my research with many great researchers: Steve Seitz, Richard Szeliski, Mario Botsch, Niloy Mitra, Miguel Ángel Otaduy, and many others. I am thankful to my thesis committee for all their help—Mark Pauly, Szymon Rusinkiewicz, Christian Theobalt, Pascal Fua, and Volkan Cevher.

I am grateful to the Swiss Nation Science Foundation for funding my research.

Many administrators at EPFL have made my Ph.D. easier. I am above all thankful to Madeleine Robert. Her passion and enthusiasm for her work is fantastic and it was always a pleasure to communicate with her.

I would like to take this opportunity to thank my close friends from Paris: Gilles Palisseau, Bastien Lhéritier, Michael Daniel, Lois Prosper that have always been here for me. And my friends from Lausanne: Giuliano Losa, Emeric Studer, Lucie Tran and

Acknowledgements

Mathieu Stephan.
Finally, my biggest thanks go to my father, my mother, my sister and my life partner Wei Li. They always supported me and brought me happiness. Without them nothing would have been possible. Your love is the greatest gift of all. Thank you for everything...

Lausanne, 26 October 2014 S. B.

Abstract

Capturing and processing human geometry, appearance, and motion is at the core of computer graphics, computer vision, and human-computer interaction. The high complexity of human geometry and motion dynamics, and the high sensitivity of the human visual system to variations and subtleties in faces and bodies make the 3D acquisition and reconstruction of humans in motion a challenging task. Digital humans are often created through a combination of 3D scanning, appearance acquisition, and motion capture, leading to stunning results in recent feature films. However, these methods typically require complex acquisition systems and substantial manual post-processing. As a result, creating and animating high-quality digital avatars entails long turn-around times and substantial production costs.

Recent technological advances in RGB-D devices, such as Microsoft Kinect, brought new hopes for realtime, portable, and affordable systems allowing to capture facial expressions as well as hand and body motions. RGB-D devices typically capture an image and a depth map. This permits to formulate the motion tracking problem as a 2D/3D non-rigid registration of a deformable model to the input data. We introduce a novel face tracking algorithm that combines geometry and texture registration with pre-recorded animation priors in a single optimization. This led to unprecedented face tracking quality on a low cost consumer level device.

The main drawback of this approach in the context of consumer applications is the need for an offline user-specific training. Robust and efficient tracking is achieved by building an accurate 3D expression model of the user's face who is scanned in a predefined set of facial expressions. We extended this approach removing the need of a user-specific training or calibration, or any other form of manual assistance, by modeling online a 3D user-specific dynamic face model.

In complement of a realtime face tracking and modeling algorithm, we developed a novel system for animation retargeting that allows learning a high-quality mapping between motion capture data and arbitrary target characters. We addressed one of the main challenges of existing example-based retargeting methods, the need for a large number of accurate training examples to define the correspondence between source and target expression spaces. We showed that this number can be significantly reduced by leveraging the information contained in unlabeled data, i.e. facial expressions in the source or target space without corresponding poses.

Abstract

Finally, we present a novel realtime physics-based animation technique allowing to simulate a large range of deformable materials such as fat, flesh, hair, or muscles. This approach could be used to produce more lifelike animations by enhancing the animated avatars with secondary effects.
We believe that the realtime face tracking and animation pipeline presented in this thesis has the potential to inspire numerous future research in the area of computer-generated animation. Already, several ideas presented in thesis have been successfully used in industry and this work gave birth to the startup company *faceshift AG*.

Key words: markerless motion capture, facial animation, physics-based animation, realtime face tracking, realtime face modeling, facial animation retargeting

Résumé

La capture et le traitement de la géométrie, de l'apparence, et des mouvements humains sont au cœur de l'informatique graphique, de l'informatique de vision, et de l'interaction homme-machine. La grande complexité de la géométrie humaine et de la dynamique des mouvements, ainsi que la haute sensibilité du système visuel humain aux variations et subtilités des visages et du corps, font de l'acquisition 3D et de la reconstruction des hommes en mouvement une tâche difficile. Les humains numériques sont souvent créés par une combinaison de numérisation 3D, d'acquisition de l'apparence, et de capture de mouvement, conduisant à des résultats étonnants dans de récents longs métrages. Cependant, ces méthodes nécessitent généralement des systèmes complexes d'acquisition et du post-traitement manuel de manière substantielle. En conséquence, la création et l'animation des avatars numériques de haute qualité implique de longs délais et des coûts de production importants.

Les récentes avancées technologiques dans les dispositifs RGB-D, tels que la Kinect de Microsoft, ont apporté de nouveaux espoirs pour la creation de systèmes temps réel, portables, et abordables, permettant de capturer les expressions faciales ainsi que les mouvements de mains et du corps. Les dispositifs RGB-D capturent habituellement une image et une carte de disparité. Ceci permet de formuler le problème de capture de movement en un problème d'alignement 2D/3D non-rigide d'un modèle déformable aux données d'entrées.

Nous avons introduit un nouvel algorithme de suivi du visage qui combine un modèle d'alignement de la géométrie et de la texture avec un modèle d'animation préenregistrés en une seule optimisation. Cela conduit à une qualité sans précédent de suivi du visage sur un dispositif à faible coût.

Le principal inconvénient de cette approche dans le cadre d'une application commerciale était la nécessité d'entraîner le système hors-ligne pour chaque utilisateur. Un suivi de visage robuste et efficace est réalisé en créant un modèle d'expression 3D précis de l'utilisateur pour un ensemble prédéfini d'expressions faciales. Nous avons étendu cette approche éliminant le besoin d'un étalonnage spécifique pour chaque utilisateur, ou de toute autre forme d'assistance manuelle, par la modélisation en ligne d'un modèle dynamique de visage 3D spécifique à l'utilisateur.

En complément d'un algorithme de suivi du visage en temps réel, nous avons développé un nouveau système pour le reciblage des animations qui permet un apprentissage de

haute qualité de la correspondance entre les données de capture de mouvement et le caractère cible. Nous avons abordé l'un des principaux défis des méthodes de reciblage basée sur l'exemple, la nécessité d'un grand nombre d'exemples précis pour calibrer le système et définir la correspondance entre l'espace source et l'espace cible. Nous avons montré que ce nombre peut être considérablement réduit en tirant parti de l'information contenue dans les données non étiquetés, c'est à dire les expressions du visage dans l'espace source ou l'espace cible sans poses correspondantes.
Finalement, nous présentons une nouvelle technique d'animation basé sur la physique en temps réel permettant de simuler une large gamme de matériaux déformables comme la graisse, la chair, les cheveux ou les muscles. Cette approche pourrait être utilisée pour produire des animations plus réalistes en améliorant les avatars animés avec des effets secondaires.
Nous croyons que le pipeline de suivi du visage et de l'animation en temps réel présenté dans cette thèse a le potentiel d'inspirer de nombreuses recherches futures dans le domaine de l'animation générée par ordinateur. Déjà, plusieurs idées présentées dans cette thèse ont été utilisés avec succès dans l'industrie et ce travail a donné naissance à la société *faceshift AG*.

Mots clefs : capture de mouvement sans marqueurs, animation faciale, animation basé sur la physique, suivi du visage en temps réel, modélisation du visage en temps réel, reciblage d'animation faciale

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

There was a time, before mail, phones, and the Internet, when all communication was face-to-face. If you wanted to talk to someone, you had to look at the person, and use your voice, gestures, and facial expressions to convey your emotions. Communication plays a fundamental role in our society. Humans are highly social beings that like to share personal experiences with others. Therefore, we had the desire to make long distance communication possible.

Unfortunately, during the process of making long distance communication feasible we lost numerous core aspects of face-to-face communication. In the beginning, we could only communicate via text messages using mail. We got back the realtime component of face-to-face communication when the telegraph was invented. We retrieved the possibility to communicate with our voices thanks to the phone. Finally, with the development of the Internet and of online video chat softwares, we can smile to the person we are talking to. So, are we done? Is video chat the best way of communicating?

We live and communicate in a three dimensional world. Therefore, to retrieve all the core aspects of real world face-to-face communication it is necessary to reintroduce the three dimensional component. What if we could take long distance communication to the next level by entering a photorealistic three dimensional virtual environment where we could interact and communicate in a similar fashion to real-world communication? To succeed this challenge, tracking the human body is key. A virtual world will not be immersive until we can see and use our own bodies, and until our brains accept three dimensional avatars as people. Consequently, it is of prime importance to develop technologies to accurately track human motions and model human appearances in a way suitable for consumer-level.

The high complexity of human geometry and motion dynamics, and the high sensitivity

of the human visual system to variations and subtleties in faces and bodies make the three dimensional acquisition and reconstruction of humans in motion a complex task. Marker-based systems, multi-camera capture devices, or intrusive scanners commonly used in high-end animation production require an expensive hardware setup, a complex calibration phase, and necessitate extensive manual assistance to setup and operate the system. These systems are therefore unusable at a consumer-level. Fortunately, recent advances in three dimensional acquisition and display technologies at consumer-level are paving the way for truly immersive virtual reality applications.

In this thesis, we present a novel end-to-end pipeline for realtime face tracking and animation on low-cost RGB-D devices that could be deployed at the consumer-level. Successfully deploying a motion capture technology at a large scale puts high demands on performance, robustness, and usability. While being affordable and accessible, the simplicity of these three dimensional acquisition devices come at the cost of high noise levels in the acquired data. It is therefore necessary to develop algorithms combining techniques from computer vision, machine learning, and computer graphics in order to achieve efficiency, robustness and accuracy.

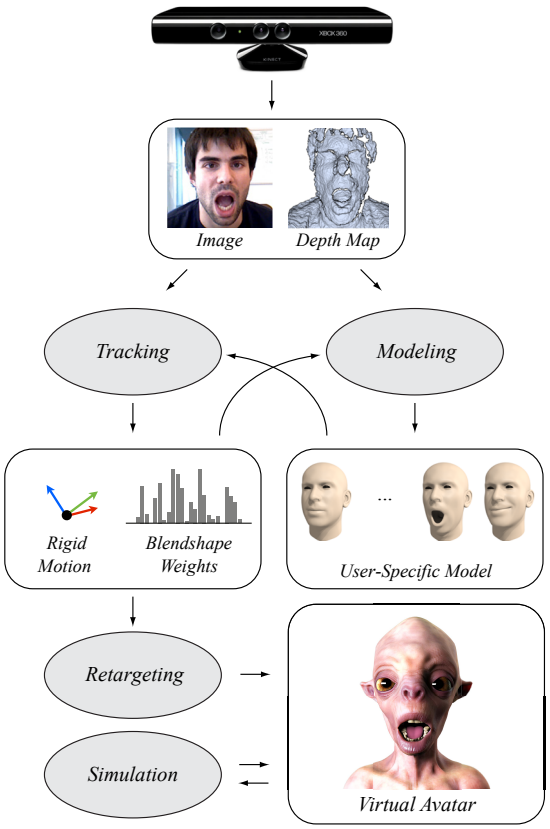


Figure 1.1.: Realtime face tracking and animation pipeline.

Our tracking and animation pipeline is illustrated in Figure 1.1 and can be decomposed into four main stages: tracking, modeling, retargeting, and simulation. Each frame coming from the acquisition device is processed with an interleaved optimization that alternates between tracking facial expressions and modeling the user's face. The tracking parameters are then retargeted to drive a virtual avatar in realtime. Finally, physics simulation is used to enhance the final animation by simulating secondary effects.

Tracking. Recent consumer-level three dimensional acquisition devices, i.e., RGB-D devices, acquire an image and a depth map of the scene. This permits to formulate the motion capture problem as a 2D/3D non-rigid registration of a deformable template face model to the input data. In 1978, Ekman et al. published the Facial Action Coding manual [64] where they explained that facial expressions can be broken up into some constituents called Action Units, which correspond to the activation of one or multiple facial muscles. In a similar spirit, our system uses a 3D facial action coding system, called a blendshape model, as a template, where each action unit is represented by a three dimensional expression model (see Figure 1.2). A novel facial expression can be generated from this blendshape model as a linear combination of the blendshape bases, i.e., the three dimensional expressions. The goal of the tracking stage is to find the rigid motion of the face and the blendshape weights for to the blendshape bases such that the resulting facial expression matches the expression of the user.

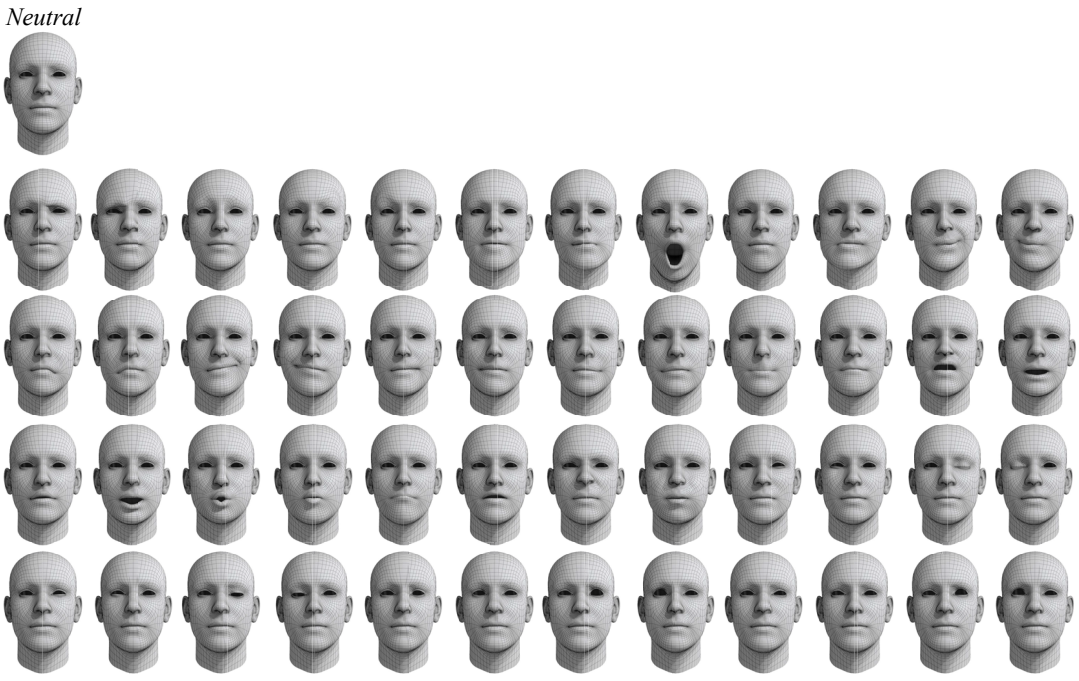


Figure 1.2.: Our parametric face model.

Modeling. Not only the blendshape weights representing the facial expression of the user are unknown but also the blendshape bases representing the geometry of the user's facial expressions. Starting from the generic blendshape model in Figure 1.2, the goal of the modeling stage is to deform the blendshape bases to match the the geometry of the user's facial expressions to create a user-specific blendshape model. As illustrated in the pipeline Figure 1.1, tracking and modeling are tightly coupled. During the registration of the face model to the input image and depth map both the shape of the face and the expression matter, i.e., improving the shape will help to more accurately retrieve the expression, and vice versa.

Retargeting. The goal of the retargeting stage is to animate a virtual target character by adapting the tracking parameters to the target parameters. Mapping the captured performance onto a virtual avatar is a highly non-trivial task, especially when the target character is not a close digital replica of the user. One common way to compute the mapping is to let the user provide a set of correspondences between tracking and target parameters, i.e., for a given recorded expression the user creates a semantically matching expression of the virtual target character. Given this set of labeled pairs, retargeting essentially becomes a problem of scattered data approximation, i.e., extrapolating the explicit correspondences into the entire expression space.

Simulation. To generate more lifelike animations of the virtual target character, physics simulation can be employed to simulate the visco-elastic properties of soft-tissues, or secondary effects such as hair motions. Physics-simulation is also important to integrate the virtual character into a virtual world where external forces and contacts could be applied on the character.

1.1. Contributions

The primary contributions of the work are:

- A novel realtime face tracking algorithm that combines 3D geometry and 2D texture registration with a dynamic data-driven prior generated from existing face animation sequences.
- An efficient method for online face modeling using an adaptive dynamic 3D expression model that combines a dynamic expression template, an identity PCA

model, and a parameterized deformation model in a low-dimensional representation suitable for online learning.

- A semi-supervised retargeting approach that significantly reduces the number of required training examples by learning a shared latent space between motion capture and character parameters to represent their underlying common structure.

- A new implicit solver for realtime physics-based animation that builds a bridge between nodal Finite Element methods and Position Based Dynamics, leading to a simple, efficient, robust, yet accurate solver that supports many different types of constraints.

Together, these technical and scientific innovations enable a range of new applications in communication, virtual reality, human-human and human-computer interaction, computer gaming, or other forms of online interactions.

1.2. Organization

The thesis presents an end-to-end pipeline for realtime face tracking and animation.

In particular,

- Chapter 2 discusses the related work relevant to this thesis.

- Chapter 3 introduces the theory of 3D registration suitable for processing depth data and presents a new unified registration framework.

- Chapter 4 presents a system for performance-based character animation that enables any user to control the facial expressions of a digital avatar in realtime. The user is recorded in a natural environment using a non-intrusive, commercially available 3D sensor. The simplicity of this acquisition device comes at the cost of high noise levels in the acquired data. To effectively map low-quality 2D images and 3D depth maps to realistic facial expressions, we introduce a novel face tracking algorithm that combines geometry and texture registration with prerecorded animation priors in a single optimization. Formulated as a maximum a posteriori estimation in a reduced parameter space, our method implicitly exploits temporal coherence to stabilize the tracking. We demonstrate that compelling 3D facial dynamics can be reconstructed in realtime without the use of face markers, intrusive lighting, or complex scanning hardware. This makes our system easy

to deploy and facilitates a range of new applications, e.g. in digital gameplay or social interactions.

- Chapter 5 extends Chapter 4 removing the need of user-specific training or calibration, or any other form of manual assistance, thus enabling a range of new applications in performance-based facial animation and virtual interaction at the consumer level. The key novelty of our approach is an optimization algorithm that jointly solves for a detailed 3D expression model of the user and the corresponding dynamic tracking parameters. Realtime performance and robust computations are facilitated by a novel subspace parameterization of the dynamic facial expression space. We provide a detailed evaluation that shows that our approach significantly simplifies the performance capture workflow, while achieving accurate facial tracking for realtime applications.

- Chapter 6 introduces a system for facial animation retargeting that allows learning a high-quality mapping between motion capture data and arbitrary target characters. We address one of the main challenges of existing example-based retargeting methods, the need for a large number of accurate training examples to define the correspondence between source and target expression spaces. We show that this number can be significantly reduced by leveraging the information contained in *unlabeled* data, i.e. facial expressions in the source or target space without corresponding poses. In contrast to labeled samples that require time-consuming and error-prone manual character posing, unlabeled samples are easily obtained as frames of motion capture recordings or existing animations of the target character. Our system exploits this information by learning a shared latent space between motion capture and character parameters in a semi-supervised manner. We show that this approach is resilient to noisy input and missing data and significantly improves retargeting accuracy. To demonstrate its applicability, we integrate our algorithm in a performance-driven facial animation system.

- Chapter 7 presents *Projective Dynamics*, a new method for implicit time integration of physical systems. This approach builds a bridge between nodal Finite Element methods and Position Based Dynamics, leading to a simple, efficient, robust, yet accurate solver that supports many different types of constraints. We propose specially designed energy potentials that can be solved efficiently using an alternating optimization approach. Inspired by continuum mechanics, we derive a set of continuum-based potentials that can be efficiently incorporated within our solver. We demonstrate the generality and robustness of our approach in many different applications ranging from the simulation of solids, cloths, and shells, to example-based simulation. Comparisons to Newton-based and Position

Based Dynamics solvers highlight the benefits of our formulation.

- Chapter 8 concludes the thesis by summarizing the main contributions, and suggesting directions of future research.

1.3. Publications

This thesis mainly covers the following publications:

- WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. Realtime performance-based facial animation. *ACM Trans. Graph.* (2011)

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

- BOUAZIZ, S., AND PAULY, M. Semi-supervised facial animation retargeting. Tech. rep., EPFL, 2014

- BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.* (2014)

A background section on registration is provided and uses parts of the following publications:

- BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. Shape-up: Shaping discrete geometry with projections. In *Computer Graphics Forum* (2012)

- BOUAZIZ, S., AND PAULY, M. Dynamic 2d/3d registration for the kinect. In *ACM SIGGRAPH Courses* (2013)

- BOUAZIZ, S., TAGLIASACCHI, A., AND PAULY, M. Sparse iterative closest point. *Computer Graphics Forum* (2013)

- BOUAZIZ, S., DENG, B., AND PAULY, M. Projection-based optimization with fast convergence. *Computer Graphics Forum* (2015). Submitted

In addition, some other publications were published during the same time period but are not explicitly addressed in this thesis:

- DENG, B., BOUAZIZ, S., DEUSS, M., ZHANG, J., SCHWARTZBURG, Y., AND PAULY, M. Exploring local modifications for constrained meshes. *Computer Graphics Forum* (2013)

- DENG, B., BOUAZIZ, S., DEUSS, M., KASPAR, A., SCHWARTZBURG, Y., AND PAULY, M. Interactive design exploration for constrained meshes. *Computer-Aided Design* (2014)

- ZHANG, J., DENG, B., LIU, Z., PATANÈ, G., BOUAZIZ, S., HORMANN, K., AND LIU, L. Local barycentric coordinates. *ACM Trans. Graph.* (2014)

- ICHIM, A. E., BOUAZIZ, S., AND PAULY, M. Dynamic facial avatar creation using handheld cameras. *ACM Trans. Graph.* (2015)

- TAGLIASACCHI, A., SCHRÖDER, M., TKACH, A., BOUAZIZ, S., BOTSCH, M., AND PAULY, M. Robust articulated-ICP for real-time hand tracking. *Computer Graphics Forum* (2015)

Chapter 2

Related Work

Facial performance capture and animation have been active research areas in recent years, with a plethora of different acquisition systems and processing pipelines that share many fundamental principles as well as specific implementation details [140]. Performance-based facial animation typically consists of a *tracking* stage to capture the facial expressions of a person. This *tracking* stage often relies on a parametric template model of the user's face created during an offline or online *modeling* process. The *tracking* stage is usually followed by a *retargeting* procedure allowing to transfer the captured facial expressions onto a virtual avatar. Finally, to improve the realism of the resulting animation, *physics-based animation* techniques can be employed to simulate hair, fat, flesh, and muscles.

2.1. Face Tracking

Animating digital characters based on facial performance capture is a well-established approach in the computer graphics industry and has been an active area of research. One fundamental tradeoff in all of the face tracking systems is the relation between the quality of the acquired data and the complexity of the acquisition setup. On one end of the spectrum are systems designed for greatest possible accuracy that lead to stunning virtual avatars suitable for movie production. Because of their robustness, marker-based techniques [191, 80, 114, 59, 20] are widely used for realtime facial animation and generally deliver sufficient motion parameters for convincing retargeting of non-human creatures or simple game characters.

Face markers significantly simplify tracking, but also limit the amount of spatial detail

that can be captured. For the realistic digitization of human faces, performance capture based on dense 3D acquisition, such as structured light scanners [199] or multi-view camera systems [70, 37, 14, 178], have been developed more recently and proven efficient to capture fine-scale dynamics (e.g. wrinkles and folds). High-resolution facial motion is generally recovered through variants of non-rigid registration and tracking algorithms across sequences of input geometry, texture, or both. With a focus on precision, these systems are not designed to achieve interactive performance in general environments, a crucial requirement for the type of consumer-level applications targeted by our work. Realtime performance can be achieved by a combination of markers and 3D scanning, while still preserving fine-scale spatial and temporal detail [118, 22, 91]. The method of Weise et al. [189] achieves realtime performance using a reduced PCA tracking model. While being able to track facial expressions in realtime, these methods involve controlled studio environments and/or highly specialized setups that need careful calibration. Therefore, none of the above methods is suitable or easily adaptable to the kind of consumer-level applications that we target, where minimal hardware setup, realtime performance, and the absence of complex manual calibration or extensive preprocessing are mandatory.

On the other end of the tradeoff between data quality and hardware complexity are passive, single camera systems that have been a focus of research in computer vision. Most commonly, 2D parametric shape models have been used for non-rigid tracking [110, 23, 65, 55, 141]. However, due to the additional challenges posed by uncontrolled lighting environments and unreliable textures, tracking is usually limited to facial features such as eyes, eyebrows, pupils, or inner and outer contours of the lips. Established methods such as active appearance models [51, 5], constrained local model [152], and Eigen-Points [53] employ a probabilistic prior model built from large sets of training data to achieve realtime performance while preventing drifts. As demonstrated in Chuang and Bregler [50], these parametric models can be used to reliably synthesize simple facial expressions for virtual avatars but inherently lack in facial details. Chai and colleagues [44] first extract 2D animation controls using feature tracking and then map these controls to 3D facial expressions using a preprocessed motion capture database to reduce tracking artifacts. Recently, Cao et al. show that 3D positions of facial landmark points [42] and 3D facial shapes [41] can be inferred by a regressor from 2D video frames leading to compelling facial animations. High-quality face tracking using a single camera has also been demonstrated [73, 155]. However, the processing times of these approaches is significant, impeding interactive frame rates.

The price to pay for the simplification of the acquisition system is often a substantially lower tracking quality leading to artifacts in the generated face animations. Our goal is to raise tracking quality while keeping the acquisition system simple enough

for consumer-level applications and avoiding any manual system calibration or training. Recent developments in RGB-D technology, such as the Microsoft Kinect or Asus Xtion Live, facilitate this goal. The method presented in [7] demonstrates how integrating depth and intensity information in a constrained local model improves tracking performance significantly compared to image-based tracking alone. We propose a real-time performance-based facial animation system [187] combining 2D and 3D non-rigid registration methods in a single optimization to achieve high-quality realtime tracking. We follow the established strategy of using existing animation data for regularization. However, instead of performing a separate post-filtering step as in most previous work, e.g. [116], we integrate an animation prior directly into the tracking optimization using a maximum a posteriori estimation. Our animation prior is based on Mixtures of Probabilistic Principal Component Analyzers (MPPCA) [173], similar in spirit to [102] who use a static pose prior for interactive design of facial geometry. In comparison to Gaussian Processes that have been successfully employed as pose prior, e.g. [79] and [95], MPPCA scales well with the size of the data set, making it particularly suitable for real-time applications.

The main drawback of our approach [187] in the context of consumer applications is the need for extensive training. Robust and efficient tracking is achieved by building an accurate 3D expression model of the user by scanning and processing a predefined set of facial expressions. Beyond being time-consuming, this preprocess is also error-prone. Users are asked to move their head in front of the sensor in a specific static pose to accumulate sufficient depth information. However, assuming and maintaining the correct pose (e.g. mouth open for a specific, pre-defined opening angle) is difficult and often requires multiple tries. Li et al. [112] improved on our system by creating on-the-fly shape correctives adjusted to the actor's expressions through incremental PCA-based learning. As a result, this system can track starting from just a single face scan of the subject in a neutral pose. In [34] we further extended our system [187] requiring no user-specific preprocessing, nor any calibration or user-assisted training, making the tracking system operational right away for any new user. Contrary to [112], we build the specific full blendshape model [108] of a user concurrently to the tracking optimization, requiring no preceding training or calibration stage.

2.2. Face Modeling

Due to the high complexity of facial morphology and heterogeneous skin materials, the most common approaches in facial modeling are data-driven. The seminal work of [24] builds a statistical (PCA) model of facial geometry by registering a template model to a

collection of laser scans. Such a PCA model can be employed to create a static model from a single image [24], from multiple images [4], from video sequences [61], or for the creation of personalized real-time tracking profiles [187, 112, 34]. However, as a compact PCA model only captures the coarse-scale characteristics of the dataset, the generated models are typically rather smooth, lacking the ability to represent fine-scale features like wrinkles and expression lines. Fine-scale detail for facial modeling has been recovered in a controlled environment with multiple calibrated DSLR cameras in the work of Beeler et al. [11]. This setup allows capturing wrinkles, skin pores, facial hair [12], and eyes [16]. The more involved system of [75] uses fixed linear polarizers in front of the cameras and enables accurate acquisition of diffuse, specular, and normal maps. While effective for high-end productions, such systems require a complex calibration within a lab environment.

A static reconstruction only recovers the geometry for a single facial expression. However, reconstructing a dynamic expression model that faithfully captures the user's specific facial movements is a necessary step for facial tracking. One approach to create such a model is to simulate facial muscle activation and model the resulting bone movements and viscoelastic skin deformations [179, 192]. However, the large computational cost and complex parameter estimation make such an approach less suitable for facial animation. Consequently, parametric models are typically employed to represent dynamic skin behavior [136, 97]. Unfortunately, such models are not only difficult to design, but are typically also custom-tuned to a particular animation rig. This makes it difficult to infer generic models for facial dynamics that can easily be adapted to specific subjects. The use of custom hardware has been the most successful way of estimating dynamic models for high-end productions. For example, the Digital Emily project [2] demonstrates how the Light Stage system enables photorealistic dynamic avatars. The work of Alexander et al. [1] recently extended this approach to enable real-time rendering of highly detailed facial rigs. Structured light and laser scanners have also been used to acquire facial geometry at the wrinkle scale [199, 118, 109, 91]. Similarly, the setup of [11, 14] is capable of reconstructing fine-scale detail using multiple calibrated/synchronized DSLR cameras. The recent techniques of [18], and [113] can re-introduce high frequency details in a coarse input animation, if a high-resolution performance database is provided. More recent work attempts to further reduce the setup complexity by only considering a *binocular* [178] or a hybrid *binocular/monocular* setup [73]. A monocular system has been presented by Shi et al. [155] using a multi-linear model. The multi-linear model introduced by [183] and then further explored in [43, 41, 155] offer an efficient way of capturing a joint space of pose and identity. Alternatively, rather than assuming a joint prior on pose and identity, we propose a novel method [34] to model dynamic geometry variations in realtime while tracking using the combination of a PCA model and Laplacian basis functions to better capture user specific details.

This compact linear model is tailored towards estimating a small set of parameters to enable realtime performance.

2.3. Facial Animation Retargeting

Since the seminal work of Williams [191], numerous methods have been devoted to facial animation retargeting. Among those methods, approaches based on correspondences between motion capture markers and target characters [21, 118, 154] have been successful when the actor and the animated faces are geometrically similar. Related to those approaches, [135, 164, 196] use dense correspondences between a source and a target mesh in order to retarget facial expression using vertex or triangle motion transfer. Numerous facial tracking and retargeting systems [54, 91, 187, 154, 34, 41] use a blendshape representation [108] based on Ekman's Facial Action Coding System [64]. However, because of the linearity of the blendshape model, reproducing subtle nonlinear motions can be difficult.

Our novel retargeting system [32] is most closely related to example-based methods [59, 159, 100, 195, 52, 146] that do not require any similarity between the source and the target face. The main difference to existing solutions is that our approach supports nonlinear retargeting of motion capture data and exploits unlabeled data to improve the retargeting accuracy with a reduced number of training examples in a semi-supervised manner. Contrary to Rhodin et al. [146] that only exploit unlabeled samples in target motion sequences, we also exploit unlabeled samples in source motion sequences.

The core of our facial animation retargeting system is based on recent works on Gaussian Process Latent Variable Models (GPLVM) [104]. GPLVM was used successfully for human body tracking [176], retargeting [193] and inverse kinematics [79]. Recently, GPLVM has been extended to support multiple observation spaces [63], missing data [133] and constrains over the latent space [177, 186]. In our work we enhance the shared GPLVM [63] with a prior over latent configurations allowing to preserve local distances of the observation spaces. This prior takes its roots in manifold alignment [86] and Gaussian random fields [202, 180].

2.4. Physics-Based Facial Animation

Physics-based simulation of deformable material has become an indispensable tool in computer animation. Since recent years, high-quality animations in video games or movies, incorporate sophisticated simulations to greatly enhance visual experience. Since the seminal work of Terzopulous and colleagues [169], models derived from continuum mechanics and Finite Element methods play an important role in physics-based animation. Such models have been used successfully for generating facial animations [170, 157, 158], and to simulate flesh [168, 106], muscles [25, 106] or hair [153]. Unfortunately, while these methods enable to reproduce realistic facial expressions [170, 157, 158], few of these approaches have found their way into realtime applications due to their high computational cost and robustness issues.

Realtime simulation of facial muscles and hair has been achieved using mass–spring systems [200, 45]. However, these methods suffer from instabilities and are not physically accurate. Robustness can be obtain using Position Based Dynamics [131, 68] but similar to mass–spring systems these approaches are also inaccurate and the simulation heavily depends on the topology and scale of the underlying mesh.

In this thesis we present *Projective Dynamics* [30], a new method that builds a bridge between Finite Element methods and Position Based Dynamics, leading to a simple, efficient, robust, yet accurate solver that supports many different types of deformable materials and constraints. This solver could be used to robustly simulate muscles or hair in realtime similar to Position Based Dynamics [131, 68] but with the accuracy of Finite Element methods [170, 157, 158, 153]. Instead of simulating an anatomically accurate face model [157, 158], augmenting facial animations with plausible physically-simulated secondary motions could be done efficiently by combining *Projective Dynamics* with the subspace approaches of Hahn et al. [83, 84].

Chapter 3

Background

Recent technological advances in RGB-D sensing devices, such as the Microsoft Kinect, facilitate numerous new and exciting applications, for example in 3D scanning and human motion tracking. While affordable and accessible, consumer-level RGB-D devices typically exhibit high noise levels. This necessitates a particular emphasis on the robustness of 3D registration algorithms. In this section we introduce the theory of 3D registration algorithms suitable for processing depth data. We focus on pairwise registration to compute the alignment of a *source* model onto a *target* model. This alignment can be rigid or non-rigid, depending on the type of object being scanned. We formulate the registration as the minimization of an energy

$$E_{\text{reg}} = E_{\text{match}} + E_{\text{prior}}. \tag{3.1}$$

The matching energy E_{match} defines a measure of how close the source is from the target. The prior energy E_{prior} quantifies the deviation from the type of transformation or deformation that the source is allowed to undergo during the registration, for example, a rigid motion or an elastic deformation. The goal of registration is to find a transformation of the source model that minimizes E_{reg} to bring the source into alignment with the target. We will first present the concept of *proximity function* which is essential to our registration framework and then explain how the proximity function can be applied to 3D registration.

3.1. Proximity Function

A central task in 3D registration is to optimize geometric shapes such that they satisfy certain constraints. For example, volume preservation can be enforced to model the behavior of certain physical materials during the registration procedure. A typical strategy to enforce such constraints is to minimize an objective function that measures their violation. We focus on geometric shapes that can be represented using their point elements, e.g., meshes with fixed connectivity such that their shapes are determined by the vertex positions. Such a shape can be naturally mapped to a high-dimensional point

$$\mathbf{x} = \left[\mathbf{p}_1^T, \mathbf{p}_2^T, \ldots, \mathbf{p}_n^T\right]^T \in \mathbb{R}^{3n},$$

where n is the number of vertices, and $\mathbf{p}_i \in \mathbb{R}^3 (i = 1, \ldots, n)$ are the vertex positions. Usually for each constraint about the geometry $\mathbf{x}$, one can search for a scalar function $c_i(\mathbf{x})$ whose zero level-set corresponds to the shapes that satisfy the constraint. Then the total constraint violation can be measured using a function

$$E(\mathbf{x}) = \sum_i w_i c_i(\mathbf{x})^2, \tag{3.2}$$

where $w_i > 0$ are weights that control the relative importance of the constraints. By minimizing this function, we obtain a new geometry $\mathbf{x}$ that satisfies the constraints as much as possible. In many cases, the functions $\{c_i\}$ are non-linear, and the solution need to be computed using a numerical solver.

Central to our registration framework is the notion of a *proximity function*, which measures the violation of a given constraint for the considered geometry. We can describe a constraint for the geometry using its *feasible set* $\mathscr{C} \subset \mathbb{R}^{3n}$, i.e., the set of shapes that satisfy the constraint. A *feasible set* is also often called a *constraint set*. The proximity function for a point $\mathbf{x} \in \mathbb{R}^{3n}$ with respect to the feasible set $\mathscr{C}$ is defined as the minimum distance from $\mathbf{x}$ to $\mathscr{C}$ as

$$d(\mathbf{x}) = \min_{\mathbf{y} \in \mathscr{C}} \|\mathbf{x} - \mathbf{y}\|_2.$$

Equivalently, the proximity function can also be written as

$$d(\mathbf{x}) = \min_{\mathbf{y}} \|\mathbf{x} - \mathbf{y}\|_2 + \delta_{\mathscr{C}}(\mathbf{y}). \tag{3.3}$$

Here, $\delta_{\mathscr{C}}(\mathbf{y})$ is an indicator function that evaluates to zero if $\mathbf{y} \in \mathscr{C}$ and to $+\infty$ otherwise, and formalizes the requirement that $\mathbf{y}$ should lie in the feasible set. This proximity

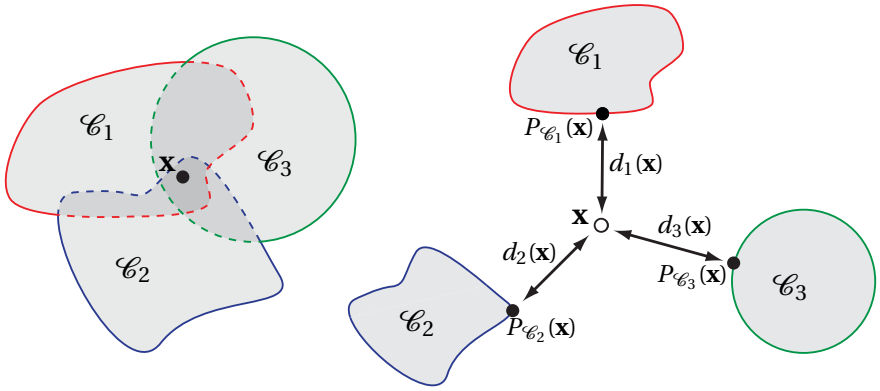


Figure 3.1.: The proximity function $\phi(\mathbf{x})$ is the weighted sum of squared distances $d_i(\mathbf{x})$ of the point $\mathbf{x}$ to the projections $P_{\mathscr{C}_i}(\mathbf{x})$ onto the respective feasible sets $\mathscr{C}_i$. Minimizing $\phi(\mathbf{x})$ yields a feasible solution if the feasible sets intersect (left), and a least-squares solution otherwise (right).

function can be reformulated only using $\mathbf{x}$ as

$$d(\mathbf{x}) = \|\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})\|_2, \tag{3.4}$$

where

$$P_{\mathscr{C}}(\mathbf{x}) = \underset{\mathbf{y}\in\mathscr{C}}{\arg\min} \|\mathbf{x} - \mathbf{y}\|_2$$

is the projection from $\mathbf{x}$ to the feasible set $\mathscr{C}$. Then the constraint can simply be expressed using Equation 3.4 as $d(\mathbf{x}) = 0$. For a collection of constraints with feasible sets $\mathscr{C}_1, \mathscr{C}_2, ..., \mathscr{C}_m$, we can measure their violation as a weighted sum

$$\phi(\mathbf{x}) = \sum_{i=1}^{m} w_i d_i(\mathbf{x})^2, \tag{3.5}$$

a shown in Figure 3.1. This function needs to be minimized to obtain a geometry with the least violation of the constraints. In the following, we first review the gradient of the proximity function. Then we derive the first-order approximations of the proximity function, which will be instrumental in developing our optimization schemes.

Gradient of the squared proximity function. For the proximity function $d(\mathbf{x})$ in (3.4), we can derive the gradient of its squared value $d(\mathbf{x})^2$ as

$$\begin{aligned}\nabla d(\mathbf{x})^2 &= 2(\mathbf{I} - \mathbf{J}_{P_{\mathscr{C}}}(\mathbf{x}))^T(\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})) \\ &= 2(\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})) - 2\,\mathbf{J}_{P_{\mathscr{C}}}(\mathbf{x})^T(\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})),\end{aligned}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{J}_{P_{\mathscr{C}}}(\mathbf{x})$ is the Jacobian of $P_{\mathscr{C}}$ with respect to $\mathbf{x}$. Note that $\mathbf{J}_{P_{\mathscr{C}}}(\mathbf{x})^T(\mathbf{x} - P_{\mathscr{C}}(\mathbf{x}))$ is the derivative of $P_{\mathscr{C}}(\mathbf{x})$ along the vector $\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})$. Since locally $P_{\mathscr{C}}(\mathbf{x})$ does not change when $\mathbf{x}$ moves towards $P_{\mathscr{C}}(\mathbf{x})$, we have $\mathbf{J}_{P_{\mathscr{C}}}(\mathbf{x})^T(\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})) = \mathbf{0}$. It follows that

$$\nabla d(\mathbf{x})^2 = 2(\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})). \tag{3.6}$$

Applying this to the target function $\phi(\mathbf{x})$ in (3.5), we obtain

$$\nabla \phi(\mathbf{x}) = 2\sum_{i=1}^{m} w_i(\mathbf{x} - P_{\mathscr{C}_i}(\mathbf{x})).$$

First-order approximation of the proximity function. Knowing the gradient of the squared distance $d(\mathbf{x})^2$, we can derive the first-order approximation of $d(\mathbf{x}) = \sqrt{d(\mathbf{x})^2}$ at $\mathbf{x}_0$ as

$$\widehat{d}(\mathbf{x})|_{\mathbf{x}_0} = \|\mathbf{x}_0 - P_{\mathscr{C}}(\mathbf{x}_0)\|_2 + \mathbf{n}(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0), \tag{3.7}$$

where

$$\mathbf{n}(\mathbf{x}_0) = \frac{\mathbf{x}_0 - P_{\mathscr{C}}(\mathbf{x}_0)}{\|\mathbf{x}_0 - P_{\mathscr{C}}(\mathbf{x}_0)\|_2}.$$

Interestingly, $\mathbf{n}(\mathbf{x}_0)$ is the normal of the feasible set $\mathscr{C}$ at the projection point $P_{\mathscr{C}}(\mathbf{x}_0)$. We can rewrite the approximation as

$$\widehat{d}(\mathbf{x})|_{\mathbf{x}_0} = \mathbf{n}(\mathbf{x}_0)^T(\mathbf{x} - P_{\mathscr{C}}(\mathbf{x}_0)) \tag{3.8}$$

by noticing that $\|\mathbf{x}_0 - P_{\mathscr{C}}(\mathbf{x}_0)\|_2 = \mathbf{n}(\mathbf{x}_0)^T(\mathbf{x}_0 - P_{\mathscr{C}}(\mathbf{x}_0))$. The approximation in (3.8) is exactly the distance from $\mathbf{x}$ to the tangent plane of the feasible set $\mathscr{C}$ at $P_{\mathscr{C}}(\mathbf{x}_0)$. Applying this to the target function (3.5) results in an approximation of ϕ at $\mathbf{x}_0$:

$$\widehat{\phi}(\mathbf{x})|_{\mathbf{x}_0} = \sum_{i=1}^{m} w_i \left[\mathbf{n}_i(\mathbf{x}_0)^T(\mathbf{x} - P_{\mathscr{C}_i}(\mathbf{x}_0))\right]^2, \tag{3.9}$$

where $\mathbf{n}_i(\mathbf{x}_0)$ is the normal of the feasible set $\mathscr{C}_i$ at $P_{\mathscr{C}_i}(\mathbf{x}_0)$ (see Figure 3.2).

