Real-Time Generative Hand Modeling and Tracking

THÈSE N^{O} 8573 (2018)

PRÉSENTÉE LE 30 AOÛT 2018

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

Anastasia TKACH

acceptée sur proposition du jury:

Prof. P. Fua, président du jury
Prof. M. Pauly, Prof. A. Tagliasacchi, directeurs de thèse
Prof. V. Lepetit, rapporteur
Dr G. Pons-Moll , rapporteur
Dr M. Salzmann, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2018

There is no justice in the laws of nature, no term for fairness in the equations of motion.
The Universe is neither evil, nor good, it simply does not care.
The stars don't care, or the Sun, or the sky.

But they don't have to. We care! There is light in the world, and it is us.

— Eliezer Yudkowsky

To my family

Acknowledgements

First of all, I would like to thank my advisors Mark Pauly and Andrea Tagliasacchi.

I am grateful to Mark for hiring me to work on hand tracking. This topic was in demand at the start of my PhD and it is even more so at the time of my graduation. Thanks to Mark, my first project was extremely well chosen. I was not starting from scratch, but working in a team of experienced researchers. The techniques that I learned fueled my entire PhD. Another crucial point that I want to thank for is freedom that I was given. I came to doctoral school with a goal to become an independent researcher which is impossible without some freedom to choose work directions and make mistakes.

I would like to express my deepest gratitude to Andrea Tagliasacchi. I was inspired by his energy, devotion to research and even more so by his long term vision and revolutionary ideas. Also I am grateful for his patience, and technical and moral support at difficult stages of my research. Andrea guided me through stagnation and dead ends to accepted papers. It motivated me a lot to see that he personally cares about success and well-being of his students and is ready to make big efforts to help. I would like to express my hope that our collaboration will go on for decades.

I am thankful to Sofien Bouaziz for supervising my Master Thesis and co-supervising my first PhD projects. Sofien was always there for me to give technical advice, answer questions or simply cheer me up. I owe Sofien my lucky choice career, since I joined the lab to continue working with him.

I would like to extend my gratitude to my collaborators Mario Botsch, Andrew Fitzgibbon, Edoardo Remelli and Matthias Schröder. The presence of Matthias brightened the deadline for our first project. The company of Edoardo and the possibility to discuss the technical details of the project on the daily basis made the last year my PhD more fun.

I would like to thank Andrew Fitzgibbon for supervising my internship at Microsoft Research. It was a privilege to have weekly discussions and look at the way he approaches research problems. I am also grateful to Federica Bogo and Tom Cashman for their help and advice.

I am thankful to Luciano Sbaiz and Wei Li for mentoring my internship at Google Research. It was an intense learning experience and their guidance and support made a big difference.

I am thankful to my thesis committee for finding the time in their busy schedule to review my

thesis and conduct the oral exam: Pascal Fua, Vincent Lepetit, Mark Pauly, Gerard Pons-Moll, Mathieu Salzmann and Andrea Tagliasacchi.

I am grateful to Merlin Nimier-David for his huge help in creating the template hand model. I also thank Timur Bagautdinov, Pierre Baqué, Jan Bednarik, Anna Biletta, Filippo Candia, Pei-I Chen, Alexis Dewaele, Giacomo Garegnani, Fabrice Guibert, Tomas Joveniaux, Isinsu Katircioglu, Mina Konakovic, Andrii Maksai, Hannah Pauly, Edoardo Remelli, Madeleine Robert, Stefano Savare and Matthieu Sauvé for participation in creating online hand calibration dataset.

I would like to thank my LGG lab mates for sharing this experience with me: Sofien Bouaziz, Duygu Ceylan, Minh Dang, Mario Deuss, Alexandru-Eugen Ichim, Mina Konakovic, Stefan Lienhard, Peng Song, Yuliy Schwartzburg, Andrea Tagliasacchi, Ziqi Wang. I am grateful to Madeleine Robert for all the help. In general I thank my fellow PhD students in EDIC for their company, especially the ones who came to board games nights and my collocation neighbors Arseniy Zaostrovnykh and Dmitrii Ustiugov.

Finally, my biggest thanks go to my parents and my life partner Andrii for their love and support. Thank you, Andrii, sharing with me every moment and making me happy.

My research has been supported by the funding from FNS project 200021_153567.

Abstract

In our everyday life we interact with the surrounding environment using our hands. A main focus of recent research has been to bring such interaction to virtual objects, such as the ones projected in virtual reality devices, or super-imposed as holograms in AR/MR headsets. For these applications, it is desirable for the tracking technology to be robust, accurate, and have a seamless deployment. In this thesis we address these requirements by proposing an efficient and robust hand tracking algorithm, introducing a hand model representation that strikes a balance between accuracy and performance, and presenting the online algorithm for precise hand calibration.

In the first part we present a robust method for capturing articulated hand motions in real time 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. We present a principled way of integrating such priors into our registration optimization to enable robust tracking without severely restricting the freedom of motion.

In the second part we propose the use of sphere-meshes as a novel geometric representation for real-time generative hand tracking. We derive an optimization to non-rigidly deform a template model to fit the user data in a number of poses. This optimization jointly captures the user's static and dynamic hand geometry, thus facilitating high-precision registration. At the same time, the limited number of primitives in the tracking template allows us to retain excellent computational performance. We confirm this by embedding our models in an open source real-time registration algorithm to obtain a tracker steadily running at 60Hz.