Remark. The above analysis shows that for a proximity function, we can evaluate its gradient and first-order approximation using only the projection operator with respect

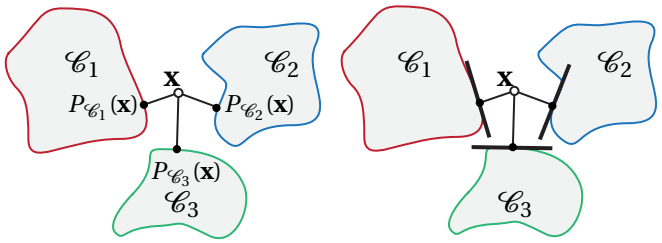


Figure 3.2.: The proximity function measures the weighted sum of squared distances of the point $\mathbf{x}$ to the projections onto the respective feasible sets (left). By linearizing the proximity function we obtain the squared distances of the point $\mathbf{x}$ to the planes tangent to the feasible sets (right).

to the feasible set, *without computing the derivatives of the projection function*.

3.1.1. Optimization Using Projection Operators

Section 3.1 shows that the local behavior of a proximity function, described by its gradient and first-order approximation, is fully determined from the projection operator. Moreover, for many constraints used in geometry processing, their projection operators can be evaluated efficiently (see e.g. [29, 30]). Based on this observation, we will now discuss three different ways of minimizing the target function (3.5), all using projection operators. In particular, we will show how the gradient information can be utilized to design projection-based Gauss-Newton type solvers with fast convergence.

Alternating minimization. A simple way to minimize the function $\phi(\mathbf{x})$ in (3.5) is to employ an alternating minimization scheme that iterates between the following two steps:

1. Compute the projections $P_{\mathscr{C}_i}(\mathbf{x})$ while keeping the current estimate $\mathbf{x}$ fixed.

2. Update $\mathbf{x}$ by minimizing $\phi(\mathbf{x})$ while keeping $P_{\mathscr{C}_i}(\mathbf{x})$ fixed.

To understand why this optimization converges, we observe that the first step weakly decreases each cost function $\|\mathbf{x} - P_{\mathscr{C}_i}(\mathbf{x})\|_2^2$ given the current estimate $\mathbf{x}$, hence $\phi(\mathbf{x})$ cannot increase. The second step minimizes the problem globally by fixing the projections, thus $\phi(\mathbf{x})$ cannot increase either. Thus we obtain a sequence where the target function value is non-increasing and bounded from below (as the sum of squared residuals cannot be negative). As a result, this algorithm always convergences monotonically to a

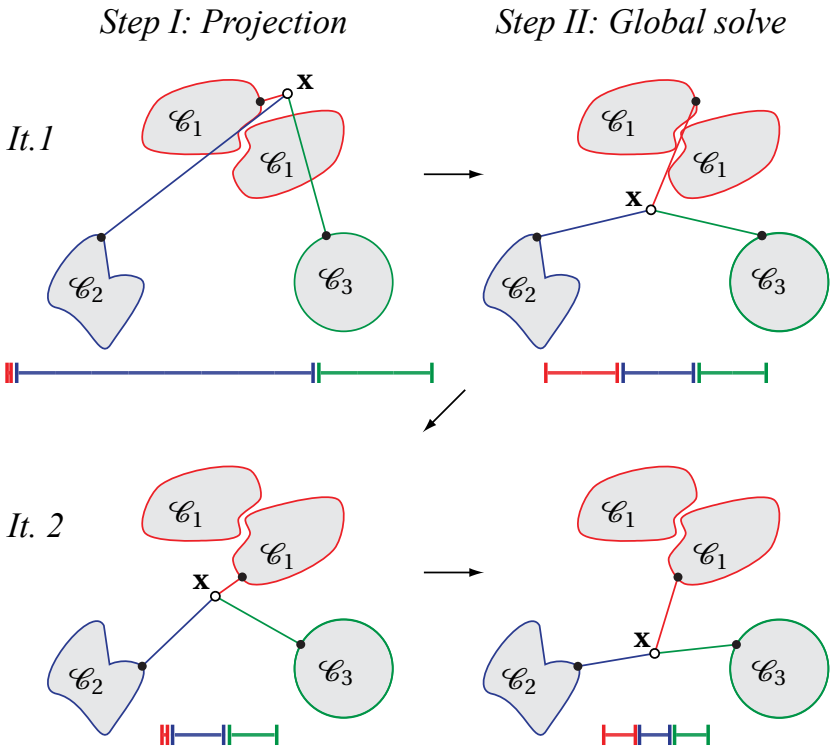


Figure 3.3.: Two iterations of the two-step minimization of the proximity function $\phi(\mathbf{x})$ with $w_i = 1$. Step I computes the projections using the current estimate $\mathbf{x}$. Step II updates $\mathbf{x}$ by minimizing $\phi(\mathbf{x})$ keeping the projections fixed. At each step, $\phi(\mathbf{x})$, illustrated by the sum of the error bars, will decrease, even if some of the individual elements increase.

stationary point (see Figure 3.3).

Interestingly, these two steps can also be expressed as the following iterative process

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} w_i \|\mathbf{x} - P_{\mathscr{C}_i}(\mathbf{x}^k)\|_2^2, \tag{3.10}$$

where $\mathbf{x}^k$ is the estimate of $\mathbf{x}$ at the k-th iteration. $\mathbf{x}^{k+1}$ can be rewritten as a simple average of the projections

$$\mathbf{x}^{k+1} = \frac{\sum_{i=1}^{m} w_i \, P_{\mathscr{C}_i}(\mathbf{x}^k)}{\sum_{i=1}^{m} w_i}. \tag{3.11}$$

Gradient descent Using the projection-based gradient of proximity functions, we can define a gradient descent scheme for $\phi(\mathbf{x})$ as

$$\mathbf{x}^{k+1} = \mathbf{x}^k - 2\lambda^k \sum_{i=1}^{m} w_i(\mathbf{x}^k - P_{\mathscr{C}_i}(\mathbf{x}^k)), \tag{3.12}$$

where λ^k is an appropriately chosen step size to ensure the decrease of $\phi(\mathbf{x})$. There is an interesting connection between alternating minimization and gradient descent: with step size

$$\lambda^k = \frac{1}{2\sum_{i=1}^{m} w_i},$$

the gradient-descent scheme (3.12) reduces to the alternating minimization scheme (3.11).

Linearization. The alternating minimization scheme and the gradient descent scheme, although being very simple, often suffer from slow final convergence. The asymptotic convergence rate of alternating minimization is sublinear [10], while for gradient descent it is linear at best [134]. As a result, it can take a large number of iterations for these methods to converge to a high-accuracy solution. To improve the final convergence rate, we note that the minimization of $\phi(\mathbf{x})$ is a non-linear least squares problem. For such problems, it is well-known that better convergence behavior can be achieved by linearizing the residual terms [120]. Such linearization is provided exactly by Equation (3.9). Therefore, we can optimize $\phi(\mathbf{x})$ using the following iterations:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} w_i \left[\mathbf{n}_i(\mathbf{x}^k)^T \left(\mathbf{x} - P_{\mathscr{C}_i}(\mathbf{x}^k)\right)\right]^2. \tag{3.13}$$

Since this problem is quadratic, it amounts to solving a linear system about $\mathbf{x}$

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}, \tag{3.14}$$

where

$$\begin{aligned} \mathbf{A} &= \sum_{i=1}^{m} w_i\, \mathbf{n}_i(\mathbf{x}^k)\, \mathbf{n}_i(\mathbf{x}^k)^T \in \mathbb{R}^{3n\times 3n}, \\ \mathbf{b} &= \sum_{i=1}^{m} w_i\, \mathbf{n}_i(\mathbf{x}^k)\, \mathbf{n}_i(\mathbf{x}^k)^T P_{\mathscr{C}_i}(\mathbf{x}^k) \in \mathbb{R}^{3n}. \end{aligned}$$

This approach corresponds to the Gauss-Newton method, a well-known non-linear least squares solver [120]. The algorithm may converge slowly or not at all if the matrix $\mathbf{A}$ is ill-conditioned. To improve its stability, we can regularize the approximation of

$\phi(\mathbf{x})$ in (3.13) using the distance from $\mathbf{x}$ to the projection points $P_{\mathscr{C}_i}(\mathbf{x}^k)$, resulting in the following iterative scheme:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} w_i \left[\left(\mathbf{n}_i(\mathbf{x}^k)^T (\mathbf{x} - P_{\mathscr{C}_i}(\mathbf{x}^k)) \right)^2 + \mu \left\| \mathbf{x} - P_{\mathscr{C}_i}(\mathbf{x}^k) \right\|_2^2 \right], \tag{3.15}$$

where $\mu > 0$ is a damping parameter. This approach is similar to the Levenberg-Marquardt solver for non-linear least squares problems, where a damped Gauss-Newton system is solved in each iteration [120]. The damping parameter μ can be used to ensure the decrease of the target function in each step. The larger μ becomes, the closer $\mathbf{x}^{k+1}$ is to the alternating minimization iteration in Equation (3.11); thus when μ is large enough, $\mathbf{x}^{k+1}$ is guaranteed to weakly decrease the function $\phi(\mathbf{x})$ just like alternating minimization. On the other hand, with a small value of μ, the step is close to a Gauss-Newton step and leads to faster convergence.

Remark. There is one important difference between our regularized scheme and the classical Levenberg-Marquardt method: our approach uses the squared distances to the projection points $\{P_{\mathscr{C}_i}(\mathbf{x}^k)\}$ as the regularization terms, while Levenberg-Marquardt uses the squared distance to the current estimate $\mathbf{x}^k$ for regularization.

3.1.2. Generalizing the Proximity Function

So far in our discussion, the proximity functions are always defined using the Euclidean distance in $\mathbb{R}^{3n}$. In a more general setting, a constraint can be imposed with respect to a point $\chi(\mathbf{x}) \in \mathbb{R}^l$, where $\chi : \mathbb{R}^{3n} \mapsto \mathbb{R}^l$ is a function that maps a shape $\mathbf{x} \in \mathbb{R}^{3n}$ to another suitable representation. In this case, we can generalize the proximity function to consider $\chi(\mathbf{x})$. Specifically, let $\mathscr{F} \in \mathbb{R}^l$ be the feasible set for $\chi(\mathbf{x})$. Then we can define a generalized proximity function as

$$h(\mathbf{x}) = \min_{\mathbf{y}} \|\chi(\mathbf{x}) - \mathbf{y}\|_2 + \delta_{\mathscr{F}}(\mathbf{y}) = \min_{\mathbf{y} \in \mathscr{F}} \|\chi(\mathbf{x}) - \mathbf{y}\|_2 = \|\chi(\mathbf{x}) - P_{\mathscr{F}}(\chi(\mathbf{x}))\|_2,$$

where $P_{\mathscr{F}}(\chi(\mathbf{x}))$ is the projection of $\chi(\mathbf{x})$ onto $\mathscr{F}$. Using the generalized proximity function, we can measure the violation of a collection of constraints with a weighted sum similar to (3.5):

$$\psi(\mathbf{x}) = \sum_{i=1}^{m} w_i \|\chi_i(\mathbf{x}) - P_{\mathscr{F}_i}(\chi_i(\mathbf{x}))\|_2^2. \tag{3.16}$$

In the following, we show that the optimization schemes in Section 3.1.1 can be naturally extended to the generalized setting.

Gradient and first-order approximation The gradient of the squared regularized proximity function $h(\mathbf{x})^2$ can be evaluated using the chain rule

$$\nabla h(\mathbf{x})^2 = \left(\nabla_\chi h(\mathbf{x})^2\right)^T \mathbf{J}_\chi(\mathbf{x}),$$

where $\nabla_\chi(\cdot)$ is the gradient with respect to χ, and $\mathbf{J}_\chi(\mathbf{x})$ is the Jacobian of χ with respect to $\mathbf{x}$. Applying Equation (3.6) to evaluate $\nabla_\chi h(\mathbf{x})^2$, we obtain

$$\nabla h(\mathbf{x})^2 = 2\left(\chi(\mathbf{x}) - P_{\mathscr{F}}(\chi(\mathbf{x}))\right)^T \mathbf{J}_\chi(\mathbf{x}). \tag{3.17}$$

If follows that

$$\nabla \psi(\mathbf{x}) = 2\sum_{i=1}^{m} w_i \left(\chi_i(\mathbf{x}) - P_{\mathscr{F}_i}(\chi_i(\mathbf{x}))\right)^T \mathbf{J}_{\chi_i}(\mathbf{x}). \tag{3.18}$$

Similarly, the first-order approximation of $h(\mathbf{x})$ at $\mathbf{x}_0$ can be computed using the chain rule as

$$\widehat{h}(\mathbf{x})|_{\mathbf{x}_0} = h(\mathbf{x}_0) + \mathbf{n}_\chi(\mathbf{x}_0)^T \mathbf{J}_\chi(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0), \tag{3.19}$$

where

$$\mathbf{n}_\chi(\mathbf{x}) = \frac{\chi(\mathbf{x}) - P_{\mathscr{F}}(\chi(\mathbf{x}))}{\|\chi(\mathbf{x}) - P_{\mathscr{F}}(\chi(\mathbf{x}))\|_2}.$$

This leads to the following approximation of $\psi(\mathbf{x})$:

$$\widehat{\psi}(\mathbf{x})|_{\mathbf{x}_0} = \textstyle\sum_{i=1}^{m} w_i \left[\, \|\chi_i(\mathbf{x}_0) - P_{\mathscr{F}_i}(\chi_i(\mathbf{x}_0))\|_2 + \mathbf{n}_{\chi_i}(\mathbf{x}_0)^T \mathbf{J}_{\chi_i}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)\right]^2. \tag{3.20}$$

Which can be rewritten as

$$\widehat{\psi}(\mathbf{x})|_{\mathbf{x}_0} = \textstyle\sum_{i=1}^{m} w_i \left[\mathbf{n}_{\chi_i}(\mathbf{x}_0)^T (\widehat{\chi}_i(\mathbf{x})|_{\mathbf{x}_0} - P_{\mathscr{F}_i}(\chi_i(\mathbf{x}_0)))\right]^2,$$

where $\widehat{\chi}_i(\mathbf{x})|_{\mathbf{x}_0} = \chi_i(\mathbf{x}_0) + \mathbf{J}_{\chi_i}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$ is the first-order approximation of $\chi_i(\mathbf{x})$ at $\mathbf{x}_0$.

Projection-based optimization The optimization schemes in Section 3.1.1 can be extended to the generalized setting, using the gradient and first-order approximation we have derived above. In the following paragraph, we discuss the alternating minimization and the linearization algorithm, since it is trivial to extend the gradient descent scheme.

Similarly to Equation (3.10), we can write the alternating minimization scheme for the

generalized target function ψ as

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} w_i \|\widehat{\chi}_i(\mathbf{x})|_{\mathbf{x}^k} - P_{\mathscr{F}_i}(\chi_i(\mathbf{x}^k))\|_2^2. \tag{3.21}$$

$\mathbf{x}^{k+1}$ can be computed by solving a linear system

$$\left(\sum_{i=1}^{m} w_i \, \mathbf{J}_{\chi_i}(\mathbf{x}^k)^T \mathbf{J}_{\chi_i}(\mathbf{x}^k)\right)\mathbf{x} = \sum_{i=1}^{m} w_i \, \mathbf{J}_{\chi_i}(\mathbf{x}^k) P_{\mathscr{F}_i}(\chi_i(\mathbf{x}^k)). \tag{3.22}$$

It is interesting to note that if $\mathbf{J}_{\chi_i}$ is not a diagonal matrix the alternating minimization scheme is different than gradient descent as a linear system needs to be solved. In this case alternating minimization often converges faster than gradient descent due to the coupling induced by the Jacobian.

For the linearization we can use the regularize approximation of ψ presented in Equation (3.15) resulting in the following iterative scheme:

$$\begin{aligned}\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} w_i \Bigg[&\left(\mathbf{n}_{\chi_i}(\mathbf{x}^k)^T (\widehat{\chi}_i(\mathbf{x})|_{\mathbf{x}^k} - P_{\mathscr{F}_i}(\chi_i(\mathbf{x}^k)))\right)^2 \\ &+ \mu \left\| \widehat{\chi}_i(\mathbf{x})|_{\mathbf{x}^k} - P_{\mathscr{F}_i}(\chi_i(\mathbf{x}^k)) \right\|_2^2 \Bigg].\end{aligned} \tag{3.23}$$

Similar to Section 3.1.1, this minimization can be computed by solving a linear system.

3.2. Matching energy

In 3D registration we want to align a source surface $\mathscr{X}$ embedded in $\mathbb{R}^3$ to a target surface $\mathscr{Y}$ in $\mathbb{R}^3$. To formalize this problem, we introduce a surface $\mathscr{Z}$ that is a transformed or deformed version of $\mathscr{X}$ that eventually aligns with $\mathscr{Y}$ (see Figure 3.4).

The matching energy measures how close the surface $\mathscr{Z}$ is to the surface $\mathscr{Y}$ and can be defined using a proximity function as

$$E_{\text{match}}(\mathscr{Z}) = \int_{\mathscr{Z}} \|\mathbf{z} - P_{\mathscr{Y}}(\mathbf{z})\|_2^2 d\mathbf{z}, \tag{3.24}$$

where $\mathbf{z} \in \mathbb{R}^3$ is a point on the surface $\mathscr{Z}$, and $P_{\mathscr{Y}}(\mathbf{z}) : \mathbb{R}^3 \to \mathbb{R}^3$ returns the closest point (using Euclidian distance) on the surface $\mathscr{Y}$ from $\mathbf{z}$. $P_{\mathscr{Y}}(\mathbf{z})$ can also be seen as the orthogonal projection of $\mathbf{z}$ onto $\mathscr{Y}$. To solve the registration problem numerically, we

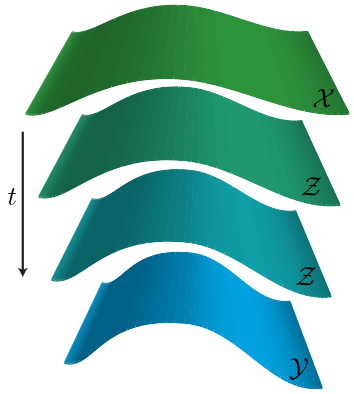


Figure 3.4.: The surface $\mathcal{Z}$ is a deformed version of the source surface $\mathcal{X}$ that eventually aligns with $\mathcal{Y}$.

represent the continuous surface $\mathcal{X}$ by a set of points $X = \{\mathbf{x}_i \in \mathcal{X}, i = 1 \dots m\}$ and define their corresponding points on the deformed surface $\mathcal{Z}$ as $Z = \{\mathbf{z}_i \in \mathcal{Z}, i = 1 \dots m\}$ (see Figure 3.5). Different sampling strategies have been presented by Rusinkiewicz and Levoy [150]. The discrete 3D registration energy is then defined as

$$E_{\text{match}}(Z) = \sum_{i=1}^{m} \|\mathbf{z}_i - P_{\mathcal{Y}}(\mathbf{z}_i)\|_2^2. \tag{3.25}$$

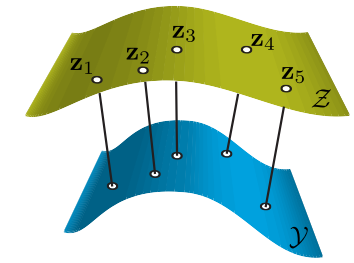


Figure 3.5.: The surface $\mathcal{Z}$ is sampled by a set of points $Z = \{\mathbf{z}_i \in \mathcal{Z}, i = 1 \dots m\}$. The projection $P_{\mathcal{Y}}(\mathbf{z}_i)$ returns the closest point on the surface $\mathcal{Y}$ from $\mathbf{z}_i$.

Iterative closest point (ICP). One way to minimize the registration energy in Equation 3.1 is to use the alternating minimization approach. The alignment is computed by solving iteratively

$$Z^{t+1} = \arg\min_{Z} \sum_{i=1}^{m} \|\mathbf{z}_i - P_{\mathcal{Y}}(\mathbf{z}_i^t)\|_2^2 + E_{\text{prior}}(Z). \tag{3.26}$$

To speed up the convergence of the optimization one can use the first-order approximation of the proximity function. The optimization can be reformulated as

$$Z^{t+1} = \arg\min_{Z} \sum_{i=1}^{m} \left(\mathbf{n}_i^T (\mathbf{z}_i - P_{\mathscr{Y}}(\mathbf{z}_i^t))\right)^2 + E_{\text{prior}}(Z), \tag{3.27}$$

where $\mathbf{n}_i$ is the normal of the surface $\mathscr{Y}$ at $P_{\mathscr{Y}}(\mathbf{z}_i^t)$.

Remark 1. The energy $E_{\text{point}}(Z) = \|\mathbf{z}_i - P_{\mathscr{Y}}(\mathbf{z}_i^t)\|_2^2$ is often referred as the point-to-point energy [19]. The energy $E_{\text{plane}}(Z) = \left(\mathbf{n}_i^T (\mathbf{z}_i - P_{\mathscr{Y}}(\mathbf{z}_i^t))\right)^2$ is typically referred as the point-to-plane energy [48]. The point-to-point and point-to-plane energies are often presented as two separate entities, but they are in fact two ways of optimizing the same proximity function. It is also interesting to notice that using the ICP method with the point-to-point or the point-to-plane energy is equivalent to Equation 3.26 and Equation 3.27.

Remark 2. Pottmann et al. [142] showed that close to the surface the point-to-plane measure is a very good approximant of the second order approximation of the squared distance function of a surface at a point $\mathbf{x}$, and inversely far from the surface the point-to-point measure is a better approximant. Our optimization strategy in Equation 3.15 encodes this fact by using the parameter μ to combine point-to-point and point-to-plane distances.

3.3. Prior energy

In this section we present several prior energies that can be used for registration. These energies can also be combined to build more sophisticated priors. Priors encode properties of the object $\mathscr{X}$ being registered. For example, when scanning rigid objects, a global rigidity prior can be used to limit the allowed transformations to rotations and translations. For deforming objects, for example a human body, geometric priors are often employed that try to mimic physical behaviors such as an elastic deformation. More complex deformation behavior can be captured using a data-driven approach. One popular method is based on a collection of sample shapes that represents the space of allowed deformations. Using dimensionality reduction, like principal component analysis, efficient linear models can be derived that are suitable for realtime registration algorithms.

3.3.1. Global Rigidity

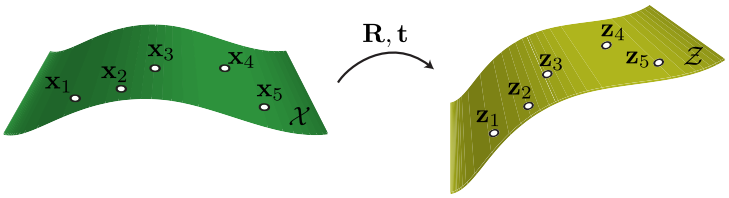
The global rigidity of the 3D registration can be measured using a proximity function

$$E_{\text{rigid}}(Z) = \sum_{i=1}^{m} \|\mathbf{z}_i - P_{\text{rigid}}(\mathbf{z_i})\|_2^2. \tag{3.28}$$

The projection $P_{\text{rigid}}(\mathbf{z_i}) = \mathbf{s}_i$ returns the closest point $\mathbf{s}_i$ from $\mathbf{z}_i$ such that

$$\mathbf{s}_i = \mathbf{R}\mathbf{x}_i + \mathbf{t}, \tag{3.29}$$

where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ a translation vector. In this case, the deformed surface $\mathcal{Z}$ tries to follow a rigid transformation of the original surface $\mathcal{X}$. The



projection $P_{\text{rigid}}(\mathbf{z_i})$ can be computed using the least-squares estimation of transformation between two point sets approach of Umeyama [175].

3.3.2. Linear Model

A 3D linear shape model can be defined using a matrix $\mathbf{B}$ containing the shape model bases, and a mean shape vector $\mathbf{m}$ [24]. A new shape $\mathbf{s}$ can be defined as

$$\mathbf{s} = \mathbf{B}\mathbf{d} + \mathbf{m}, \tag{3.30}$$

where $\mathbf{d}$ is a vector containing the bases coefficients. The linear model energy can be formulated as the deviation of the vertices from the linear model

$$E_{\text{linear}}(Z) = \|\mathbf{Z} - P_{\text{linear}}(\mathbf{Z})\|_2^2, \tag{3.31}$$

where $\mathbf{Z} = [\mathbf{z}_1^T, \mathbf{z}_2^T, \ldots, \mathbf{z}_k^T]^T$ is a vector that stacks all vertices $\mathbf{z}_1, \ldots, \mathbf{z}_k \in \mathbb{R}^3$ involved in the shape constraint. The projection $P_{\text{linear}}(\mathbf{Z}) = \mathbf{s}$ returns the closest shape $\mathbf{s}$ from $\mathbf{Z}$. Therefore, $P_{\text{linear}}(\mathbf{Z})$ can be computed in closed-form as

$$P_{\text{linear}}(\mathbf{Z}) = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T(\mathbf{Z} - \mathbf{m}) + \mathbf{m}. \tag{3.32}$$

Such linear model could be a blendshape model or a PCA model, for example.

3.3.3. General Shapes

The key observation of this section is that the proximity function is ideally suited to encode general geometric shape constraints. The projection of a set of vertices onto a geometric shape is found by minimizing the sum of the squared distances of the vertices to the corresponding feasible set. This minimum is computed through shape matching, i.e. by finding the least-squares fit of the constraint shape onto the set of vertices. We formulate the shape proximity function as

$$E_{\text{shape}}(Z) = \|\mathbf{NZ} - P_{\text{shape}}(\mathbf{NZ})\|_2^2. \tag{3.33}$$

The matrix $\mathbf{N}$ is used to center the vertices of $\mathbf{Z}$ at their mean and is defined as

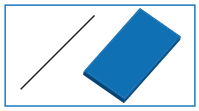
$$\mathbf{N} = (\mathbf{I}_{k\times k} - \frac{1}{k}\mathbf{1}_{k\times k}) \otimes \mathbf{I}_{3\times 3}, \tag{3.34}$$

where $\otimes$ is the Kronecker product and $\mathbf{1}_{k\times k}$ is a $k \times k$ matrix of ones. Subtracting the mean allows translational motion as a degree of freedom during the optimization. This introduces a global solve, but considerably improves convergence [29, 30]. This formulation is possible because shape projections are invariant under rigid motion, and therefore invariant under translation.

As mentioned above, we find the minimal displacement of vertices by projecting them onto the least-squares fit of the shape over those vertices. In this section we present a variety of different shape projections that can be combined, adapted, or extended to formulate new geometric optimization solutions. The original vertex positions are denoted by $\mathbf{X} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T]^T$, and the projected vertex positions by $P_{\text{shape}}(\mathbf{NZ}) = \mathbf{Z}^* = [\mathbf{z}_1^{*T}, \ldots, \mathbf{z}_k^{*T}]^T$. To simplify notation, we assume that the vertices stacked in $\mathbf{Z}$ and $\mathbf{X}$ are already mean centered.

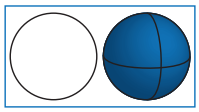
We describe three classes of constraints. *Continuous shapes*, such as planes or circles, *polygonal shapes*, such as line segments, regular polygons, or rectangles, and *relative shapes*. The latter encode the class of transformations that the shapes of the original geometry, e.g. polygons, tetrahedra, one-ring neighborhoods, etc., can undergo during the optimization. This allows the preservation of geometric properties such as lengths or angles of the original model.

3.3.3.1. Continuous Shapes



Line - Plane. This constraint specifies that the vertices of $\mathbf{Z}$ should all lie on a continuous line or plane.

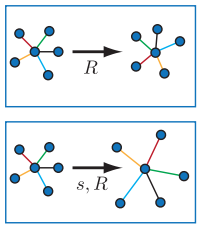
Projection: We can efficiently solve for the projection by first computing the sorted eigenvectors $\mathbf{U} = \left[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\right]$ of the 3×3 covariance matrix $\mathbf{C}\mathbf{C}^T$ where $\mathbf{C} = \left[\mathbf{z}_1, \ldots, \mathbf{z}_k\right]$. We remove the last column of $\mathbf{U}$ for plane projection and the last two columns for line projection. The projected vertices are then given as $\left[\mathbf{v}_1^*, ..., \mathbf{v}_n^*\right] = \mathbf{U}\mathbf{U}^T\mathbf{C}$.



Circle - Sphere. This constraint specifies that the vertices of $\mathbf{Z}$ should all lie on a 2D circle or a 3D sphere.

Projection: Since the direct projection of 3D vertices to their 2D least-squares circle can be computationally expensive, we apply an approximate projection. We first project the vertices onto their least-squares plane (see above) and then fit a 2D circle within that plane. Circle fitting is achieved by minimizing $\sum_j (\|\mathbf{z}_j - \mathbf{c}\|_2^2 - r^2)^2$, where r and $\mathbf{c}$ are the unknown radius and center of the circle, respectively. We solve for these parameters using the closed-form solution of [171] and project the vertices of $\mathbf{Z}$ onto this circle to obtain $\mathbf{Z}^*$. The projection onto a sphere is computed by minimizing the same equation directly on the 3D points.

3.3.3.2. Relative Shapes



Rigid - Similar. These constraints are defined relative to the original vertex set $\mathbf{X}$, i.e. they constrain the type of transformation that the vertex set can undergo. *Rigid* aims at restricting the deformations to isometries, while *Similar* aims for a conformal deformation.

Projection: Finding the closest rigid transform or similarity that maps the original vertices $\mathbf{X}$ onto the current set $\mathbf{Z}$ can be solved using the method described in [175]. The algorithm computes the rigid transformation and uniform scale using least-squares fitting and allows a minimal and maximal scale constraint by keeping the rigid transformation as is and clamping the scale to the desired range.

While this approach works well, we also propose a faster projection operator for 2D shapes. The idea is to first project the vertices onto their least-squares plane and then formulate the fitting in 2D. We denote the projected 2D points by a bar, e.g. $\bar{\mathbf{x}}_j$ is the

projection of the original vertex $\mathbf{x}_j$ onto the least-squares plane.

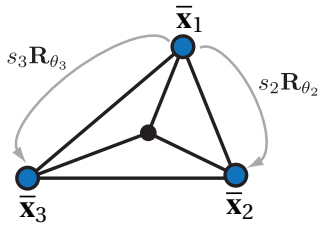
Let $\mathbf{M}$ be all the sets of points conformal to the 2D original points $\overline{\mathbf{X}} = [\overline{\mathbf{x}}_1^T, \ldots, \overline{\mathbf{x}}_k^T]^T$. We find the point set $\overline{\mathbf{Z}}^* = [\overline{\mathbf{z}}_1^{*T}, \ldots, \overline{\mathbf{z}}_k^{*T}]^T \in \mathbf{M}$ closest to $\overline{\mathbf{Z}} = [\overline{\mathbf{z}}_1^T, \ldots, \overline{\mathbf{z}}_k^T]^T$, i.e. solve for

$$\{\overline{\mathbf{z}}_1^*, \ldots, \overline{\mathbf{z}}_n^*\} = \underset{\overline{\mathbf{Z}}^* \in \mathbf{M}}{\arg\min} \sum_{j=1}^{k} \|\overline{\mathbf{z}}_j^* - \overline{\mathbf{z}}_j\|_2^2. \tag{3.35}$$

As explained in [90], at the minimum of Equation 3.35 the centroids of $\overline{\mathbf{Z}}$ and $\overline{\mathbf{Z}}^*$ coincide. Therefore, if $\overline{\mathbf{Z}}$ is centered, Equation 3.35 can be expressed as

$$\underset{\overline{\mathbf{z}}_1^*}{\arg\min} || \underbrace{\begin{bmatrix} \mathbf{I}_{2\times 2} \\ s_2\mathbf{R}_{\theta_2} \\ \vdots \\ s_n\mathbf{R}_{\theta_n} \end{bmatrix}}_{\mathbf{A}} \overline{\mathbf{z}}_1^* - \begin{bmatrix} \overline{\mathbf{z}}_1 \\ \overline{\mathbf{z}}_2 \\ \vdots \\ \overline{\mathbf{z}}_k \end{bmatrix} ||_2^2, \tag{3.36}$$

where $s_i\mathbf{R}_{\theta_i}$ represent the scale and rotation mapping the first point to the ith point in the original centered set $\overline{\mathbf{X}}$.

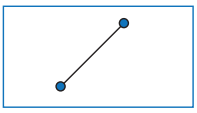


The minimum of Equation 3.36 is obtained by solving the normal equation

$$\overline{\mathbf{z}}_1^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\overline{\mathbf{Z}}. \tag{3.37}$$

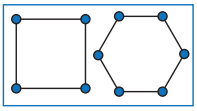
We can then express the projection as a linear operator $\mathbf{P} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, which maps the current point set $\overline{\mathbf{Z}}$ to the closest point set $\overline{\mathbf{Z}}^*$ in $\mathbf{M}$, i.e., $\overline{\mathbf{Z}}^* = \mathbf{P}\overline{\mathbf{Z}}$. The matrix $\mathbf{P}$ depends only on the original point set $\overline{\mathbf{X}}$ and can thus be precomputed. If $\mathbf{P}$ is applied to any point set in $\mathbf{M}$, by the idempotence property of the projection operator, the result is unchanged. Since $\mathbf{A}^T\mathbf{A}$ is a 2×2 matrix, this projection operator has a closed form expression.

3.3.3.3. Polygonal Shapes



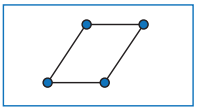
Line Segment. For a pair of vertices $\{\mathbf{z}_1, \mathbf{z}_2\}$, this constraint specifies the allowed value for their relative distance.

Projection: Let $d = \|\mathbf{z}_1 - \mathbf{z}_2\|_2$ be the current distance between the vertices and d^* the desired length of the line segment. Then the projection $\{\mathbf{z}_1^*, \mathbf{z}_2^*\}$ is computed as $\mathbf{z}_1^* = \frac{d^*}{d}\mathbf{z}_1$ and $\mathbf{z}_2^* = -\mathbf{z}_1^*$.

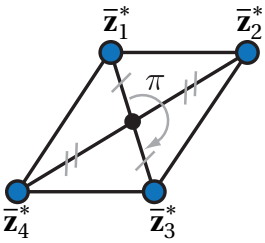


Regular Polygon. This constraint specifies that the vertex set $\mathbf{Z}$ should assume the shape of a regular polygon, i.e. have all angles be equal and all sides be of equal length.

Projection: Since a regular polygon is invariant only under similarity transformations, we can use the same projection method as described above for *relative shapes*. We simply replace the original vertex set $\mathbf{X}$ by the vertices of the regular polygon of the corresponding order.



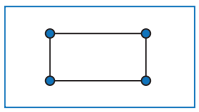
Parallelogram. This constraint specifies that a quadrilateral should become a parallelogram, i.e. have two pairs of parallel sides.



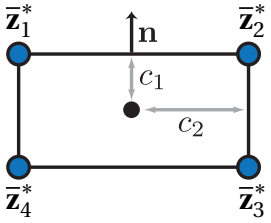
Projection: We formulate the parallelogram fitting by extending the projection for *relative shapes* as described above. We first project the vertices onto their least-squares plane, then formulate the optimization as

$$\underset{\bar{\mathbf{z}}_1^*, \bar{\mathbf{z}}_2^*}{\arg\min} \| \underbrace{\begin{bmatrix} \mathbf{I}_{4\times4} \\ -\mathbf{I}_{4\times4} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \bar{\mathbf{z}}_1^* \\ \bar{\mathbf{z}}_2^* \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{z}}_1 \\ \bar{\mathbf{z}}_2 \\ \bar{\mathbf{z}}_3 \\ \bar{\mathbf{z}}_4 \end{bmatrix} \|_2^2. \tag{3.38}$$

As previously, the solution of this optimization is $\mathbf{Z}^* = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Z}$.



Rectangle. This constraint specifies that a quadrilateral should become a rectangle, i.e. have only right angles.



Projection: We first project the vertices onto their least-squares plane and then fit the rectangle in 2D. Unlike the other polygonal shapes, we compute the equation of the four lines that define the rectangle by solving

$$\underset{c_1,c_2,\mathbf{n}}{\arg\min} || \underbrace{\begin{bmatrix} 1 & 0 & \bar{\mathbf{z}}_{1x} & \bar{\mathbf{z}}_{1y} \\ 1 & 0 & \bar{\mathbf{z}}_{2x} & \bar{\mathbf{z}}_{2y} \\ 0 & 1 & \bar{\mathbf{z}}_{2y} & -\bar{\mathbf{z}}_{2x} \\ 0 & 1 & \bar{\mathbf{z}}_{3y} & -\bar{\mathbf{z}}_{3x} \\ -1 & 0 & \bar{\mathbf{z}}_{3x} & \bar{\mathbf{z}}_{3y} \\ -1 & 0 & \bar{\mathbf{z}}_{4x} & \bar{\mathbf{z}}_{4y} \\ 0 & -1 & \bar{\mathbf{z}}_{4y} & -\bar{\mathbf{z}}_{4x} \\ 0 & -1 & \bar{\mathbf{z}}_{1y} & -\bar{\mathbf{z}}_{1x} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} c_1 \\ c_2 \\ \mathbf{n}_x \\ \mathbf{n}_y \end{bmatrix} ||_2^2 \quad \text{s.t} \quad \|\mathbf{n}\|_2^2 = 1. \tag{3.39}$$

This optimization is minimized by taking the QR decomposition of $\mathbf{A}$ and solving a 2×2 eigenvalue problem as described in [71]. We then find the projected points by computing the intersection of these four lines.

3.4. Robust Registration

3.4.1. Robust Functions

In registration, outliers are not only introduced by corrupted sensor measurements, but also by partial overlaps -– many samples on the source simply do not have an ideal corresponding point on the target shape. To address this problem, various techniques rely on a set of heuristics to either *prune* or *downweigh* low quality correspondences.

Typical criteria include discarding correspondences that are too far from each other, have dissimilar normals, or involve points on the boundary of the geometry; see [150] for details. As we will see next these heuristics are related to the optimization of robust functions. In this section we will consider robust functions as alternatives to the Euclidean metric and introduce a suitable optimization technique to use them efficiently.

In previous sections, we always considered least squares energies composed by terms like $\varphi(\epsilon(\mathbf{x}))$, where $\varphi(\epsilon) = \epsilon^2$ and $\epsilon(\mathbf{x})$ is the euclidean norm of the *residual* vector with parameters $\mathbf{x}$. This *squared Euclidian distance* metric is ideal for data corrupted by Gaussian noise [36, Sec. 7.1.1]. However, the squared Euclidian distance is not robust to outliers which are common in real world data acquired by RGB-D devices.

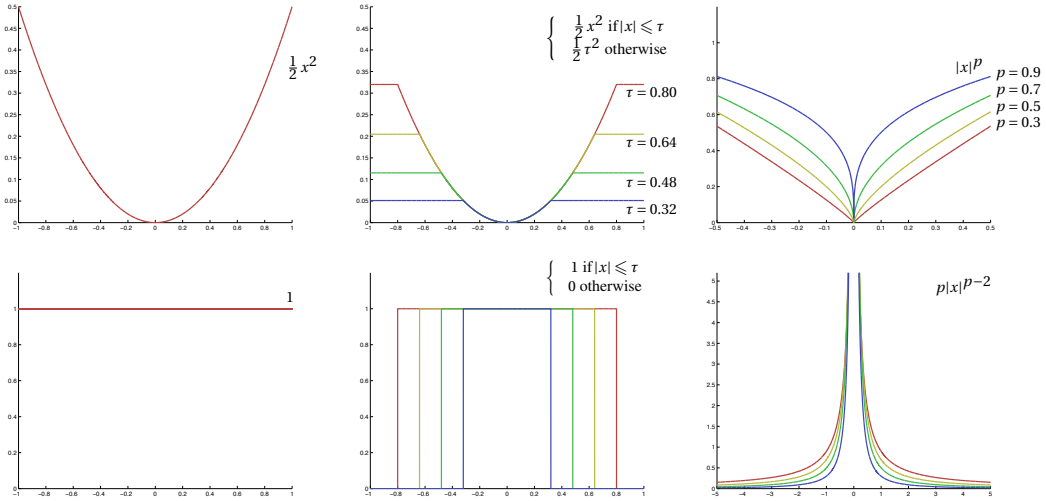


Figure 3.6.: The penalty functions φ (top) . The weight functions w (bottom).

In registration, robustness can be obtained by exploiting robust functions [125]. In this framework, $\varphi(\epsilon)$ acts as a "penalty" function – a function measuring the influence that a certain residual has in the optimization. Given one of these functions, our robust optimization can be expressed as

$$\underset{\mathbf{x}}{\arg\min} \quad \sum_{i=1}^{m} \varphi(\epsilon_i(\mathbf{x})). \tag{3.40}$$

In Figure 3.6 we show a few commonly used penalty functions. These functions possess properties like radial monotonicity and symmetry [67].

The optimization problem in Equation 3.40 can be solved using *Iteratively Re-Weighted*

Least Squares (IRLS) by solving a sequence of problems of the form

$$\arg\min_{\mathbf{x}} \quad \sum_{i=1}^{m} \alpha_i \epsilon_i(\mathbf{x})^2. \tag{3.41}$$

To understand how to compute the weights α_i first notice that the optima of Equation 3.40 can be obtained by vanishing its gradient, which can be computed by a simple application of the chain rule (note we only look at one element of the sum)

$$\frac{\partial \varphi(\epsilon(\mathbf{x}))}{\partial \mathbf{x}} = \psi(\epsilon(\mathbf{x})) \frac{\partial \epsilon(\mathbf{x})}{\partial \mathbf{x}} = w(\epsilon(\mathbf{x}))\epsilon(\mathbf{x}) \frac{\partial \epsilon(\mathbf{x})}{\partial \mathbf{x}}, \tag{3.42}$$

where $\psi(x) = \partial \varphi(x)/\partial x$ for compactness of notation and $w(x) = \psi(x)/x$ is the so called *weighting function*. Interestingly, the gradient of Equation 3.41 is

$$\frac{\partial \alpha\, \epsilon(\mathbf{x})^2}{\partial \mathbf{x}} = \alpha\, \epsilon(\mathbf{x}) \frac{\partial \epsilon(\mathbf{x})}{\partial \mathbf{x}}. \tag{3.43}$$

We can now see that by setting $\alpha = w(\epsilon(\mathbf{x}))$ the two gradients become equal. However, as the optimal weights $\alpha_i^* = w(\epsilon_i(\mathbf{p}^*))$ are not available, we use an iterative approach where at each iteration the weights are computed using the previous iteration

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} w(\epsilon_i(\mathbf{x}^t))\epsilon_i(\mathbf{x})^2. \tag{3.44}$$

This scheme is know as *Iteratively Re-Weighted Least Squares (IRLS)* and is related to majorization-minimization. The basic idea of majorization-minimization is to iteratively minimize a function always larger or equal to the objective function and with at least one point in common. If these requirements are fulfilled the algorithm converges to a stationary point [181].

Remark. In this derivation $\varphi(\cdot)$ needs to be a non-decreasing function on $\mathbb{R}^+$. In this case $\varphi(\|.\|_2)$ achieves its minimum value at the same points as $\|.\|_2$. Therefore,

$$P_{\mathscr{C}}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathscr{C}} \varphi(\|\mathbf{x}-\mathbf{y}\|_2) = \arg\min_{\mathbf{y}\in\mathscr{C}} \|\mathbf{x}-\mathbf{y}\|_2.$$

The robust proximity function can in turn be written as

$$d(\mathbf{x}) = \min_{\mathbf{y}\in\mathscr{C}} \varphi(\|\mathbf{x}-\mathbf{y}\|_2) = \varphi(\|\mathbf{x} - P_{\mathscr{C}}(\mathbf{x})\|_2),$$

and solved using IRLS and one of the optimization approaches described in Section 3.1.

3.4.2. Trimmed Metrics.

Discarding unreliable correspondences is undoubtedly the simplest and most common way of dealing with outliers [150]. Interestingly, this can as well be formulated by Equation 3.40, as it corresponds to a weight function similar to the one in Figure 3.6 (bottom-middle) whose corresponding penalty function is a truncated squared euclidean norm Figure 3.6 (top-middle). Even though this is trivial to implement, the local support of the weight function is problematic: if the source surface is too far from the target surface the registration process will not proceed as all the weights would be zero valued. A possible solution is to *dynamically* adapt the threshold value by analyzing the distribution of residuals. For example, when the ratio of outliers versus inliers is known a priori, then the threshold can be readily estimated [49].

3.4.3. Sparse Metrics

The shortcomings of trimmed metrics can be overcome by considering sparse metrics [33]. Our formulation is based on recent advances in sparsity-inducing penalties [6, 121] that have been successfully applied in compressive sensing [40]. The registration problem is formulated as recovering a transformation that maximizes the number of zero distances between correspondences. This can be achieved by minimizing the ℓ_0 norm of the vector of error residuals, or by using non-convex ℓ_p relaxations of the ℓ_0 norm where $p \leq 1$, i.e., the penalty function takes the form $\varphi(\epsilon) = |\epsilon|^p$, see Figure 3.6 (bottom-right). An important observation is that the weight functions of p-norms tend to infinity as we approach zero giving a very large reward to inliers. Moreover, contrary to trimmed metrics, p-norms weakly penalize outliers leading to a more stable approach when target and source are far apart. Problems involving ℓ_p norms can be approached by IRLS techniques [47] as presented in Section 3.4.1. In practice, however, IRLS can suffer from instability when the residuals vanish, as the weight function goes to infinity. To optimize registration problems involving the ℓ_p norm of the proximity function in a robust manner, we transform the minimization problem

$$\min_{\mathbf{x},\mathbf{y}_i} \sum_{i=1}^{m} \|\chi_i(\mathbf{x}) - \mathbf{y}_i\|_2^p + \delta_{\mathscr{F}_i}(\mathbf{y}_i) \tag{3.45}$$

to an equivalent problem

$$\min_{\mathbf{x},\mathbf{y}_i,\mathbf{s}_i} \sum_{i=1}^{m} \|\mathbf{s}_i\|_2^p + \delta_{\mathscr{F}_i}(\mathbf{y}_i) \quad \text{s.t} \quad \mathbf{s}_i = \chi_i(\mathbf{x}) - \mathbf{y}_i. \tag{3.46}$$

As detailed in Appendix A.1, the *augmented Lagrangian* method is an effective tool to approach this constrained optimization. The augmented Lagrangian function for Equation 3.46 is defined as

$$\begin{aligned}\mathscr{L}_A &= \sum_{i=1}^{m} \|\mathbf{s}_i\|_2^p + \boldsymbol{\lambda}_i^T \boldsymbol{\delta}_i + \tfrac{\mu}{2}\|\boldsymbol{\delta}_i\|_2^2 + \delta_{\mathscr{F}_i}(\mathbf{y}_i) \\ &= \sum_{i=1}^{m} \|\mathbf{s}_i\|_2^p + \tfrac{\mu}{2}\|\boldsymbol{\delta}_i + \boldsymbol{\lambda}_i/\mu\|_2^2 - \tfrac{1}{2\mu}\|\boldsymbol{\lambda}_i\|_2^2 + \delta_{\mathscr{F}_i}(\mathbf{y}_i),\end{aligned} \tag{3.47}$$

where $\boldsymbol{\delta}_i = \chi_i(\mathbf{x}) - \mathbf{y}_i - \mathbf{s}_i$, $\boldsymbol{\lambda}_i$ is a vector of Lagrange multipliers, and $\mu > 0$ is a penalty weight. We can optimize this function by employing the *Alternating Direction Method of Multipliers* (ADMM); see Appendix A.2. ADMM effectively decomposes this problem into three simple steps:

Step 1: $$\mathbf{s}_i = \underset{\mathbf{s}_i}{\arg\min} \|\mathbf{s}_i\|_2^p + \tfrac{\mu}{2}\|\mathbf{s}_i - \mathbf{h}_i\|_2^2 \tag{3.48}$$

Step 2: $$\{\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m\} = \underset{\mathbf{x},\mathbf{y}_i}{\arg\min} \sum_{i=1}^{m} \|\chi_i(\mathbf{x}) - \mathbf{y}_i - \mathbf{c}_i\|_2^2 + \delta_{\mathscr{F}_i}(\mathbf{y}_i) \tag{3.49}$$

Step 3: $$\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_i + \mu\boldsymbol{\delta}_i \tag{3.50}$$

where $\mathbf{h}_i = \chi_i(\mathbf{x}) - \mathbf{y}_i + \boldsymbol{\lambda}_i/\mu$ and $\mathbf{c}_i = \mathbf{s}_i - \boldsymbol{\lambda}_i/\mu$.

In *Step 1*, the problem is separable and each $\mathbf{s}_i$ can be optimized independently. Each sub-problem can then be solved efficiently by applying the following *shrinkage operator* [137]:

$$\mathbf{s}_i = \begin{cases} \mathbf{0} & \text{if} \quad \|\mathbf{h}_i\|_2 \le \tilde{h}_i \\ \beta_i \mathbf{h}_i & \text{if} \quad \|\mathbf{h}_i\|_2 > \tilde{h}_i \end{cases} \tag{3.51}$$

The values of β_i and $\tilde{h}_i$ are detailed in Appendix A.3. The shrinkage operator can be interpreted as a *classifier* acting on the residual vector. For example, when $p = 0$, β_i will always evaluate to one; this results in a *binary* classification: the operator either rejects the value $\mathbf{h}$ or accepts it fully.

In *Step 2*, the optimization problem can be simplified to

$$\mathbf{x} = \underset{\mathbf{x}}{\arg\min} \sum_{i=1}^{m} \|(\chi_i(\mathbf{x}) - \mathbf{c}_i) - P_{\mathscr{F}}(\chi_i(\mathbf{x}) - \mathbf{c}_i)\|_2^2, \tag{3.52}$$

as $\mathbf{y}_i = P_{\mathscr{F}}(\chi_i(\mathbf{x}) - \mathbf{c}_i)$. Therefore, this problem can be solved using one of the optimization approaches described in Section 3.1.

Chapter 4

Face Tracking

Figure 4.1.: Our system captures and tracks the facial expression dynamics of the users (grey renderings) in realtime and maps them to a digital character (colored renderings) on the opposite screen to enable engaging virtual encounters in cyberspace.

4.1. Foreword

Capturing and processing human geometry, appearance, and motion is at the core of modern computer animation. Digital actors are often created through a combination of 3D scanning, appearance acquisition, and motion capture, leading to stunning results in recent feature films. However, these methods typically require complex acquisition systems and substantial manual post-processing. As a result, creating high-quality character animation entails long turn-around times and substantial production costs. Recent developments in gaming technology, such as the Nintendo Wii and the Kinect system of

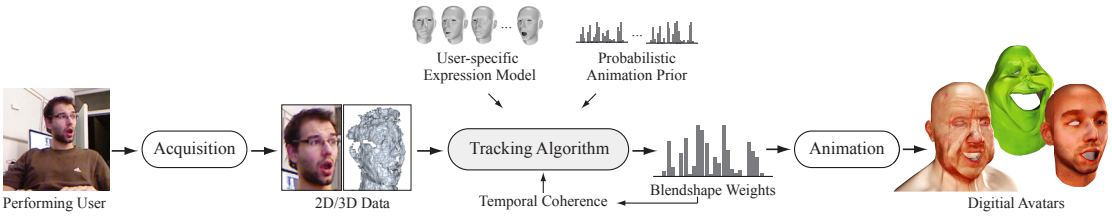


Figure 4.2.: Overview of the online processing pipeline. The blendshape weights that drive the digital avatar are estimated by matching a user-specific expression model to the acquired 2D image and 3D depth map. A probabilistic animation prior learned from existing blendshape sequences regularizes the tracking. Temporal coherence is exploited by considering a window of consecutive frames.

Microsoft, focus on robust motion tracking for compelling realtime interaction, while geometric accuracy and appearance are of secondary importance. Our goal is to leverage these technological advances and create a low-cost facial animation system that allows arbitrary users to enact a digital character with a high level of realism.

We emphasize *usability*, *performance*, and *robustness*. Usability in our context means ease of deployment and non-intrusive acquisition. These requirements put severe restrictions on the acquisition system which in turn leads to tradeoffs in the data quality and thus higher demands on the robustness of the computations. We show that even a minimal acquisition system such as the Kinect can enable compelling realtime facial animations. Any user can operate our system after recording a few standard expressions that are used to adapt a facial expression model.

Contributions. Our main contribution is a novel face tracking algorithm that combines 3D geometry and 2D texture registration in a systematic way with dynamic blendshape priors generated from existing face animation sequences. Formulated as a probabilistic optimization problem, our method successfully tracks complex facial expressions even for very noisy inputs. This is achieved by mapping the acquired depth maps and images of the performing user into the space of realistic facial expressions defined by the animation prior. Realtime processing is facilitated by a reduced facial expression model that can be easily adapted to the specific expression space and facial geometry of different users. We integrate these components into a complete framework for realtime, non-intrusive, markerless facial performance capture and animation (Figure 4.1).

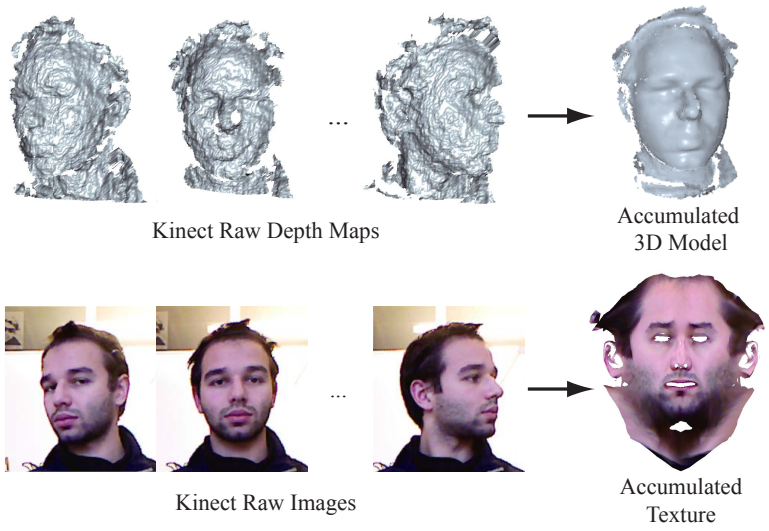


Figure 4.3.: Acquisition of user expressions for offline model building. Aggregating multiple scans under slight head rotation reduces noise and fills in missing data.

4.2. Overview

Performance-driven facial animation requires solving two main technical challenges: We need to accurately track the rigid and non-rigid motion of the user's face, and map the extracted tracking parameters to suitable animation controls that drive the virtual character. Our approach combines these two problems into a single optimization that solves for the most likely parameters of a user-specific expression model given the observed 2D and 3D data. We derive a suitable probabilistic prior for this optimization from pre-recorded animation sequences that define the space of realistic facial expressions. Figure 4.2 gives an overview of our pipeline.

Blendshape Representation. To integrate tracking and animation into one optimization, we represent facial expressions as a weighted sum of blendshape meshes. This design choice offers a number of advantages: A blendshape model provides a compact representation of the facial expression space, thus significantly reducing the dimensionality of the optimization problem. In addition, we can use existing blendshape animations, that are ubiquitous in movie and game production, to define the dynamic expression priors. The underlying hypothesis here is that the blendshape weights of a human facial animation sequence provide a sufficient level of abstraction to enable expression transfer between different characters. Finally, the output generated by our algorithm, a temporal sequence of blendshape weights, can be directly imported into commercial

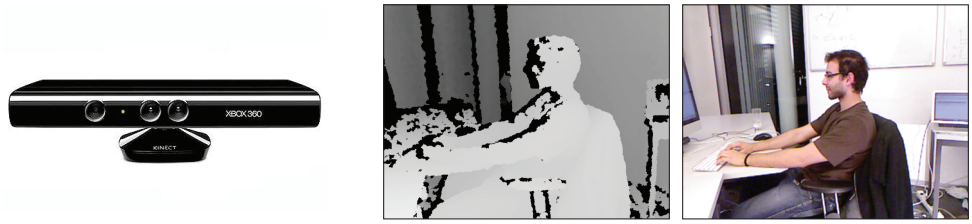


Figure 4.4.: The Kinect simultaneously captures a 640 × 400 color image and corresponding depth map at 30 Hertz, computed via triangulation of an infrared projector and camera.

animation tools, thus facilitating integration into existing production workflows.

Acquisition Hardware. All input data is acquired using the Kinect system, i.e. no other hardware such as laser scanners is required for user-specific model building. The Kinect supports simultaneous capture of a 2D color image and a 3D depth map at 30 frames per second, based on invisible infrared projection (Figure 4.4). Essential benefits of this low-cost acquisition device include ease of deployment and sustained operability in a natural environment. The user is neither required to wear any physical markers or specialized makeup, nor is the performance adversely affected by intrusive light projections or clumsy hardware contraptions. However, these key advantages come at the price of a substantial degradation in data quality compared to state-of-the-art performance capture systems based on markers and/or active lighting. Ensuring robust processing given the low resolution and high noise levels of the input data is the primary challenge that we address in this section.

4.3. Facial Expression Model

A central component of our tracking algorithm is a facial expression model that provides a low-dimensional representation of the user's expression space. We build this model in an offline preprocessing step by adapting a generic blendshape model with a small set of expressions performed by the user. These expressions are captured with the Kinect prior to online tracking and reconstructed using a morphable model combined with non-rigid alignment methods. Figure 4.5 summarizes the different steps of our algorithm for building the facial expression model. We omit a detailed description of previous methods that are integrated into our algorithm. Please refer to the cited papers for parameter settings and implementation details.

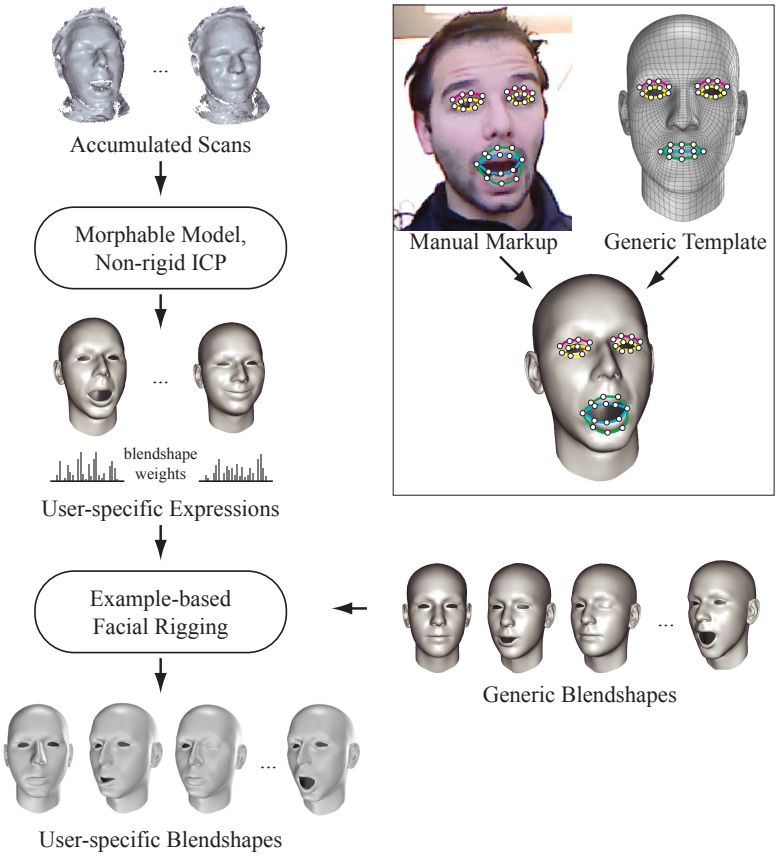


Figure 4.5.: Offline pre-processing for building the user-specific expression model. Predefined example poses of the user with known blendshape weights are scanned and registered to a template mesh to yield a set of user-specific expressions. An optimization solves for the user-specific blendshapes that maintain the semantics of a generic blendshape model. The inset shows how manually selected feature correspondences guide the reconstruction of user-specific expressions.

Data Capture. To customize the generic blendshape rig, we record a pre-defined sequence of example expressions performed by the user. Since single depth maps acquired with the Kinect exhibit high noise levels, we aggregate multiple scans over time using the method described in [188] (see Figure 4.3). The user is asked to perform a slight head rotation while keeping the expression fixed. Besides exposing the entire face to the scanner, this rotational motion has the additional benefit of alleviating reconstruction bias introduced by the spatially fixed infrared dot pattern projected by the Kinect. We use the method of [182] to detect the face in the first frame of the acquisition and accumulate the acquired color images to obtain the skin texture using Poisson reconstruction [139].

Expression Reconstruction. We use the morphable model of Blanz and Vetter [24] to represent the variations of different human faces in neutral expression. This linear PCA model is first registered towards the recorded neutral pose to obtain a high-quality template mesh that roughly matches the geometry of the user's face. We then warp this template to each of the recorded expressions using the non-rigid registration approach of [109]. To improve registration accuracy, we incorporate additional texture constraints in the mouth and eye regions. For this purpose, we manually mark features as illustrated in Figure 4.5. The integration of these constraints is straightforward and easily extends the framework of [109] with positional constraints.

Blendshape Reconstruction. We represent the dynamics of facial expressions using a generic blendshape rig based on Ekman's Facial Action Coding System (FACS) [64]. To generate the full set of blendshapes of the user we employ example-based facial rigging as proposed by Li et al. [111]. This method takes as input a generic blendshape model, the reconstructed example expressions, and approximate blendshape weights that specify the appropriate linear combination of blendshapes for each expression. Since the user is asked to perform a fixed set of expressions, these weights are manually determined once and kept constant for all users. Given this data, example-based facial rigging performs a gradient-space optimization to reconstruct the set of user-specific blendshapes that best reproduce the example expressions (Figure 4.5). We use the same generic blendshape model with $m = 39$ blendshapes in all our examples.

4.4. Realtime Tracking

The user-specific blendshape model defines a compact parameter space suitable for realtime tracking. We decouple the rigid from the non-rigid motion and directly estimate the rigid transform of the user's face before performing the optimization of blendshape weights. We found that this decoupling not only simplifies the formulation of the optimization, but also leads to improved robustness of the tracking.

Rigid Tracking. We align the reconstructed mesh of the previous frame with the acquired depth map of the current frame using ICP with point-plane constraints. To stabilize the alignment we use a pre-segmented template (Figure 4.6, left) that excludes the chin region from the registration as this part of the face typically exhibits the strongest deformations. As illustrated in Figure 4.7 this results in robust tracking even for large occlusions and extreme facial expressions. We also incorporate a temporal filter to ac-

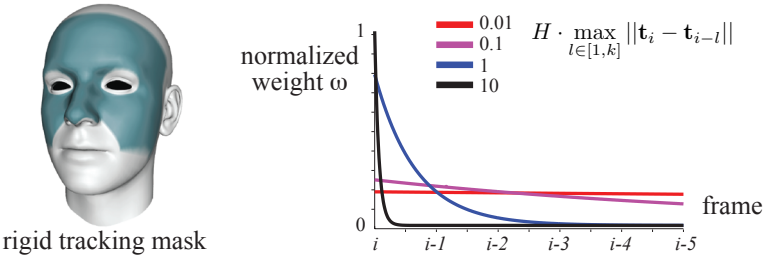


Figure 4.6.: The colored region on the left indicates the portion of the face used for rigid tracking. The graph on the right illustrates how temporal filtering adapts to the speed of motion.

count for the high-frequency flickering of the Kinect depth maps. The filter is based on a sliding window that dynamically adapts the smoothing coefficients in the spirit of the exponentially weighted moving average method [148] to reduce high frequency noise while avoiding disturbing temporal lags. We independently filter the translation vector and quaternion representation of the rotation. For a translation or quaternion vector $\mathbf{t}_i$ at the current time frame i, we compute the smoothed vector as weighted average in a window of size k as

$$\mathbf{t}_i^\star = \frac{\sum_{j=0}^{k} w_j \mathbf{t}_{i-j}}{\sum_{j=0}^{k} w_j} \tag{4.1}$$

where $\mathbf{t}_{i-j}$ denotes the vector at frame $i-j$. The weights w_j are defined as

$$w_j = e^{-j \cdot H \cdot \max_{l \in [1,k]} ||\mathbf{t}_i - \mathbf{t}_{i-l}||}, \tag{4.2}$$

with a constant H that we empirically determine independently for rotation and translation based on the noise level of a static pose. We use a window size of $k = 5$ for all our experiments.

Scaling the time scale with the maximum variation in the temporal window ensures that less averaging occurs for fast motion, while high-frequency jitter is effectively removed from the estimated rigid pose (Figure 4.6, right). As shown in the video, this leads to a stable reconstruction when the user is perfectly still, while fast and jerky motion can still be recovered accurately.

Non-rigid Tracking. Given the rigid pose, we now need to estimate the blendshape weights that capture the dynamics of the facial expression of the recorded user. Our goal is to reproduce the user's performance as closely as possible, while ensuring that the reconstructed animation lies in the space of realistic human facial expressions. Since blendshape parameters are agnostic to realism and can easily produce nonsensi-

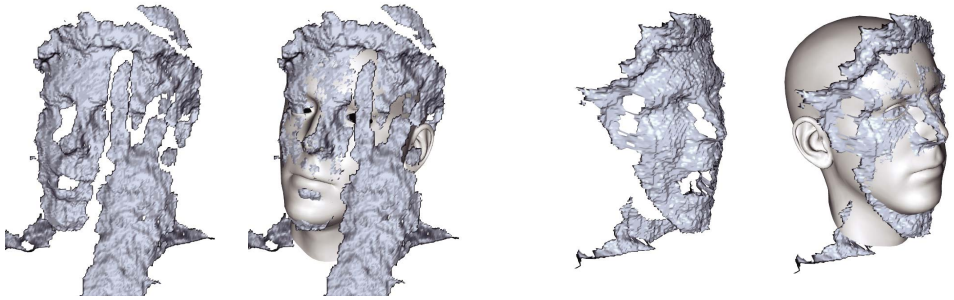


Figure 4.7.: Robustly tracking the rigid motion of the face is crucial for expression reconstruction. Even with large occlusions and fast motion, we can reliably track the user's global pose.

cal shapes, parameter fitting using geometry and texture constraints alone will typically not produce satisfactory results, in particular if the input data is corrupted by noise (see Figure 4.8). Since human visual interpretation of facial imagery is highly sophisticated, even small tracking errors can quickly lead to visually disturbing artifacts.

4.4.1. Statistical Model

We prevent unrealistic face poses by regularizing the blendshape weights with a dynamic expression prior computed from a set of existing blendshape animations $\mathscr{A} = \{A_1, \ldots, A_l\}$. Each animation A_j is a sequence of blendshape weight vectors $\mathbf{a}_j^i \in \mathbb{R}^m$ that sample a continuous path in the m-dimensional blendshape space. We exploit temporal coherence of these paths by considering a window of n consecutive frames, yielding an effective prior for both the geometry and the motion of the tracked user.

MAP Estimation. Let $D_i = (G_i, I_i)$ be the input data at the current frame i consisting of a depth map G_i and a color image I_i. We want to infer from D_i the most probable blendshape weights $\mathbf{x}_i \in \mathbb{R}^m$ for the current frame given the sequence $X_n^i = \mathbf{x}_{i-1}, \ldots, \mathbf{x}_{i-n}$ of the n previously reconstructed blendshape vectors. Dropping the index i for notational brevity we formulate this inference problem as a maximum a posteriori (MAP) estimation

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} p(\mathbf{x}|D, X_n), \tag{4.3}$$

where $p(\cdot|\cdot)$ denotes the conditional probability. Using Bayes' rule we obtain

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} p(D|\mathbf{x}, X_n)\, p(\mathbf{x}, X_n). \tag{4.4}$$

Assuming that D is conditionally independent of X_n given $\mathbf{x}$, we can write

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} \underbrace{p(D|\mathbf{x})}_{\text{likelihood}} \underbrace{p(\mathbf{x}, X_n)}_{\text{prior}}. \tag{4.5}$$

Prior Distribution. To adequately capture the nonlinear structure of the dynamic expression space while still enabling realtime performance, we represent the prior term $p(\mathbf{x}, X_n)$ as a Mixtures of Probabilistic Principal Component Analyzers (MPPCA) [173]. Probabilistic principal component analysis (PPCA) (see [174]) defines the probability density function of some observed data $\mathbf{x} \in \mathbb{R}^s$ by assuming that $\mathbf{x}$ is a linear function of a latent variable $\mathbf{z} \in \mathbb{R}^t$ with $s > t$, i.e.,

$$\mathbf{x} = \mathbf{C}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{4.6}$$

where $\mathbf{z} \sim \mathcal{N}(0, I)$ is distributed according to a unit Gaussian, $C \in \mathbb{R}^{s \times t}$ is the matrix of principal components, $\boldsymbol{\mu}$ is the mean vector, and $› \sim \mathcal{N}(0, \sigma^2 I)$ is a Gaussian-distributed noise variable. The probability density of $\mathbf{x}$ can then be written as

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C}\mathbf{C}^T + \sigma^2\mathbf{I}). \tag{4.7}$$

Using this formulation, we define the prior in Equation 4.5 as a weighted combination of K Gaussians

$$p(\mathbf{x}, X_n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}, X_n|\boldsymbol{\mu}_k, \mathbf{C}_k\mathbf{C}_k^T + \sigma_k^2\mathbf{I}). \tag{4.8}$$

with weights π_k. This representation can be interpreted as a reduced-dimension Gaussian mixture model that attempts to model the high-dimensional animation data with locally linear manifolds modeled with PPCA.

Learning the Prior. The unknown parameters in Equation 4.8 are the means $\boldsymbol{\mu}_k$, the covariance matrixes $\mathbf{C}_k\mathbf{C}_k^T$, the noise parameters σ_k, and the relative weights π_k of each PPCA in the mixture model. We learn these parameters using the Expectation Maximization (EM) algorithm based on the given blendshape animation sequences $\mathscr{A}$. To increase the robustness of these computations, we estimate the MPPCA in a latent space of the animation sequences $\mathscr{A}$ using principal component analysis. By keeping 99% of the total variance we can reduce the dimensionality of the training data by two-thirds allowing a more stable learning phase with the EM algorithm. Equation 4.8 can

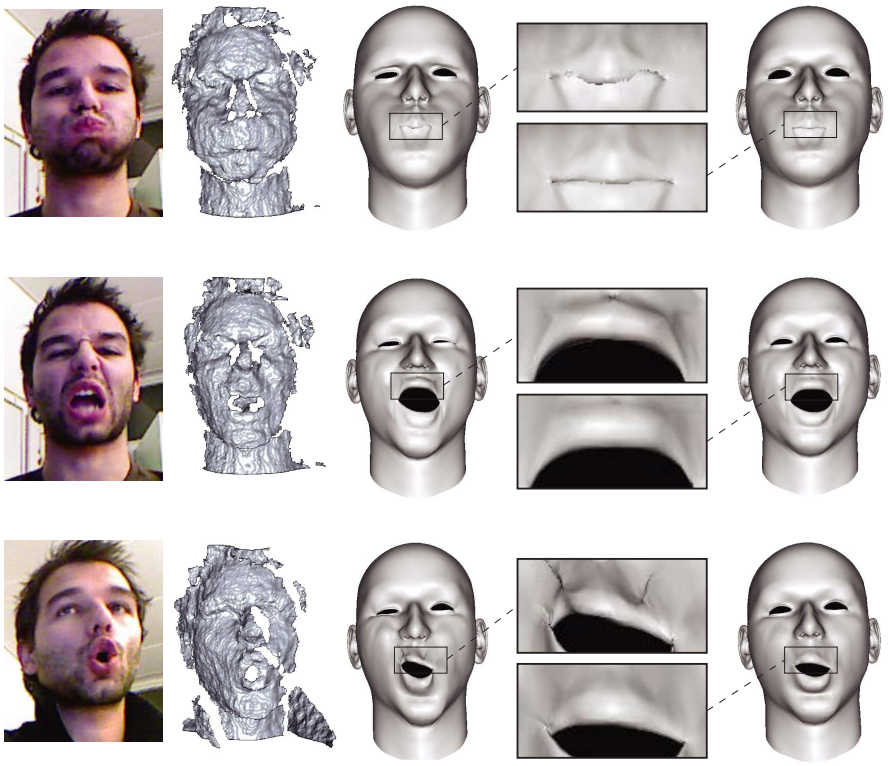


Figure 4.8.: Without the animation prior, tracking inaccuracies lead to visually disturbing self-intersections. Our solution significantly reduces these artifacts. Even when tracking is not fully accurate as in the bottom row, a plausible pose is reconstructed.

thus be rewritten as

$$p(\mathbf{x}, X_n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}, X_n | \mathbf{P}\boldsymbol{\mu}_k + \boldsymbol{\mu}, \mathbf{PMP}^T), \tag{4.9}$$

where $\mathbf{M} = (\mathbf{C}_k\mathbf{C}_k^T + \sigma_k^2\mathbf{I})$ is the covariance matrix in the latent space, $\mathbf{P}$ is the principal component matrix, and $\boldsymbol{\mu}$ the mean vector. Since the EM algorithm converges to local minima, we run the algorithm 50 times with random initialization to improve the learning accuracy. We use 20 Gaussians to model the prior distribution and we use one-third of the latent space dimension for the PPCA dimension. More details on the implementation of the EM algorithm can be found in [124].

Likelihood Distribution. By assuming conditional independence, we can model the likelihood distribution in Equation 4.5 as the product $p(D|\mathbf{x}) = p(G|\mathbf{x})p(I|\mathbf{x})$. The

two factors capture the alignment of the blendshape model with the acquired depth map and texture image, respectively. We represent the distribution of each likelihood term as a product of Gaussians, treating each vertex of the blendshape model independently.

Let V be the number of vertices in the template mesh and $\mathbf{B} \in \mathbb{R}^{V \times m}$ the blendshape matrix. Each column of B defines a blendshape base mesh such that $\mathbf{Bx}$ generates the blendshape representation of the current pose. We denote with $\mathbf{v}_i = (\mathbf{Bx})_i$ the i-th vertex of the reconstructed mesh. The likelihood term $p(G|\mathbf{x})$ models a geometric registration in the spirit of non-rigid ICP by assuming a Gaussian distribution of the per-vertex point-plane distances

$$p(G|\mathbf{x}) = \prod_{i=1}^{V} \frac{1}{(2\pi\sigma_{\text{geo}}^2)^{\frac{3}{2}}} \exp\left(-\frac{||\mathbf{n}_i^T(\mathbf{v}_i - \mathbf{v}_i^*)||^2}{2\sigma_{\text{geo}}^2}\right), \tag{4.10}$$

where $\mathbf{v}_i^*$ is the corresponding closest point in the depth map G, and $\mathbf{n}_i$ is the surface normal at $\mathbf{v}_i^*$.

The likelihood term $p(I|\mathbf{x})$ models texture registration. Since we acquire the user's face texture when building the facial expression model (Figure 4.3), we can integrate model-based optical flow constraints [56], by formulating the likelihood function using per-vertex Gaussian distributions as

$$p(I|\mathbf{x}) = \prod_{i=1}^{V} \frac{1}{2\pi\sigma_{\text{im}}^2} \exp\left(-\frac{||\nabla I_i^T(\mathbf{p}_i - \mathbf{p}_i^*)||^2}{2\sigma_{\text{im}}^2}\right), \tag{4.11}$$

where $\mathbf{p}_i$ is the projection of $\mathbf{v}_i$ into the image I, $\mathbf{p}_i^*$ is the corresponding point in the rendered texture image, and ∇I_i is the gradient of I at $\mathbf{p}_i^*$.

4.4.2. Optimization

In order to solve the MAP problem as defined by Equation 4.5 we minimize the negative logarithm, i.e.,

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} -\ln p(G|\mathbf{x}) - \ln p(I|\mathbf{x}) - \ln p(\mathbf{x}, X_n). \tag{4.12}$$

Discarding constants, we write

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} E_{\text{geo}} + E_{\text{im}} + E_{\text{prior}}, \tag{4.13}$$

where

$$E_{\text{prior}} = -\ln p(\mathbf{x}, X_n), \tag{4.14}$$

$$E_{\text{geo}} = \frac{1}{\sigma_{\text{geo}}^2} \sum_{i=1}^{V} ||\mathbf{n}_j^T (\mathbf{v}_i - \mathbf{v}_i^*)||^2, \text{and} \tag{4.15}$$

$$E_{\text{im}} = \frac{1}{\sigma_{\text{im}}^2} \sum_{i=1}^{V} ||\nabla I_i^T (\mathbf{p}_i - \mathbf{p}_i^*)||^2. \tag{4.16}$$

The parameters σ_{geo} and σ_{im} model the noise level of the data that controls the emphasis of the geometry and image likelihood terms relative to the prior term. Since our system provides realtime feedback, we can experimentally determine suitable values that achieve stable tracking performance. For all our results we use the same settings $\sigma_{\text{geo}} = 1$ and $\sigma_{\text{im}} = 0.45$.

The optimization of Equation 4.13 can be performed efficiently using an iterative gradient solver, since the gradients can be computed analytically (see the derivations in the Appendix). In addition, we precompute the inverse covariance matrices and the determinants of the MPPCA during the offline learning phase. We use a gradient projection algorithm based on the limited memory BFGS solver [117] in order to enforce that the blendshape weights are between 0 and 1. The algorithm converges in less that 6 iterations as we can use an efficient warm starting with the previous solution. We then update the closest point correspondences in E_{geo} and E_{im}, and re-compute the MAP estimation. We found that 3 iterations of this outer loop are sufficient for convergence.

4.5. Results

We present results of our realtime performance capture and animation system and illustrate potential applications. The output of the tracking optimization is a continuous stream of blendshape weight vectors $\{\mathbf{x}_i\}$ that drive the digital character. Figures 4.1 and 4.9 illustrates how our system can be applied in interactive applications, where the user controls a digital avatar in realtime. Blendshape weights can be transmitted in realtime to enable virtual encounters in cyberspace. Since the blendshape representation facilitates animation transfer, the avatar can either be a digital representation of the user himself or a different humanoid character, assuming compatible expression spaces.

While we build the user-specific blendshape model primarily for realtime tracking, our technique offers a simple way to create personalized blendshape rigs that can be used in traditional animation tools. Since the Kinect is the only acquisition device required,

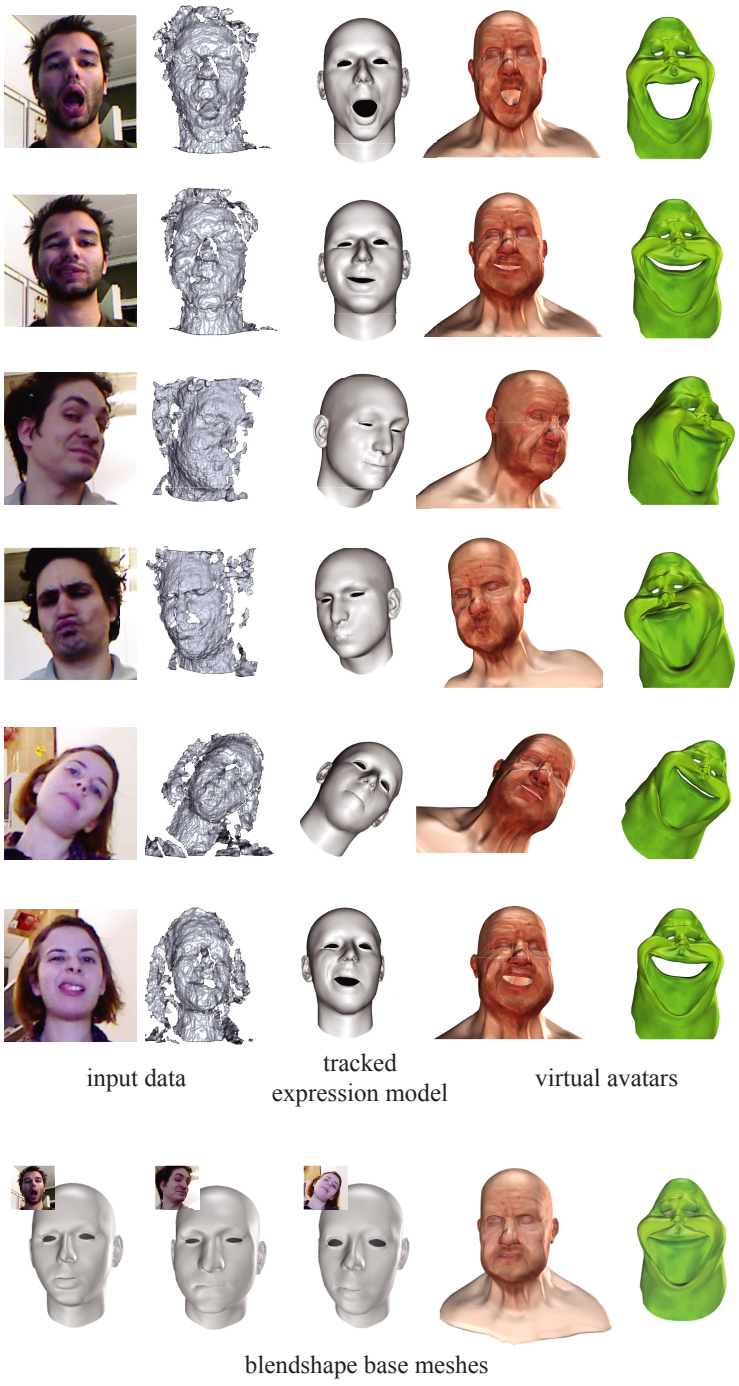


Figure 4.9.: The user's facial expressions are reconstructed and mapped to different target characters in realtime, enabling interactive animations and virtual conversations controlled by the performance of the tracked user. The smile on the green character's base mesh gives it a happy countenance for the entire animation.

generating facial rigs becomes accessible for non-professional users.

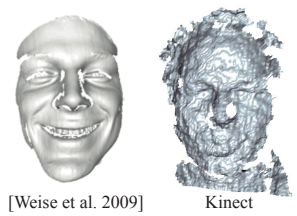
Statistics. We use 15 user-specific expressions to reconstruct 39 blendshapes for the facial expression model. Manual markup of texture constraints for the initial offline model building requires approximately 2 minutes per expression. Computing the expression model given the user input takes less than 10 minutes. We pre-compute the Gaussian mixture model that defines the dynamic expression prior from a total of 9,500 animation frames generated on the generic template model by an animation artist. Depending on the size of the temporal window, these computations take between 10 and 20 minutes.

Our online system achieves sustained framerates of 20 Hertz with a latency below 150 ms. Data acquisition, preprocessing, rigid registration, and display take less than 5 ms. Nonrigid registration including constraint setup and gradient optimization require 45 ms per frame. All timing measurements have been done on a Intel I7 2.8Ghz with 8 GBytes of main memory and a ATI Radeon HD 4850 graphics card.

4.6. Evaluation

We focus our evaluation on the integration of 2D and 3D input data and the effect of animation training data. We also comment on limitations and drawbacks of our approach.

Geometry and Texture. Figure 4.10 evaluates the interplay between the geometry and texture information acquired with the Kinect.


[Weise et al. 2009] Kinect

Tracking purely based on geometry as proposed in [189] is not successful due to the high noise level of the Kinect data. Integrating model-based optical flow constraints reduces temporal jitter and stabilizes the reconstruction. In our experiments, only the combination of both modalities yielded satisfactory results. Compared to purely image-based tracking as e.g. in [44], direct access to 3D geometry offers two main benefits: We can significantly improve the robustness of the rigid pose estimation in particular for non-frontal views (see also Figure 4.7). In addition, the expression template mesh generated during preprocessing much more closely matches the geometry of

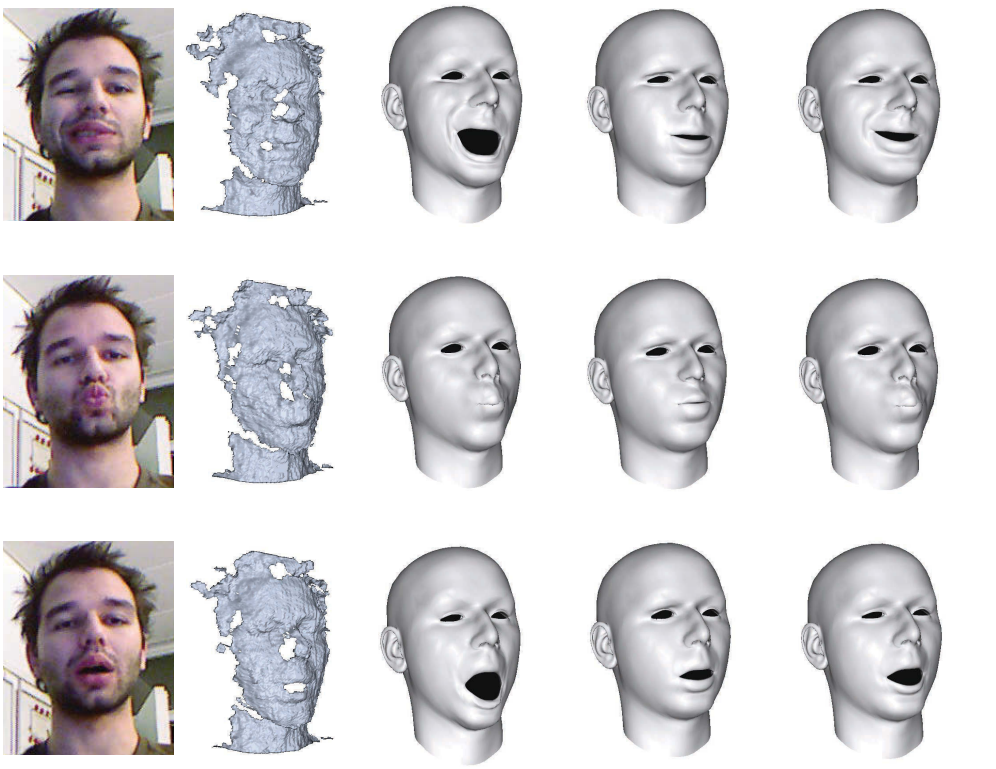


Figure 4.10.: The combination of geometric and texture-based registration is essential for realtime tracking. To isolate the effects of the individual components, no animation prior is used in this example.

the user, which further improves tracking accuracy. Figure 4.11 shows difficult tracking configurations and provides an indication of the limits of our algorithm.