In the third part we introduce an online hand calibration method that learns the geometry as the user performs live in front of the camera, thus enabling seamless virtual interaction at the consumer level. The key novelty in our approach is an online optimization algorithm that jointly estimates pose and shape in each frame, and determines the uncertainty in such estimates. This knowledge allows the algorithm to integrate per-frame estimates over time, and build a personalized geometric model of the captured user. Our approach can easily be integrated in state-of-the-art continuous generative motion tracking software. We provide a detailed evaluation that shows how our approach achieves accurate motion tracking for real-

time applications, while significantly simplifying the workflow of accurate hand performance capture.

Keywords: non-rigid registration, realtime hand tracking, realtime hand calibration, sphere-meshes, markerless motion capture

Résumé

Dans notre vie quotidienne, nous interagissons avec l'environnement en utilisant nos mains. Un objectif principal de la recherche récente a été d'apporter une telle interaction à des objets virtuels, tels que ceux projetés dans des dispositifs de réalité virtuelle, ou super-imposés comme des hologrammes dans les casques AR / MR. Pour ces applications, il est souhaitable que la technologie de suivi soit robuste, précise et transparente dans le déploiement. Dans cette thèse, nous répondons à ces exigences en fournissant un algorithme de suivi manuel efficace et robuste, en introduisant une représentation manuelle du modèle qui équilibre la précision et la performance, et en présentant l'algorithme en ligne pour un étalonnage manuel précis.

Dans la première partie, nous présentons une méthode robuste pour capturer les mouvements de la main articulée en temps réel en utilisant une caméra de profondeur unique. Notre système est basé sur un processus d'enregistrement en temps réel qui reconstruit avec précision les poses de la main en ajustant un modèle de main 3D articulé aux images de profondeur. Nous enregistrons le modèle de la main en utilisant la profondeur, la silhouette et l'information temporelle. Pour mapper efficacement des cartes de profondeur de basse qualité à des poses de mains réalistes, nous régularisons l'enregistrement avec des priors cinématiques et temporels, ainsi qu'un préréglage basé sur des données construit à partir d'une base de données de poses réalistes. Nous présentons une méthode basée sur des principes pour intégrer ces priors dans notre optimisation d'enregistrement pour permettre un suivi robuste sans restreindre de manière significative la liberté de mouvement.

Dans la seconde partie, nous proposons l'utilisation de mailles-sphères comme nouvelle représentation géométrique pour le suivi génératif en temps réel. Nous dérivons une optimisation pour déformer de manière non rigide un modèle étalon pour adapter les données de l'utilisateur dans un certain nombre de poses. Cette optimisation capture conjointement la géométrie de la main statique et dynamique de l'utilisateur, facilitant ainsi l'enregistrement de haute précision. En même temps, le nombre limité de primitives dans le modèle de suivi nous permet de maintenir d'excellentes performances de calcul. Nous confirmons cela en intégrant nos modèles dans un algorithme d'enregistrement en temps réel et code source ouvert pour obtenir un tracker fonctionnant régulièrement à 60Hz.

Dans la troisième partie, nous introduisons une méthode de calibrage manuel en ligne qui apprend la géométrie lorsque l'utilisateur se produit en direct devant la caméra, permettant

ainsi une interaction virtuelle transparente au niveau du consommateur. La nouveauté clé dans notre approche est un algorithme d'optimisation en ligne qui estime conjointement la pose et la forme dans chaque trame, et détermine l'incertitude dans de telles estimations. Cette connaissance permet à l'algorithme d'intégrer les estimations d'images dans le temps et de construire un modèle géométrique personnalisé de l'utilisateur capturé. Notre approche peut facilement être intégrée dans un logiciel de suivi de mouvement continu, à la pointe de la technologie. Nous fournissons une évaluation détaillée qui montre comment notre approche réalise un suivi de mouvement précis pour les applications en temps réel, tout en simplifiant grandement le flux de travail pour une capture précise des performances de la main.

Mots-clés : enregistrement non-rigide, suivi des mains en temps réel, étalonnage manuel en temps réel, maillages de sphères, capture de mouvement sans marqueur

Contents

1 Introduction

1.1 Motivation

Tracking humans in motion is a fundamental problem in computer graphics and computer vision. A particularly important question is how to accurately reconstruct the shape and articulation of human hands. Firstly, because in our everyday life we interact with the surrounding environment using our hands [Bullock et al., 2013], [Dollar, 2014]. Secondly, hand motion is a crucial component of non-verbal communication [Goman, 2009], [Goman, 2011]. The digital world applications of hand tracking follow from these two functions of hands in the real world.

Performance capture. Performance capture is essential in film and game production for pre-visualization, where motion can be transferred in real-time to a virtual avatar. This allows directors to plan shots more effectively, reduce turn-around times and hence costs. The captured motion can also be analyzed for purposes like re-training after stroke, automatically translating sign language, or giving feedback to piano students.

Remote communication. Being an important part of our body language, hand motion plays a significant role in the animation of humanoid avatars. The first steps towards commercial avatar-based communication were made by *Microsoft Holoportation* [1] and *Apple Animoji* [2].

Gesture control. Gesture control, a simplified version of hand tracking, becomes increasingly popular as a replacement of remote control for home appliances. It is currently used in such consumer products as *Samsung Smart TV* [3] and *Singlecue* [4]. A few other similar products are currently under development.