Animation Prior. Figure 4.12 studies the effectiveness of our probabilistic tracking algorithm when varying the amount of training data used for the reconstruction. The figure illustrates that if the training data does not contain any sequences that are sufficiently close to the captured performance, the reconstruction can differ substantially from the acquired data. With more training data, the tracked model more closely matches the performing user. What the prior achieves in any case is that the reconstructed pose is plausible, even if not necessarily close to the input geometrically (see also Figure 4.8). We argue that this is typically much more tolerable than generating unnatural or even physically impossible poses that could severely degrade the visual perception of the avatar. In addition, our approach is scalable in the sense that if the reconstructed animation does not well represent certain expressions of the user, we can manually correct the sequence using standard blendshape animation tools and add the corrected sequence

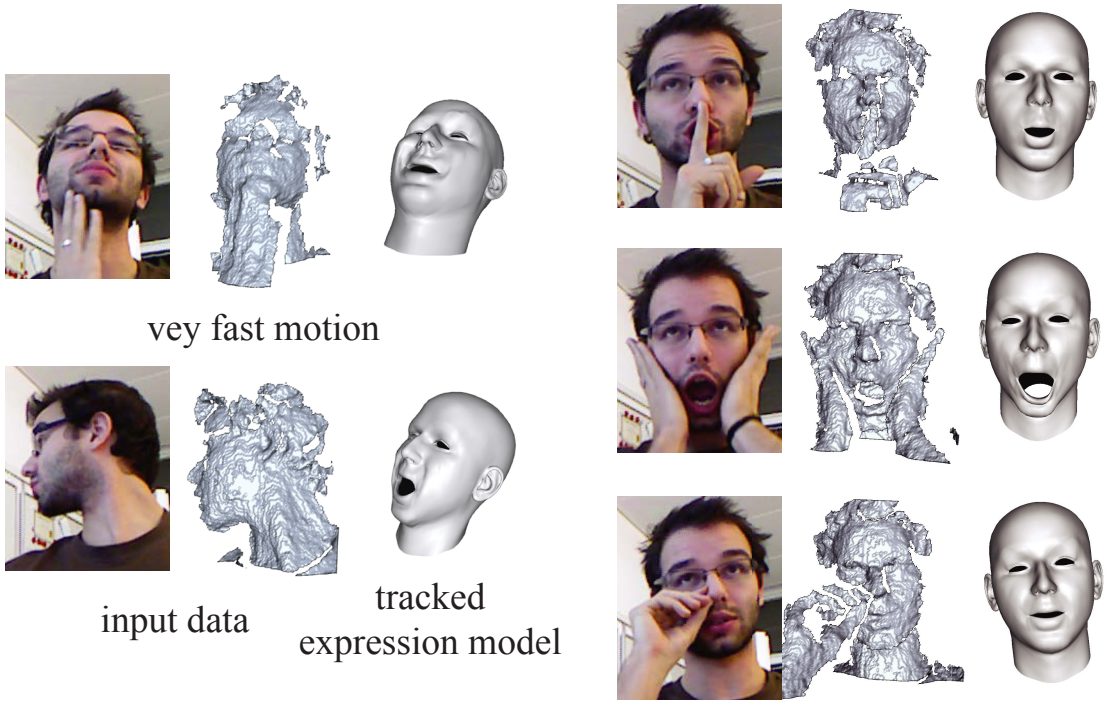


Figure 4.11.: Difficult tracking configurations. Right: despite the occlusions by the hands, our algorithm successfully tracks the rigid motion and the expression of the user. Left: with more occlusion or very fast motion, tracking can fail.

to the training data set. This allows to successively improve the animation prior in a bootstrapping manner. For the temporal window X_n used in the animation prior, we found a window size of $3 \leq n \leq 5$ to yield good results in general. Longer temporal spans raise the dimensionality and lead to increased temporal smoothing. If the window is too small, temporal coherence is reduced and discontinuities in the tracking data can lead to artifacts.

Limitations. The resolution of the acquisition system limits the amount of geometric and motion detail that can be tracked for each user, hence slight differences in expressions will not be captured adequately. This limitation is aggravated by the wide-angle lens of the Kinect installed to enable full-body capture, which confines the face region to about 160 × 160 pixels or less than 10% of the total image area. As a result, our system cannot recover small-scale wrinkles or very subtle movements. We also currently do not model eyes, teeth, tongue, or hair.

In our current implementation, we require user support during pre-processing in the

form of manual markup of lip and eye features to register the generic template with the recorded training poses (see Figure 4.5). In future work, we want to explore the potential of generic active appearance models similar to [51] to automate this step of the offline processing pipeline as well.

While offering many advantages as discussed in Section 4.2, the blendshape representation also has an inherent limitation: The number of blendshapes is a tradeoff between expressiveness of the model and suitability for tracking. Too few blendshapes may result in user expressions that cannot be represented adequately by the pose space of the model. Introducing additional blendshapes to the rig can circumvent this problem, but too many blendshapes may result in a different issue: Since blendshapes may become approximately linearly dependent, there might not be a unique set of blendshape weights for a given expression. This can potentially result in unstable tracking due to overfitting of the noisy data. While the prior prevents this instability, a larger number of blendshapes requires a larger training database and negatively affects performance.

4.7. Additions and Remarks

Enhancing the tracking performance using realtime speech analysis, or integrating secondary effects such as simulation of hair are further areas of future research that could help increase the realism of the generated virtual performances. More fundamentally, being able to deploy our system at a massive scale can enable interesting new research in human communication and paves the way for new interaction metaphors in performance-based game play. In this chapter we focused on facial acquisition and ignore other important aspects of human communication, such as hand gestures, which pose interesting technical challenges due to complex occlusion patterns. In [166] we show that a similar optimization can be used for hand tracking. We present a robust method for capturing articulated hand motions in realtime using a single depth camera. Our system is based on a realtime registration process that accurately reconstructs hand poses by fitting a 3D articulated hand model to depth images.We register the hand model using depth, silhouette, and temporal information. To effectively map low-quality depth maps to realistic hand poses, we regularize the registration with kinematic and temporal priors, as well as a data-driven prior built from a database of realistic hand poses.

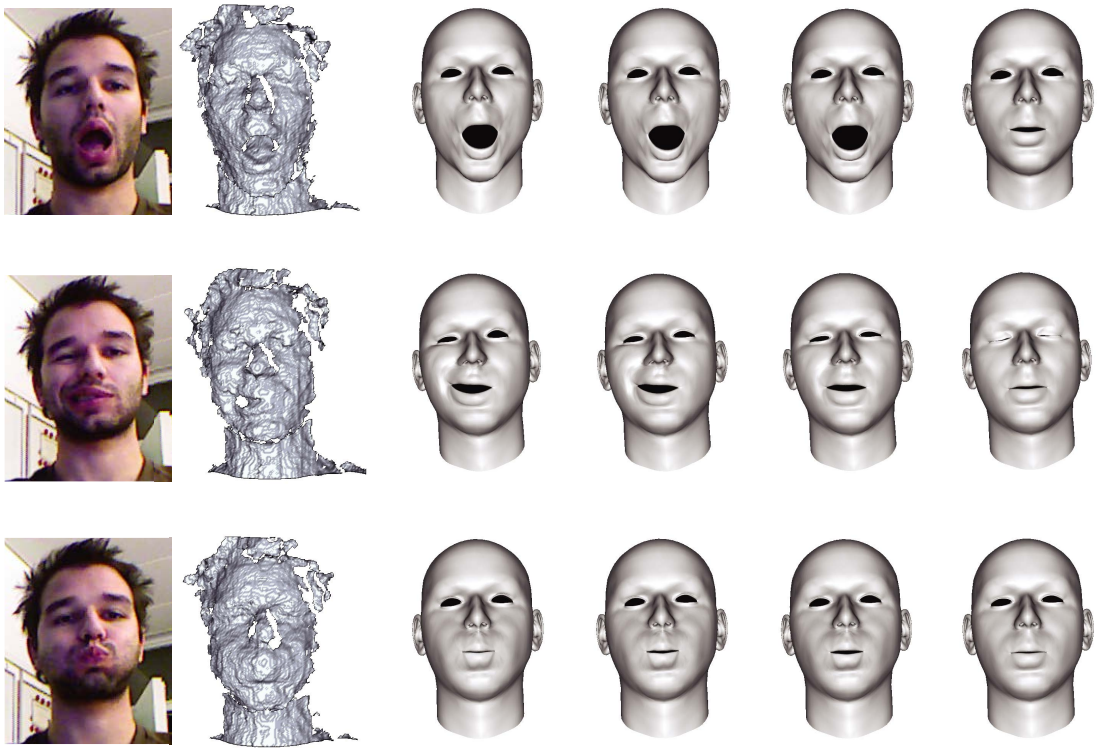


Figure 4.12.: Effect of different amounts of training data on the performance of the tracking algorithm. We successively delete blendshapes from the input animation sequences, which removes entire portions of the expression space. With only 25% of the blendshapes in the training data the expressions are not reconstructed correctly.

Animation prior using Gaussian Mixture Regression. In Section 4.4.1, we formulate the inference problem as a maximum a posteriori estimation

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} p(D|\mathbf{x})\, p(\mathbf{x}, X_n). \tag{4.17}$$

Using Bayes' rule this maximization can also be reformulated as

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} p(D|\mathbf{x})\, p(\mathbf{x}|X_n). \tag{4.18}$$

We can use Gaussian Mixture Regression [165] to create a Gaussian predictive distribution

$$\tilde{p}(\mathbf{x}|X_n) = \mathcal{N}(\mathbf{x}|E\{p(\mathbf{x}|X_n)\}, V\{p(\mathbf{x}|X_n)\}), \tag{4.19}$$

where $E\{p(\mathbf{x}|X_n)\}$ and $V\{p(\mathbf{x}|X_n)\}$ denote the expectation and the variance of $\mathbf{x}$ given X_n. This expectation and variance can be estimated from the joint density built using

the Gaussian Mixture model in Equation 4.9 (see [165] for more details). By using a Gaussian predictive distribution as prior the minimization in Equation 4.13 only consists of least-squares energies and the minimum can be computed efficiently using a direct solver instead of BFGS [117] .

Chapter 5

Face Modeling

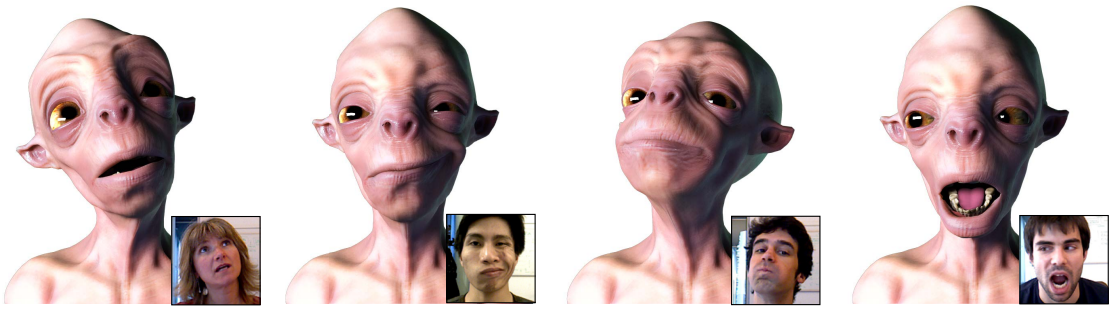


Figure 5.1.: Realtime tracking and retargeting of the facial expressions of the user (inset) captured with an RGB-D sensor.

5.1. Foreword

Recent advances in realtime performance capture have brought within reach a new form of human communication. Capturing dynamic facial expressions of a user and retargeting these expressions to a digital character in realtime allows enacting arbitrary virtual avatars with live feedback. Compared to communication via recorded video streams that only offer limited ability to alter one's appearance, such technology opens the door to fascinating new applications in computer gaming, social networks, television, training, customer support, or other forms of online interactions.

Successfully deploying such a technology at a large scale puts high demands on performance and usability. Facial tracking needs to be accurate and fast enough to create plausible and responsive animations that faithfully match the performance of the captured

user. Ease-of-use affects both hardware and system handling. Marker-based systems, multi-camera capture devices, or intrusive scanners commonly used in high-end animation production are not suitable for consumer-level applications. Equally inappropriate are methods that require complex calibration or necessitate extensive manual assistance to setup or operate the system.

Several realtime methods for face tracking have been proposed that require only a single video camera [44, 5, 152] or consumer-level RGB-D camera, such as the Microsoft Kinect [187, 7]. Video-based methods typically track a few facial features and often lack fine-scale detail, which limits the quality of the resulting animations. Tracking performance can also degrade in difficult lighting situations that commonly occur in a home environment, for example. Additionally exploiting 3D depth information obtained by active IR sensing improves tracking accuracy and robustness. This is commonly achieved using a 3D template model [37, 178] or building a dynamic 3D expression model (DEM) that represents the 3D geometry of the individual facial expressions of the user [187]. The DEM allows formulating facial tracking as a non-rigid registration problem in a low-dimensional parameter space, thus facilitating robust and efficient tracking.

However, current methods have one major drawback: The DEM must be created a priori during a controlled training stage, where each user is scanned in several pre-defined expressions. Manual corrections and parameter tuning is often required to achieve satisfactory tracking results. While appropriate for professionals in animation content creation, such user-specific calibration is a severe impediment for deployment in consumer-level applications. We propose an algorithm that addresses this problem.

Contributions. We introduce an adaptive DEM that combines a dynamic expression template, an identity PCA model, and a parameterized deformation model in a low-dimensional representation suitable for online learning. We show how this generic model can be adapted to a specific user on-the-fly without any manual assistance. Our core algorithmic contribution integrates online DEM learning directly into the tracking method. As more and more of the user's expression space is observed during tracking, the generic DEM is progressively adapted to the facial features of the specific user, which in turn will lead to more accurate tracking. Combined with state-of-the-art registration methods, our algorithm yields a fully automatic, realtime face tracking and animation system suitable for consumer-level applications (see Figures 5.1 and 5.12).

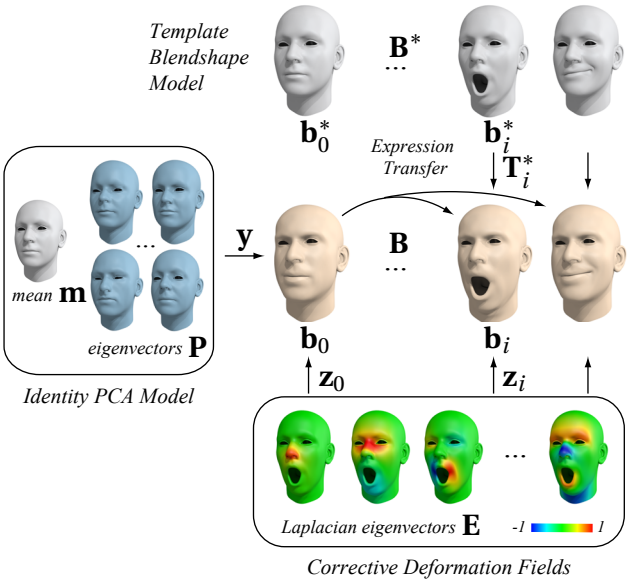


Figure 5.2.: Adaptive DEM. The user-specific blendshape model **B** is created using a combination of identity PCA model, expression transfer from the template model $\mathbf{B}^*$, and corrective deformation fields for each blendshape.

5.2. Overview.

The input to our system comes from a consumer-level RGB-D device, such as the Microsoft Kinect or the Asus Xtion Live, that provides a color image and 3D depth map of 640x480 resolution at 30 Hz. Due to the wide-angle lens, the face is confined to a region of about 160x160 pixels. Our goal is to estimate expression parameters that accurately capture the facial dynamics of the observed user in a representation that is appropriate for animating digital avatars. Similar to previous work, e.g. [187, 91], we employ a 3D blendshape model that offers a compact representation suitable for realtime tracking. In our system we build the specific blendshape model of a user concurrently to the tracking optimization, requiring no preceding training or calibration stage. Starting from a rough initial estimate, the dynamic expression model (DEM) is continuously refined as tracking progresses. As soon as each blendshape has been observed sufficiently many times, the DEM converges to a steady state.

We first describe our adaptive DEM that can be customized on the fly to the particular expression space of the user (Section 5.3). Then in Section 5.4 we show how realtime tracking can be achieved by registering the DEM with the observed image and depth map data, while concurrently refining the DEM to match the geometry of the observed user. Section 5.5 provides a detailed evaluation to demonstrate that our realtime performance-based animation system achieves accurate tracking results.

5.3. Adaptive Dynamic Expression Model

Blendshapes. We represent a DEM as a set of blendshape meshes $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_n]$, where $\mathbf{b}_0$ is the neutral pose and the $\mathbf{b}_i, i > 0$ define specific base expressions. All blendshapes have the same static mesh combinatorics and are represented by stacked coordinate vectors. A new facial expression is generated as $\mathbf{F}(\mathbf{x}) = \mathbf{b}_0 + \Delta\mathbf{B}\mathbf{x}$, where $\Delta\mathbf{B} = [\mathbf{b}_1 - \mathbf{b}_0, \ldots, \mathbf{b}_n - \mathbf{b}_0]$, and $\mathbf{x} = [x_1, \ldots, x_n]^T$ are blendshape weights bounded between 0 and 1. The blendshape representation is well suited for realtime performance capture because it reduces tracking to estimating the rigid head alignment and the n blendshape weights for each frame. As an additional benefit, the blendshapes $\mathbf{b}_i$ can be chosen to match pre-defined semantics of common face animation controllers, e.g. *mouth-open*, *smile*, *frown*, etc., which simplifies post-editing and animation retargeting.

We denote with $\mathbf{B}^* = [\mathbf{b}_0^*, \ldots, \mathbf{b}_n^*]$ a template blendshape model that is given a priori, in our case modeled by hand (see additional material for a complete list of blendshapes). This template model defines the expression semantics that we want to transfer onto the DEM of the tracked user during online model building as described in Section 5.4. Next, we introduce the main ingredients to achieve this dynamic adaptation: an identity PCA model, an expression transfer operator, and corrective deformation fields (Figure 5.2).

Identity PCA model. We capture variations of face geometry across different users with a morphable model as proposed in [24]. Given a large set of meshes of different human faces with one-to-one vertex correspondence in neutral expression, we build a reduced representation using PCA on the stacked vertex coordinate vectors. Let $\mathbf{m}$ be the resulting mean face and $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_l]$ the first l PCA eigenvectors. With such an orthonormal basis, a specific face model in neutral expression can be approximated as $\mathbf{b}_0 = \mathbf{m} + \mathbf{P}\mathbf{y}$ with suitable linear coefficients $\mathbf{y} = [y_1, \ldots, y_l]^T$.

Expression transfer operator. For a given neutral expression $\mathbf{b}_0$ we define approximations for all other blendshapes using a variant of deformation transfer [164], see also [111]. Using the template $\mathbf{B}^*$, we transfer the known deformation of the neutral expression $\mathbf{b}_0^*$ to a specific blendshape expression $\mathbf{b}_i^*$, onto the neutral expression $\mathbf{b}_0$ in order to obtain $\mathbf{b}_i$. Our formulation defines $\mathbf{b}_i$ as a linear transformation $\mathbf{T}_i^*\mathbf{b}_0$ of the neutral expression $\mathbf{b}_0$. Contrary to previous formulations of deformation transfer [164, 27], the operator $\mathbf{T}_i^*$ does not depend on $\mathbf{b}_0$, which allows the model refinement optimization to be formulated as the solution to a linear system that can be computed efficiently and robustly (see Section 5.4.2). A derivation of our expression transfer operator $\mathbf{T}_i^*$ is given

in the appendix.

Corrective deformation fields. The PCA model represents the large-scale variability of facial geometries in the neutral expression, but might not capture user-specific details. Similarly, deformation transfer copies expressions from the template without accounting for the particular facial dynamics of the user. We therefore apply additional surface deformation fields to each reconstructed blendshape mesh $\mathbf{b}_i \in \mathbf{B}$ to obtain a more faithful reconstruction of the user's facial expression space.
Per-vertex displacements are modeled using a spectral representation defined by the k last eigenvectors $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_k]$ of the graph Laplacian matrix $\mathbf{L}$ computed on the 3D face mesh, see [107] for more details. A smooth deformation field can then be defined as a linear combination $\mathbf{Ez}$, where $\mathbf{z} = [z_1, \ldots, z_k]^T$ are the spectral coefficients. The spectral basis offers two main advantages in our setting: We can optimize for the corrective deformations in a low-dimensional space, requiring only k variables to represent a deformation of a blendshape mesh. In addition, the built-in smoothness of the low-frequency eigenvectors helps to avoid over-fitting when aligning the blendshapes to noisy depth maps.

Parameterized DEM. With all this machinery in place, we can now define a parameterized DEM that can be adapted to a particular user (see Figure 5.2). The neutral expression is given as $\mathbf{b}_0 = \mathbf{m} + \mathbf{Py} + \mathbf{Ez}_0$, i.e., a combination of identity PCA model and a corrective deformation field. The remaining blendshapes $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are parameterized as

$$\mathbf{b}_i = \mathbf{T}_i^* \mathbf{b}_0 + \mathbf{Ez}_i = \mathbf{T}_i^* (\mathbf{m} + \mathbf{Py} + \mathbf{Ez}_0) + \mathbf{Ez}_i,$$

i.e., combining expression transfer of the template $\mathbf{B}^*$ to the neutral expression $\mathbf{b}_0$ with expression-specific deformation fields.

5.4. Optimization

The adaptive DEM described in the previous section is at the core of our tracking optimization algorithm. The goal of this optimization is to compute accurate tracking parameters, while at the same time refining the user-specific DEM in realtime.

More precisely, our algorithm solves for

- the rigid alignment of the face model to the input depth map defined by a rotation

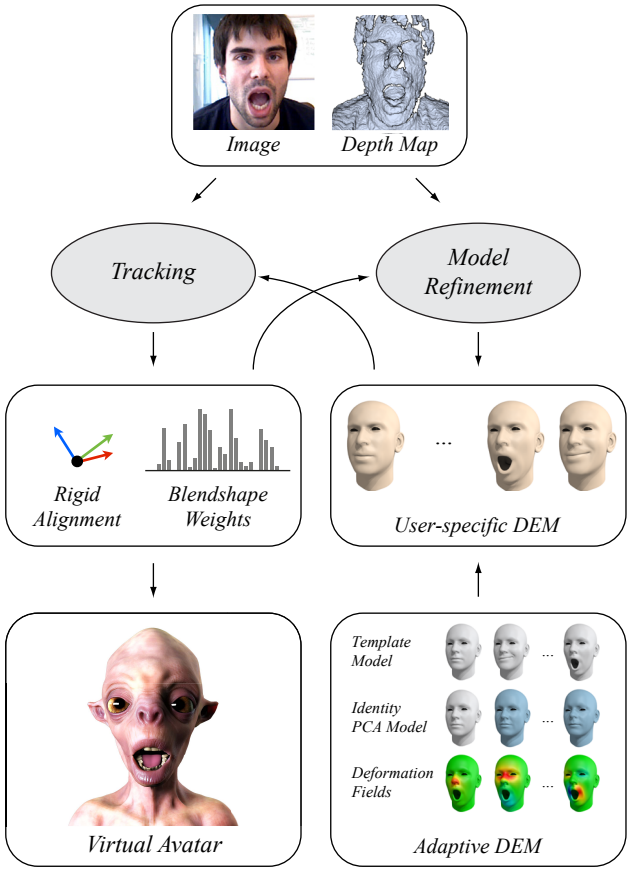


Figure 5.3.: Optimization pipeline. Each frame of the input data (color image and depth map), is processed with our interleaved optimization that alternates tracking and model refinement. The output are tracking parameters (rigid alignment, blendshape weights) per frame that can be used to drive a virtual avatar in realtime. Concurrently, the user-specific DEM is adapted according to the facial characteristics of the observed user.

matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ at each frame t,

- the blendshape weights $\mathbf{x} = [x_1, \ldots, x_n]^T$ for each frame t,
- the identity PCA parameters $\mathbf{y} = [y_1, \ldots, y_l]^T$ for the neutral face expression $\mathbf{b}_0$ of the user, and
- the deformation coefficients $\mathbf{Z} = \{\mathbf{z}_0, \ldots, \mathbf{z}_n\}$ for each blendshape $\mathbf{b}_i$, where $\mathbf{z}_i = [z_{i,1}, \ldots z_{i,k}]^T$.

We use superscripts to refer to specific time frames, e.g. $\mathbf{x}^t$ denotes the blendshape weights at frame $t \in \mathbb{N}$, where $t = 1$ denotes the first frame. To simplify notation, we

omit the superscripts when irrelevant or clear from the context.

The optimization alternates between two stages as shown in Figure 5.3. Stage I estimates the rigid alignment and blendshape weights, keeping the DEM fixed. Stage II refines the user-specific DEM by solving for the PCA parameters $\mathbf{y}$ and deformation coefficients $\mathbf{Z}$, keeping the blendshape weights fixed. We bootstrap this alternating minimization by initializing the DEM with the PCA reconstruction for the neutral expression and deformation transfer of the template DEM as described next.

Initialization. Our system requires the user to enter the sensor's field of view in a neutral facial expression. We use the method of [182] to detect the face and crop the depth map to obtain a 3D scan of the neutral expression. From this initial face scan, we compute a first approximation of $\mathbf{b}_0$ by aligning the parameterized neutral expression to the depth map. This means that we solve for the PCA coefficients $\mathbf{y}$ and deformation coefficients $\mathbf{z}_0$, as well as the rigid head pose $(\mathbf{R}, \mathbf{t})$, by minimizing the common ICP energy with point-plane constraints [150]. More specifically, we solve for

$$\underset{\mathbf{R},\mathbf{t},\mathbf{y},\mathbf{z}_0}{\arg\min} \|\mathbf{A}_0(\mathbf{R}\mathbf{b}_0 + \mathbf{t}) - \mathbf{c}_0\|_2^2 + \beta_1 \|\mathbf{D_P}\mathbf{y}\|_2^2 + \beta_2 \|\mathbf{D_E}\mathbf{z}_0\|_2^2 + \beta_3 \|\mathbf{z}_0\|_2^2. \tag{5.1}$$

Here $(\mathbf{A}_0, \mathbf{c}_0)$, is the matrix resp. right-hand side summarizing the ICP constraint equations in the first term of the objective function (see [187] for details). The remaining summands are regularization terms with corresponding positive scalar weights β_1, β_2, β_3. The term $\mathbf{D_P}\mathbf{y}$ regularizes the PCA weights, where $\mathbf{D_P}$ is a diagonal matrix containing the inverse of the standard deviation of the PCA basis. The term $\mathbf{D_E}\mathbf{z}_0$ regularizes the deformation coefficients by measuring the bending of the deformation. $\mathbf{D_E}$ is the diagonal matrix of eigenvalues corresponding to the eigenvectors in $\mathbf{E}$ of the Laplacian matrix $\mathbf{L}$ [26]. The last summand penalizes the magnitude of the deformation vectors.

The optimization is solved using the Gauss-Newton method [120]. We initialize the solver with $\mathbf{y} = \mathbf{z}_0 = 0$, the initial face location is retrieved from the face detector with the user assumed to be front-facing. Given the reconstruction of $\mathbf{b}_0^1$ at the first frame $(t = 1)$, we initialize the additional blendshapes by applying the deformation transfer operator, i.e. $\mathbf{b}_i^1 = \mathbf{T}_i^* \mathbf{b}_0^1$ for $i = 1, \ldots, n$.

5.4.1. Tracking

The tracking stage of the optimization assumes that the DEM is fixed and solves for the rigid motion $(\mathbf{R}, \mathbf{t})$ and blendshape weights $\mathbf{x}$ at timeframe t.

Rigid motion tracking. We first estimate $\mathbf{R}$ and $\mathbf{t}$ by directly aligning the static reconstructed mesh of the previous frame with the acquired depth map of the current frame using ICP with point-plane constraints. To stabilize the rigid motion, the constraints are only defined for the front head and nose region of the reconstructed mesh as illustrated in blue on the right.

Estimating blendshape weights. Given the rigid pose and the current set of blendshapes $\mathbf{B}$, we now need to estimate the blendshape weights $\mathbf{x}$ that best match the input data of the current frame. We formulate this problem as a combined 2D/3D registration. The 2D registration is formulated using optical flow constraints, while the 3D registration is using ICP as above. This yields a fitting energy of the form

$$E_{\text{fit}} = \|\mathbf{A}(\mathbf{b}_0 + \Delta\mathbf{B}\mathbf{x}) - \mathbf{c}\|_2^2, \tag{5.2}$$

where $(\mathbf{A}, \mathbf{c})$ summarize the registration constraints on a subset of the face vertices as indicated in blue on the left. For brevity we omit the specific formulas here, detailed derivations of the constraint terms can be found in [187]. Our optimization iteratively minimizes the following energy

$$\arg\min_{\mathbf{x}} E_{\text{fit}} + \lambda_1 E_{\text{smooth}} + \lambda_2 E_{\text{sparse}}. \tag{5.3}$$

Two additional terms, E_{smooth} and E_{sparse} with non-negative weights λ_1 and λ_2, are added for regularization. Temporal smoothness is enforced by penalizing the second-order difference

$$E_{\text{smooth}} = \|\mathbf{x}^{t-2} - 2\mathbf{x}^{t-1} + \mathbf{x}^t\|_2^2,$$

where t denotes the current timeframe. We also apply the 1-norm regularization

$$E_{\text{sparse}} = \|\mathbf{x}\|_1$$

on the blendshape coefficients. We found that this sparsity-inducing energy is very important to stabilize the tracking (see Figure 5.4). Because the blendshape basis are not linearly independent, the same expression could in principle be represented by dif-

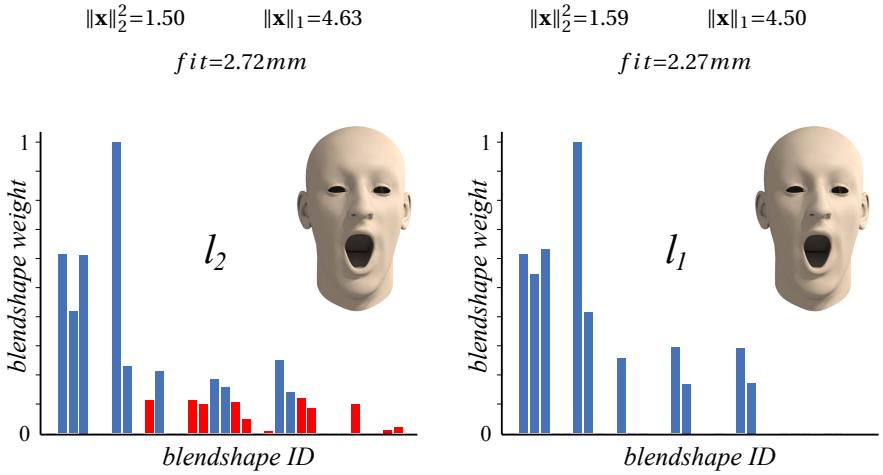


Figure 5.4.: Comparison between l_1 and l_2 regularization for the blendshape weight optimization of Equation 5.3. The l_1 regularization leads to a lower average fitting error (denoted by fit), but more importantly, significantly reduces the number of non-zero blendshape weights. The red bars on the left show the additionally activated blendshapes under l_2 norm regularization.

ferent blendshape combinations. Favoring a reconstruction with as few blendshapes as possible avoids potential blendshape compensation artifacts and better matches the blendshape weights a human animator would chose, which can be advantageous for retargeting. In addition, the l_1 regularization leads to a significant speed-up of the subsequent model refinement stage (Section 5.4.2), since blendshape refinement is only performed on blendshapes with non-zero blendshape weight (see also Figure 5.6).
The optimization is performed using a warm started shooting method [69]. The blendshape weights $\mathbf{x} = [x_1, \ldots, x_n]^T$ are bounded between 0 and 1 by projection over the constraint set at each iteration.

5.4.2. DEM Refinement

The refinement stage of the optimization adapts the blendshape model by solving for the PCA parameters $\mathbf{y}$ and deformation coefficients $\mathbf{z}_0, \ldots, \mathbf{z}_n$, keeping the rigid pose $(\mathbf{R}, \mathbf{t})$ and the blendshape weights $\mathbf{x}$ computed in the previous stage fixed.

We rewrite the fitting energy in Equation 5.2 as

$$E_{\text{fit}} = \|\mathbf{A}(\mathbf{b}_0 + \Delta\mathbf{B}\mathbf{x}) - \mathbf{c}\|_2^2 = \|\mathbf{A}[\bar{x}\mathbf{b}_0 + \sum_{i=1}^{n} x_i\mathbf{b}_i] - \mathbf{c}\|_2^2,$$

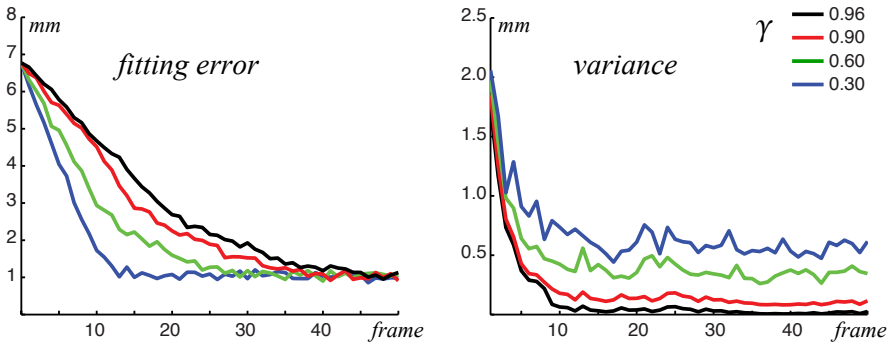


Figure 5.5.: Effect of the temporal decay factor γ in Equation 5.5. Lower values lead to faster reduction in fitting error, measured as the mean non-rigid ICP error for each frame, but incur more variance, measured as the mean per-vertex difference between consecutive frames.

where $\bar{x} = 1 - \sum_{i=1}^{n} x_i$. With $\mathbf{b}_0 = \mathbf{m} + \mathbf{P}\mathbf{y} + \mathbf{E}\mathbf{z}_0$ and $\mathbf{b}_i = \mathbf{T}_i^* \mathbf{b}_0 + \mathbf{E}\mathbf{z}_i$, this term can then be reformulated as $E_{\text{fit}} = \|\bar{\mathbf{A}}\mathbf{u} - \bar{\mathbf{c}}\|_2^2$, where

$$\bar{\mathbf{A}} = \mathbf{A}[(\bar{x}\mathbf{I} + \sum_{i=1}^{n} x_i \mathbf{T}_i^*)\mathbf{P}, (\bar{x}\mathbf{I} + \sum_{i=1}^{n} x_i \mathbf{T}_i^*)\mathbf{E}, x_1\mathbf{E}, \ldots, x_n\mathbf{E}],$$

$$\mathbf{u} = [\mathbf{y}^T, \mathbf{z}_0^T, \ldots, \mathbf{z}_n^T]^T, \text{ and } \bar{\mathbf{c}} = \mathbf{c} - \mathbf{A}(\bar{x}\mathbf{I} + \sum_{i=1}^{n} x_i \mathbf{T}_i^*)\mathbf{m}.$$

As previously, we regularize the PCA coefficients $\mathbf{y}$ and deformation coefficients $\mathbf{z}_i$, leading to the model refinement energy

$$E_{\text{ref}} = \|\bar{\mathbf{A}}\mathbf{u} - \bar{\mathbf{c}}\|_2^2 + \beta_1 \|\mathbf{D}_\mathbf{P}\mathbf{y}\|_2^2 + \sum_{i=0}^{n} (\beta_2 \|\mathbf{D}_\mathbf{E}\mathbf{z}_i\|_2^2 + \beta_3 \|\mathbf{z}_i\|_2^2). \tag{5.4}$$

Temporal Aggregation. The optimization of the DEM should not only depend on the current frame, but consider the entire history of observed expressions. However, directly optimizing over all frames would quickly become prohibitive in terms of memory and computation overhead. We therefore introduce an aggregation scheme that keeps the memory cost constant. The optimization is formulated as

$$\underset{\mathbf{y},\mathbf{z}_0,\ldots,\mathbf{z}_n}{\arg\min} \sum_{j=1}^{t} \frac{\gamma^{t-j}}{\sum_{j=1}^{t} \gamma^{t-j}} E_{\text{ref}}^j, \tag{5.5}$$

where t is the current frame and $0 \leq \gamma \leq 1$ defines an exponential decay over the frame history. E_{ref}^j denotes the refinement energy of Equation 5.4 at time j. The optimal

Algorithm 1: Blendshape Refinement at frame t

1 **Initialization:** $\mathbf{M}^1 = \mathbf{0}, \mathbf{y}^1 = \mathbf{0}, s^1 = 0$
2 $s^t = \gamma s^{t-1} + 1$
3 $\mathbf{M}^t = \gamma \frac{s^{t-1}}{s^t} \mathbf{M}^{t-1} + \frac{1}{s^t} (\bar{\mathbf{A}}^t)^T \bar{\mathbf{A}}^t$
4 $\mathbf{y}^t = \gamma \frac{s^{t-1}}{s^t} \mathbf{y}^{t-1} + \frac{1}{s^t} (\bar{\mathbf{A}}^t)^T \bar{\mathbf{c}}^t$
5 **Output:** $\mathbf{u}^t = GaussSeidel(\mathbf{M}^t + \mathbf{D}, \mathbf{y}^t, \mathbf{u}^{t-1})$

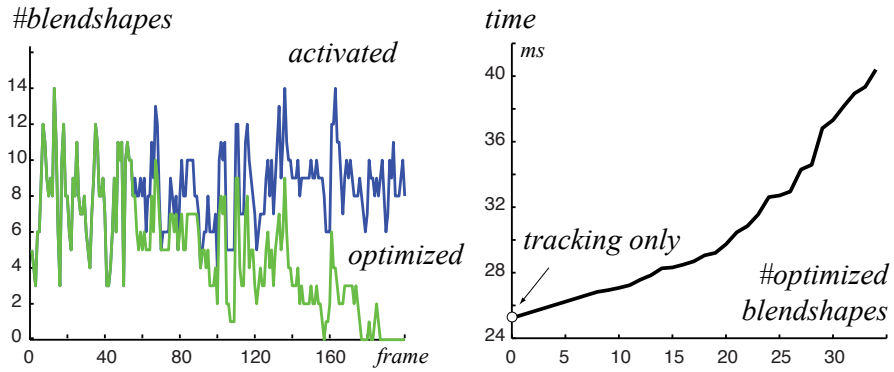


Figure 5.6.: Optimization performance. Left: The number of blendshapes optimized during DEM refinement gradually decreases as more blendshapes reach the coverage threshold. Right: total computation time per frame as a function of the number of blendshapes that are optimized in each frame.

solution of this minimization can be found by solving

$$(\mathbf{D} + \sum_{j=1}^{t} \frac{\gamma^{t-j}}{\sum_{j=1}^{t} \gamma^{t-j}} (\bar{\mathbf{A}}^j)^T \bar{\mathbf{A}}^j)\mathbf{u} = \sum_{j=1}^{t} \frac{\gamma^{t-j}}{\sum_{j=1}^{t} \gamma^{t-j}} (\bar{\mathbf{A}}^j)^T \bar{\mathbf{c}}^j, \tag{5.6}$$

where $\mathbf{D}$ is a diagonal matrix containing the regularization terms of Equation 5.4. To solve this system, we propose an online algorithm based on warm-started Gauss-Seidel optimization [9]. Our algorithm allows optimizing over the entire history of frames with a fixed memory overhead, as we do not need to store each frame separately (see Algorithm 1).

Figure 5.5 illustrates the tradeoff between fitting error and temporal variance as a function of the parameter γ. We found $\gamma = 0.9$ to provide a good balance and use this value for all our experiments.

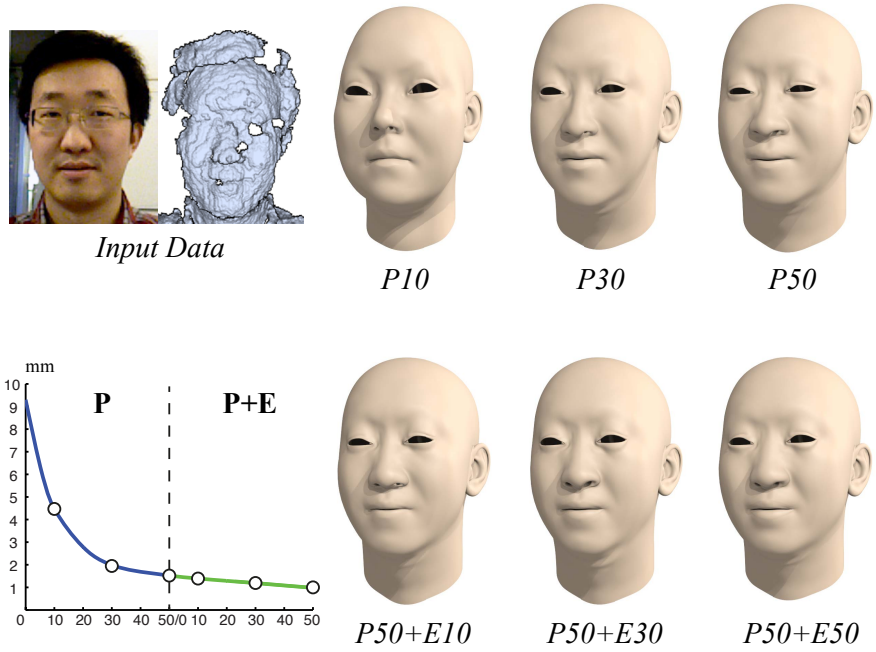


Figure 5.7.: Evaluation of the initial estimation of the neutral expression $\mathbf{b}_0$ when varying the number of PCA basis in **P** and the number of Laplacian eigenvector in **E**. The graph shows the mean non-rigid ICP error averaged over a sequence of 440 frames.