Virtual interaction. Recently the field of virtual and augmented reality (VR/AR) has made a large step forward. A number of VR/AR headsets were released, including *Oculus*, *Vive*, *Samsung Gear VR*, *Microsoft Hololens*, *PlayStation VR*, *Google Daydream*, *Microsoft Mixed*

[1] Holoportation: https://www.microsoft.com/en-us/research/project/holoportation-3/, accessed on 27.11.2017
[2] Apple Animoji: https://support.apple.com/en-us/HT208190, accessed on 27.11.2017
[3] Samsung Smart TV: http://www.samsung.com/ph/smarttv/motion_control.html, accessed on 27.11.2017
[4] Singlecue: https://singlecue.com/, accessed on 27.11.2017

Reality Headset, *Intel Project Alloy* and *Meta 2 AR Headset*. The technology is incomplete without providing the user a way to interact with a virtual environment. Most of the listed headsets started with dedicated controller devices. However, there is a trend in the field of replacing controller devices by markerless hand control. *Microsoft Hololens* [5], *Meta 2 AR Headset* [6] and *Intel Project Alloy* [7] are already released in a hand-controlled version and the other main manufacturers are also currently developing similar technology. There are several reasons why VR/AR helmets benefit from hand control. Firstly, according to the user study conducted by *Leap Motion* [8], interacting with your own hands creates a more immersive experience. Secondly, it takes time to get used to the controller device and to remember the functionality assigned to each button. Moreover, hands control can potentially be more expressive and subtle than a dedicated controller device.

Commercial hand control devices are still an emerging technology, because hand tracking is challenging and remains a research problem. This was even more so at the start of my doctoral studies in 2014. The challenges are described below.

Requirements

Any consumer application relies on robustness of the tracker. However, the applications listed above have different requirements in terms of precision, efficiency and output format of the hand tracking algorithm.

A *gesture control* system is only required to classify a gesture, thus inferring exact hand poses is not necessary. For *performance capture* it is acceptable to have slower than real time performance. In *remote communication*, the hand motion may be re-targeted to an avatar hand. In that case it is only required to track joint positions as opposed to an entire hand surface.

Virtual interaction is the most demanding, yet most promising application. It requires exact tracking of hand movements. As explained below, accurate tracking is only possible if the model is precisely calibrated to the user. Moreover, to be suitable for consumer application, it is undesirable for the calibration to take a long time or require user input. Physically plausible interaction with a virtual object requires the system to infer not just hand joint positions, but its full 3D geometry.

[5] Microsoft Hololens: https://www.microsoft.com/en-us/hololens, accessed on 27.11.2017
[6] Meta 2 AR Headset: https://www.metavision.com/, accessed on 27.11.2017
[7] Intel Project Alloy: https://newsroom.intel.com/press-kits/project-alloy/, accessed on 27.11.2017
[8] Leap Motion Blog:http://blog.leapmotion.com/image-hands-bring-your-own-hands-into-virtual-reality/, accessed on 27.11.2017

Challenges

Tracking challenges. Accurate hand tracking with a non-invasive sensing device in real-time is a challenging scientific problem. Human hands are highly articulated and therefore require models with sufficiently many degrees of freedom to adequately describe the corresponding motion space. Hand motion is often fast and exhibits intricate geometric configurations with complex contact patterns among fingers. With a single-camera RGBD setup, we are faced with incomplete data due to self-occlusions and high noise levels.

Calibration challenges. High precision model based tracking is difficult without calibrating the model to the specific user. The main challenge comes from the fact that tracking and calibration procedures are interdependent. High quality tracking requires good calibration and, to accurately calibrate the model, the motion needs to be precisely tracked. Moreover, hand calibration is bound to consider multiple frames, since from a single frame only a subset of the shape degrees of freedom can be estimated. For example, it is difficult to estimate the length of a phalanx when observing a straight finger.

Setup

Tracking setup. Over the past two decades a number of techniques have been explored to address the hand tacking problem, from expensive and unwieldy marker-based mocap [Welch and Foxlin, 2002] to instrumented gloves [Dipietro et al., 2008] as well as imaging systems [Erol et al., 2007]. Multi-camera imaging systems can recover the hand pose and hand-objects interactions with high accuracy [Ballan et al., 2012], but the only system capable to approach interactive applications is the 10Hz system of [Sridhar et al., 2013]. Conversely, in this thesis we focus on hand motion tracking with a single RGBD sensor (e.g. Intel RealSense or Microsoft Kinect), commonly predicted to be readily available in a typical AR/VR consumer experience. This setup does not require the user to wear a glove or markers. Such single-camera acquisition is particularly advantageous as it is cheap, does not require any sensor calibration, and does not impede user movements.

Tracking: discriminative vs. generative. Modern systems for real-time tracking from RGBD data [Sridhar et al., 2015, Sharp et al., 2015] rely on a combination of discriminative approaches like [Keskin et al., 2012], and generative approaches such as [Oikonomidis et al., 2011]. The per-frame re-initialization of discriminative methods prevents error propagation by offering a continuous recovery from tracking failure. As these discriminative models are learnt from data, they are typically limited in their precision by dataset annotation accuracy. Annotating joint locations is challenging because it needs to be done in 3D, and because the joints are situated inside the hand. These difficulties affect the labeling quality. Therefore, generative models are used to refine the estimate by aligning a geometric template of the user hand to the measured point cloud, as well as to regularize its motion through time. It is not surprising that the quality of the template directly affects the quality of pose refinement.