Blendshape coverage. In principle, DEM refinement could run indefinitely to continuously optimize the blendshape model as tracking progresses. However, we can improve computational performance with a simple heuristic. Blendshapes that have been optimized sufficiently many times can be considered "saturated" and are removed from the optimization. We define a coverage coefficient $\sigma_i = \sum_{j=1}^{t} x_i^j$ that measures how well each blendshape $\mathbf{b}_i$ has been observed until the current frame t. As soon as $\sigma_i > \bar{\sigma}$ for some fixed threshold $\bar{\sigma}$, the corresponding blendshape $\mathbf{b}_i$ is considered saturated and remains constant for the subsequent optimization. Since the neutral expression $\mathbf{b}_0$ plays a special role as the source for expression transfer, we always run the full optimization until $\sum_{j=1}^{t} \max(\bar{x}^j, 0) > \bar{\sigma}$. In practice, this does not affect performance significantly, since $\mathbf{b}_0$ is the blendshape that is typically most often observed. Figure 5.6 gives an indication of the computational overhead of DEM refinement. Since the computational cost gradually decreases as more blendshapes reach their coverage thresholds, DEM refinement quickly becomes negligible compared to the tracking stage of the optimization.

5.4.3. Implementation

In our current implementation, we employ a blendshape model of 34 blendshapes (see additional material for a complete list). The identity PCA model is computed from the dataset of [24] that consists of 100 male and 100 female head scans of young adults. We use 50 PCA basis vectors to approximate the neutral expression for all our examples. The corrective deformation fields are represented by 50 Laplacian eigenvectors for each coordinate (see also Section 5.5). We empirically determined the parameters $\beta_1 = 0.5$, $\beta_2 = 0.1$, and $\beta_3 = 0.001$ for Equations 5.1 and 5.4, $\lambda_1 = 10$ and $\lambda_2 = 20$ for Equation 5.3, and $\bar{\sigma} = 10$ for the coverage threshold, and use the same fixed settings for all our examples.

Our software is implemented in C++ and parallelized using OpenMP. We use the Eigen library for linear algebra computations and OpenCV for the face detector [182] and image processing operations. In order to speed-up the system we do not optimize for the unknowns of the blendshapes with 0 weight, keeping them fixed during the Gauss-Seidel optimization. Another speed improvement is achieved by building $(\bar{\mathbf{A}}^t)^T\bar{\mathbf{A}}^t$ per block as numerous blocks are similar up to a scalar factor, and blocks corresponding to the blendshapes with 0 weights are 0.

To complete the face tracking algorithm, we have implemented a separate image-based eye tracker. Since the rigid and the non-rigid alignment accurately determine the location of the eyes in the color image, we can apply a k-nearest neighbor search in a database of labeled eyes by cropping, rectifying and normalizing the input image. This k-nearest neighbor search is implemented using the OpenCV library. The final result is a weighted average of the labels of the k neighbors. The result of the eye tracker drives 14 supplementary blendshapes (see additional material) localized around the eyes. These blendshapes are computed using expression transfer only and are not part of the model refinement optimization.
Our system achieves sustained framerates of 25 Hz with a latency of 150 ms on a MacBook Pro with an Intel Core i7 2.7Ghz processor, 16 GBytes of main memory, and an NVIDIA GeForce GT 650M 1024MB graphics card.

5.5. Evaluation

In this section we present several experiments that we have performed to analyze our optimization algorithm, and discuss limitations of our approach.

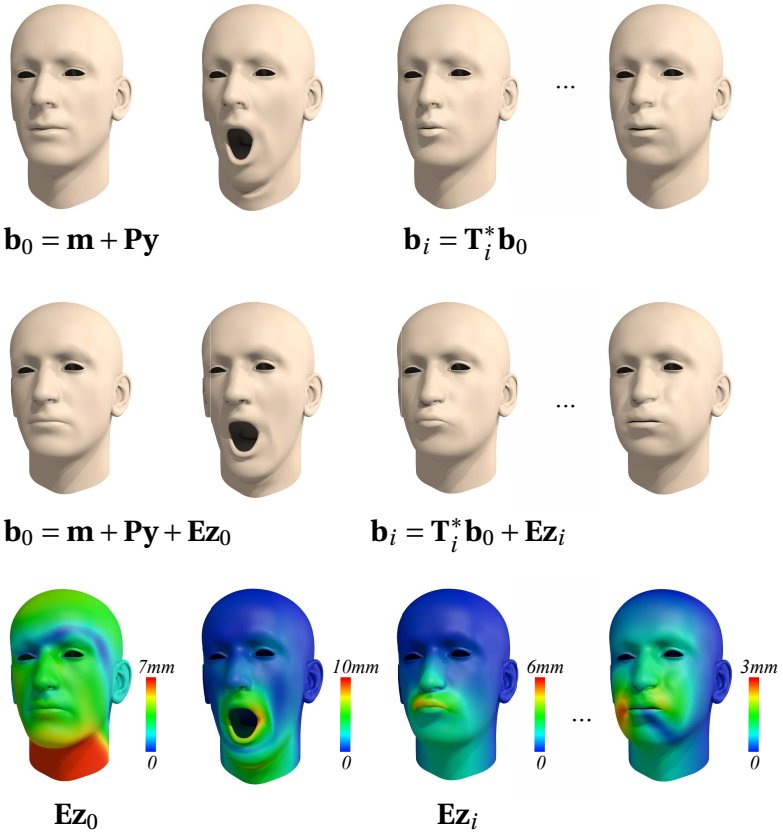


Figure 5.8.: Effect of corrective deformation fields. PCA and expression transfer only (top), additional deformation fields for both $\mathbf{b}_0$ and the $\mathbf{b}_i$ (middle), color-coded vertex displacements due to the deformation fields $\mathbf{E}\mathbf{z}_i$ (bottom).

Dynamic expression model. Figure 5.7 shows how the optimization of the neutral face depends on the number of basis vectors used for the identity PCA model and the corrective deformation fields, respectively. Due to the limited number of input samples (200 head models total), we observed no significant improvement beyond 50 basis vectors for the PCA model. For the deformation fields we found that 50 Laplacian eigenvectors are sufficient to obtain accurate reconstructions while still enabling real-time performance. The effect of the deformation fields is also shown for several blendshapes in Figure 5.8, where notable changes can be observed in the mouth region and around the nostrils. In general, we found these per-vertex deformations to be important to capture geometric detail, in particular the asymmetries common in many faces.

Tracking and DEM refinement. Figure 5.5 and Figure 5.9 show how the fitting error decreases over time as the DEM is refined concurrently to tracking. The corresponding adaptation of the blendshapes is illustrated in Figure 5.10. Figure 5.11 provides

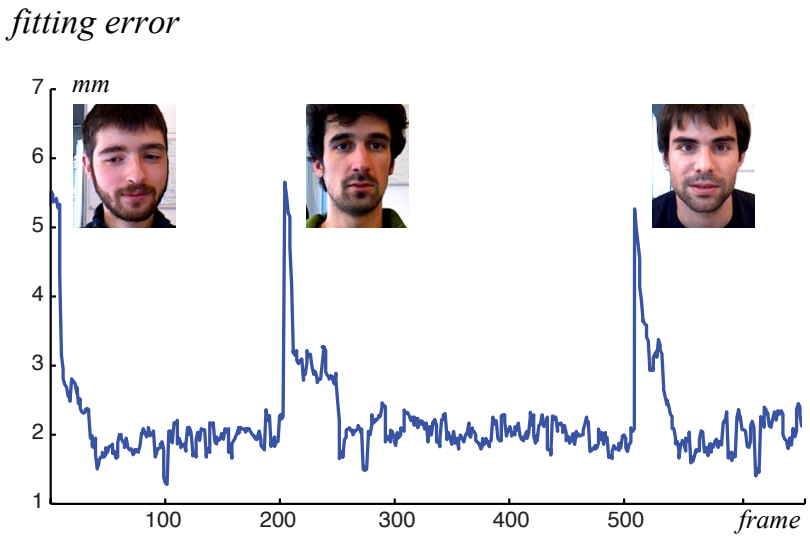


Figure 5.9.: Dynamic adaptation of the DEM model for three different users. The vertical spikes in fitting error indicate when a new user enters the field of view of the sensor. The DEM quickly adapts to the new facial geometry. High tracking accuracy is typically achieved within a second of using the system.

a comparison with a commercial software [66] that requires significant user-specific training and manual assistance to create the DEM. As the plot illustrates, our approach achieves comparable accuracy, while requiring no training or pre-processing.

Retargeting. The expression transfer operator (see Section 5.3 and Appendix), ensures that the user-specific DEM retains the blendshape semantics of the template model. The blendshape weights computed during tracking can therefore be used directly to drive a compatible face rig with the same blendshape configuration. This simple retargeting incurs no extra cost, which is particularly important for realtime applications. Figures 5.1 and 5.12 show that virtual avatars with significantly different facial features than the tracked user can be animated faithfully with our method.

Limitations. Our performance capture system is limited by the resolution and noise levels of the input device. While future hardware developments are likely to improve the performance of our system, tracking accuracy is inherently limited by the geometric detail of the template blendshape model. This representation is built on the premise that the location of facial features is spatially consistent across different users, an assumption that is no longer valid at small scales. For example, wrinkles typically appear at different locations for different people. As a consequence, such fine-scale features are

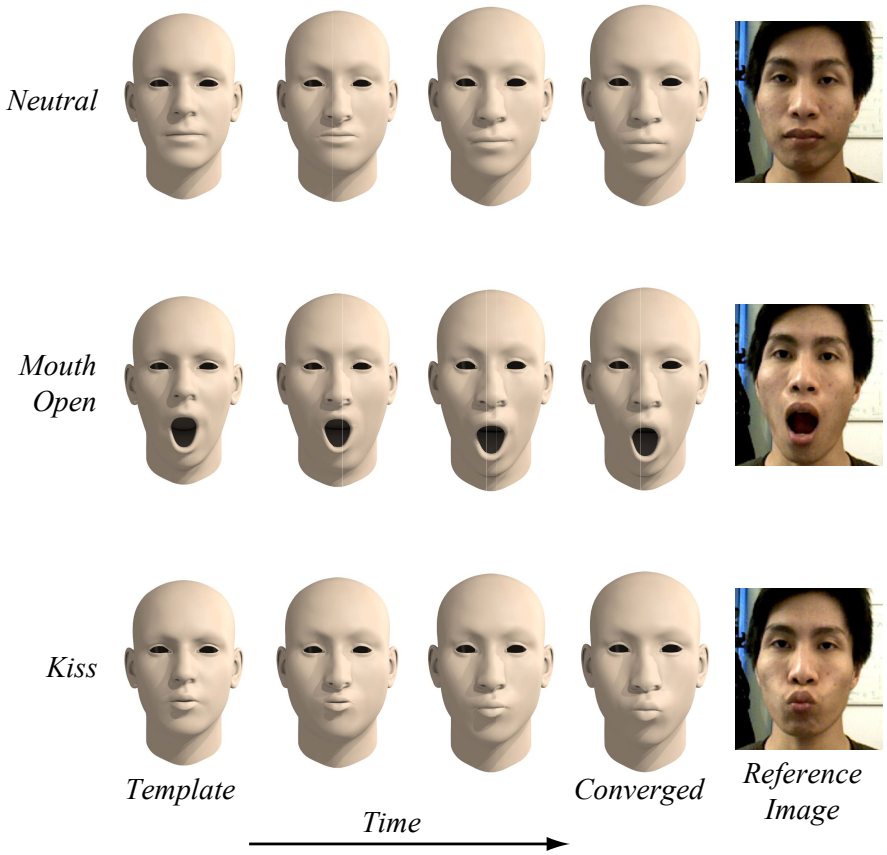


Figure 5.10.: Progressive DEM refinement. Each row shows the temporal evolution of a specific blendshape. The input image on the right is provided for reference. For this experiment we omit the PCA initialization to illustrate the robustness of the DEM refinement even when large deformations are required to match the face geometry of the tracked user.

not modeled adequately with our current approach.

In general, the same dynamic expression template might not be optimal for all tracked users. For example, children have significantly different facial dynamics than adults, and it might be more appropriate to apply different template models to different age groups. Since the identity PCA model of [24] that we currently use does not contain any children or older people, we did not yet investigate this hypothesis further.

The template blendshape model we currently use has been created in an iterative, empirical process. Starting with the most commonly used expressions, such as *mouth open*, *smile*, etc., we successively extended the model to include more blendshapes to obtain a more accurate expression space. This extension has been done with the advice of professional animators in order to match the established conventions for blendshape

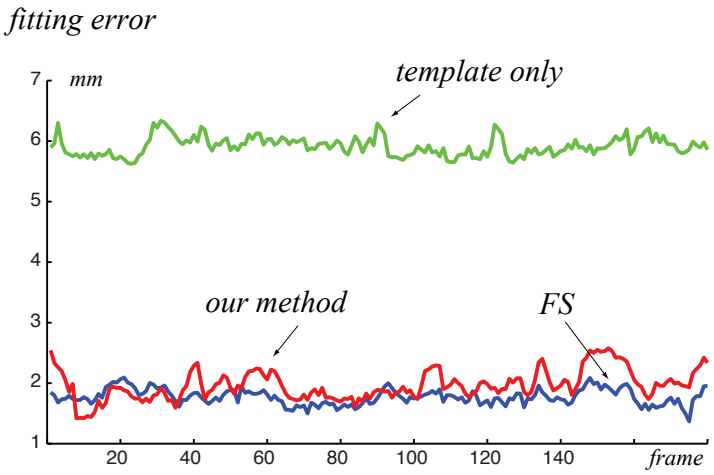


Figure 5.11.: Comparison of average fitting error for different tracking methods. DEM refinement significantly improves tracking accuracy compared to tracking with the template only. After convergence of the DEM, our method is comparable to the commercial software Faceshift Studio (FS) that depends on user-specific training. For this test, FS requires 11 static face scans of the user to create the expression model, as well as some manual work to assist the reconstruction, while our approach is completely automatic.

controllers. However, what constitutes the optimal blendshape model for tracking is not clear. A systematic study answering this question could be interesting future work.

5.6. Additions and Remarks

Future work will focus on further improving tracking accuracy. One interesting possibility would be to integrate speech analysis for better lip synching.Another promising avenue for future work is online avatar creation. The user-specific DEM that we build automatically already constitutes a fully rigged geometric avatar. Adding reconstruction of texture and other facial features such as hair would allow building complete digital avatars that can directly be integrated into online applications. Finally, we would like to investigate the application of similar online optimizations to other linear models such as Active Appearance Models or Active Shape Models. In [94] we show an extension of this work. We present a complete pipeline for creating fully rigged, personalized 3D facial avatars from hand-held video. Our system faithfully recovers facial expression dynamics of the user by adapting a blendshape template to an image sequence of recorded expressions using an optimization that integrates feature tracking, optical flow, and shape from shading. Fine-scale details such as wrinkles are captured separately in

Figure 5.12.: *Mimicry*, an application case study using our approach. An observer can simply step in front of the picture frame and the character depicted in the virtual painting will start mimicking the person's facial expression in realtime. The sensor is embedded in the frame.

normal maps and ambient occlusion maps. From this user- and expression-specific data, we learn a regressor for on-the-fly detail synthesis during animation to enhance the perceptual realism of the avatars. Our system demonstrates that the use of appropriate reconstruction priors yields compelling face rigs even with a minimalistic acquisition system and limited user assistance. This facilitates a range of new applications in computer animation and consumer-level online communication based on personalized avatars.

Chapter 6

Facial Animation Retargeting

Figure 6.1.: Our facial animation retargeting system learns a mapping from motion capture data to arbitrary character parameters.

6.1. Foreword

Creating realistic facial animations is a complex task that usually requires a significant time commitment of highly skilled animators. Recent developments in facial motion capture systems allow speeding up this process by accurately capturing the performance of an actor, thereby shifting the complexity of facial animation towards retargeting. However, mapping the captured performance onto a virtual avatar is a highly non-trivial task, especially when the target character is not a close digital replica of the actor, as for example in the movie *King-Kong*. Low-level automatic methods are bound to fail, since establishing the correspondence between facial expressions of largely different characters requires high-level semantic knowledge of their expressions spaces. A common strategy is thus to provide a set of explicit point correspondences between these two spaces. For example, for a given recorded smile of the actor, an animator would create a semantically matching smile of the virtual target character. Given a set of such labeled

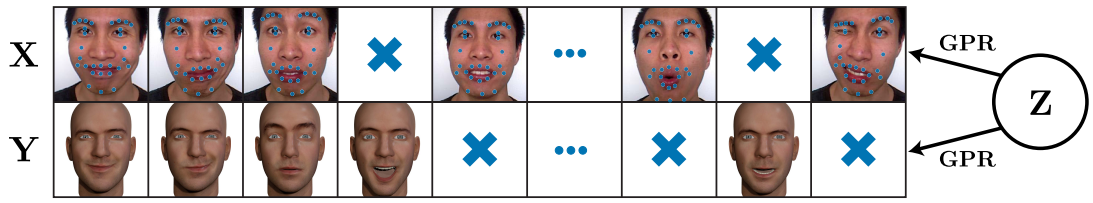


Figure 6.2.: Our algorithm learns a shared latent space **Z** from a space **X** of motion capture parameters and a space **Y** of character parameters. Gaussian Process Regressors (GPR) are used to model the mappings from the latent space onto the observation spaces. In order to train the GPRs only few pairwise correspondences between **X** and **Y** need to be specified. A key feature of our algorithm is that we also incorporate unlabeled data points for which no correspondence is given.

pairs, retargeting essentially becomes a problem of scattered data approximation, i.e., extrapolating the explicit correspondences into the entire expression space. The main difficulty in this type of example-based retargeting is creating the examples. Typically a large number of correspondences needs to be established to adequately capture the subtleties of facial expressions. In addition, posing a character to match a recorded expression can be very difficult, as subtle motions, e.g. a slight raise of the eyebrows, are often overlooked. These minor inaccuracies can quickly lead to noticeable disturbances in the animations of the target character.

Contribution. In this section, we present a novel example-based retargeting approach that significantly reduces the number of required training examples. Our method learns a shared latent space between motion capture and character parameters to represent their underlying common structure. Given a small set of manually specified correspondences between actor performance and target character expressions, the latent space is learned in a semi-supervised manner by using these labeled key poses, as well as the complete actor performance and previous animations of the target character. By adding this additional information we can increase the learning accuracy and stability, while the number of required training examples is reduced. We demonstrate that our system is resilient to noise and missing data, and can deal with high dimensional representations common in production-level facial rigs.

6.2. Learning

Classical example-based retargeting establishes a mapping from the source to the target space by computing an interpolation function from the point-wise correspondences defined by the labeled examples. Our method is based on one key observation: *unlabeled* frames can provide valuable information to establish this mapping. With unlabeled frames we mean poses in the captured sequence for which no corresponding expression for the target has been specified. For motion capture data, these unlabeled data points are abundant, since typically many hundreds of frames are recorded and only few are manually labelled. The main advantage of incorporating unlabeled data is that they provide important information about the local structure of the expressions space, which leads to better alignment of source and target spaces when computing the mapping. We can even go further and also incorporate unlabeled expressions of the target character, which help to constrain the mapping function by defining the space of semantically correct expressions of the target. Unlabeled target character samples are often available in the form of pre-existing animations that, for example, have been generated by an artist.

We employ shared GPLVM [63] to learn a mapping between motion capture and character parameters. The main hypothesis here is that both parameter spaces are (non-linearly) generated from a common low-dimensional manifold. Shared GPLVM (sGPLVM) learns a shared latent space by training Gaussian Process Regressors (GPR) to model the generative mappings from the latent space onto the observation spaces as illustrated in Figure 6.2. Gaussian Process Regressors can be trained robustly from small training sets and their parameters can be learned by maximizing the marginal likelihood of the training data. This is more efficient than techniques that use cross-validation to infer the parameter values when the training set is small, since the training dataset does not need to be reduced further [145].

6.2.1. Shared GPLVM Learning

Assume we are given two sets of corresponding observations $\mathbf{X} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_n\right]^T$ and $\mathbf{Y} = \left[\mathbf{y}_1, \ldots, \mathbf{y}_n\right]^T$, where $\mathbf{x}_i \in \mathbb{R}^{d_x}$ and $\mathbf{y}_i \in \mathbb{R}^{d_y}$. In our retargeting system $\mathbf{X}$ represents the space of source motion capture parameters and $\mathbf{Y}$ the space of target virtual character parameters. Let $\mathbf{Z} = \left[\mathbf{z}_i, \ldots, \mathbf{z}_n\right]^T$, $\mathbf{z}_i \in \mathbb{R}^{d_z}$ denote the corresponding (unknown) shared latent points. We model the generative mapping from the latent space onto the

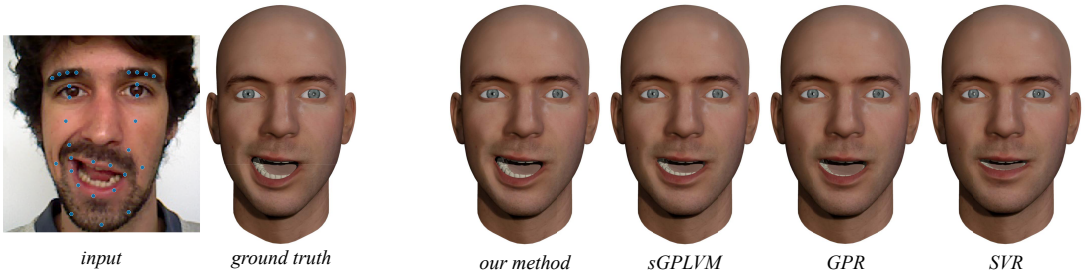


Figure 6.3.: Our method retargets accurately the facial expressions of the actor. With a small number of labels SVR has tendency to damp the facial expressions. In our examples, GPR gives results similar or slightly less accurate than sGPLVM, which we further improve in our method by incorporating unlabeled data.

observation spaces with Gaussian processes using the conditional probabilities

$$P(\mathbf{X}|\mathbf{Z}) = \frac{1}{\sqrt{2\pi^{nd_x}|\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}_X}|^{d_x}}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}_X}^{-1}\mathbf{X}\mathbf{X}^T\right)\right), \tag{6.1}$$

$$P(\mathbf{Y}|\mathbf{Z}) = \frac{1}{\sqrt{2\pi^{nd_y}|\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}_Y}|^{d_y}}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}_Y}^{-1}\mathbf{Y}\mathbf{Y}^T\right)\right). \tag{6.2}$$

The vector $\boldsymbol{\Phi} = \{\theta_1, \theta_2, \theta_3\}$ defines the parameters of the kernel $\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}}$ given as

$$\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}}^{i,j} = k_{\boldsymbol{\Phi}}(\mathbf{z}_i, \mathbf{z}_j) = \theta_1 \exp\left(-\frac{\theta_2}{2}||\mathbf{z}_i - \mathbf{z}_j||_2^2\right) + \theta_3^{-1}\delta_{i,j}, \tag{6.3}$$

where $\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}}^{i,j}$ is the element located at the i-th line and j-th column of the kernel matrix $\mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}}$ and $\delta_{i,j}$ is the Kronecker delta. Learning a shared GPLVM amounts to estimating the latent positions and kernel parameters by maximizing

$$\underset{\mathbf{Z},\boldsymbol{\Phi}_X,\boldsymbol{\Phi}_Y}{\arg\max}\, P(\mathbf{Z}|\mathbf{X},\mathbf{Y}) = \underset{\mathbf{Z},\boldsymbol{\Phi}_X,\boldsymbol{\Phi}_Y}{\arg\max}\, P(\mathbf{X}|\mathbf{Z})P(\mathbf{Y}|\mathbf{Z})P(\mathbf{Z}). \tag{6.4}$$

Semi-supervised learning. An important benefit of the shared GPLVM is that it can directly incorporate extra data points that do not need to be in correspondence. We can thus learn the shared GPLVM using $\mathbf{X} = \left[\mathbf{X}_l^T, \mathbf{X}_u^T, \circ\right]^T$ and $\mathbf{Y} = \left[\mathbf{Y}_l^T, \circ, \mathbf{Y}_u^T\right]^T$, where labeled pairs are denoted by $\mathbf{X}_l \in \mathbb{R}^{l\times d_x}$ and $\mathbf{Y}_l \in \mathbb{R}^{l\times d_y}$, and unlabeled samples are given by $\mathbf{X}_u \in \mathbb{R}^{m\times d_x}$, $\mathbf{Y}_u \in \mathbb{R}^{n\times d_y}$ with the $\circ$ indicating the missing correspondences (see Figure 6.2).

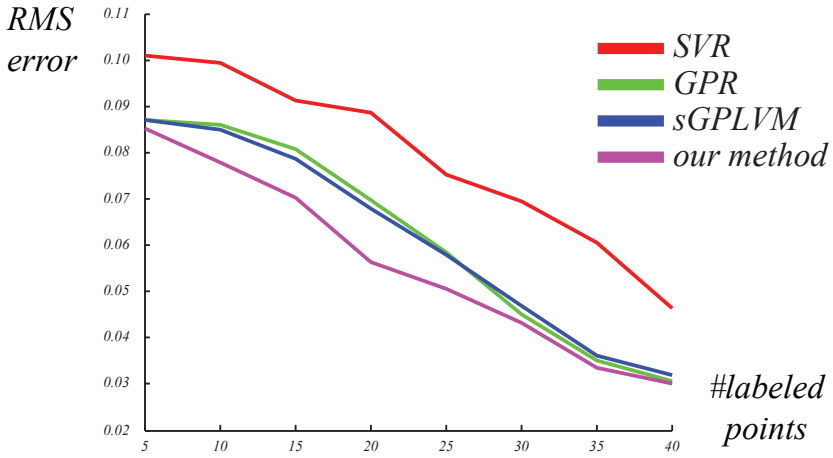


Figure 6.4.: A quantitative comparison of different learning approaches shows the root mean square (RMS) distance to the ground truth as a function of the number of training examples.

By using smooth mappings from the latent space to the observation spaces, sGPLVM ensures that close points in the latent space remain close in the observation spaces. However, the inverse is not necessarily true, i.e., points close in the observation spaces may be far apart in the latent space. In order to preserve the local topological structure of $\mathbf{X}$ and $\mathbf{Y}$ in the latent space, we therefore define a prior based on local linear embedding (LLE) [149] over the latent configurations. LLE assumes that each data point of the observation spaces and its neighbors are close to a locally linear patch on the manifold. The local geometry of these patches can then be encoded by linear coefficients w_{ij} that reconstruct each data point from its neighbors. By enforcing that the reconstruction of each latent point from its neighbors follows the same set of coefficients than their corresponding high dimensional point, the local structure of the observation spaces can be preserved in the latent space. We model this concept with a prior over the latent configuration using a Gaussian process

$$P(\mathbf{Z}) = \frac{1}{\sqrt{2\pi^{(l+m+n)d_z}|\mathbf{L}^{-1}|^{d_z}}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{L}\mathbf{Z}\mathbf{Z}^T\right)\right), \tag{6.5}$$

where $\mathbf{L} = \mathbf{M}^T\mathbf{M} + \mathbf{I}$ and $\mathbf{M}$ is a matrix in which each line encodes one reconstruction constraint and is defined as

$$\mathbf{M} = \begin{bmatrix} (\mathbf{I}-\mathbf{C}_X)^{:,1:l} & (\mathbf{I}-\mathbf{C}_X)^{:,(l+1):(l+m)} & 0 \\ (\mathbf{I}-\mathbf{C}_Y)^{:,1:l} & 0 & (\mathbf{I}-\mathbf{C}_Y)^{:,(l+1):(l+n)} \end{bmatrix}. \tag{6.6}$$

In the formulation above, $\mathbf{A}^{:,i:j}$ denotes a block a the matrix $\mathbf{A}$ going from column i to

column j and

$$\mathbf{C}_U^{i,j} = c_U(\mathbf{u}_i, \mathbf{u}_j) = \begin{cases} w_{ij} \text{ if } j \in N_i, \\ 0 \text{ otherwise.} \end{cases} \tag{6.7}$$

N_i are the indices of the k-nearest neighbor of $\mathbf{u}_i$ and the coefficients w_{ij} are defined as

$$w_{ij} = \arg\min_{w_{ij}} ||\mathbf{u}_i - \sum_{j \in N_i} w_{ij}\mathbf{u}_j||_2^2 \quad \text{s.t.} \sum_{j \in N_i} w_{ij} = 1. \tag{6.8}$$

Incorporating this prior of the local structure of the observation spaces helps to better constrain the position of the points with missing correspondences in the latent space. We also found that it helps increase the robustness of the training to bad initialization of the latent coordinates.

6.2.2. Computing the Mapping Function

The mapping from motion capture parameters to character parameters is done in two steps. We first solve for the latent position $\mathbf{z}_k^*$ given the motion capture observation $\tilde{\mathbf{x}}_k$. We call this part *source mapping*. Given the latent position $\mathbf{z}_k^*$, the subsequent *target mapping* part solves for the character parameters $\mathbf{y}_k^*$.

Source mapping. The source mapping not only solves for the latent position $\mathbf{z}_k^*$, but also for the most likely capture parameters $\mathbf{x}_k^*$ given the observation $\tilde{\mathbf{x}}_k$, the optimized motion capture parameters $\mathbf{x}_{k-1}^*$ of the previous frame, and the training data $\mathbf{X}$ and $\mathbf{Z}$. Thus we optimize

$$\arg\max_{\mathbf{x}_k^*, \mathbf{z}_k^*} P(\mathbf{x}_k^*, \mathbf{z}_k^* | \mathbf{x}_{k-1}^*, \tilde{\mathbf{x}}_k, \mathbf{X}, \mathbf{Z}). \tag{6.9}$$

We approximate the above probability density function by assuming that $\mathbf{z}_k^*$ is independent of $\mathbf{x}_{k-1}^*$, $\tilde{\mathbf{x}}_k$, $\mathbf{X}$, and $\mathbf{Z}$. This allows us to reformulate the optimization as

$$\arg\max_{\mathbf{x}_k^*, \mathbf{z}_k^*} P(\mathbf{x}_k^* | \mathbf{z}_k^*, \mathbf{x}_{k-1}^*, \tilde{\mathbf{x}}_k, \mathbf{X}, \mathbf{Z}) P(\mathbf{z}_k^*), \tag{6.10}$$

which can be extended to

$$\underset{\mathbf{x}_k^*,\mathbf{z}_k^*}{\arg\max} P(\mathbf{x}_k^*, \mathbf{x}_{k-1}^*, \tilde{\mathbf{x}}_k | \mathbf{z}_k^*, \mathbf{X}, \mathbf{Z}) P(\mathbf{z}_k^*). \tag{6.11}$$

By further assuming that $\tilde{\mathbf{x}}_k$ and $\mathbf{x}_{k-1}^*$ are independent of $\mathbf{z}_k^*$, $\mathbf{X}$, and $\mathbf{Z}$ given $\mathbf{x}_k^*$, and $\tilde{\mathbf{x}}_k$ is independent of $\mathbf{x}_{k-1}^*$ given $\mathbf{x}_k^*$, we obtain our final optimization objective

$$\underset{\mathbf{x}_k^*,\mathbf{z}_k^*}{\arg\max} P(\mathbf{x}_k^* | \mathbf{z}_k^*, \mathbf{X}, \mathbf{Z}) P(\tilde{\mathbf{x}}_k | \mathbf{x}_k^*) P(\mathbf{x}_{k-1}^* | \mathbf{x}_k^*) P(\mathbf{z}_k^*). \tag{6.12}$$

The likelihoods $P(\tilde{\mathbf{x}}_k|\mathbf{x}_k^*)$ and $P(\mathbf{x}_{k-1}^*|\mathbf{x}_k^*)$ represent closeness to the observation and temporal smoothness, respectively, and are modeled by two Gaussian distributions as

$$P(\tilde{\mathbf{x}}_k | \mathbf{x}_k^*) = \mathcal{N}(\tilde{\mathbf{x}}_k | \mathbf{x}_k^*, \sigma_c^2 \mathbf{I}), \tag{6.13}$$

$$P(\mathbf{x}_{k-1}^* | \mathbf{x}_k^*) = \mathcal{N}(\mathbf{x}_{k-1}^* | \mathbf{x}_k^*, \sigma_t^2 \mathbf{I}). \tag{6.14}$$

The two probabilities $P(\mathbf{x}_k^*|\mathbf{z}_k^*, \mathbf{X}, \mathbf{Z})$ and $P(\mathbf{z}_k^*)$ act as priors over motion capture parameters and latent position and are defined as

$$P(\mathbf{x}_k^* | \mathbf{z}_k^*, \mathbf{X}, \mathbf{Z}) = \mathcal{N}(\mathbf{x}_k^* | \boldsymbol{\mu}, \sigma_p^2 \mathbf{I}), \tag{6.15}$$

$$\boldsymbol{\mu} = \mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}_X}^{-1} \mathbf{X} \mathbf{k}_{\boldsymbol{\Phi}_X}(\mathbf{z}_k^*), \tag{6.16}$$

$$\sigma_p^2 = k_{\boldsymbol{\Phi}_X}(\mathbf{z}_k^*, \mathbf{z}_k^*) - \mathbf{k}_{\boldsymbol{\Phi}_X}(\mathbf{z}_k^*)^T \mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}_X}^{-1} \mathbf{k}_{\boldsymbol{\Phi}_X}(\mathbf{z}_k^*), \tag{6.17}$$

where $\mathbf{k}_{\boldsymbol{\Phi}}(\mathbf{z}_k^*)$ is a vector whose i-th element is $k_{\boldsymbol{\Phi}}(\mathbf{z}_k^*, \mathbf{z}_i)$ and $P(\mathbf{z}_k^*) = \mathcal{N}(\mathbf{z}_k^*|0, \mathbf{I})$. One advantage of this formulation is that missing dimensions of $\tilde{\mathbf{x}}_k$ can be retrieved during the optimization by setting $\sigma_c^2 = \infty$ in Equation 6.13 for these dimensions.

Target mapping. The second step of the mapping process is to find the character parameters $\mathbf{y}^*$ given the latent position $\mathbf{z}^*$ by maximizing

$$\underset{\mathbf{y}_k^*}{\arg\max} P(\mathbf{y}_k^* | \mathbf{z}_k^*, \mathbf{Y}, \mathbf{Z}) = \mathbf{K}_{\mathbf{Z},\boldsymbol{\Phi}_Y}^{-1} \mathbf{Y} \mathbf{k}_{\boldsymbol{\Phi}_Y}(\mathbf{z}_k^*). \tag{6.18}$$

Implementation. In our implementation, we first mean center the observation spaces and rescale them by dividing by their maximum variance. For the learning phase, we empirically found $\boldsymbol{\Phi} = \{1, 1, 100\}$ to be good initial kernel parameters for the optimization for all our examples. We fix σ_c^2 and σ_t^2 by estimating the noise level of the motion

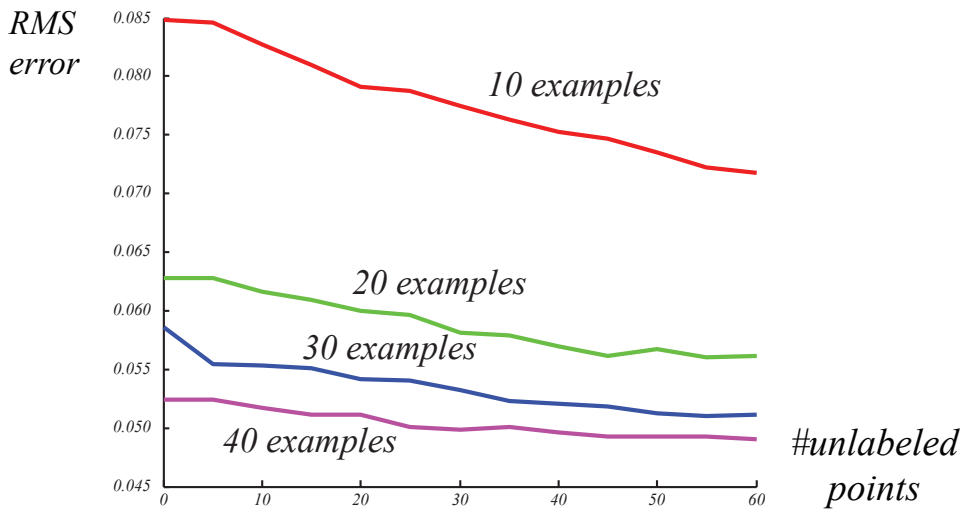


Figure 6.5.: Unlabeled data points help to increase retargeting accuracy, in particular when working with few training examples.

capture system [187] and chose $k = 8$ nearest neighbors for LLE and 8 dimensions for the latent space. The latent coordinates are initialized using the semi-supervised manifold alignment technique presented in [86]. For the mapping phase, we initialize $\mathbf{x}_k^*$ with the motion capture observation $\tilde{\mathbf{x}}_k$ and $\mathbf{z}_k^*$ with the latent position corresponding to the closest $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_k$. We use scaled conjugate gradient [126] as optimizer and minimize the negative logarithm of the probabilities.

6.3. Evaluation

For our evaluation experiments, we use the *faceshift* tracking system (www.faceshift.com). Given a recorded sequence of a human actor, this system produces an animated 3D mesh represented in a blendshape basis that matches the actor's performance. We select a set of vertices on the mesh as marker positions to generate motion capture input and perform a retargeting of these marker points onto the blendshape basis of the animated target character. This setup allows measuring and comparing the performance of our algorithm since the blendshape parameters provided by the tracking system can be treated as ground truth for the evaluation. Note that all other retargeting sequences use target characters (the models shown in Figure 6.1, see also video) for which no such ground truth data is available.

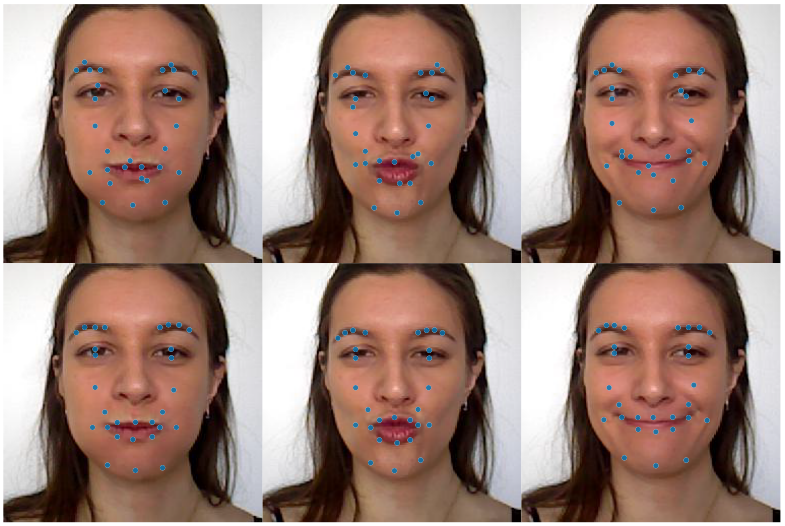


Figure 6.6.: Resilience to noise. Our learning approach is able to compute accurate marker positions (bottom row) by automatically correcting the noisy input points (top row).

Comparison. We compare our algorithm with Support Vector Regression (SVR) [62], Gaussian Process Regression (GPR) [190] and the supervised shared GPLVM (sGPLVM) [63]. We recorded sequences of approximatively 2000 frames of different actors. The different algorithms are applied 20 times over those sequences by random selection of labelled and unlabeled points, using 100 unlabeled data points for both observation spaces. The averaged results shown in Figures 6.3 and 6.4 demonstrate that our algorithm improves the retargeting accuracy by up to 20%, especially when the number of labeled expression correspondences is small. Our algorithm also preserves motion dynamics significantly better than the other approaches.

Unlabeled points. Figure 6.5 illustrates the effect of using unlabeled points for establishing the retargeting mapping function. As the curves indicate, when using about 50 unlabeled points we can achieve the same retargeting accuracy with 20 training examples as with 30 examples and no additional unlabeled points. Compared to the time-consuming and error-prone labeling, the latter come essentially for free, allowing for significant savings in manual labor. Unlabeled points are particularly useful for small sets of manually specified examples as the given correspondences do not span the full animation space.

Noise and missing data. One advantage of our formulation is its robustness to noise (Figure 6.6) and missing data (Figure 6.7). Our system models a probability

Figure 6.7.: Missing markers can be handled by our retargeting system. The optimization jointly retrieves the location of the missing markers (green) and the target character parameters.

distribution function over motion capture parameters and latent positions allowing to retrieve the most probable set of markers given the possible noisy or incomplete input observation.

Character posing. The resilience of our algorithm to missing data is not limited to the input space. We can exploit the regularization of our probabilistic framework to also complete missing data in the target space, which offers a simple but effective approach to character posing. The animator can specify only a subset of the target animation parameters and our algorithm will automatically infer the most probable pose matching the specified values (see Figure 6.8). This type of guided character posing is particularly advantageous for complex animation models, where many parameters only induce subtle pose variations that are thus difficult to specify, but nevertheless important for the expression.

Discussion and Limitations. When the number of examples is small, example-based retargeting methods have a tendency to infer a wrong correlation between parts of the face as for example mouth open and eyebrows up. This effect is reduced in our approach by taking into account unlabeled data. One additional solution is to split the face (e.g. upper part and lower part) and to learn the retargeting independently for those parts, similar to recent linear 3D face models [167].

In our work, we use a set of key poses, rather than sequences, to learn the retargeting function. Learning a latent dynamical system as in [186] with different motion style is challenging especially with a small set of sequences. Nevertheless, motion sequences can additionally be used in our approach by taking into account temporal closeness when building the matrix in Equation 6.6.

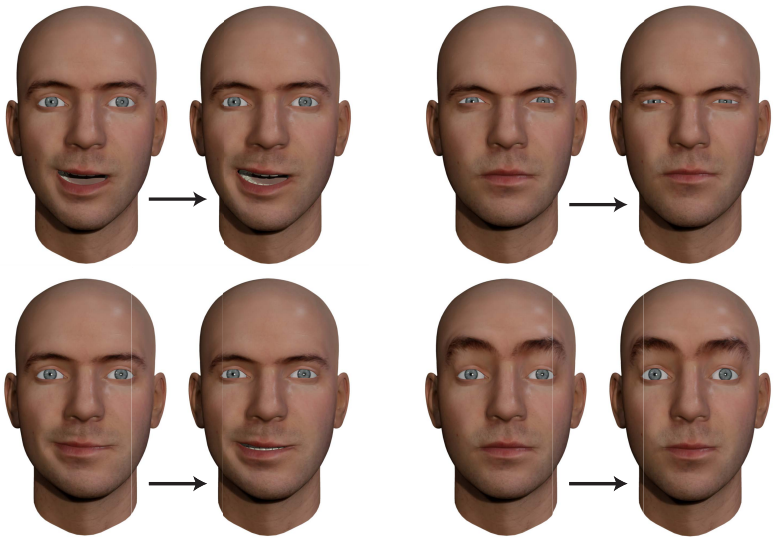


Figure 6.8.: Character posing can be simplified by optimizing for the missing animation parameters. In these examples, the animator only needs to specify 2-3 animation parameters (left) and the system automatically infers the most likely pose matching this input (right), activating about 20 additional blendshape parameters.

A drawback of the Gaussian Process Regressor model is its time complexity, which is $O(N^3)$ for the training phase and $O(N^2)$ for evaluating the mapping, where N is the number of points in the training data. Sparse approximations [103, 105] allow to reduce the training complexity to a more manageable $O(k^2N)$ where k is the number of active points retained in the sparse representation. In practice, our current implementation supports realtime retargeting for a training set of a few hundred data points for each observation space. The training time of our system for 40 examples and 100 unlabeled points is around 1-2 minutes and the mapping between 30 to 40ms.

In our current implementation the dimension of the latent space is chosen empirically. Recent works in non-linear dimensionality reduction [74, 151] introduced a rank prior that allows to automatically determine the dimension of the latent space. This work should also be applicable for our approach.

6.4. Additions and Remarks

We believe that the main features of our approach will be applicable in other retargeting applications and see several avenues for future research. A promising idea is to further explore manifold alignment algorithms [194, 184, 197] to define a prior over latent con-

figurations and for the initialization of the shared GPLVM. Our statistical framework is also well suited for active learning. We expect further improvements in retargeting accuracy when automatically suggesting new poses for labeling based on an online analysis of the uncertainty of the current retargeting mapping function.

Active learning. A side-benefit of our approach is that it supports active learning. For each set of motion capture parameters $\mathbf{x}_i$ in $\mathbf{X}$ we can compute a corresponding latent position $\mathbf{z}_i$ as explained in the Section 6.2.2. We can then compute the Gaussian probability distribution $P(\mathbf{y}_i^* | \mathbf{z}_i^*, \mathbf{Y}, \mathbf{Z})$ where the variance provides a quantitative estimate of the uncertainty of the mapping from the latent position to the character parameters. We can thus apply a greedy strategy to iteratively propose to the animator the next pose for labeling. We simply select the captured expression $\mathbf{x}_i$ with most uncertain mapping and request the animator to create the corresponding target pose. We then update the mapping taking this new example into account to reduce the overall uncertainty.

Chapter 7

Physics-Based Animation

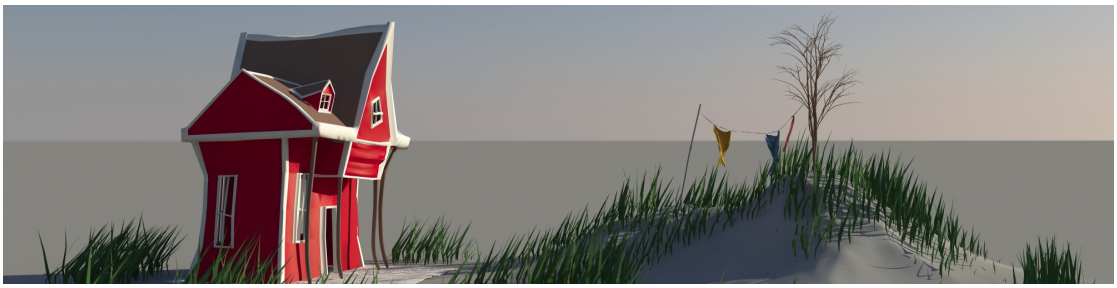


Figure 7.1.: We propose a new "projection-based" implicit Euler integrator that supports a large variety of geometric constraints in a single physical simulation framework. In this example, all the elements including building, grass, tree, and clothes (49k DoFs, 43k constraints), are simulated at 3.1ms/iteration using 10 iterations per frame.

7.1. Foreword

Physics-based simulation of deformable material has become an indispensable tool in many areas of computer graphics. Virtual worlds, and more recently character animations, incorporate sophisticated simulations to greatly enhance visual experience, e.g., by simulating muscles, fat, hair, clothing, or vegetation. These models are often based on finite element discretizations of continuum-mechanics formulations, allowing highly *accurate* simulation of complex non-linear materials.

Besides realism and accuracy, a number of other criteria are also important in computer graphics applications. By *generality* we mean the ability to simulate a large spectrum of

behaviors, such as different types of geometries (solids, shells, rods), different material properties, or even art-directable extensions to classic physics-based simulation. *Robustness* refers to the capability to adequately handle difficult configurations, including large deformations, degenerate geometries, and large time steps. Robustness is especially important in real-time applications where there is no "second chance" to re-run a simulation, such as in computer games or medical training simulators. The *simplicity* of a solver is often important for its practical relevance. Building on simple, easily understandable concepts – and the resulting lightweight codebases – eases the maintenance of simulators and makes them adaptable to specific application needs. *Performance* is a critical enabling criterion for realtime applications. However, performance is no less important in offline simulations, where the turnaround time for testing new scenes and simulation parameters should be minimized.

Current continuum mechanics approaches often have unfavorable trade-offs between these criteria for certain computer graphics applications, which led to the development of alternative methods, such as Position Based Dynamics (PBD).

Due to its generality, simplicity, robustness, and efficiency, PBD is now implemented in a wide range of high-end products including PhysX, Havok Cloth, Maya nCloth, and Bullet. While predominantly used in realtime applications, PBD is also often used in offline simulation. However, the desirable qualities of PBD come at the cost of limited accuracy, because PBD is not rigorously derived from continuum mechanical principles.

We propose a new implicit integration solver that bridges the gap between continuum mechanics and PBD. The key idea is to introduce energy potentials with a specific structure. More precisely, our potentials consist of a convex quadratic distance measure from a *constraint*. The constraints are general nonlinear functions that express the desired state of an element, for example, that the volume of a tetrahedron must remain within given bounds. The distance measure quantifies how much individual constraints are violated in a given deformed configuration. While our solver can handle arbitrary geometric constraints, we propose a specific set of constraints derived from continuous deformation energies. These continuum-based constraints are very practical because they considerably simplify parameter tuning especially when dealing with meshes of different resolutions and non-uniform tessellation.

The main advantage of our constraint-based potentials is that their structure enables an efficient local/global optimization (block coordinate descent). Specifically, the local step consists of projecting every element onto the constraint manifold, i.e., solving a small nonlinear problem per element. The global step combines the results of individuals projections, finding a compromise between all of the individual constraints, while

also taking into account global effects such as inertia and external forces.

The local/global approach allows us to formulate an implicit integration solver that is guaranteed to weakly decrease the energy in every iteration without requiring any specific precautions. This contrasts with classical Newton's method which requires line search strategies and safeguards against singular or indefinite Hessians to guarantee robustness.

Furthermore, with a fixed set of constraints, we can pre-factor the linear system of the global step, which greatly reduces computation time.

The local steps consists of small independent optimization problems, which can be all executed in parallel.

To our knowledge, our method is the first to apply local/global optimization to simulate general dynamical systems. We demonstrate that this solution provides a robust and efficient approach to implicit integration, often significantly outperforming the classical Newton method. The connection between PBD and our solver reveals new insights on how PBD relates to traditional approaches based on finite element methods and Newtonian mechanics.

Contributions. Since the pioneering work of Terzopulous and colleagues [169], models derived from continuum mechanics play an important role in physics-based animation. The basic principle is that the resistance of an elastic object to deformations is quantified using an elastic potential energy – a scalar function whose variational derivative leads to the elastic force [156]. Unfortunately, the elastic forces are usually non-linear even for basic material models, which complicates time integration of the resulting equations of motion.

The simplest time integration schemes used in computer graphics are explicit and very fragile to large time steps [143]. Implicit Euler methods significantly improve robustness [8], but at the cost of solving a system of non-linear equations at every step. As shown in [122], this can be equivalently formulated as a non-convex optimization problem that operates directly on elastic potentials instead of forces. One of the main shortcomings of implicit Euler integration is artificial numerical damping. This motivated the development of symplectic integrators [85, 99] and mixed implicit-explicit methods (IMEX) [38, 162], featuring better energy conservation properties. Another approach is *energy budgeting* [163] which enforces energy conservation explicitly. However, implicit Euler integration continues to be one of the popular choices in applications of physics-based animation where robustness is an important criterion and numerical

damping is not a major concern. Our solver is derived from the variational form of implicit Euler integration [122] as it gives an intuitive way of thinking about time integration in our framework – simply by adding another constraint to the system. This further allows us to draw connections between PBD and the implicit Euler integration scheme and results in a robust and efficient approach that is stable under large time steps.

Regardless of the particular flavor and formulation of implicit integration, Newton's method remains the computational workhorse for solving the system of non-linear equations. However, its robust implementation requires precautions such as conservative line search procedures and safeguards against indefinite Hessians [36]. From a performance standpoint, a serious drawback of Newton's method is the fact that the Hessian matrix and the gradient change at every iteration. Quasi-Newton methods therefore employ approximate Hessians, trading faster linear system solves for suboptimal descent directions (and therefore slower convergence) as demonstrated by [60, 83]. A similar strategy, explored in the context of co-rotated elasticity, is to use carefully scheduled updates of sparse Cholesky factorization [88]. Recently, Liu and colleagues [115] presented a method for efficient implicit time integration of mass-spring systems by introducing auxiliary variables that enable alternating local/global optimization. This approach, also known as block coordinate descent, has been previously used with great success in geometry processing [160, 29]. We also employ local/global alternation in our approach, but contrary to [115], which is limited to mass-spring systems and assumes *only* linear springs (Hooke's law), we show how to generalize this concept employing projection onto constraint sets to simulate *general* nodal dynamical systems.

Our constraint-based formulation bears some similarity with recent non-traditional approaches based on constraint projection. The idea of constraint projection is central to the Nucleus system [161] and Position Based Dynamics [128, 15]. In contrast to our solution, these methods do not treat the constraints in a global manner, but iteratively project onto them in a (non-linear) Gauss-Seidel-like fashion [128]. While the resulting algorithm is very easy to implement, this approach has a number of shortcomings: the Gauss-Seidel optimization does not converge very rapidly, the material stiffness depends on the number of iterations, and the result depends on the traversal order. In contrast, our method uses constraints to formulate elastic potentials that are rigorously combined with inertial terms as dictated by Newton's laws of motion. Our solver first computes all constraint projections separately and then finds the best compromise between them, which makes the solution independent of the order of constraints. To obtain faster convergence, constraints are expressed using differential coordinates, which often yields satisfactory results after just a few iterations. Furthermore, our solver converges to a true implicit Euler solution with our elastic energy, in contrast to Position Based

Dynamics which converges to completely inelastic behavior.

Another closely related concept is shape matching [129, 147] where, in contrast to our method, constraint projections are used to directly build elastic forces instead of potentials to simulate deformable objects. Constraint projections were also used in *strain limiting* [144, 76, 172, 185, 132] not as a standalone simulation technique but rather as a way to improve handling of stiff systems with standard time integration methods. In our approach we can also perform strain limiting but it is directly included in the implicit solver.

7.2. Continuum Mechanics View

In this section we introduce the special structure of our potentials that form the basis of our method. We start with the implicit time integration of FEM-discretized elastic models.

7.2.1. Implicit Euler Solver

Let us briefly review the variational form of implicit Euler integration [122]. We assume a mesh consisting of m vertices with positions $\mathbf{q} \in \mathbb{R}^{m\times 3}$ and velocities $\mathbf{v} \in \mathbb{R}^{m\times 3}$. The system evolves in time according to Newton's laws of motion through a discrete set of time samples $t_1, t_2, \ldots$. At time t_n, the system is defined as $\{\mathbf{q}_n, \mathbf{v}_n\}$. The sum of the external forces is defined as $\mathbf{f}_{\text{ext}}$ and the sum of internal forces as $\mathbf{f}_{\text{int}}$. We consider position dependent internal forces such that $\mathbf{f}_{\text{int}}(\mathbf{q}) = -\sum_i \nabla W_i(\mathbf{q})$, where $W_i(\mathbf{q})$ is a scalar potential energy function. Implicit Euler time integration results in the following update rule:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\mathbf{v}_{n+1} \tag{7.1}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\mathbf{M}^{-1}(\mathbf{f}_{\text{int}}(\mathbf{q}_{n+1}) + \mathbf{f}_{\text{ext}}) \tag{7.2}$$

where $\mathbf{M}$ is the mass-matrix and h represents the simulation step size. Note that $\mathbf{f}_{\text{ext}}$ and $\mathbf{M}$ are held constant for any given time step. Using these equations we can derive

$$\mathbf{M}(\mathbf{q}_{n+1} - \mathbf{q}_n - h\mathbf{v}_n) = h^2(\mathbf{f}_{\text{int}}(\mathbf{q}_{n+1}) + \mathbf{f}_{\text{ext}}). \tag{7.3}$$

This system can be converted to an optimization problem

$$\min_{\mathbf{q}_{n+1}} \frac{1}{2h^2} \|(\mathbf{q}_{n+1} - \mathbf{s}_n)\|^2_{\mathbf{M}} + \sum_i W_i(\mathbf{q}_{n+1}), \tag{7.4}$$

where $\mathbf{s}_n = \mathbf{q}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$ and $\|.\|_F$ denotes the Frobenius norm. Intuitively, this minimization problem describes the compromise between the *momentum potential*

$$\frac{1}{2h^2} \|\mathbf{q}_{n+1} - \mathbf{s}_n\|^2_{\mathbf{M}}, \tag{7.5}$$

which states that the solution should follow its momentum (plus external forces), and the elastic potential, that requires the solution to minimize the elastic deformation. The corresponding weighting terms, i.e., the mass distribution in $\mathbf{M}$, the time step h and the material stiffness of W, determine which potential has more importance in this balance. Furthermore, according to Noether's theorem, linear and angular momenta are always conserved when the elastic potential is rigid motion invariant.

The minimization of Equation 7.4 is commonly performed using careful implementations of Newton's method [122]. However, this is quite costly because at each iteration a different linear system needs to be solved, as the Hessian changes from one iteration to the next. To simplify notation, we will drop below the subscript in $\mathbf{q}_{n+1}$ and just use $\mathbf{q}$.

7.2.2. Nonlinear Elasticity

We analyze the classical form of FEM-based nonlinear elastic energies to reveal how we can restrict the elastic potentials in Equation 7.4 to a structure that will allow deriving our novel solver.

Nonlinear elastic potentials. In nonlinear continuum mechanics the deformation from a rest state is measured using a discrete, elemental strain $\mathbf{E}(\mathbf{q})$, e.g., the quadratic Green's strain [96]. Numerous elastic potentials used in practice are formulated as a function of the strain using a (often nonlinear) material model $\Psi(\cdot)$, resulting in elastic potentials $W(\mathbf{q}) = \Psi(\mathbf{E}(\mathbf{q}))$. From a geometric point of view, we can observe that $\mathbf{E}(\mathbf{q}) = \mathbf{0}$ defines a constraint manifold of all possible undeformed configurations, while $\Psi(\mathbf{E}(\mathbf{q}))$ measures how far the deformed configuration is from this manifold (level sets in Figure 7.2). Our key observation is that these two concepts can be decoupled; the distance metric does not have to be a complicated nonlinear function because the non-

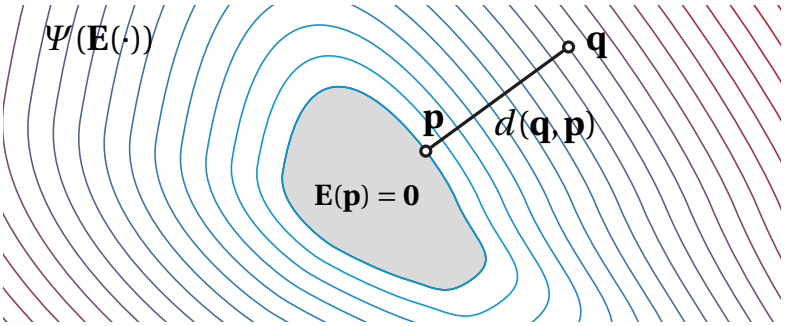


Figure 7.2.: The function $\Psi(\mathbf{E}(\cdot))$ defines both the constraint manifold $\mathbf{E}(\cdot) = \mathbf{0}$ as its zero level set and the elastic potential given by its isolines. By introducing a projection variable $\mathbf{p}$ in the manifold, we can decouple the manifold definition from the elastic potential, modeled as the distance function $d(\mathbf{q},\mathbf{p})$.

linearities are already captured by the constraint manifold.

Decoupling distance measure and constraint manifold. We introduce potential functions W that make use of an auxiliary variable $\mathbf{p}$ as

$$W(\mathbf{q},\mathbf{p}) = d(\mathbf{q},\mathbf{p}) + \delta_{\mathbf{E}}(\mathbf{p}). \tag{7.6}$$

Here, $\delta_{\mathbf{E}}(\mathbf{p})$ is an indicator function that evaluates to zero if $\mathbf{E}(\mathbf{p}) = \mathbf{0}$ and to $+\infty$ otherwise, and formalizes the requirement that $\mathbf{p}$ should lie on the constraint manifold. The function $d(\mathbf{q},\mathbf{p})$ then measures a distance between $\mathbf{q}$ and $\mathbf{p}$. Minimizing Equation 7.6 over $\mathbf{p}$ corresponds to a projection of $\mathbf{q}$ onto the constraint manifold, as illustrated in Figure 7.2. An elastic potential analogous to $\Psi(\mathbf{E}(\mathbf{q}))$ can therefore be defined as $\tilde{W}(\mathbf{q}) = \min_{\mathbf{p}} W(\mathbf{q},\mathbf{p})$.

Quadratic distance measures. With this separation in mind, we can build a solver that alternates between distance minimization and projection. An important advantage of this formulation is that the distance measure can be freely chosen. The *constraint nonlinearity* (also known as geometric nonlinearity) is already taken care of by the projection on the constraint set, so the distance metric can be kept simple, trading general *material nonlinearity* against efficiency and robustness. Specifically, we consider distance metrics leading to the following potentials:

$$W(\mathbf{q},\mathbf{p}) = \frac{w}{2}\|\mathbf{A}\mathbf{q} - \mathbf{B}\mathbf{p}\|_F^2 + \delta_{\mathbf{C}}(\mathbf{p}), \tag{7.7}$$

Algorithm 2: Projective Implicit Euler Solver

$\mathbf{s}_n = \mathbf{q}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$
$\mathbf{q}_{n+1} = \mathbf{s}_n$
loop *solverIteration* **times**
  forall the *constraints i* **do**
    $\mathbf{p}_i = ProjectOnConstraintSet(\mathbf{C}_i, \mathbf{q}_{n+1})$
  end
  $\mathbf{q}_{n+1} = SolveLinearSystem(\mathbf{s}_n, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots)$
end
$\mathbf{v}_{n+1} = (\mathbf{q}_{n+1} - \mathbf{q}_n)/h$

where $\mathbf{A}$ and $\mathbf{B}$ are constant matrices and w is a nonnegative weight. The distance to the constraint set is thus modeled by a *quadratic* function in $\mathbf{q}$ and $\mathbf{p}$, which allows us to deploy an efficient solver. Moreover, we are not restricted to Green's strains but can use any constraint definition $\mathbf{C}(\mathbf{q}) = \mathbf{0}$ for the set of desired configurations [8], e.g., describing desired bending angles between triangles, goal volumes for tetrahedrons, or boundary conditions, as discussed below.

7.2.3. Projective Implicit Euler Solver

Using simplified potentials as given in Equation 7.7, we can reformulate the implicit integration defined in Equation 7.4 as the minimization of

$$\frac{1}{2h^2}\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q}-\mathbf{s}_n)\|_F^2 + \sum_i \frac{w_i}{2}\|\mathbf{A}_i\mathbf{S}_i\mathbf{q} - \mathbf{B}_i\mathbf{p}_i\|_F^2 + \delta_{\mathbf{C}_i}(\mathbf{p}_i) \tag{7.8}$$

over $\mathbf{q}$ and the auxiliary variables $\mathbf{p}_i$, where $\mathbf{S}_i$ is a constant selection matrix that selects the vertices involved in the ith constraint. We minimize Equation 7.8 using a local/-global alternating minimization technique.

Local solve. First, we minimize Equation 7.8 over the auxiliary variables keeping the positions fixed. Since each constraint has its own set of auxiliary variables $\mathbf{p}_i$, the minimization can be performed independently for each constraint as

$$\min_{\mathbf{p}_i} \frac{w_i}{2}\|\mathbf{A}_i\mathbf{S}_i\mathbf{q} - \mathbf{B}_i\mathbf{p}_i\|_F^2 + \delta_{\mathbf{C}_i}(\mathbf{p}_i), \tag{7.9}$$

which allows massive parallelization of the local step. We will discuss specific constraint types in Section 7.4.

Global solve. Second, we minimize Equation 7.8 over the positions, keeping the auxiliary variables fixed. Since Equation 7.8 is quadratic in the unknowns $\mathbf{q}$, we can minimize it with a single linear solve. Requiring that the gradient vanishes at the critical point leads to the linear system

$$(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i)\mathbf{q} = \frac{\mathbf{M}}{h^2}\mathbf{s}_n + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{B}_i \mathbf{p}_i. \tag{7.10}$$

The system matrix is constant as long as the constraints are not changing and therefore can be prefactored at initialization, allowing for very efficient global solves. The right hand side requires recomputation in each iteration after the projection variables have been updated in the local step. Note that the objective is bounded below and that both local and global steps are guaranteed to weakly decrease it, even for non-convex sets. Consequently, the optimization converges, making safeguards unnecessary.

Algorithm. We summarize our optimization procedure in Algorithm 2. On line 2 we warm start the optimization using the momentum estimate $\mathbf{s}_n$. We observe that this is favorable when using only few solver iterations, leading to less damped systems than when using the last time step's solution as starting point. After solving multiple local/global iterations the velocities are updated in line 9.

Choice of A and B. If we choose $\mathbf{A}_i = \mathbf{B}_i = \mathbf{I}$, Equation 7.7 measures the squared *Euclidean* distance from $\mathbf{S}_i\mathbf{q}$ to its closest point on the constraint set. With diagonal matrices, the Hessian of the global solve ends up being diagonal as well, leading to a trivial linear system to solve. However, this choice corresponds to working directly with absolute positions, which results in a poor convergence rate because changes propagate slowly through the (usually locally) coupled points [29].

The convergence can be greatly improved if we make use of the fact that internal physical constraints are translation invariant (i.e., applying a common translation to all involved points in the constraints does not change the values of the constraints). In this case, we can choose $\mathbf{A}_i = \mathbf{B}_i$ as differential coordinate matrices (global translation in their null space). Various such matrices can be used, for example one can subtract the mean [29] or simply one of the vertices involved in the constraint [115]. Note that the choice of $\mathbf{A}_i$ and $\mathbf{B}_i$ only impacts the numerical solution procedure and does not affect the conservation of momentum.

Using such differential coordinates greatly improves the convergence speed of the resulting local/global solver [29]. However, without further precautions, the resulting

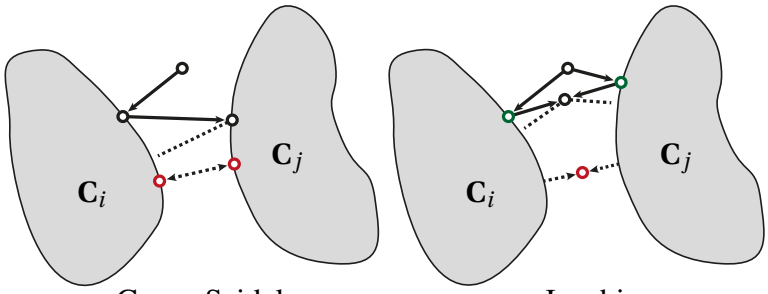


Figure 7.3.: Gauss-Seidel vs. Jacobi. The Gauss-Seidel algorithm used in PBD consecutively projects the current estimate on each constraint set ($\mathbf{C}_i$ and $\mathbf{C}_j$ in this case). If there is no feasible solution, i.e., the constraint sets do not overlap, the Gauss-Seidel algorithm will oscillate between the different constraints (between the two red points). On the contrary, the Jacobi algorithm projects the current estimate on each constraint set in parallel (green points) and reaches a consensus in a second step. This allows the Jacobi algorithm to converge (red point).

behavior is tessellation and resolution dependent. We show in Section 7.4 that in certain cases the $\mathbf{A}_i$ and $\mathbf{B}_i$ matrices can be derived from continuum formulations in order to avoid these shortcomings.

7.3. Position Based Dynamics View

While [115] hint at the similarity between general variational implicit Euler and PBD, in this section, we derive the exact relationship not just between implicit Euler and PBD, but also between the local/global formulation and PBD, for general constraints. This analysis highlights the close connections of PBD to our solver, but also identifies fundamental differences that explain the higher accuracy of results obtained with our approach.

7.3.1. Gauss-Seidel Solver

A classical PBD solver [128] performs three steps. In the first step, the positions are initialized by an explicit Euler step, ignoring internal forces. In the second step, the positions are updated by projecting the current configuration consecutively on each constraint set respecting the mass weighting. In the last step, the velocities are updated as $\mathbf{v}_{n+1} = (\mathbf{q}_{n+1} - \mathbf{q}_n)/h$.

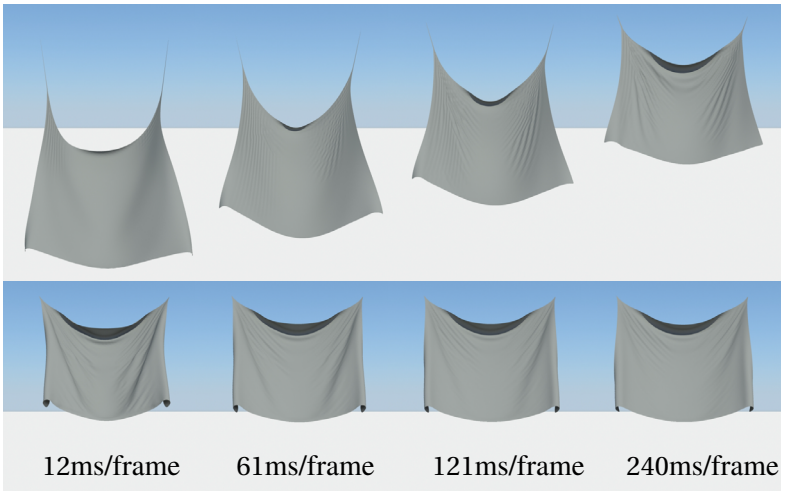


Figure 7.4.: For a piece of cloth with 19683 DoFs and 19360 edge constraints, PBD exhibits different material stiffness depending on the allowed time budget for a time step (top). Due to the additional momentum term and the differential coordinate formulation, our simulation behaves consistently even for different number of iterations (bottom).

We can show that the constraint resolution strategy of PBD actually implements a Gauss-Seidel type minimization on the energy

$$\frac{1}{2}\sum_i \|\mathbf{M}_i^{\frac{1}{2}}(\mathbf{S}_i\mathbf{q}-\mathbf{p}_i)\|_F^2 + \delta_{C_i}(\mathbf{p}_i), \tag{7.11}$$

using a lumped mass matrix $\mathbf{M}_i$ only involving the constraint's points. A Gauss-Seidel approach minimizes this energy by optimizing each summand sequentially, i.e., minimizing potentials of the form $\frac{1}{2}\|\mathbf{M}^{\frac{1}{2}}\Delta\mathbf{q}\|_F^2 + \delta_C(\mathbf{q}+\Delta\mathbf{q})$ where we introduce corrections $\Delta\mathbf{q} = \mathbf{p}-\mathbf{q}$ to simplify the derivation. Using Lagrange multipliers for the linearized constraint $C(\mathbf{q}) + \mathrm{tr}(\nabla C(\mathbf{q})^T\Delta\mathbf{q}) = 0$, we can define the Lagrangian

$$\frac{1}{2}\|\mathbf{M}^{\frac{1}{2}}\Delta\mathbf{q}\|_F^2 + \lambda\left(C(\mathbf{q}) + \mathrm{tr}(\nabla C(\mathbf{q})^T\Delta\mathbf{q})\right). \tag{7.12}$$

Using the critical point condition w.r.t. $\Delta\mathbf{q}$, we find the optimal direction $\Delta\mathbf{q} = -\lambda\mathbf{M}^{-1}\nabla C(\mathbf{q})$. The Lagrange multiplier λ can then be found by requiring that the linearized constraint vanishes in this direction, i.e., $C(\mathbf{q}) - \lambda\|\mathbf{M}^{-\frac{1}{2}}\nabla C(\mathbf{q})\|_F^2 = 0$, leading to the final update

$$\Delta\mathbf{q} = -\mathbf{M}^{-1}\nabla C(\mathbf{q})\frac{C(\mathbf{q})}{\|\mathbf{M}^{-\frac{1}{2}}\nabla C(\mathbf{q})\|_F^2}, \tag{7.13}$$

corresponding exactly to the mass-weighted update rule of PBD [15].

Discussion. Theoretically, Gauss-Seidel has good convergence, however only for feasible constraint sets. For non-feasible sets, lacking a global view on the optimization problem, Gauss-Seidel will oscillate between the incompatible sets (see Figure 7.3). As an example, when simulating the compression of an elastic material with stretch constraints and boundary conditions or collisions, the constraints can become unfeasible and thus the solution will oscillate and not converge.

More severely, the same is true for the momentum estimation performed in the first step, which consists of first solving the constraint given in Equation 7.5. If added as a true constraint to the optimization, it could lead to completely incompatible constraint sets and make convergence even worse. By solving the momentum constraint first with the initial explicit Euler step, it is possible to maintain the linear momentum of the entire object, however the individual momenta of the points are washed away the longer the optimization iterates – contrary to finding a compromise between momentum and internal elasticity as suggested by the Implicit Euler solver that we propose (see Figure 7.4).

7.3.2. Jacobi Solver

In the view of Equation 7.11, we can solve these issues in a straightforward manner by performing two steps. First, we replace the Gauss-Seidel by a Jacobi solver (see Figure 7.3) that is able to deal with incompatible constraints. Jacobi solvers have in general slower convergence than Gauss-Seidel solvers [172]. However, they allow the use of differential coordinate representations for faster convergence and efficient parallelization of the constraint projections that resolve this shortcoming. Second, we introduce the momentum constraint into the optimization to take into account the inertia of each point. As seen in the continuum mechanics view, to achieve a correct behavior we need to add back the inertia of each point by integrating the momentum constraint term defined in Equation 7.5

$$\frac{1}{2h^2}\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q}-\mathbf{s}_n)\|_F^2+\sum_i\frac{w_i}{2}\|\mathbf{M}_i^{\frac{1}{2}}(\mathbf{S}_i\mathbf{q}-\mathbf{p}_i)\|_F^2+\delta_{\mathbf{C}_i}(\mathbf{p}_i). \tag{7.14}$$

The Jacobi solver then becomes a two-step optimization: In the local step, the current solution $\mathbf{q}$ is first projected onto the constraints independently by solving Equation 7.11 for all $\mathbf{p}_i$. Then, a consensus can be reached between the different solutions by solving the global step over $\mathbf{q}$.

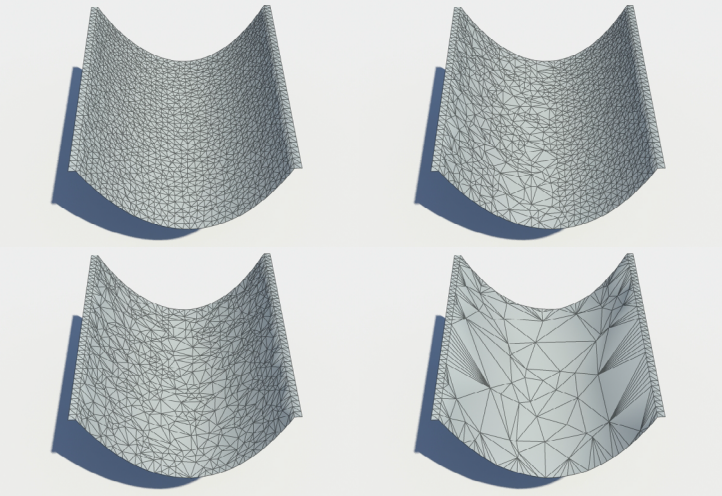


Figure 7.5.: For a given continuous surface, discretizing our continuum based constraints on piecewise simplicial approximations of different resolutions results in very similar qualitative behaviors.

Connection to Projective Implicit Euler. At this stage, we can see how close this Jacobi solver is to our projective implicit solver procedure presented in the last section – we recover this solver by choosing $\mathbf{A}_i = \mathbf{B}_i = \mathbf{M}_i^{\frac{1}{2}}$. By deriving constraints from a continuum principle in the next section we furthermore achieve better independence on mesh tessellation and convergence than with the simpler mass-based weighting used in PBD (see Figure 7.5).

7.4. Continuum-Based Constraints

Differential representations are important for our local/global solver to improve convergence. In geometry processing the gradient and the Laplace-Beltrami operators play an essential role in the design of efficient and robust models. In this section we will present a set of continuous energies based on these operators that allow the control of the differential properties of the material under deformation. We will show that their discretizations will have a form similar to Equation 7.7 that allow for correct behavior under mesh refinement and non-uniform discretizations. The local optimization of the discrete potentials will be discussed in Appendix D.1.

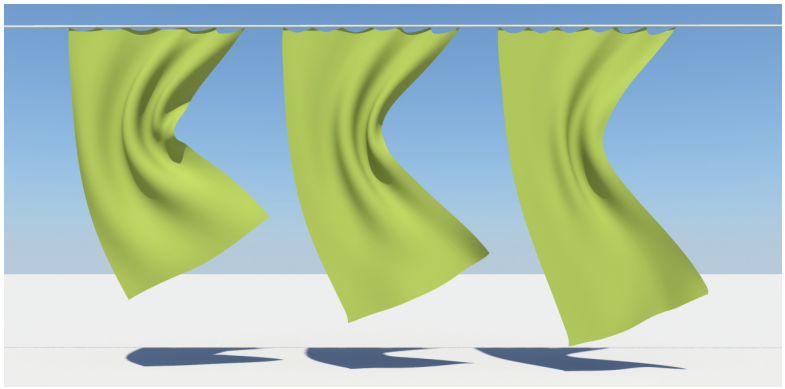


Figure 7.6.: Starting from the same mesh, strain limiting allows simulating material that can undergo small to moderate amount of stretching. From left to right, we use strain limits of [-10%, +10%], [-20%, +20%] and [-30%, +30%]. Notice how the cloth stretches and how the folds get absorbed when the limit increases.

7.4.1. Strain

Continuous energy. Strain energies are important for simulating materials that can stretch. We first discuss 2-manifold surfaces and then extend the results to volumes and curves. Let the undeformed surface be a differentiable 2-manifold surface S embedded in $\mathbb{R}^3$. We define the piecewise linear coordinate function of the undeformed surface by $\mathbf{g}: S \to \mathbb{R}^3$ and its deformed counterpart by $\mathbf{f}: S \to \mathbb{R}^3$. Introducing a set M of desired point-wise transformations $\mathbf{T}$, we formulate an energy measuring the change of local variation between the deformed and the undeformed surface as

$$E(\mathbf{f}, \mathbf{T}) = \frac{w}{2} \int_S \|\nabla_S \mathbf{f} - \mathbf{T}\nabla_S \mathbf{g}\|_F^2 + \delta_M(\mathbf{T}) \, \mathrm{d}A, \tag{7.15}$$

where ∇_S is the gradient operator defined on the manifold surface S. The choice of M determines all allowed rest configurations $\mathbf{T}\nabla_S \mathbf{g}$. If M is the set of rotation matrices $SO(3)$, we are simply measuring the local deviation from a rigid motion. In this case this energy is identical to the deformation model presented by Chao et al. [46]. If M is the set of matrices with bounded singular values $\sigma_{\min} < \sigma < \sigma_{\max}$, we can also achieve *isotropic strain limiting* similar to Wang et al. [185]. This could be further extended to anisotropic material by using reference frames following Hernandez et al. [89].

Discrete potential. If S is a 2-manifold simplicial complex this energy can be discretized over triangles using a piecewise linear hat basis [26]. The integral is then

transformed to a sum of per triangle potentials of the form

$$W(\mathbf{q}, \mathbf{T}) = \frac{w}{2} A \|\mathbf{X_f} \mathbf{X_g}^{-1} - \mathbf{T}\|_F^2 + \delta_M(\mathbf{T}), \tag{7.16}$$

where A is the triangle area, $\mathbf{X_f} = [\mathbf{q}_j - \mathbf{q}_i, \mathbf{q}_k - \mathbf{q}_i] \in \mathbb{R}^{2\times 2}$ contains the triangle edges of the current configuration isometrically embedded in 2D, and similarly $\mathbf{X_g}$ contains the triangle edges of the rest configuration. Note that this discrete potential has the same form as the one in Equation 7.7 where $\mathbf{A}$ is a function of the rest state edges and the area and $\mathbf{B}$ only depends on the rest state area. Figure 7.6 shows the strain limiting constraint applied to a curtain example.

Volumes and curves. This potential can be defined in a similar way for volumes: If S is a 3-manifold simplicial complex the energy can be discretized over tetrahedrons replacing the areas of the triangles by the volumes of the tetrahedrons and having 3×3 edge matrices. Note that if we perform a 1D discretization of this energy over a set of edges, we arrive at a model similar to the fast simulation of mass spring models of Liu et al. [115] where, in addition, the edge potentials are now properly weighted by the edge length.

7.4.2. Area and Volume Preservation

Area and volume preservation is important for simulating incompressible materials. Using the continuous energy of Equation 7.15 we can define M as the set of matrices with bounded determinants $\sigma_{\min} < \det(\mathbf{T}) < \sigma_{\max}$, effectively enabling us to control the amount of volume change. If $\sigma_{\min} < 1$ the modeled material allows for compression and similarly if $\sigma_{\max} > 1$ then the material allows for expansion. Figure 7.7 shows the combination of volume preservation and strain constraints.

7.4.3. Example-Based

Example-based simulation allows modeling artistic elastic material behavior by supplying a few deformation examples that the material should follow [122, 101, 98]. We use an energy comparable to Equation 7.15 defined on 3-manifold surfaces as

$$E(\mathbf{f}, \mathbf{R}, \mathbf{w}) = \frac{w}{2} \int_S \|\nabla_S \mathbf{f} - \mathbf{R} \nabla_S \mathbf{h}(\mathbf{w})\|_F^2 + \delta_{SO(3)}(\mathbf{R}) \, \mathrm{d}V. \tag{7.17}$$

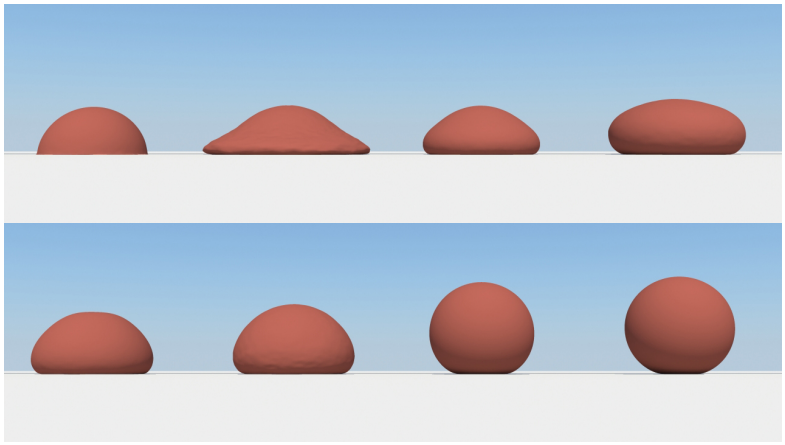


Figure 7.7.: Varying weight combinations of volume preservation and strain constraints allow the simulation of different types of materials for volumetric objects.

where $\mathbf{h}(\mathbf{w})$ is a parametrized rest shape defined by the examples. We formulate the rest shape as $\mathbf{h}(\mathbf{w}) = \mathbf{g} + \sum_i w_i(\mathbf{R}_i\mathbf{g}_i - \mathbf{g})$, where the $\mathbf{g}_i$ define the piecewise linear coordinate functions of the examples and $\mathbf{R}_i$ are precomputed rotation matrices defined point-wise such that it rotates $\mathbf{g}_i$ locally to best align with the undeformed configuration $\mathbf{g}$, similar in spirit to Koyama et al. [101].

We can discretize this continuous energy using a piecewise linear hat basis leading to a sum of per tetrahedron potentials

$$W(\mathbf{q}, \mathbf{R}, \mathbf{w}) = \frac{w}{2} V \|\mathbf{X_f}\mathbf{X_g}^{-1} - \mathbf{R}\mathbf{X_h}(\mathbf{w})\mathbf{X_g}^{-1}\|_F^2 + \delta_{SO(3)}(\mathbf{R}), \tag{7.18}$$

where $\mathbf{X_h}(\mathbf{w}) = \mathbf{X_g} + \sum_i w_i(\mathbf{R}_i\mathbf{X_{g_i}} - \mathbf{X_g})$. Note that the example weights $\mathbf{w}$ can either be defined locally per element or globally, resulting in local or global coupling of the deformation, respectively. An example of three colliding cars using this constraint can be found in Figure 7.8.

7.4.4. Bending

Continuous energy. Thin shells and thin plates are commonly simulated using a bending energy based on dihedral angles across edges [78]. More recently, efficient models for bending of inextensible surfaces relating the Laplace-Beltrami operator to the mean curvature normal have been presented [17, 72]. We introduce a bending energy

Figure 7.8.: Adding the deformation examples (top) to the simulation using the example-based constraint allows the simulation of complex artistic materials. In this scene, three cars collide and react in a cartoonish manner following the prescribed examples (bottom).

measuring the squared difference of absolute mean curvatures

$$E(\mathbf{f}) = \frac{w}{2} \int_S (|H_\mathbf{f}| - |H_\mathbf{g}|)^2 \, \mathrm{d}A, \tag{7.19}$$

where $H_\mathbf{f}$ and $H_\mathbf{g}$ are the mean curvature functions of the deformed and undeformed surface, respectively. For an isometric deformation (inextensible surface) we can then rewrite the energy using auxiliary rotation matrices as

$$E(\mathbf{f}, \mathbf{R}) = \frac{w}{2} \int_S \|\Delta_S \mathbf{f} - \mathbf{R}\Delta_S \mathbf{g}\|_2^2 + \delta_{SO(3)}(\mathbf{R}) \, \mathrm{d}A, \tag{7.20}$$

where Δ_S is the Laplace-Beltrami operator defined on the manifold surface S. This is because the mean curvature vector is equal to the surface's Laplace-Beltrami operator applied to the coordinate function. For an isometric deformation the Laplace-Beltrami operator does not change and therefore can be defined on the undeformed surface. Please notice how similar Equation 7.20 is to Equation 7.15 replacing the gradient by the Laplace-Beltrami. It could therefore be interesting to apply the strain limiting and example-based concepts to the bending energy as well.