Calibration setup. The process of accurately generating a user-specific tracking model from input data is referred to in the literature as *calibration* or *personalization*. Calibrating a template from a set of static poses is a standard component in facial performance capture [Weise et al., 2011, Cao et al., 2015], and the work of [Taylor et al., 2014] pioneered it within the realm of hand tracking. However, current methods such as [Taylor et al., 2016] suffer a major drawback: the template must be created during a controlled calibration stage where the hand is scanned in several static poses (i.e. offline). While appropriate for professional use, a calibration session is a severe drawback for seamless deployment in consumer-level applications.

1.2 Contributions

This dissertation is based on and uses parts of the following papers published in the course of my PhD:

1. TAGLIASACCHI A., SCHROEDER M., TKACH A., BOUAZIZ S., BOTSCH M., PAULY M.: Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum(Proc. of the Symposium on Geometry Processing). 2015.*

2. TKACH A., PAULY M., TAGLIASACCHI A.: Sphere-meshes for real-time hand modeling and tracking. *In ACM Trans. Graph. (Proc. SIGGRAPH Asia). 2016.*

3. TKACH A., TAGLIASACCHI A., REMELLI E., PAULY M., FITZGIBBON A.: Online generative model personalization for hand tracking. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia). 2017.*

The accompanying videos, Video1 [9], Video2 [10], and Video3 [11], illustrate the real-time tracking performance of the presented systems.

The following paper, also published during my PhD, is not discussed in this thesis, since its contributions are contained within the later work.

REMELLI E., TKACH A., TAGLIASACCHI A., PAULY M.: Low-Dimensionality Calibration through Local Anisotropic Scaling for Robust Hand Model Personalization. *Proceedings of the International Conference on Computer Vision. 2017.*

In summary, the contributions of this dissertation are:

- **Robust real-time model-based hand tracking algorithm.** We develope a robust model-based hand tracking algorithm that efficiently integrates data and regularization priors

[9] Please find the accompanying Video1 at http://lgg.epfl.ch/publications/2015/Htrack_ICP/new_video.mp4.
[10] Please find the accompanying Video2 at http://lgg.epfl.ch/publications/2016/HModel/video.mp4.
[11] Please find the accompanying Video3 at http://lgg.epfl.ch/publications/2017/HOnline/video.mp4.

into a unified real-time solver running at 60 FPS. The fist key component of the algorithm is an efficient combined 2D/3D registration method to align the 3D hand model to the acquired depth map and extracted silhouette image. The second key feature is a new way of computing data-to-model correspondences that accounts for occlusions and significantly improves the robustness of the tracking

- **Sphere-meshes model for efficient and accurate hand shape representation.** We present a sphere-meshes hand model and demonstrate that it provides superior hand tracking performance for single-view depth sensors. We introduced an optimization approach that allows adapting our tracking model to different human hands with a high level of accuracy. The improved geometric fidelity compared to existing representations leads to quantifiable reductions in registration error and allows accurate tracking even for intricate hand poses and complex motion sequences that previous methods have difficulties with. At the same time, due to a very compact model representation and closed-form correspondence queries, our generative model retains high computational performance, leading to sustained tracking at 60 FPS.

- **Online hand model calibration.** We introduce a principled way of integrating per-frame information into an online real-time pose/shape tracking algorithm: one that estimates the hand's pose, while simultaneously refining its shape. That is, the more of the user's hand and articulation is observed during tracking, the more the tracking template is progressively adapted to match the performer, which in turn results in more accurate motion tracking. Our technique automatically estimates the confidence in per-frame parameter computations, and leverages this information to build a tracking model that selectively accumulates confident parameter estimates over time. Assuming a reasonable performance by the user, our system typically constructs a fully calibrated model within a few seconds, while simultaneously tracking the user in real time.

- **Open Source.** Another important contribution is that we fully disclosed our source code. We believe that publishing our code will not only ensure reproducibility of our results, but also facilitate future research in this domain.

1.3 Overview

The remainder of the thesis describes our steps in solving the problem of precise model-based hand tracking. The problem consists of two inter-dependent components: tracking and calibration.

Section 1.4 presents a detailed review of existing real-time single view hand tracking systems that are using depth input.

Chapter 2 describes our initial hand tracking system that uses the cylinders hand model. In Sections 2.1 we place our work in a broader context. In Section 2.2 we address the challenges of robust hand tracking by proposing a regularized articulated ICP-like optimization that carefully balances data fitting with suitable priors. Our data fitting performs a joint 2D-3D optimization. The 3D alignment ensures that every point measured by the sensor is sufficiently close to the tracked model. Simultaneously, as we cannot create such constraints for occluded parts of the hand, we integrate a 2D registration that pushes the tracked model to lie within the estimated foreground. In Section 2.3 we detail a carefully chosen set of priors that regularize

the solution to ensure the recovered pose is plausible. After discussing some implementation details in Section 2.4, we analyze tracking performance by providing a comparison to several state-of-the-art solutions in Section 2.5.

Chapter 3 addresses the choice of hand model representation that is suitable both for efficient tracking and for accurate calibration. In Sections 3.1 and 3.2 we motivate the work, discuss related literature and our contributions. In Section 3.3 we detail how our novel formulation fits into previous generative real-time hand tracking technique, while still enabling efficient correspondence computation. Section 3.4 explains how we build our template model from 3D scans acquired either through multi-view stereo or from depth maps. In Section 3.5 we analyze the performance of our model for realtime tracking and provide comparisons to the state-of-the-art.