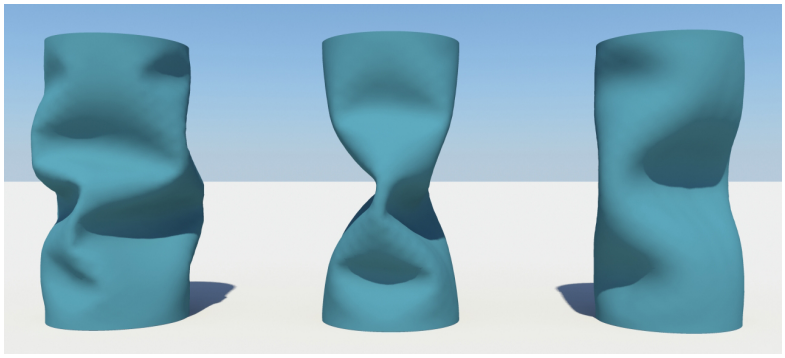


Figure 7.9.: Simulation of a thin shell cylinder using increasing bending weights from left to right. When the cylinder is compressed, buckling patterns of different frequencies appear.

Discrete potential. If S is a 2-manifold simplicial complex Equation 7.20 can be discretized using a piecewise linear hat basis leading to per vertex potentials of the form

$$W(\mathbf{q}, \mathbf{R}) = \frac{w}{2} A \|\mathbf{X_f c} - \mathbf{R X_g c}\|_2^2 + \delta_{SO(3)}(\mathbf{R}), \tag{7.21}$$

where A is the Voronoi area of the vertex, and $\mathbf{X_f}$ and $\mathbf{X_g}$ contain the one-ring edges of the vertex for the current configuration and for the rest configuration, respectively. The vector $\mathbf{c}$ stores the common cotangent weights divided by the Voronoi area [26]. An example of the bending constraint can be found in Figure 7.9. As can be seen in the appendix this bending constraint allows for a very efficient local solve as it can be implemented just as a simple normalization of the mean curvature vector of the deformed configuration.

7.5. Discrete Constraints

The constraints derived from continuous energies presented in the previous section allow modeling a large variety of elastic bodies. For practical animation systems additional constraints are equally important. We model these directly as discrete constraints.

Positional constraints. As seen earlier, individual DoFs can be directly constrained by simply choosing $\mathbf{A}_i = \mathbf{B}_i = \mathbf{I}$ in Equation 7.7. *Dirichlet boundary conditions* can then be realized by defining the constraint set as the desired goal positions, in order to fix objects or create interactive handles.

Collisions. Handling collisions in an implicit manner fits naturally into our general solver and allows respecting the equilibrium of momentum and internal constraints during the collision resolution. When detecting a collision, we dynamically add new unilateral plane constraints. As for positional constraints, we again choose $\mathbf{A}_i = \mathbf{B}_i = \mathbf{I}$ in Equation 7.7. For a colliding point $\mathbf{q}_c$ we first find the closest surface point $\mathbf{b}$ with normal $\mathbf{n}$, defining a collision plane, such that the constraint set $\mathbf{C}$ is defined by the half space $\mathbf{n}^T(\mathbf{q}-\mathbf{b}) \geq 0$. The projection into this half space in the local step is trivial as it is either a plane projection or the identity map. Note that defining the collision constraint unilaterally allows us to overcome the commonly known sticking problems in implicit collisions handling. Similar to PBD, we handle friction and restitution by changing the velocities of colliding vertices when updating velocities. A simple damping model can also be implemented by filtering velocities [128].

More constraints. General types of geometric constraints, as for example bending constraints using hinge angles [15], can be easily incorporated into our solver. The local solve can be performed in a general manner by minimizing Equation 7.7 over the auxiliary variables. For many geometric constraints closed-form solutions for this minimization can be found [29]. If no closed-form solutions exist, the optimization can be solved using sequential quadratic programming (SQP) [134]. As shown in Section 7.3.1, for the case of $\mathbf{A} = \mathbf{B} = \mathbf{M}^{\frac{1}{2}}$ one step of SQP is similar to the PBD update [15].

7.6. Results

7.6.1. Generality

Our solver does not rely on any particular type of constraint and is able to deal with any variety of geometric constraints within the same setup, making it possible to simulate complex sceneries using a single solver and to also handle object interactions robustly in an implicit manner. In Figure 7.1 we show such a complex scene with different constraint types, where the objects are also coupled together. For example, the tree and the house are modeled with volumetric strain constraints whereas the washing line, the cloth, the grass and the leaves use edge strain and bending constraints.

Cloths and shells. In Figure 7.10 we simply use edge strain constraints to model the behavior of a pirate flag. Wind forces are added as a function of wind direction

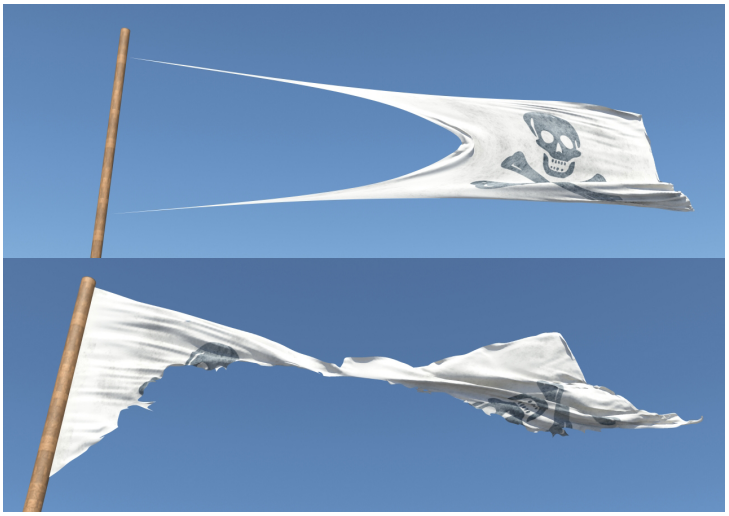


Figure 7.10.: Even under extreme wind forces our projective implicit solver remains stable. The solver weakly decreases the energy at each iteration making any safeguards unnecessary (top). The pirate flag is torn by the wind in real-time using dynamic updates of the constraints (bottom).

and triangle normal. When the wind forces are too strong, the pirate flag is torn. This is realized by removing edge constraints when the strain exceeds a certain threshold. More complex cloths that can undergo small to moderate amounts of stretching can also be modeled using a limit on the triangle strain in combination with bending constraints (see Figure 7.6). By varying the weights of the strain and bending constraints other types of materials such as thin plates and thin shells can be simulated. In Figure 7.9 we can see an example of a thin cylinder compressed from the top and showing different buckling patterns due to different ratios of strain and bending constraint stiffnesses.

Solids and exampled-based simulation. We simulate solids by using a combination of strain and volume constraints applied on tetrahedral meshes. As shown in Figure 7.7, different type of materials can be modeled by varying the weights combining these constraints. While we cannot model arbitrary non-linear materials, we are able to approximate some non-linear behaviour by combining weak strain constraints with stronger strain limiting constraints. Then, the material is soft for small deformations while becoming stiffer when the deformation reaches the strain limit and the second constraint becomes active – a behavior commonly modeled by nonlinear material models. Combining different quadratic potentials has been used earlier for collision handling in [87] but also suits very well our framework to model non-linear material behavior.

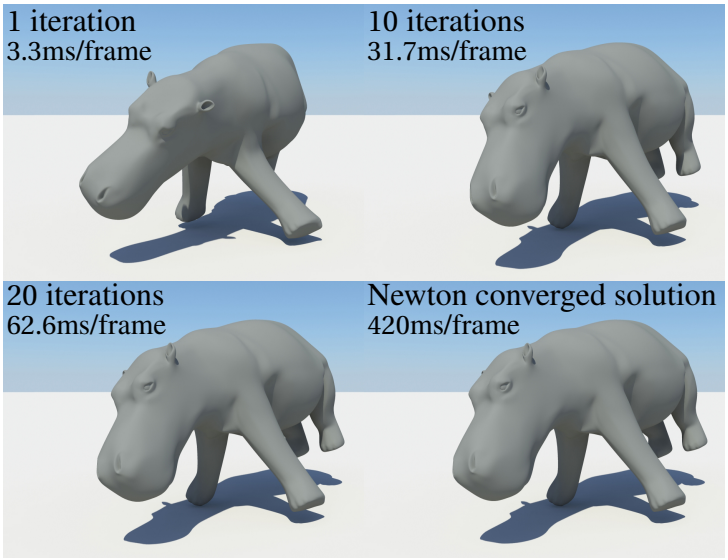


Figure 7.11.: This volumetric hippopotamus with 7161 DoFs and 8406 strain constraints is simulated with 1, 10, and 20 iterations of our local/global solver. It is interesting to notice that already after 10 iterations our approach looks very similar to the converged solution computed using Newton's method for a fraction of the computational cost.

Example-based simulation of volumetric meshes is also possible in our formulation. This allows an artistic control over the physical simulation. In Figure 7.8 three cars deform in a cartoonish manner following the input examples after colliding. Similar to [122] the car surface is embedded into a volumetric mesh, which is then deformed using our solver.

7.6.2. Robustness and Simplicity

One important advantage of our approach is numerical stability. In Figure 7.10 we show that even under extreme forces our solver stays robust. Similarly, our method remains reliable in situations where the mesh elements degenerate. The only requirement of our approach is that the mesh elements of the input model are well behaved in order to compute the discretization of the gradient and Laplace-Beltrami operators of the original manifold.

We illustrate the simplicity of our approach by laying out our optimization procedure in Algorithm 2. By removing line 7 and changing $\mathbf{p}_i$ to $\mathbf{q}_{n+1}$ in line 5, we are able to completely recover the structure of the original PBD algorithm [128]. Moreover, notice that introducing a new constraint only requires the definition of the constraint

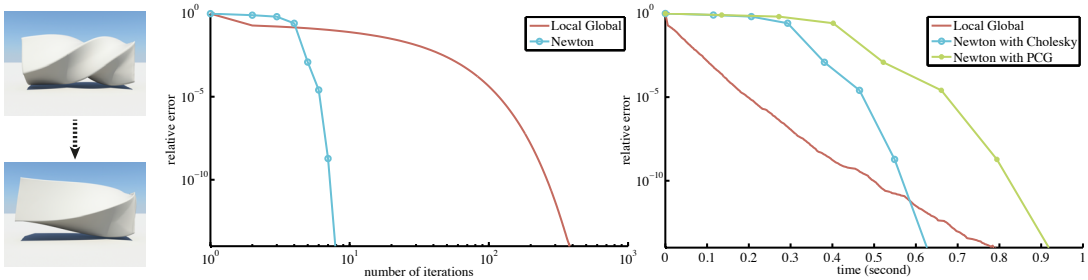


Figure 7.12.: By comparing the decrease of the relative error with respect to the iteration count, we observe that Newton's method converges faster than our local/global approach. However, this does not reflect the cost of each iteration as for each Newton iteration a changing linear system needs to be solved. Looking at the decrease of the relative error with respect to the computation time, we notice that our local/global approach exhibits a better performance up to a relative error of 10^{-10} making our approach particularly attractive for interactive applications. In these curves, the relative error is defined as the normalized error relative to the optimal solution $(\epsilon(\mathbf{q}_i) - \epsilon(\mathbf{q}^*))/(\epsilon(\mathbf{q}_0) - \epsilon(\mathbf{q}^*))$ and measured for a twisting bar example (left) with 4290 DoFs and 4099 tetrahedral strain constraints.

projection used in the local solve (either exact if known or the general approximate projection scheme given in [128] if not) and the definition of suitable quadratic distance metrics (matrices $\mathbf{A}_i$ and $\mathbf{B}_i$).

7.6.3. Accuracy and Performance

Comparison with Newton. In Figure 7.12 we compare the performance of our local/global solver to Newton's method when solving the discretization of Equation 7.15 for $M = SO(3)$ similar to [46]. As shown in Figure 7.12 the local/global approach converges slower in number of iterations. This is perfectly logical as Newton's method exhibits quadratic convergence while local/global solvers (block coordinate descent methods) have linear convergence. However, when looking at the convergence in terms of computational time, we notice that our approach is faster than Newton's method for interactive applications. For 1 Newton iteration approximatively 30 local/global iterations can be performed.This is due to the fact that at each Newton iteration the Hessian needs to be recomputed and therefore a new linear system needs to be solved.

Moreover, in Figure 7.11 we observe that with approximatively 10 iterations the simulation looks visually similar to the converged one using Newton's method making our scheme a better choice for realtime applications where high accuracy is not the main

focus. This type of behavior has already been observed in some of the previous local/global solvers used in geometry processing and simulation [130, 115]. Note that implementing Newton's method for the continuous energies presented in Section 7.4 is nontrivial as one needs to differentiate SVD [123] and new Hessian matrices have to be computed in each time step. Moreover, some safeguards need to be integrated in the optimization as the Hessian matrix may become indefinite and a line search procedure is also needed to avoid overshooting.

Comparison with Position Based Dynamics. We also compared our approach to PBD using edge strain constraints. As explained in Section 7.3, PBD does not include the momentum constraints making the material stiffness dependent of the number of iterations. This can be seen in Figure 7.4, where for different number of iterations the stiffness of the material simulated by PBD drastically changes. This is not the case in our approach where the material stiffness is much less dependent of the number of iterations.

Meshing independence. In Section 7.4 we presented a set of new constraints derived from continuous energies. As shown in Figure 7.5, these new constraints allow our solver to maintain the deformation behavior under different piecewise simplicial approximations of the same underlying surface. This is an important property for computer graphics applications and interactive environments where mesh resolutions can frequently change during development and where geometric levels of detail are widely employed to increase performance. The lack of convergence of PBD approaches makes it difficult to handle geometric level of detail properly due to the dependence of the material behavior on the underlying meshing and of the number of iterations [82].

7.7. Implementation

The complete framework presented in this section is implemented in C++. We use OpenMP to parallelize the local step and we solve the global step in parallel for the x, y and z coordinates by prefactorizing the linear system using sparse Cholesky factorization and performing three times back-substitution in parallel. Dynamic constraints are handled by rank updates and downdates of the linear system. The Eigen library (eigen.tuxfamily.org) is used for dense and sparse linear algebra. We use either the standard simplicial mass discretization [92] or its lumped version to compute the mass matrix without any noticeable difference.

Timing. For simulation of medium sized models (< 30K constraints and < 30K DoFs), 5-10 iterations are usually sufficient. At 1-6ms per iteration, this enables real-time simulation on a MacBook Pro 2.7 GHz Intel Quad-core i7 with 16GB of memory.

7.8. Limitations and Future Work

While our implicit Euler solver is efficient and robust, it exhibits implicit damping. In the near future we plan to extend our approach to symplectic integrators [99] which provide better energy behavior. Damping can also be observed when the optimization is terminated early. This is due to the fact that external forces may not be able to propagate fully through the mesh if the optimization is not run for enough iterations. This effect is accentuated in large meshes as more iterations are needed until convergence. As a future work, we would like to improve the speed of our solver by implementing a GPU version of our code and focus on topological changes (cutting, fracturing) that result in dynamically changing constraints. While the local steps remain simple to solve on the GPU, the global system is changing, making it even more involved to solve efficiently. This problem becomes even more accentuated if we want to extend our approach to fluid simulation similar to [119] where neighborhood relations always change.

We are trading hard constraints for simplicity and efficiency. Treating all constraints in a soft manner allows us to handle them in a unified and effective manner. However, in certain situations, being able to enforce hard constraints, such as for collision handling or boundary conditions, would be advantageous. Hard constraints can still be approximated by increasing the weight of the constraints. However, this can degrade the conditioning of the linear system and can result in locking artifacts.

Another interesting area of further research is enlarging our set of constraints. One direction we want to explore is modeling more complex deformation behaviors such as in anisotropic and non-linear materials. Furthermore, it would also be attractive to integrate rigid bodies into the same simulation framework.

7.9. Additions and Remarks

We introduce a new implicit constraint-based solver for real-time simulation. Our approach is based on an abstract, constraint-based description of the physical system making our method *general* in its use to simulate a large variety of different geometries and

materials. To solve the constraint problem, we apply a local/global solver that is guaranteed to weakly decrease the energy making any safeguards unnecessary and giving us *robustness*. Our *simple* constraint-based formulation only requires the definition of a projection operator for a given constraints (local solve), making it very easy to implement and to introduce new models into the solver. Furthermore, the global solve only requires solving a linear system, where the system matrix is constant if the number of constraints is kept fixed, leading to *efficient* computation. Due to the independence of the local solves, the approach is also very well suited for parallelism, further boosting performance. We derive a broad set of constraints directly from continuous energies using proper discretization that make the solver robust to non-uniform meshing with different resolutions. With these qualities in mind we believe that our approach strikes the right balance between the simplicity, generality, robustness and performance of position-based simulations with the rigor and accuracy of continuum mechanics. We believe this makes our method suitable for many applications in both realtime and offline simulation in computer graphics.

Stiff constraints. As mentioned in Section 7.8, hard constraints can be approximated by increasing the weight of the constraints. However, this can degrade the conditioning of the linear system and can result in locking artifacts. This problem can be partially overcome by using the robust penalty approach presented in [77]. Recall that our global solve consists in solving

$$\frac{\mathbf{M}}{h^2}(\mathbf{q}-\mathbf{s}_n)+\sum_i \frac{1}{\mu_i}\mathbf{S}_i^T\mathbf{A}_i^T(\mathbf{A}_i\mathbf{S}_i\mathbf{q}-\mathbf{B}_i\mathbf{p}_i)=0, \tag{7.22}$$

where $\mu_i = \frac{1}{w_i}$. Using a set of auxiliary variables, Equation 7.22 can be rewritten as

$$\begin{cases} \frac{\mathbf{M}}{h^2}(\mathbf{q}-\mathbf{s}_n)+\sum_i \mathbf{S}_i^T\mathbf{A}_i^T\boldsymbol{\lambda}_i=0, \\ \quad \mu_i\boldsymbol{\lambda}_i=\mathbf{A}_i\mathbf{S}_i\mathbf{q}-\mathbf{B}_i\mathbf{p}_i. \end{cases} \tag{7.23}$$

This results in a symmetric linear system

$$\begin{bmatrix} \frac{\mathbf{M}}{h^2} & \mathbf{S}_1^T\mathbf{A}_1^T & \mathbf{S}_2^T\mathbf{A}_2^T & \dots \\ \mathbf{A}_1\mathbf{S}_1 & -\mu_1 & \mathbf{0} & \dots \\ \mathbf{A}_2\mathbf{S}_2 & \mathbf{0} & -\mu_2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{M}}{h^2}\mathbf{s}_n \\ \mathbf{B}_1\mathbf{p}_1 \\ \mathbf{B}_2\mathbf{p}_2 \\ \vdots \end{bmatrix} \tag{7.24}$$

For stiff constraints, i.e., when the μ_i vanish, this linear system does not suffer from the

ill conditioning present in the direct solution of Equation 7.22 [77]. It is interesting to note that when the μ_i vanish the $\boldsymbol{\lambda}_i$ are in fact equivalent to Lagrange multipliers. This linear system becomes indefinite for very stiff constraints and can be prefactored and solved using sparse LDLT Cholesky factorization.

Chapter 8

Conclusion

This thesis presents a complete pipeline for realtime facial tracking and animation. We have demonstrated that high-quality facial animation can be done in realtime using a simple low-cost RGB-D sensor and showed the potential of our system for applications in virtual reality, human interaction, live virtual TV shows, and computer gaming [66].

In Chapter 4, robust realtime tracking is achieved by building suitable user-specific blendshape models and exploiting the different characteristics of the acquired 2D image and 3D depth map data for registration. We found that learning the dynamic expression space from existing animations is essential. Combining these animation priors with effective geometry and texture registration in a single MAP estimation is our key contribution to achieve robust tracking even for highly noisy input data. While foreseeable technical advances in acquisition hardware will certainly improve data quality in coming years, numerous future applications, e.g. in multi-people tracking, acquisition with mobile devices, or performance capture in difficult lighting conditions, will produce even worse data and will thus put even higher demands on robustness. Our algorithm provides a systematic framework for addressing these challenging problems.

The main drawback of this tracking approach in the context of consumer applications is the need for an offline user-specific training. In Chapter 5, we have demonstrated that online model building can replace user-specific training and manual calibration for facial performance capture systems, while maintaining high tracking accuracy.

In Chapter 6, we have introduced a novel statistical approach to high-quality facial animation retargeting that achieves better results than other non-linear regression techniques. By leveraging the information contained in unlabeled data, a key novelty in our retargeting approach, we can reduce the number of required training examples. We have shown that our approach is well suited to retargeting facial animations from motion

capture data, as posing a character is time consuming, while unlabeled data is easily obtained by tracking the actor. Since our method implicitly learns a low-dimensional representation, our system has no difficulty dealing with complex, high-dimensional input or output data commonly used in studio productions. At the same time, the robustness of our approach to noise and missing data makes the method particularly suitable for low-cost motion capture systems [66]. In addition, our method can simplify character posing by exploiting the correlation between the different character parameters.

In Chapter 7 we presented a novel implicit solver for realtime physics-based animation. We believe that our approach strikes the right balance between accuracy, simplicity, generality, robustness and performance. This makes our method suitable for realtime applications and a good candidate to complement our realtime facial animation system. Muscles, fat, hair, or flesh could be simulated to enhance the animated avatars with secondary effects. Moreover, our solver could be used to include contact and collision of the face with external objects.

The research progress outlined in this thesis brings performance-based facial animation within reach of consumer level applications. However, a number of principle and fundamental problems remain, and the uncanny valley [127] poses numerous challenges when aiming for realistic human avatars.

Our current facial tracking system uses a simple blendshape representation [108] trading-off tracking accuracy in return for increased performance. Because of the linearity of the blendshape model, reproducing non-linear motions is difficult. It is currently an open question as to which parametric face model would have the best trade-off between accuracy and performance. One promising path of research is on the automatic adaptation of a high resolution, anatomically accurate face model (bones, tendons, flesh, muscles and skin) to motion sequences captured from a subject [3, 201]. Not only the anatomical model could be adapted to the user's face but muscle activations could be digitized [157]. This approach would complicate the tracking process but would provide non-linear behaviors not captured by approaches that linearly blend basis functions. To achieve realtime performance, our implicit solver [30] could be used instead of a complex nonlinear finite element method [157].

While accurate digitization of static facial geometry has seen a tremendous increase in quality in the last few years [11, 12, 16], dynamic face models have still a long way to go to become indistinguishable from reality. Creating a fully rigged avatar of a person is currently a highly complex process that requires a complicated acquisition setup and significant manual work [2, 1]. The approach described in this thesis [34], and its extension presented in [94], allow to model a fully rigged model of the face. However, we do

not model hair, teeth or tongue. Unfortunately, these details are crucial to generate a realistic avatar of a person and are currently missing in our system. Modeling and tracking fine scale details using a low-cost device is a challenging task and an interesting topic of future research.

We presented a novel implicit solver allowing for the realtime simulation of a large range of deformable materials, e.g., muscles, fat, hair, flesh. One of the main challenges we are facing is to create an accurate anatomical face model suitable for physics simulation. Creating such a model is complex and necessitates precise facial measurements (e.g., facial soft tissue thickness) and significant manual work [157]. These measurements could be done using MRI or CT scans [157, 3, 13]. Numerous studies examine facial tissue depth in different populations and report precise facial measurements with average thickness values for different landmarks as well as standard deviations [93, 81, 39, 138]. An interesting avenue of future work is to create a statistical model from these databases to build an adaptable and accurate face model suitable for physics simulation.

To conclude, this thesis proposes a complete system for realtime facial tracking and animation. We believe that our system will enable a variety of new applications in human communication, such as in-game puppetry, virtual avatars for social networks, or computer-assisted realtime training applications, and will be the basis for substantial follow-up research. There is still a large number of problems and open questions that need to be solved before being able to seamlessly communicate with 3D avatars. These problems not only matter for communication and virtual reality, but solving such a principle problem than accurately capturing the motions and appearances of humans would enable numerous other applications in robotics, security, human-computer interaction, and medical. In the future, 3D acquisition and display technologies will become smaller, cheaper and more accurate, compute power will keep increasing, and we will see numerous advances in computer graphics, computer vision and machine learning algorithms. When virtual reality technologies will be ready, we will be able to experience whatever we want, in any places we can imagine, with anyone in the world. This will change everything about the way we live and communicate.

Bibliography

[1] ALEXANDER, O., FYFFE, G., BUSCH, J., YU, X., ICHIKARI, R., JONES, A., DEBEVEC, P., JIMENEZ, J., DANVOYE, E., ANTIONAZZI, B., EHELER, M., KYSELA, Z., AND VON DER PAHLEN, J. Digital ira: Creating a real-time photoreal digital actor. In *ACM SIGGRAPH 2013 Posters* (2013).

[2] ALEXANDER, O., ROGERS, M., LAMBETH, W., CHANG, M., AND DEBEVEC, P. The digital emily project: photoreal facial modeling and animation. *ACM SIGGRAPH Courses* (2009).

[3] ALI-HAMADI, D., LIU, T., GILLES, B., KAVAN, L., FAURE, F., PALOMBI, O., AND CANI, M.-P. Anatomy transfer. *ACM Trans. Graph.* (2013).

[4] AMBERG, B., BLAKE, A., FITZGIBBON, A. W., ROMDHANI, S., AND VETTER, T. Reconstructing high quality face-surfaces using model based stereo. In *ICCV* (2007).

[5] AMBERG, B., BLAKE, A., AND VETTER, T. On compositional image alignment, with an application to active appearance models. In *Computer Vision and Pattern Recognition* (2009).

[6] BACH, F. R., JENATTON, R., MAIRAL, J., AND OBOZINSKI, G. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning* (2012).

[7] BALTRUŠAITIS, T., ROBINSON, P., AND MORENCY, L.-P. 3d constrained local model for rigid and non-rigid facial tracking. In *Computer Vision and Pattern Recognition* (2012).

[8] BARAFF, D., AND WITKIN, A. Large steps in cloth simulation. In *Proc. of ACM SIGGRAPH* (1998).

[9] BARRETT, R., BERRY, M., CHAN, T. F., DEMMEL, J., DONATO, J., DONGARRA, J., EIJKHOUT, V., POZO, R., ROMINE, C., AND DER VORST, H. V.

Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition. SIAM, 1994.

[10] BECK, A., AND TETRUASHVILI, L. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization* (2013).

[11] BEELER, T., BICKEL, B., BEARDSLEY, P., SUMNER, B., AND GROSS, M. High-quality single-shot capture of facial geometry. *ACM Trans. Graph.* (2010).

[12] BEELER, T., BICKEL, B., NORIS, G., BEARDSLEY, P., MARSCHNER, S., SUMNER, R. W., AND GROSS, M. Coupled 3d reconstruction of sparse facial hair and skin. *ACM Trans. Graph.* (2012).

[13] BEELER, T., AND BRADLEY, D. Rigid stabilization of facial expressions. *ACM Trans. Graph.* (2014).

[14] BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph.* (2011).

[15] BENDER, J., MÜLLER, M., OTADUY, M. A., AND TESCHNER, M. Position-based methods for the simulation of solid objects in computer graphics. In *EG State of the Art Reports* (2013).

[16] BÉRARD, P., BRADLEY, D., NITTI, M., BEELER, T., AND GROSS, M. High-quality capture of eyes. *ACM Trans. Graph.* (2014).

[17] BERGOU, M., WARDETZKY, M., HARMON, D., ZORIN, D., AND GRINSPUN, E. A quadratic bending model for inextensible surfaces. In *Computer Graphics Forum* (2006).

[18] BERMANO, A. H., BRADLEY, D., BEELER, T., ZÜND, F., NOWROUZEZAHRAI, D., BARAN, I., SORKINE, O., PFISTER, H., SUMNER, R. W., BICKEL, B., AND GROSS, M. Facial performance enhancement using dynamic shape space analysis. *ACM Trans. Graph.* (2014).

[19] BESL, P. J., AND MCKAY, N. D. A method for registration of 3D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (1992).

[20] BHAT, K. S., GOLDENTHAL, R., YE, Y., MALLET, R., AND KOPERWAS, M. High fidelity facial animation capture and retargeting with contours. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2013).

[21] BICKEL, B., BOTSCH, M., ANGST, R., MATUSIK, W., OTADUY, M., PFISTER, H., AND GROSS, M. Multi-scale capture of facial geometry and motion. *ACM Trans. Graph.* (2007).

[22] BICKEL, B., LANG, M., BOTSCH, M., OTADUY, M. A., AND GROSS, M. Pose-space animation and transfer of facial details. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2008).

[23] BLACK, M. J., AND YACOOB, Y. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *International Conference on Computer Vision* (1995).

[24] BLANZ, V., AND VETTER, T. A morphable model for the synthesis of 3d faces. In *Proc. of ACM SIGGRAPH* (1999).

[25] BLEMKER, S., TERAN, J., SIFAKIS, E., FEDKIW, R., AND DELP, S. Fast 3d muscle simulations using a new quasistatic invertible finite-element algorithm. In *International Symposium on Computer Simulation in Biomechanics* (2005).

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

[27] BOTSCH, M., SUMNER, R., PAULY, M., AND GROSS, M. Deformation Transfer for Detail-Preserving Surface Editing. In *Vision, Modeling, Visualization* (2006).

[28] BOUAZIZ, S., DENG, B., AND PAULY, M. Projection-based optimization with fast convergence. *Computer Graphics Forum* (2015). Submitted.

[29] BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. Shape-up: Shaping discrete geometry with projections. In *Computer Graphics Forum* (2012).

[30] BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.* (2014).

[31] BOUAZIZ, S., AND PAULY, M. Dynamic 2d/3d registration for the kinect. In *ACM SIGGRAPH Courses* (2013).

[32] BOUAZIZ, S., AND PAULY, M. Semi-supervised facial animation retargeting. Tech. rep., EPFL, 2014.

[33] BOUAZIZ, S., TAGLIASACCHI, A., AND PAULY, M. Sparse iterative closest point. *Computer Graphics Forum* (2013).

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

[35] BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* (2011).

[36] BOYD, S., AND VANDENBERGHE, L. *Convex optimization*. Cambridge University Press, 2004.

[37] BRADLEY, D., HEIDRICH, W., POPA, T., AND SHEFFER, A. High resolution passive facial performance capture. *ACM Trans. Graph.* (2010).

[38] BRIDSON, R., MARINO, S., AND FEDKIW, R. Simulation of clothing with folds and wrinkles. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2003).

[39] BULUT, O., SIPAHIOGLU, S., AND HEKIMOGLU, B. Facial soft tissue thickness database for craniofacial reconstruction in the turkish adult population. *Forensic science international* (2014).

[40] CANDÈS, E. J., AND WAKIN, M. B. An introduction to compressive sampling. *IEEE Signal Process. Mag.* (2008).

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

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

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

[44] CHAI, J. X., XIAO, J., AND HODGINS, J. Vision-based control of 3d facial animation. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2003).

[45] CHAI, M., ZHENG, C., AND ZHOU, K. A reduced model for interactive hairs.

ACM Trans. Graph. (2014).

[46] CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. A simple geometric model for elastic deformations. *ACM Trans. Graph.* (2010).

[47] CHARTRAND, R., AND YIN, W. Iteratively reweighted algorithms for compressive sensing. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (2008).

[48] CHEN, Y., AND MEDIONI, G. Object modeling by registration of multiple range images. In *IEEE International Conference on Robotics and Automation* (1991).

[49] CHETVERIKOV, D., SVIRKO, D., STEPANOV, D., AND KRSEK, P. The trimmed iterative closest point algorithm. In *International Conference on Pattern Recognition* (2002).

[50] CHUANG, E., AND BREGLER, C. Performance driven facial animation using blendshape interpolation. Tech. rep., Stanford University, 2002.

[51] COOTES, T., EDWARDS, G., AND TAYLOR, C. Active appearance models. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2001).

[52] COSTIGAN, T., PRASAD, M., AND MCDONNELL, R. Facial retargeting using neural networks. In *International Conference on Motion in Games* (2014).

[53] COVELL, M. Eigen-points: Control-point location using principle component analyses. In *International Conference on Automatic Face and Gesture Recognition* (1996).

[54] CURIO, C., BREIDT, M., KLEINER, M., VUONG, Q. C., GIESE, M. A., AND BÜLTHOFF, H. H. Semantic 3d motion retargeting for facial animation. In *Symposium on Applied Perception in Graphics and Visualization* (2006).

[55] DECARLO, D., AND METAXAS, D. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Computer Vision and Pattern Recognition* (1996).

[56] DECARLO, D., AND METAXAS, D. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision* (2000).

[57] DENG, B., BOUAZIZ, S., DEUSS, M., KASPAR, A., SCHWARTZBURG,

Y., AND PAULY, M. Interactive design exploration for constrained meshes. *Computer-Aided Design* (2014).

[58] DENG, B., BOUAZIZ, S., DEUSS, M., ZHANG, J., SCHWARTZBURG, Y., AND PAULY, M. Exploring local modifications for constrained meshes. *Computer Graphics Forum* (2013).

[59] DENG, Z., CHIANG, P.-Y., FOX, P., AND NEUMANN, U. Animating blendshape faces by cross-mapping motion capture data. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2006).

[60] DESBRUN, M., SCHRÖDER, P., AND BARR, A. Interactive animation of structured deformable objects. In *Graphics Interface* (1999).

[61] DIMITRIJEVIC, M., ILIC, S., AND FUA, P. Accurate face models from uncalibrated and ill-lit video sequences. In *Computer Vision and Pattern Recognition* (2004).

[62] DRUCKER, H., BURGES, C. J. C., KAUFMAN, L., SMOLA, A. J., AND VAPNIK, V. Support vector regression machines. In *Conference on Neural Information Processing Systems* (1996).

[63] EK, C. *Shared Gaussian Process Latent Variable Models*. PhD thesis, Oxford Brookes University, 2009.

[64] EKMAN, P., AND FRIESEN, W. *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, 1978.

[65] ESSA, I., BASU, S., DARRELL, T., AND PENTLAND, A. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proc. Computer Animation* (1996).

[66] FACESHIFT, 2015. http://www.faceshift.com.

[67] FOX, J. *An R and S-Plus companion to applied regression*. Sage, 2002.

[68] FRATARCANGELI, M. Position-based facial animation synthesis. *Comput. Animat. Virtual Worlds* (2012).

[69] FU, W. J. Penalized Regressions: The Bridge versus the Lasso. *J. Comp. Graph. Stat.* (1998).

[70] FURUKAWA, Y., AND PONCE, J. Dense 3d motion capture for human faces. In

Computer Vision and Pattern Recognition (2009).

[71] GANDER, W., AND HREBICEK, J. *Solving Problems in Scientific Computing Using Maple and MATLAB*. Springer-Verlag New York, 1995.

[72] GARG, A., GRINSPUN, E., WARDETZKY, M., AND ZORIN, D. Cubic shells. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2007).

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

[74] GEIGER, A., URTASUN, R., AND DARRELL, T. Rank priors for continuous non-linear dimensionality reduction. In *Computer Vision and Pattern Recognition* (2009).

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

[76] GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. Efficient simulation of inextensible cloth. *ACM Trans. Graph.* (2007).

[77] GOULD, N. I. M. On the accurate determination of search directions for simple differentiable penalty functions. *IMA Journal of Numerical Analysis* (1986).

[78] GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. Discrete shells. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2003).

[79] GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. Style-based inverse kinematics. *ACM Trans. Graph.* (2004).

[80] GUENTER, B., GRIMM, C., WOOD, D., MALVAR, H., AND PIGHIN, F. Making faces. *IEEE Computer Graphics and Applications* (1993).

[81] GUYOMARC'H, P., SANTOS, F., DUTAILLY, B., AND COQUEUGNIOT, H. Facial soft tissue depths in french adults: Variability, specificity and estimation. *Forensic science international* (2013).

[82] HÄGGSTRÖM, O. Interactive real time cloth simulation with adaptive level of detail. Master's thesis, Umea University, 2009.

[83] HAHN, F., MARTIN, S., THOMASZEWSKI, B., SUMNER, R., COROS, S., AND GROSS, M. Rig-space physics. *ACM Trans. Graph.* (2012).

[84] HAHN, F., THOMASZEWSKI, B., COROS, S., SUMNER, R. W., AND GROSS, M. Efficient simulation of secondary motion in rig-space. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2013).

[85] HAIRER, E., LUBICH, C., AND WANNER, G. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2002.

[86] HAM, J. H., LEE, D. D., AND SAUL, L. K. Semisupervised alignment of manifolds. In *Proc. of the 10th International Workshop on Artificial Intelligence and Statistics* (2005).

[87] HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. Asynchronous contact mechanics. In *ACM Trans. Graph.* (2009).

[88] HECHT, F., LEE, Y. J., SHEWCHUK, J. R., AND O'BRIEN, J. F. Updated sparse cholesky factors for corotational elastodynamics. *ACM Trans. Graph.* (2012).

[89] HERNANDEZ, F., CIRIO, G., PEREZ, A. G., AND OTADUY, M. A. Anisotropic strain limiting. In *Proc. of Congreso Español de Informática Gráfica* (2013).

[90] HORN, B. Closed-form solution of absolute orientation using unit quaternions. *J. of the Opt. Society of America A* (1987).

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

[92] HUGHES, T. J. R. *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[93] HWANG, H.-S., PARK, M.-K., LEE, W.-J., CHO, J.-H., KIM, B.-K., AND WILKINSON, C. M. Facial soft tissue thickness database for craniofacial reconstruction in korean adults. *Journal of forensic sciences* (2012).

[94] ICHIM, A. E., BOUAZIZ, S., AND PAULY, M. Dynamic facial avatar creation using handheld cameras. *ACM Trans. Graph.* (2015).

[95] IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. Generalizing motion edits with

gaussian processes. *ACM Trans. Graph.* (2009).

[96] IRVING, G., TERAN, J., AND FEDKIW, R. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2004).

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

[98] JONES, B., POPOVIC, J., MCCANN, J., LI, W., AND BARGTEIL, A. Dynamic sprites. In *International Conference on Motion in Games* (2013).

[99] KHAREVYCH, L., YANG, W., TONG, Y., KANSO, E., MARSDEN, J. E., SCHRÖDER, P., AND DESBRUN, M. Geometric, variational integrators for computer animation. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2006).

[100] KHOLGADE, N., MATTHEWS, I., AND SHEIKH, Y. Content retargeting using parameter-parallel facial layers. *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2011).

[101] KOYAMA, Y., TAKAYAMA, K., UMETANI, N., AND IGARASHI, T. Real-time example-based elastic deformation. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2012).

[102] LAU, M., CHAI, J., XU, Y.-Q., AND SHUM, H.-Y. Face poser: interactive modeling of 3d facial expressions using model priors. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2007).

[103] LAWRENCE, N., SEEGER, M., AND HERBRICH, R. Fast sparse gaussian process methods: The informative vector machine. In *Conference on Neural Information Processing Systems* (2003).

[104] LAWRENCE, N. D. Gaussian process latent variable models for visualisation of high dimensional data. In *Conference on Neural Information Processing Systems* (2004).

[105] LAWRENCE, N. D. Learning for larger datasets with the gaussian process latent variable model. In *Proc. of Int. Workshop on Artificial Intelligence and Statistics* (2007).

[106] LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. Comprehensive biomechan-

ical modeling and simulation of the upper body. *ACM Trans. Graph.* (2009).

[107] LEVY, B., AND ZHANG, R. H. Spectral geometry processing. In *ACM SIGGRAPH Courses* (2010).

[108] LEWIS, J. P., ANJYO, K., RHEE, T., ZHANG, M., PIGHIN, F., AND DENG, Z. Practice and Theory of Blendshape Facial Models. In *Eurographics State of the Art Reports* (2014), S. Lefebvre and M. Spagnuolo, Eds.

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

[110] LI, H., ROIVAINEN, P., AND FORCHEIMER, R. 3-d motion estimation in model-based facial image coding. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (1993).

[111] LI, H., WEISE, T., AND PAULY, M. Example-based facial rigging. *ACM Trans. Graph.* (2010).

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

[113] LI, J., XU, W., CHENG, Z., XU, K., AND KLEIN, R. Lightweight wrinkle synthesis for 3d facial modeling and animation. *Computer-Aided Design* (2015).

[114] LIN, I.-C., AND OUHYOUNG, M. Mirror mocap: Automatic and efficient capture of dense 3d facial motion parameters from video. *The Visual Computer* (2005).

[115] LIU, T., BARGTEIL, A. W., O'BRIEN, J. F., AND KAVAN, L. Fast simulation of mass-spring systems. *ACM Trans. Graph.* (2013).

[116] LOU, H., AND CHAI, J. Example-based human motion denoising. *IEEE Trans. on Visualization and Computer Graphics* (2010).

[117] LU, P., NOCEDAL, J., ZHU, C., BYRD, R. H., AND BYRD, R. H. A limited-memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* (1994).

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

Trans. Graph. (2008).

[119] MACKLIN, M., AND MÜLLER, M. Position based fluids. *ACM Trans. Graph.* (2013).

[120] MADSEN, K., NIELSEN, H., AND TINGLEFF, O. Methods for non-linear least squares problems. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

[121] MARJANOVIC, G., AND SOLO, V. On ℓ_q optimization and matrix completion. *IEEE Trans. Signal Process.* (2012).

[122] MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. Example-based elastic materials. In *ACM Trans. Graph.* (2011).

[123] MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.* (2011).

[124] MCLACHLAN, G. J., AND KRISHNAN, T. *The EM Algorithm and Extensions*. Wiley-Interscience, 1996.

[125] MIRZA, M., AND BOYER, K. Performance evaluation of a class of m-estimators for surface parameter estimation in noisy range data. *IEEE Transactions on Robotics and Automation* (1993).

[126] MOLLER, M. F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks* (1993).

[127] MORI, M., MACDORMAN, K. F., AND KAGEKI, N. The uncanny valley [from the field]. *Robotics & Automation Magazine, IEEE* (2012).

[128] MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. Position based dynamics. *J. Vis. Comun. Image Represent.* (2007).

[129] MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. Meshless deformations based on shape matching. In *ACM Trans. Graph.* (2005).

[130] MYLES, A., AND ZORIN, D. Global parametrization by incremental flattening. *ACM Trans. Graph.* (2012).