In Chapter 4 we reconsider offline calibration, aiming to enhance user experience and push calibration quality further. In Sections 4.1 and 4.2 we introduce the topic, explain the relevance of our work and position it with respect to other approaches in the area. In Section 4.3 we describe our joint calibration and tracking algorithm, which combines the Levenberg-style optimization of previous hand trackers with the uncertainty bookkeeping of the Kalman filter. In Section 4.4, to evaluate the technical validity of our approach, we corroborate the formulation of our optimization on a synthetic 3D dataset, analyze its robustness by randomly perturbing the algorithm initialization, and attest how our method achieves state-of-the-art performance on publicly available datasets. In Section 4.6 we introduce the Kalman filter with its extensions and derive the equivalence of the proposed online calibration scheme with a recent tool from control theory – the Levenberg-Marquardt Kalman Filter.

1.4 Related Works

In this section we summarize the main works in real-time single view hand tracking from depth input. The works from the other areas relevant to the subsequent chapters of this thesis are reviewed in the related literature sections of the corresponding chapters.

Tracking algorithms can be roughly divided into two main classes: discriminative and generative.

- *Discriminative* methods directly predict hand pose from image features. State-of-the-art approaches learn the mapping between the image and hand pose from the annotated training data. The most widely used learning algorithms are Random Decision Forest (RDF) [Keskin et al., 2012] and Convolutional Neural Networks (CNN) [Tompson et al., 2014]. Discriminative algorithms regress a small number of key features, like joint positions or angles, as opposed to the full hand geometry. The predicted hand pose can afterwards be used to drive a hand model, however the surface of the model is often not exactly aligned with the data.

- *Generative* methods minimize the discrepancy between the hand model and the input

data by solving a data-model alignment problem. The main algorithms used for this task are gradient descent [Taylor et al., 2016] and Particle Swarm Optimization (PSO) [Oikonomidis et al., 2011]. There are also some new works that use CNNs [Dibra et al., 2017], [Wan et al., 2017]. Gradient Descent and PSO require initialization, which is either obtained from hand pose at the previous frame or from a discriminative method.

1.4.1 Discriminative Methods

[Keskin et al., 2012] estimate hand pose by predicting the hand part labels probabilities for each pixel. The labels prediction is done using an RDF. The centers of the hand parts are inferred by representing each label with a gaussian and finding the maximum on the resulting surface. This is under the assumption that the pixel with maximum probability value for the given hand part is situated in the center of that hand part. The hand skeleton is obtained by connecting the joints according to their configuration in the hand. To improve performance, the training set is split in clusters of similar hand poses. The results from different clusters are aggregated by an expert network.

[Tang et al., 2014] present a method similar to the one introduced by [Keskin et al., 2012]. Differently from the former, instead of using an RDF for predicting hand parts, they adopt Latent Regression Forest (LRF). In LRF the non-leaf nodes correspond to groupings of hand parts. The method performs structured coarse-to-fine search, starting with the entire hand and recursively splitting it, until locating all the skeletal joints. This work has superior performance with respect to [Keskin et al., 2012], where one of the reasons is greater robustness to occlusions.

[Tompson et al., 2014] pioneered using CNNs for discriminative hand tracking. Their work (and numerous subsequent methods) are enabled by the automatically labeled dataset that they have constructed. The authors trained a CNN to generate a set of heat-map images for key hand features, taking multi-resolution depth images as an input. At each resolution the network contains two convolution layers; each convolution is followed by RELU and max pooling. The concatenated outputs of convolution layers are fed to two fully connected layers. The final kinematically valid hand pose is obtained by applying an inverse kinematic model on the heat-maps.

[Sun et al., 2015] use cascaded regression for predicting hand pose. In the cascaded regression framework, the pose is estimated iteratively by a sequence of regressors. Each regressor uses the output of the previous one, progressively decreasing the error. The regressors are learned with RDF. The authors modify offset features, widely used for RDF, to make them invariant to 3D transformations. They also propose a hierarchical approach to regress hand pose. Firstly, the palm transformation is regressed. The inverse of this transformation is afterwards applied to the fingers before estimating their poses. This approach is shown to perform better than estimating the pose holistically, as it reduces appearance variations for the fingers.

[Tang et al., 2015] propose to estimate hand pose hierarchically starting with the parameters at the base of hand kinematic chain and inferring the parameters at each next layer conditioned on the previous layer (layer 1 – wrist translation, layer 2 – wrist rotation, and so on along the kinematic chain). For efficiency they formulate a cost function in terms of joint positions only. Advantageously, evaluation of this cost function does not require rendering the model or computing closest point correspondences. Moreover, this cost function can also be evaluated for partial poses. The proposed hierarchical optimization framework generates several samples of the partial pose at each layer, the sample with the minimal value of cost function is then selected. To generate the samples, the authors train an RDF for predicting partial poses. They use standard features for RDF on depth images. The system generates multiple hypotheses using the described approach, the final pose is selected by evaluating the "golden energy" suggested by [Sharp et al., 2015]. This approach outperforms the other works that use hierarchical hand pose estimation algorithms, such as [Tang et al., 2014] and [Sun et al., 2015].

[Li et al., 2015] extend the work of [Keskin et al., 2012] and [Tang et al., 2015] by proposing another variant of RDF. Similarly to [Tang et al., 2014], the method performs structured coarse-to-fine search, starting with entire hand and splitting it recursively to joints. Differently from [Tang et al., 2014] the division hierarchy of hand parts may not be the same for different poses. The work achieves superior performance on the ICVL dataset ([Tang et al., 2014]).

[Oberweger et al., 2015a] compare several CNN architectures and find that the best performance is given by a deeper architecture that takes depth images at several scales as an input. The rationale is that using multiple scales helps capturing contextual information. The authors also propose to regress hand pose parameters in a lower-dimensional subspace. After the initial estimation phase follows a refinement step. To enhance the location estimate provided by the first stage, they use a different network for each joint. The per-joint networks look at several patches of different sizes centered on the predicted joint location. The refinement step is repeated several times, each iteration is centered on a newly predicted location.

[Ge et al., 2016] propose to project the input depth image onto orthogonal planes and use the resulting views to predict 2D heat-maps of joint locations on each plane. These 2D heat-maps are then fused to produce the final 3D hand pose. The fusion step is expected to correct the imprecisions using the predictions from complementary viewpoints. The authors use a multi-resolution CNN on each view with architecture similar to the one introduced by [Tompson et al., 2014]. Given the 2D heat maps from the three views, they find the hand pose parameters in a lower dimensional PCA subspace, such that the total heat map confidence at the joint locations on the three views is maximized.

[Sinha et al., 2016] exploit activation features from a hidden layer of a trained CNN. The assumption is that augmenting an output activation feature by a pool of its nearest neighbors brings more reliable information about the hand pose. Drawing on the fact that CNNs are less robust for regression than for classification, the authors compute the activation features

from classifying joint angles into bins with a CNN (as opposed to regressing the exact values of the joint angles). Since the number of quantized hand poses is very large, they propose a two-stage classification. On the first stage global hand rotation is classified. Next, for each rotation bin, five separate CNNs are trained to classify the poses of the fingers. At run time, given the activation features, a pool of their nearest neighbors is efficiently retrieved from a database. The final hand pose is computed from the assumption that a matrix of stacked neighboring activation features concatenated with stacked corresponding hand poses has a low rank. The unknown current hand pose is computed by matrix completion[12].

[**Zhou et al., 2016**] integrate domain knowledge about hand motion into a CNN. This is done by adding a non-parametric layer that encodes a forward kinematic mapping from joint angles to joint locations. Since the forward kinematic function is differentiable, it can be used in a neural network for gradient-descent like optimization. This approach guarantees that the predicted hand pose is valid. The remaining network architecture is similar to the one introduced by [Oberweger et al., 2015a].

[**Guo et al., 2017**] propose to use a hierarchically-structured Region Ensemble Network (REN) for hand pose inference. This architecture is inspired by the widely used approach of averaging predictions from different crops of an original image. The averaging is beneficial since it decreases the variance of image classification; however, it is computationally expensive. The authors propose a solution that retains the advantages while cutting the costs. They suggest to split the input image in several regions, predict the whole hand pose separately from each region and aggregate regional results afterwards. The REN architecture starts with six convolutional layers augmented with two residual connections. The region-wise prediction is implemented through dividing the output of the convolutional layers into a uniform grid. Each grid cell is fed into fully connected layers. Subsequently the outputs of all the cells are concatenated together and used to predict the final hand pose. This approach has state-of-the-art performance on the NYU and ICVL datasets.

[**Madadi et al., 2017**] propose a hierarchical tree-like CNN that mimics the kinematic structure of human hand. The branches of the network are trained to become specialized in predicting the locations of subsets of hand joints (local pose), while the parameters closer to the tree root are shared for all hand parts. The network contains a loss term for each local pose. Additionally, the outputs of the tree branches are concatenated and fed to the fully-connected layer for estimating the final pose. The authors argue the later step allows to learn higher order dependencies among joints. The loss function also contains the terms that penalize predicting joint locations outside of data hull and encourage all joints from one finger to be co-planar.

[**Mueller et al., 2017**] present a method for predicting hand pose in egocentric view. Their system is designed for hand-object interaction scenarios and is robust to occlusions. They

[12] Matrix completion is the task of filling in the missing entries of a partially observed matrix. One of the variants of the matrix completion problem is to find the lowest rank matrix X which matches the matrix M, which we wish to recover, for all entries in the set E of observed entries. "Matrix completion." Wikipedia: The Free Encyclopedia. Wikimedia Foundation, https://en.wikipedia.org/wiki/Matrix_completion, [accessed 30 January 2018].

estimate hand pose in several steps. Firstly, to localize the hand, a heat map of the hand root position is regressed. Given the hand root, the input image is normalized and feed into a joint regression network. This network outputs 2D heat maps and 3D positions of the joints. As the last step, a kinematically valid hand pose is computed by optimizing a sum-of-energies cost function. The cost function includes the closeness of optimized joint locations to the CNN-predicted joint locations, joint limits and temporal smoothness term. Both networks are trained on synthetic data generated by accurately tracked hand motion with existing tracker and retargeting it to a virtual hand model.

[Oberweger and Lepetit, 2017] extend their previous work [Oberweger et al., 2015a]. They carry out an extensive evaluation to show that the improved method achieves superior or comparable performance to all recent works on three main benchmarks of hand tracking (NUY, ICVL and MSRA). The authors introduce the following improvements: firstly, the training data is augmented to 10M samples (by translating, rotating and scaling). The second enhancement is training a CNN that regresses hand root for accurate hand localization. Finally, the new pose network architecture is similar to ResNet: a convolution layer is followed by four residual modules, that are in turn followed by several fully connected layers with dropout.