[131] MÜLLER, M., KIM, T.-Y., AND CHENTANEZ, N. Fast simulation of inextensible hair and fur. In *Workshop on Virtual Reality Interaction and Physical*

Simulation (2012), J. Bender, A. Kuijper, D. W. Fellner, and E. Guerin, Eds.

[132] NARAIN, R., SAMII, A., AND O'BRIEN, J. F. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.* (2012).

[133] NAVARATNAM, R., FITZGIBBON, A. W., AND CIPOLLA, R. The joint manifold model for semi-supervised multi-valued regression. In *International Conference on Computer Vision* (2007).

[134] NOCEDAL, J., AND WRIGHT, S. J. *Numerical optimization*. Springer Verlag, 2006.

[135] NOH, J.-Y., AND NEUMANN, U. Expression cloning. In *Proc. of ACM SIGGRAPH* (2001).

[136] OAT, C. Animated wrinkle maps. In *ACM SIGGRAPH 2007 courses* (2007).

[137] PARIKH, N., AND BOYD, S. Proximal Algorithms. *Found. and Trends in Optimization* (2013).

[138] PECKMANN, T. R., HARRIS, M., HUCULAK, M., PRINGLE, A., AND FOURNIER, M. In vivo facial tissue depth for canadian mi'kmaq adults: A case study from nova scotia, canada. *Journal of forensic and legal medicine* (2015).

[139] PÉREZ, P., GANGNET, M., AND BLAKE, A. Poisson image editing. *ACM Trans. Graph.* (2003).

[140] PIGHIN, F., AND LEWIS, J. P. Performance-driven facial animation. In *ACM SIGGRAPH Courses* (2006).

[141] PIGHIN, F., SZELISKI, R., AND SALESIN, D. Resynthesizing facial animation through 3d model-based tracking. *International Conference on Computer Vision* (1999).

[142] POTTMANN, H., HUANG, Q.-X., YANG, Y.-L., AND HU, S.-M. Geometry and convergence analysis of algorithms for registration of 3D shapes. *International Journal of Computer Vision* (2006).

[143] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[144] PROVOT, X. Deformation constraints in a mass-spring model to describe rigid

cloth behavior. In *Graphics Interface* (1995).

[145] RASMUSSEN, C. E., AND WILLIAMS, C. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[146] RHODIN, H., TOMPKIN, J., IN KIM, K., VARANASI, K., SEIDEL, H.-P., AND THEOBALT, C. Interactive motion mapping for real-time character control. In *Computer Graphics Forum* (2014).

[147] RIVERS, A., AND JAMES, D. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.* (2007).

[148] ROBERTS, S. Control chart tests based on geometric moving averages. In *Technometrics* (1959).

[149] ROWEIS, S. T., AND SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science* (2000).

[150] RUSINKIEWICZ, S., AND LEVOY, M. Efficient variants of the ICP algorithm. *International Conference on 3D Digital Imaging and Modeling* (2001).

[151] SALZMANN, M., EK, C. H., URTASUN, R., AND DARRELL, T. Factorized orthogonal latent spaces. *Journal of Machine Learning Research* (2010).

[152] SARAGIH, J. M., LUCEY, S., AND COHN, J. F. Real-time avatar animation from a single image. In *International Conference on Automatic Face and Gesture Recognition* (2011).

[153] SELLE, A., LENTINE, M., AND FEDKIW, R. A mass spring model for hair simulation. *ACM Trans. Graph.* (2008).

[154] SEOL, Y., LEWIS, J., SEO, J., CHOI, B., ANJYO, K., AND NOH, J. Spacetime expression cloning for blendshapes. *ACM Trans. Graph.* (2012).

[155] SHI, F., WU, H.-T., TONG, X., AND CHAI, J. Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM Trans. Graph.* (2014).

[156] SIFAKIS, E., AND BARBIC, J. Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses* (2012).

[157] SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph.*

(2005).

[158] SIFAKIS, E., SELLE, A., ROBINSON-MOSHER, A., AND FEDKIW, R. Simulating speech with a physics-based facial muscle model. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2006).

[159] SONG, J., CHOI, B., SEOL, Y., AND NOH, J. Characteristic facial retargeting. *Computer Animation and Virtual Worlds* (2011).

[160] SORKINE, O., AND ALEXA, M. As-rigid-as-possible surface modeling. In *Computer Graphics Forum* (2007).

[161] STAM, J. Nucleus: towards a unified dynamics solver for computer graphics. In *IEEE Int. Conf. on CAD and Comput. Graph.* (2009).

[162] STERN, A., AND GRINSPUN, E. Implicit-explicit variational integration of highly oscillatory problems. *Multiscale Modeling & Simulation* (2009).

[163] SU, J., SHETH, R., AND FEDKIW, R. Energy conservation for the simulation of deformable bodies. *Trans. on Visualization and Computer Graphics* (2013).

[164] SUMNER, R. W., AND POPOVIĆ, J. Deformation transfer for triangle meshes. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2004).

[165] SUNG, H. G. *Gaussian mixture regression and classification*. PhD thesis, Rice University, 2004.

[166] TAGLIASACCHI, A., SCHRÖDER, M., TKACH, A., BOUAZIZ, S., BOTSCH, M., AND PAULY, M. Robust articulated-ICP for real-time hand tracking. *Computer Graphics Forum* (2015).

[167] TENA, J. R., TORRE, F. D. L., AND MATTHEWS, I. Interactive region-based linear 3d face models. *ACM Trans. Graph.* (2011).

[168] TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2005).

[169] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. Elastically deformable models. In *Computer Graphics (Proc. SIGGRAPH)* (1987).

[170] TERZOPOULOS, D., AND WATERS, K. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation*

(1990).

[171] THOMAS, S., AND CHAN, Y. A simple approach for the estimation of circular arc center and its radius. *Computer Vision, Graphics, and Image Processing* (1989).

[172] THOMASZEWSKI, B., PABST, S., AND STRASSER, W. Continuum-based strain limiting. In *Computer Graphics Forum* (2009).

[173] TIPPING, M. E., AND BISHOP, C. M. Mixtures of probabilistic principal component analyzers. *Neural Computation* (1999).

[174] TIPPING, M. E., AND BISHOP, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B* (1999).

[175] UMEYAMA, S. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (1991).

[176] URTASUN, R., FLEET, D. J., AND FUA, P. 3d people tracking with gaussian process dynamical models. In *Computer Vision and Pattern Recognition* (2006).

[177] URTASUN, R., FLEET, D. J., AND LAWRENCE, N. D. Modeling human locomotion with topologically constrained latent variable models. In *Proc. Conf. on Human Motion* (2007).

[178] VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph.* (2012).

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

[180] VERBEEK, J. J., AND VLASSIS, N. Gaussian fields for semi-supervised regression and correspondence learning. *Pattern Recogn.* (2006).

[181] VERBOON, P. *Majorization with iteratively reweighted least squares: a general approach to optimize a class of resistant loss functions*. University of Leiden, 1990.

[182] VIOLA, P., AND JONES, M. Rapid object detection using a boosted cascade of

simple features. In *Computer Vision and Pattern Recognition* (2001).

[183] VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. Face transfer with multilinear models. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2005).

[184] WANG, C., AND MAHADEVAN, S. A general framework for manifold alignment. In *AAAI Symposium on Manifold Learning and its Applications* (2009).

[185] WANG, H., O'BRIEN, J., AND RAMAMOORTHI, R. Multi-resolution isotropic strain limiting. *ACM Trans. Graph.* (2010).

[186] WANG, J. M., FLEET, D. J., AND HERTZMANN, A. Gaussian process dynamical models for human motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2008).

[187] WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. Realtime performance-based facial animation. *ACM Trans. Graph.* (2011).

[188] WEISE, T., LEIBE, B., AND GOOL, L. V. Accurate and robust registration for in-hand modeling. In *Computer Vision and Pattern Recognition* (2008).

[189] WEISE, T., LI, H., GOOL, L. V., AND PAULY, M. Face/off: Live facial puppetry. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2009).

[190] WILLIAMS, C. K. I., AND RASMUSSEN, C. E. Gaussian processes for regression. In *Conference on Neural Information Processing Systems* (1995).

[191] WILLIAMS, L. Performance-driven facial animation. In *Computer Graphics (Proc. SIGGRAPH)* (1990).

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

[193] YAMANE, K., ARIKI, Y., AND HODGINS, J. Animating non-humanoid characters with human motion data. In *Proceedings of the EG/SIGGRAPH Symposium on Computer Animation* (2010).

[194] YANG, G., XU, X., AND ZHANG, J. Manifold alignment via local tangent space alignment. In *Proc. of Int. Conf. on Comp Sc. and Soft. Eng.* (2008).

[195] ZEILER, M. D., TAYLOR, G. W., SIGAL, L., MATTHEWS, I., AND FERGUS, R. Facial expression transfer with input-output temporal restricted boltzmann

machines. In *Conference on Neural Information Processing Systems* (2011).

[196] ZELL, E., AND BOTSCH, M. Elastiface: Matching and blending textured faces. In *Symposium on Non-Photorealistic Animation and Rendering* (2013).

[197] ZHAI, D., LI, B., CHANG, H., SHAN, S., CHEN, X., AND GAO, W. Manifold alignment via corresponding projections. In *British Machine Vision Conference* (2010).

[198] ZHANG, J., DENG, B., LIU, Z., PATANÈ, G., BOUAZIZ, S., HORMANN, K., AND LIU, L. Local barycentric coordinates. *ACM Trans. Graph.* (2014).

[199] ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph.* (2004).

[200] ZHANG, Y., PRAKASH, E. C., AND SUNG, E. Efficient modeling of an anatomy-based face and fast 3d facial expression synthesis. *Computer Graphics Forum* (2003).

[201] ZHU, L., HU, X., AND KAVAN, L. Adaptable anatomical models for realistic bone motion reconstruction. *Computer Graphics Forum* (2015).

[202] ZHU, X., LAFFERTY, J., AND GHAHRAMANI, Z. Semi-supervised learning: From gaussian fields to gaussian processes. Tech. rep., School of CS, CMU, 2003.

Appendix A

Robust Optimization

A.1. Augmented Lagrangian Method (ALM)

We briefly discuss constrained optimization methods to provide a suitable background for the optimization approach taken in this paper. Consider the *equality-constrained* optimization problem having $x \in \mathbb{R}^k$:

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to } c_i(\mathbf{x}) = 0 \quad i = 1 \dots n. \tag{A.1}$$

Given a vector of Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^n$ and $\mu \in \mathbb{R}^+$, we can transform the problem into an *unconstrained* optimization expressed by the *augmented Lagrangian* function

$$\mathscr{L}_A(\mathbf{x}, \boldsymbol{\lambda}, \mu) = f(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i c_i(\mathbf{x}) + \frac{\mu}{2} \sum_{i=1}^{n} c_i^2(\mathbf{x}). \tag{A.2}$$

A close inspection of this expression reveals that it is a combination of a *linear* penalty typical of *dual ascent methods* [137, Ch. 2] and a *quadratic* one from *penalty methods* [134, Ch. 17]

$$\mathscr{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(x) + \sum_{i=1}^{n} \lambda_i c_i(\mathbf{x}) \quad Q(\mathbf{x}, \mu) = f(\mathbf{x}) + \frac{\mu}{2} \sum_{i=1}^{n} c_i^2(\mathbf{x}).$$

Intuitively, dual ascent methods exploit the fact that the Lagrange dual function $g(\boldsymbol{\lambda}) = \min_{\mathbf{x}} \mathscr{L}(\mathbf{x}, \boldsymbol{\lambda})$ is a *lower bound* on the primal problem [36, pp. 216]. A gradient ascent w.r.t. $\boldsymbol{\lambda}$ of appropriate size gradually closes the *duality gap*, effectively optimizing Equation A.1 as shown in Equation A.4. Conversely, penalty methods gradually increase the penalty parameter μ, resulting in progressively increased constrain satis-

faction. Unfortunately, both approaches suffer severe limitations; dual ascent methods place strong assumptions on the problem, like strict convexity and boundedness of $g(.)$; penalty methods offer limited control on constraint satisfaction, running into numerical issues as μ is increased to very large values. These issues are addressed by the augmented Lagrangian approach; the optimization involves an algorithm strongly resembling dual ascent:

$$\textbf{Step 1:} \qquad \mathbf{x}^{t+1} := \arg\min_{x} \mathscr{L}(\mathbf{x}, \boldsymbol{\lambda}^t, \mu) \tag{A.3}$$

$$\textbf{Step 2:} \qquad \lambda_i^{t+1} := \lambda_i + \mu\, c_i(\mathbf{x}^{t+1}) \quad i = 1 \dots n \tag{A.4}$$

providing an asymptotic approximate constraint satisfaction in the form [134, Thm 17.2]

$$c_i(x) \approx (\lambda_i^* - \lambda_i)/\mu. \tag{A.5}$$

Consequently, constraints can be satisfied by either increasing μ, or alternatively by providing multipliers λ having values close to the optimal λ^*; note that this is achieved by the Lagrange multiplier ascent step inherited from dual ascent; see Equation A.4.

A.2. Alternating Direction Method of Multipliers

The ADMM method is an extension of the augmented Lagrangian method to optimize a compound problem $f(\mathbf{x}) = \tilde{f}(\tilde{\mathbf{x}}) + \hat{f}(\hat{\mathbf{x}})$ by decoupling it into simpler sub-problems [35]. Under appropriate conditions [137, Ch. 3], as the optimizer is separated in $\mathbf{x} = [\tilde{\mathbf{x}}, \hat{\mathbf{x}}]$, the problem can be approached by iteratively solving the following three steps:

$$\textbf{Step 1:} \qquad \tilde{\mathbf{x}}^{t+1} := \arg\min_{\tilde{\mathbf{x}}} \mathscr{L}([\tilde{\mathbf{x}}, \hat{\mathbf{x}}^t], \boldsymbol{\lambda}^t, \mu) \tag{A.6}$$

$$\textbf{Step 2:} \qquad \hat{\mathbf{x}}^{t+1} := \arg\min_{\hat{\mathbf{x}}} \mathscr{L}([\tilde{\mathbf{x}}^{t+1}, \hat{\mathbf{x}}], \boldsymbol{\lambda}^t, \mu) \tag{A.7}$$

$$\textbf{Step 3:} \qquad \lambda_i^{t+1} := \lambda_i^t + \mu\, c_i(\mathbf{x}^{t+1}) \quad i = 1 \dots n \tag{A.8}$$

Note that alternately looping over *Steps 1/2* until convergence would simply correspond to a *block coordinate descent* decomposition of the joint minimization in Equation A.3. Consequently, ADMM can be interpreted as applying a single step of block coordinate descent optimization applied to the augmented Lagrangian problem.

A.3. Shrink operator for $f(z) = \|\mathbf{z}\|_2^p + \frac{\mu}{2}\|\mathbf{z}-\mathbf{h}\|_2^2$

The minimization of $f(z)$ can be simplified into a scalar problem by noticing that $\mathbf{z}$ can be expressed as $\mathbf{z} = \alpha\mathbf{h}$ (see Appendix A.4) leading to

$$\min_{\alpha} \|\alpha\mathbf{h}\|_2^p + \tfrac{\mu}{2}\|\alpha\mathbf{h}-\mathbf{h}\|_2^2 = \min_{\alpha} \|\mathbf{h}\|_2^{q-2}|\alpha|^p + \tfrac{\mu}{2}|\alpha-1|^2. \tag{A.9}$$

As proven in [121], the optimal α^* is given by:

$$\alpha^* = \begin{cases} 0 & \text{if } \|\mathbf{h}\|_2 \le \tilde{h} \\ \beta & \text{if } \|\mathbf{h}\|_2 > \tilde{h} \end{cases} \tag{A.10}$$

where the threshold $\tilde{h}$ is computed as

$$\tilde{h} = \alpha_a + \tfrac{p}{\mu}\alpha_a^{p-1}, \quad \alpha_a = \left(\tfrac{2}{\mu}(1-p)\right)^{\frac{1}{2-p}}. \tag{A.11}$$

β is found by using the following update scheme

$$\beta_{t+1} = 1 - \tfrac{p}{\mu}\|\mathbf{h}\|_2^{p-2}\beta_t^{p-1} \tag{A.12}$$

by initializing $\beta_0 \in [\alpha_a\|\mathbf{h}\|_2^{-1}, 1]$. Similarly to [121], we noticed that this scheme converges in two or three iterations. The optimal $\mathbf{z}$ is then given by $\mathbf{z}^* = \alpha^*\mathbf{h}$.

A.4. Scalar version of $f(z) = \|\mathbf{z}\|_2^p + \frac{\mu}{2}\|\mathbf{z}-\mathbf{h}\|_2^2$

The minimization of $f(z)$ can be re-interpreted into a simpler scalar problem where the optimal solution can be expressed as $\mathbf{z}^* = \alpha^*\mathbf{h}$, $\alpha \in \mathbb{R}$. We verify this by expressing $\mathbf{z}$ as a linear combination of a component $\alpha\mathbf{h}$ lying in the space defined by $\mathbf{h}$, and κ orthogonal to it, that is, $\kappa^T\mathbf{h} = 0$. All that is necessary is to prove that $\forall\kappa \in \mathbb{R}^3, f(\alpha\mathbf{h}+\kappa) \ge f(\alpha\mathbf{h})$. This can be verified by checking that both of the two following inequalities $\|\alpha\mathbf{h}+\kappa\|_2^p \ge \|\alpha\mathbf{h}\|_2^p$ and $\|\alpha\mathbf{h}+\kappa-\mathbf{h}\|_2^2 \ge \|\alpha\mathbf{h}-\mathbf{h}\|_2^2$ hold. We can easily convert the first of these into one involving quadratic exponents only. We first raise both sides to the power $2/p$ and then expand the norm as an inner product obtaining $(\alpha\mathbf{h}+\kappa)^t(\alpha\mathbf{h}+\kappa) \ge \|\alpha\mathbf{h}\|^2$. As $\mathbf{h}$ and κ are orthogonal we get $\|\alpha\mathbf{h}\|_2^2 + \|\kappa\|_2^2 \ge \|\alpha\mathbf{h}\|_2^2$; this is always verified as $\|\kappa\|_2^2 \ge 0$. To verify the second expression, we again expand the norm as an inner product, then, after removing $\|\mathbf{h}\|_2^2$ from both sides, we obtain $\|\alpha\mathbf{h}+\kappa\|_2^2 - (\alpha\mathbf{h}+\kappa)^T\mathbf{h} \ge \|\alpha\mathbf{h}\|_2^2 - (\alpha\mathbf{h})^T\mathbf{h}$. Similarly to what we did before, we expand the norm and exploit orthogonality, simplifying the expression to $\|\kappa\|^2 \ge 0$, that, again, is always verified.

Appendix B

Face Tracking

B.1. Gradients

We derive the gradients for the optimization of Equation 4.13. The energy terms for geometry registration E_{geo} and optical flow E_{im} can both be written in the form

$$f(\mathbf{x}) = \frac{||A\mathbf{x} - b||^2}{2\sigma^2} \tag{B.1}$$

hence the gradients can easily be computed analytically as

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \frac{A^T(A\mathbf{x} - b)}{\sigma^2}. \tag{B.2}$$

The prior term is of the form

$$E_{\text{prior}} = -\ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}, X_n | \text{—}_k, \Sigma_k), \tag{B.3}$$

where Σ_k is the covariance matrix. The Gaussians $\mathcal{N}(\mathbf{x}, X_n | \text{—}_k, \Sigma_k)$ model the combined distribution of the current blendshape vector $\mathbf{x} \in \mathbb{R}^m$ and the n previous vectors X_n, hence the Σ_k are matrices of dimension $(n+1)m \times (n+1)m$. Since we are only interested in the gradient with respect to $\mathbf{x}$, we can discard all components that do not depend on this variable. We split the mean vectors as $\mu_k = (\mu_k^1, \mu_k^n)$, corresponding to $\mathbf{x}$

and X_n, respectively. We can write the inverse of Σ_k as

$$\Sigma_k^{-1} = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] = \left[\begin{array}{c|c} (m \times m) & (m \times nm) \\ \hline (nm \times m) & (nm \times nm) \end{array}\right] \tag{B.4}$$

with $B_k = C_k^T$. We then obtain for the gradient of the prior energy term

$$\frac{\partial E_{\text{prior}}}{\partial \mathbf{x}} = \tag{B.5}$$

$$\frac{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}, X_n | —_k, \Sigma_k)[(\mathbf{x} - —_k^1)^T A_k + (X_n - —_k^n)^T C_k]}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}, X_n | —_k, \Sigma_k)}.$$

The complete gradient is the sum of the three energy gradients derived above

$$g(\mathbf{x}) = \frac{\partial E_{\text{geo}}}{\partial \mathbf{x}} + \frac{\partial E_{\text{im}}}{\partial \mathbf{x}} + \frac{\partial E_{\text{prior}}}{\partial \mathbf{x}}. \tag{B.6}$$

Appendix C

Face Modeling

C.1. Expression Transfer

In this section we describe our formulation of deformation transfer that we use to deform the neutral expression $\mathbf{b}_0$ to an expression $\mathbf{b}_i$ by transferring the deformation from the neutral expression $\mathbf{b}_0^*$ to the expression $\mathbf{b}_i^*$ of a template model. We first compute the set of affine transformations $\{\mathbf{S}_1^*,\ldots,\mathbf{S}_p^*\}$ deforming the p triangles of $\mathbf{b}_0^*$ to the corresponding ones of $\mathbf{b}_i^*$. As an affine transformation is not fully characterized by the deformation of a triangle we instead use tetrahedrons to compute the affine transformations where the fourth vertex is added in the direction perpendicular to the triangle [164]. The affine transformation $\mathbf{S}^*$ of a tetrahedron $\{\mathbf{v}_{01}^*,\mathbf{v}_{02}^*,\mathbf{v}_{03}^*,\mathbf{v}_{04}^*\}$ of $\mathbf{b}_0^*$ to the corresponding tetrahedron $\{\mathbf{v}_{i1}^*,\mathbf{v}_{i2}^*,\mathbf{v}_{i3}^*,\mathbf{v}_{i4}^*\}$ of $\mathbf{b}_i^*$ is computed as $\mathbf{S}^* = \mathbf{S}_i^*\mathbf{S}_0^{*(-1)}$, where $\mathbf{S}_i^* = [\mathbf{v}_{i2}^*-\mathbf{v}_{i1}^*,\mathbf{v}_{i3}^*-\mathbf{v}_{i1}^*,\mathbf{v}_{i4}^*-\mathbf{v}_{i1}^*]$ and $\mathbf{S}_0^* = [\mathbf{v}_{02}^*-\mathbf{v}_{01}^*,\mathbf{v}_{03}^*-\mathbf{v}_{01}^*,\mathbf{v}_{04}^*-\mathbf{v}_{01}^*]$. The deformation transfer problem can then be formulated as

$$\arg\min_{\mathbf{b}_i} \sum_{j=1}^{p} \|\mathbf{S}_j^*\mathbf{t}_{0j} - \mathbf{t}_{ij}\|_2^2 + \mu\|\mathbf{F}(\mathbf{b}_i - \mathbf{b}_0)\|_2^2,$$

where $\mathbf{t}_{ij} = [\mathbf{v}_{i2}-\mathbf{v}_{i1},\mathbf{v}_{i3}-\mathbf{v}_{i1}]_j$ represents two edges of the triangle j of $\mathbf{b}_i$, $\mathbf{F}$ is a diagonal matrix defining the vertices that need to be fixed between $\mathbf{b}_0$ and $\mathbf{b}_i$ shown in blue on the left, and μ is a weight factor that we fix to $\mu = 100$ for all our computations. This optimization can be reformulated as

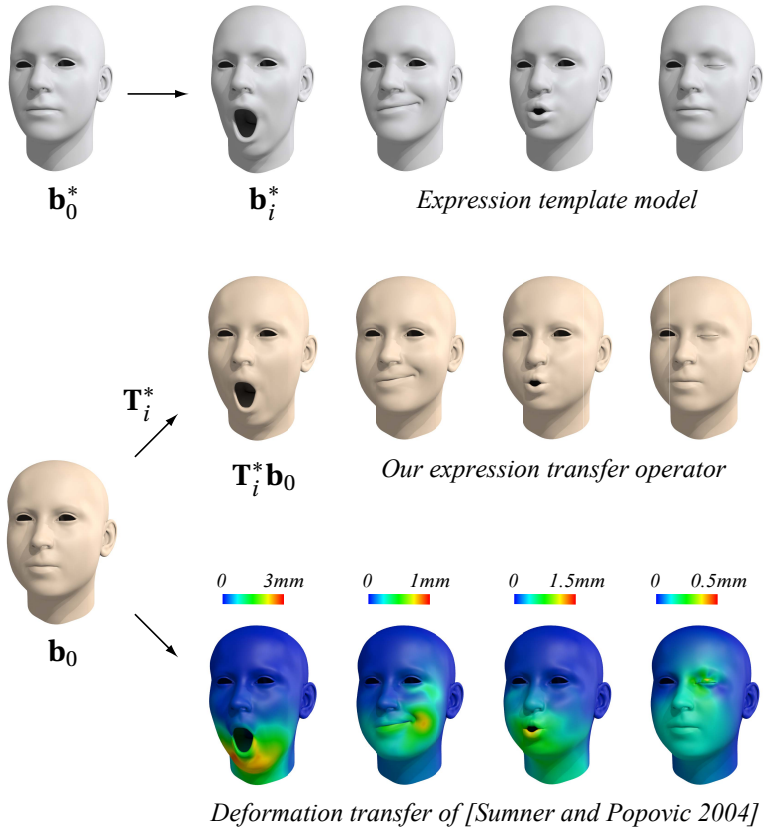


Figure C.1.: Expression transfer from a template model (top) to the user-specific model (middle). Our approach gives comparable results to the method of [Sumner and Popovic 2004] (bottom), but can express the transfer operation as a linear transformation.

$$\arg\min_{\mathbf{b}_i} \|\mathbf{H}_i^*\mathbf{G}\mathbf{b}_0 - \mathbf{G}\mathbf{b}_i\|_2^2 + \mu\|\mathbf{F}(\mathbf{b}_i - \mathbf{b}_0)\|_2^2, \tag{C.1}$$

where $\mathbf{G}$ is a matrix transforming vertices to edges and $\mathbf{H}_i^*$ is a matrix containing the affine transformations mapping each edge of the template neutral expression $\mathbf{b}_0^*$ to the template expression $\mathbf{b}_i^*$. The optimal solution of this problem is $\mathbf{b}_i = \mathbf{T}_i^*\mathbf{b}_0$ where $\mathbf{T}_i^* = (\mathbf{G}^T\mathbf{G}+\mathbf{F})^{-1}(\mathbf{G}^T\mathbf{H}_i^*\mathbf{G}+\mathbf{F})$ is a linear operator defining the transformation from the neutral expression $\mathbf{b}_0$ to an expression $\mathbf{b}_i$ that matches the transformation of $\mathbf{b}_0^*$ to $\mathbf{b}_i^*$. The main difference of our formulation compared to previous approaches proposed in [164] or [27] is that $\mathbf{T}_i^*$ does not depend on $\mathbf{b}_0$. Effectively, our formulation uses a graph Laplacian instead of a cotan Laplacian, which avoids the weighting factor of triangle areas of $\mathbf{b}_0$ in $\mathbf{T}_i^*$, which would make $\mathbf{T}_i^*\mathbf{b}_0$ non-linear with respect to $\mathbf{b}_0$. Since our face

meshes are uniformly tessellated, this simplification has little effect on the resulting deformations (see Figure C.1). However, it allows the DEM refinement optimization to be formulated as a simple linear system (see Equation 5.4) which makes it fast and robust to solve.

Appendix D

Physics-Based Animation

D.1. Local Solves

Strain. When minimizing over $\mathbf{T}$ while keeping $\mathbf{q}$ fixed in the local step

$$\min_{\mathbf{T}} \|\mathbf{X_f}\mathbf{X_g}^{-1} - \mathbf{T}\|_F^2 + \delta_M(\mathbf{T}), \tag{D.1}$$

the optimization can be reformulated as

$$\min_{\hat{\circ}} \|\hat{\circ} - \hat{\circ}^*\|_F^2 \quad \text{s.t.} \quad \sigma_{min} \leq \hat{\circ}^*_{ii} \leq \sigma_{max}, \tag{D.2}$$

where $\mathbf{X_f}\mathbf{X_g}^{-1} = \mathbf{U}\hat{\circ}\mathbf{V}^T$ and $\mathbf{T} = \mathbf{U}\hat{\circ}^*\mathbf{V}^T$. The optimal solution can be computed as $\hat{\circ}^*$ being the singular values $\hat{\circ}$ clamped between σ_{min} and σ_{max}. For tetrahedrons, if $\det(\mathbf{X}_f\mathbf{X}_g^{-1}) < 0$, the last singular value is negated to avoid reflections.

Area and Volume. Similar to the strain constraint the local minimization of the volume constraint can be reformulated as

$$\min_{\hat{\circ}} \|\hat{\circ} - \hat{\circ}^*\|_F^2 \quad \text{s.t.} \quad \sigma_{min} \leq \prod_i \hat{\circ}^*_{ii} \leq \sigma_{max}. \tag{D.3}$$

This problem can be further transformed in

$$\min_{\mathbf{D}} \|\mathbf{D}\|_2^2 \quad \text{s.t.} \quad \prod_i (\hat{\circ}_{ii} + \mathbf{D}_i) = \sigma, \tag{D.4}$$

with ${}^{\circ}_{ii}{}^{*} = {}^{\circ}_{ii} + \mathbf{D}_i$ and where $\sigma = \sigma_{min}$ when $\prod_i {}^{\circ}_{ii}{}^{*} < \sigma_{min}$ and $\sigma = \sigma_{max}$ when $\prod_i {}^{\circ}_{ii}{}^{*} > \sigma_{max}$. This constrained minimization can be solved by iteratively solving a quadratic programming problem by linearizing the constraint leading to a simple update rule

$$\mathbf{D}^{k+1} = \frac{\nabla \mathbf{C}(\mathbf{D}^k)^T \mathbf{D}^k - \mathbf{C}(\mathbf{D}^k)}{\|\nabla \mathbf{C}(\mathbf{D}^k)\|_2^2} \nabla \mathbf{C}(\mathbf{D}^k), \tag{D.5}$$

where $\mathbf{C}(\mathbf{D}) = \prod_i ({}^{\circ}_{ii} + \mathbf{D}_i) - \sigma$.

Example-Based. We solve the optimization

$$\min_{\mathbf{R},\mathbf{w}} \|\mathbf{X_f} \mathbf{X_g}^{-1} - \mathbf{R}\mathbf{X_h}(\mathbf{w})\mathbf{X_g}^{-1}\|_F^2 + \delta_{SO(3)}(\mathbf{R}), \tag{D.6}$$

using a local/global approach by minimizing over $\mathbf{R}$ and $\mathbf{w}$ iteratively. The minimization over $\mathbf{R}$ is solved using SVD following [160] and solving over $\mathbf{w}$ corresponds to solve a simple linear system.

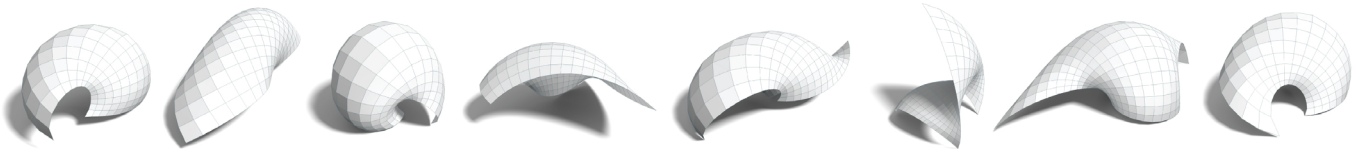
Bending. The local solve of the bending constraint can be formulated as

$$\min_{\mathbf{R}} \|\mathbf{v}_f - \mathbf{R}\mathbf{v}_g\|_2^2 + \delta_{SO(3)}(\mathbf{R}), \tag{D.7}$$

where $\mathbf{v}_f = \mathbf{X}_f \mathbf{c}$ and $\mathbf{v}_g = \mathbf{X}_g \mathbf{c}$. This corresponds in finding a rotation $\mathbf{R}$ such that the rotated vector $\mathbf{v}_g$ matches best the vector $\mathbf{v}_f$. While $\mathbf{R}$ could be found using SVD [160] this problem has an easier closed form solution where $\mathbf{R}\mathbf{v}_g$ can be replaced by $\frac{\mathbf{v}_f \|\mathbf{v}_g\|_2}{\|\mathbf{v}_f\|_2}$.

Sofien Bouaziz

⌂ Chemin des Retraites 11, 1004 Lausanne, Switzerland
🛈 French citizenship. Born in 1986.
@ http://lgg.epfl.ch/~bouaziz
✉ sofien.bouaziz@epfl.ch
☎ (+41) 76 494 32 07



Education

Swiss Federal Institute of Technology in Lausanne (EPFL) — Lausanne, Switzerland
Doctor of Philosophy in Computer Graphics (Advisor: Prof. Mark Pauly)
September 2010 — August 2015

Swiss Federal Institute of Technology in Lausanne (EPFL) — Lausanne, Switzerland
Master of Science in Computer Science
September 2007 — September 2009

Graduate School of Electronics and Electrical Engineering (ESIEE) — Paris, France
Bachelor of Science in Electronics and Electrical Engineering
September 2004 — September 2007

Paris-Est University — Paris, France
Bachelor of Science in Mathematics
September 2004 — September 2007

Startup Company

Faceshift AG — Zurich, Switzerland
Co-Founder
May 2012 — Present

Our software analyzes the face motions, and describes them as a combination of basic expressions, plus head orientation and gaze. This description is then used to animate virtual characters for use in movies or games. With currently 15 employees distributed in the US, the UK, and Switzerland, faceshift provides a unique technology used by some of the most creative companies and individuals. We have an astonishing real time tracking and offline post-processing technology that we provide in a single, convenient application. As a co-founder and a co-author of the software one of my roles is to build the future of the company technology and vision.

Journal Articles

Projection-Based Optimization with Fast Convergence
S. Bouaziz, B. Deng, M. Pauly
Computer Graphics Forum, 2015 (currently in submission)

Robust Articulated-ICP for Real-Time Hand Tracking
A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, M. Pauly
Computer Graphics Forum, 2015

Dynamic 3D Avatar Creation from Hand-held Video Input
A. E. Ichim, S. Bouaziz, M. Pauly
Transactions on Graphics, 2015

Local Barycentric Coordinates
J. Zhang, B. Deng, Z. Liu, G. Patane, S. Bouaziz, K. Hormann, L. Liu
Transactions on Graphics, 2014

Projective Dynamics: Fusing Constraint Projections for Fast Simulation
S. Bouaziz, S. Martin, T. Liu, L. Kavan, M. Pauly
Transactions on Graphics, 2014

Interactive Design Exploration for Constrained Meshes
B. Deng, S. Bouaziz, M. Deuss, A. Kaspar, Y. Schwartzburg, M. Pauly
Computer-Aided Design, 2014

Sparse Iterative Closest Point
S. Bouaziz, A. Tagliasacchi, M. Pauly
Computer Graphics Forum, 2013

Online Modeling For Realtime Facial Animation
S. Bouaziz, Y. Wang, M. Pauly
Transactions on Graphics, 2013

Exploring Local Modifications for Constrained Meshes
B. Deng, S. Bouaziz, M. Deuss, J. Zhang, Y. Schwartzburg, M. Pauly
Computer Graphics Forum, 2013

Shape-Up: Shaping Discrete Geometry with Projections
S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, M. Pauly
Computer Graphics Forum, 2012

Semi-Supervised Dimensionality Reduction for Analyzing High-Dimensional Data with Constraints
S. Yan, S. Bouaziz, D. Lee, J. Barlow
Neurocomputing Journal, 2012

Realtime Performance-Based Facial Animation
T. Weise, S. Bouaziz, H. Li, M. Pauly
Transactions on Graphics, 2011

Conference Papers

The Light-Path Less Traveled
S. Ramalingam, S. Bouaziz, P. Sturm, P. Torr
Computer Vision and Pattern Recognition, 2011

Pose Estimation Using Both Points and Lines for Geo-Localization
S. Ramalingam, S. Bouaziz, P. Sturm
International Conference on Robotics and Automation, 2011

DSP: Robust Semi-Supervised Dimensionality Reduction using Dual Subspace Projections
S. Yan, S. Bouaziz, D. Lee
Web Intelligence, 2010

SKYLINE2GPS: Localization in Urban Canyons Using Omni-Skylines
S. Ramalingam*, S. Bouaziz*, P. Sturm, M. Brand
International Conference on Intelligent Robots and Systems, 2010

Other Publications

Semi-Supervised Facial Animation Retargeting
S. Bouaziz, M. Pauly
EPFL Technical Report #202143, 2014

Faceshift: Real-Time Facial Performance Capture
S. Bouaziz, T. Weise, B. Amberg, M. Pauly
Symposium on Computer Animation, 2012 *(Poster)*

Geolocalization Using Skylines From Omni-Images
S. Ramalingam, S. Bouaziz, P. Sturm, M. Brand
Search in 3D and Video ICCV Workshop, 2009

Geolocalization Using Skylines From Omni-Images
S. Bouaziz
Master Thesis EPFL, 2009

Teaching - Service

Courses

Dynamic 2D/3D Registration
S. Bouaziz, A. Tagliasacchi, M. Pauly
Symposium on Geometry Processing Graduate School, 2015

Dynamic 2D/3D Registration
S. Bouaziz, A. Tagliasacchi, M. Pauly
Eurographics Tutorial, 2014

Dynamic 2D/3D Registration for the Kinect
S. Bouaziz, M. Pauly
SIGGRAPH Course, 2013

Assistant

CS-341 Introduction to Computer Graphics at EPFL, Fall, 2010-2014
CS-446 Digital 3D Geometry Processing at EPFL, Spring, 2011-2012
CS-470 Advance Computer Graphics at EPFL, Fall, 2013

Supervisor

Realtime Physically Plausible Deformation of Articulated Meshes R. G. Schneider
Animation Database for Realtime Motion Reconstruction S. Richert
Face and Hair Reconstruction using RGB-D Cameras A. E. Ichim
Optimization of Scale-Like Structure on Meshes L. Gosmino
Realtime Smoothed Particle Hydrodynamics M. L. F. Hernán
Face Tracking with Constrained Local Models Y. Messerli
Morphable Models for Facial Expressions I. Dewancker
High-Quality Capture of Facial Geometry A. Giroux
Mapping of Facial Representations D. Firmenich
Photorealistic Facial Texture Synthesis A. Tkach
Realtime Realistic Hair Rendering D. Chappuis
Realtime Realistic Skin Rendering D. Chappuis
Robust Optical Flow T. Droxler
Face Hallucination P. H. J. Kim

Reviewer

Short Papers IPC Eurographics 2015, **SIGGRAPH** 2015-2011, **SIGGRAPH Asia** 2015-2011, **Eurographics** 2015-2014, **PG** 2015, **IEEE CG&A** 2013, **TVCJ** 2013, **VR**2012, **ACCV** 2009,

Chair

Short Papers Geometry Session Eurographics 2015

Industry Experience

Adobe Research — Seattle, USA
Research Intern
June 2013 — September 2013

E-ON Software — Paris, France
R&D Engineer
January 2010 — September 2010

Mitsubishi Electric Research Labs — Boston, USA
Research Intern
March 2009 — December 2009

Eugen System — Paris, France
Research Intern
July 2006 — October 2006

US Patents

#13/912,378 **Online Modeling for Realtime Facial Animation**
#13/323,231 **A Method for Facial Animation**
#12/495,655 **Method for Determining a Location From Images Acquired of an Environment with an Omni-Directional Camera**

Prizes

Grant **SNF Early Postdoc Mobility Grant**

Awards **EPFL/Logitech Master Thesis Innovation Award**
ESIEE/Texas Instruments Bachelor Thesis Innovation Award

Honors **Bachelor Degree with Honors (Top of my Class)**
Scientific High School Diploma with Honors

Invited Talks

Proximal Splitting Methods in Computer Graphics Seggau, Austria
International Geometry Workshop
July 11, 2015

Digital Humans Saarbruecken, Germany
Max Planck Institute
June 19, 2015

Digital Humans Haifa, Israel
Technion
May 10, 2015

Digital Humans New York, USA
Cornell Tech
April 22, 2015

Digital Humans Ithaca, USA
Cornell
April 20, 2015

Digital Humans San Francisco, USA
Adobe
April 7, 2015

Digital Humans Zurich, Switzerland
Disney Research
March 4, 2015

The Future of 3D Acquisition and Design Hefei, China
University of Science and Technology of China
December 8, 2014

Digital Humans Zurich, Switzerland
Zurich Minds
December 11, 2013

Realtime Facial Animation Seattle, USA
Adobe
July 12, 2013

Realtime Facial Animation Seattle, USA
Microsoft
July 11, 2013

Realtime Facial Animation Seattle, USA
Google
July 10, 2013

From Research to Industry Lausanne, Switzerland
Swiss Federal Institute of Technology
November 21, 2012

Realtime Face Tracking Lausanne, Switzerland
The Machine Learning Workshop
November 19, 2012

Shaping Meshes with Projections Obergurgl, Austria
International Geometry Workshop
June 21, 2011

Conference Presentations

Projective Dynamics: Fusing Constraint Projections for Fast Simulation Vancouver, Canada
SIGGRAPH
August 14, 2014

Online Modeling For Realtime Facial Animation Anaheim, USA
SIGGRAPH
July 22, 2013

Faceshift: Real-Time Facial Performance Capture Lausanne, Switzerland
Symposium on Computer Animation
July 30, 2012

Shape-Up: Shaping Discrete Geometry with Projections Tallinn, Estonia
Symposium on Geometry Processing
July 17, 2012

Be your Avatar: Real-time Facial Animation using the Kinect Hong Kong, China
SIGGRAPH Asia *(Emerging Technologies)*
December 15, 2011

Pose Estimation Using Both Points and Lines for Geo-Localization Shanghai, China
International Conference on Robotics and Automation
May 12, 2011

SKYLINE2GPS: Localization in Urban Canyons Using Omni-Skylines Taipei, Taiwan
International Conference on Intelligent Robots and Systems
October 20, 2010

Open Source Projects

Shape Op – `http://shapeop.org`
ShapeOp is a C++ library for static and dynamic geometry processing, using a unified framework.

Sparse ICP – `https://github.com/opengp/sparseicp`
Sparse ICP is a C++ library for robust alignment of 3D scans.

LBC – `https://github.com/bldeng/LBC`
LBC is a C++ library for computing local barycentric coordinates.

HTrack – `https://github.com/OpenGP/htrack`
HTrack is a C++ library for hand tracking..

Other Activities

- I like to play piano and guitar, and I enjoy composing music.
- I am a member of **Zurich.Minds** a non-profit foundation promoting inspirational exchange of ideas.