1.4.2 Generative Methods

[Oikonomidis et al., 2011] present a generative tracking approach. Their algorithm minimizes the difference between the sensor data and the rendered capsules model. The optimization is performed using Particle Swarm Optimization. The method runs at 15 fps on GPU and does not include any re-initialization component in case of tracking failure.

[Melax et al., 2013] show compelling 60 fps realtime performance using gradient-based optimization. The authors introduce a convex polyhedral model and track it with a rigid body dynamics solver. The rigid bodies from the model are constrained to come into alignment with the point cloud. The hand parts are attached together by constraints of a larger strength. Thus, in contrast with the majority of model-based systems, their technique does not use Inverse Kinematics. Each data point adds a constraint on the closest articulated component of the hand. The model is also constrained to stay within 3D hull of the point cloud by adding collision planes constraints on the boundaries of the convex hull.

[Oikonomidis et al., 2014] extend their previous work [Oikonomidis et al., 2011] by introducing a more advanced sampling strategy that improves tracking efficiency without compromising quality. They sample the hand-pose vectors using quasi-random sequence that covers multi-dimensional spaces better than random sampling. However, gradient-based optimization approaches converge faster and more accurately than PSO when close to the solution.

[Qian et al., 2014] modify the PSO algorithm employed by [Oikonomidis et al., 2011] by adding a gradient-based component to it. Each particle takes an additional ICP-like gradient descent step in each PSO generation. This is intended to combine advantages and mitigate drawbacks

of PSO and ICP. The authors demonstrate that their system has superior performance to [Oikonomidis et al., 2011]. The presented system is hybrid, it uses a spheres model for ICP-PSO optimization and detects fingertips with flood fill for re-initialization. Apart from closeness of the model to the data, the cost function also includes a term that constrains the model to lie within the sensor visual hull and behind the data.

[Schroder et al., 2014] formulate the optimization in a subspace of likely hand poses, rather than resorting to reinitialization for robustness. They capture a dataset of human hand movements with a Vicon motion tracking system. The dataset is employed as the ground truth for deriving natural hand synergies based on principal component analysis. While the lower number of optimization variables leads to efficient computations, tracking accuracy can be limited by the reduced pose complexity induced by the subspace. The authors use a cylinder hand model driven by Inverse Kinematics and apply ICP algorithm for aligning the model with the data.

[Fleishman et al., 2015] present a system that uses capsules hand model and ICP-IK algorithm for data-model alignment. For initialization they train an RDF classifier to label data pixels with hand parts. To increase the robustness, the system generates several hypotheses of hand pose from the labeled data. In the final step, they apply ICP-IK algorithm to each skeleton hypothesis (with each finger being straight or bent). The closest-point correspondences are only created between the same parts of the data and model. The authors show that ICP-IK algorithm gives superior performance with respect to their implementation of PSO.

[Oberweger et al., 2015b] design a convolutional neural network capable of directly synthesizing hand depth images. The motivation for this work is replacing hand model for hybrid tracking. As a first step they use a CNN to predict an initial pose from the depth input. The initial pose is used to synthesize a depth image. The synthesized image and the input image are fed to an updater CNN. The updater learns to predict updates, which would improve the pose estimate, given the input and the synthesized depth. This process is repeated for several iterations. The synthesizer network consists of several fully-connected layers followed by several unpooling and convolution layers. The updater network has a siamese architecture. It consists of two identical paths of several convolutional layers. The final feature maps are concatenated and fed into a fully connected network.

[Poier et al., 2015] initialize the proposed hybrid tracking system by regressing hand joint locations with an RDF. The authors consider several top predictions for each joint along with the confidence score. The kinematic parameters of a 3D hand model are determined by selecting a proposal for each joint location, such that the chosen locations for all joints form an anatomically valid pose. They apply PSO algorithm for optimizing the described cost function. For efficiency, the authors split the full PSO problem into sub-problems, solving for the pose of each finger independently. Differently from [Oikonomidis et al., 2011], this approach does not require rendering the model, thus it can run on CPU.

[Sharp et al., 2015] introduce a hybrid approach that minimizes the “golden energy” - the

reconstruction error between a rendered 3D hand model and the observed depth image. The rendered image has a potential to match the observed image, since they use a detailed triangular mesh hand model instead of spheres/cylinders. The model is not differentiable; thus the authors apply PSO algorithm for optimization. For re-initialization they use Kinect-provided skeleton and train a two-staged RDF regressor. The first stage only deals with predicting quantized global hand rotation, while the second stage refines the rotation and regresses the pose. The system is robust and works well at a distance of several meters and in moving camera scenarios.

[Sridhar et al., 2015] encode the model with a predefined mixture of Gaussians. The data is also represented as a mixture of Gaussians. This is done through decomposing the depth image into regions of homogeneous depth (using a quad-tree) and fitting a Gaussian to each region. The authors optimize the closeness of model to the data with gradient descent. A Gaussian mixture representation allows, instead of computing closest point correspondences, to match data mixtures of Gaussians with the model. For robustness the system generates multiple hypotheses of hand pose and chooses the best one based on pose fitting energy. One of the hypothesis comes from an RDF hand parts classifier. For that hypothesis a different type of energy is optimized: each Gaussian in the data is given a part label which is most frequent among its pixels; the model is aligned with the data according to hand part labels.

[Taylor et al., 2016] present a continuous registration framework for tracking hands with triangular meshes. The control mesh is augmented with a continuous Loop subdivision surface that provides gradients for optimization. Similar to [Tagliasacchi et al., 2015] they define a differentiable cost function as a weighted sum of several terms, including data energy, joint limits, pose prior, temporal prior, etc. For the data energy term they introduce an alternative to the ICP algorithm. To compute closest point correspondences, they define a set of corresponding variables that are optimized jointly with the model pose. Compared to ICP, the proposed algorithm requires less iterations and has a wider convergence basin.

[Dibra et al., 2017] propose the first CNN-based approach that does not require an annotated hand-motion dataset for training. As a first step, they train a network to predict an approximate hand pose from synthetic depth images. As a second step, they refine the network by training it on the unlabeled data. The loss function on unlabeled data is an L1 error norm between the input depth image and a synthesized depth image, given the current hand pose. To enable backpropagation of the error, the authors introduce a differentiable algorithm for "rendering" the hand model. The algorithm applies linear blend skinning to the point cloud that was uniformly sampled from the hand model. The authors also propose a differentiable method for rendering only the visible part of the model, which relies on defining a support circle for each model point. The presented system achieves performance comparable to state of the art methods without requiring costly annotation.

[Taylor et al., 2017] introduce a new hand model representation that avoids the compromise between efficiency and accuracy. This is achieved by constructing an articulated signed dis-

tance function that provides closed-form distances to the model surface and is differentiable. In more details, the hand model is driven by a linear blend skinned tetrahedral mesh, that deforms a precomputed signed distance field into a given pose. The closest point correspondences are computed in efficient and parallelizable manner. This allows the system to run at ultra-high frame rates on GPU (1000Hz). Due to its efficiency and robustness, this system accurately tracks complex interaction of two hands.

[Wan et al., 2017] propose a framework for learning from unlabeled data in a semi-supervised manner. They learn a shared latent space where each point can be mapped both to a synthetic depth image and to the corresponding hand pose parameters. The hand pose is regressed by training a discriminator to predict a posterior of the latent pose given the input depth image. The depth image generator and discriminator are trained jointly in order to improve generalization. To avoid overfitting during posterior estimation the authors add additional loss terms that share first several convolutional layers with pose estimation.

1.4.3 Comparison of Previous Works

The comparative summary of previous hand tracking works is presented in Table 1.1, while the partial ranking of their accuracy on hand tracking benchmarks in shown in Table 1.2.

Comparative Summary. Table 1.1 includes the type of applied discriminative and/or generative algorithm, the type of hand model, the time complexity of the system (GPU/CPU and FPS) as well as domain knowledge (priors) incorporated in each method and the type of final output. The final output can be different from the model type for some discriminative approaches that perform a model-fitting step after regressing joint locations.

We use the following naming conventions:

- *inconsistent joints* - model type for discriminative methods that predict a set joint locations per-frame. Without an additional model fitting step, joint locations are not guaranteed to correspond to a skeleton with consistent length of phalanges. Thus, they can overfit to the input data and get higher performance score, but cannot be directly used to drive a hand model.
- *skeleton* - model type for discriminative methods that regress joint angles. These methods use a skeleton with constant joint length to pose it with the predicted joint angles.
- <*description*> *model* - (volumetric) model type for generative methods, where the <*description*> names model components, such as capsules, spheres, triangular mesh, Gaussians, ect.
- *point cloud* - model type for generative methods that train a CNN to regress an image/point-cloud of the hand.

Paper	Discriminative Algorithm	Generative Algorithm	Model Type	GPU/ CPU	FPS	Priors	Final Output
[Oikonomidis et al., 2011]	none	PSO	capsules model	GPU	15	fingers have similar abduction	surface
[Keskin et al., 2012]	classification of pixels with RDF	none	inconsistent joints	CPU	30	none	joint locations
[Melax et al., 2013]	none	rigid body dynamics solver	convex polyhedral model	CPU	60	abduction limits, sensor visual hull	surface
[Oikonomidis et al., 2014]	none	PSO	capsules model	GPU	30-60	collisions	surface
[Qian et al., 2014]	fingers detection by flood fill	ICP-PSO	spheres model	CPU	25	sensor visual hull	surface
[Schroder et al., 2014]	none	ICP-IK	capsules model	CPU	25	pose subspace, joint limits	surface
[Tompson et al., 2014]	regression of heat-maps with CNN	none	skeleton	GPU	30	none	skeleton
[Fleishman et al., 2015]	classification of pixels with RDF	ICP-IK	capsules model	CPU	25	none	surface
[Oberweger et al., 2015b]	regression of joint locations with CNN	generation of model surface with CNN	skeleton and point cloud	GPU	400	none	surface
[Poier et al., 2015]	regression of joint locations with RDF	PSO	skeleton	CPU	23	joint limits	skeleton
[Sharp et al., 2015]	regression of joint angles with RDF	PSO	triangular mesh model	GPU	30	none	surface
[Sridhar et al., 2015]	classification of pixels with RDF	gradient descent	gaussian mixture model	CPU	50	collisions, motion smoothness, joint limits	surface

[Sun et al., 2015]	cascaded regression of joint angles with RDF	none	skeleton	CPU	300	none	skeleton
[Tang et al., 2015]	regression of joint locations with LRF	none	inconsistent joints	CPU	62.5	none	joint locations
[Li et al., 2015]	regression of joint locations with RDF	none	inconsistent joints	CPU	55.5	none	joint locations
[Oberweger et al., 2015a]	regression of joint locations with CNN	none	skeleton	CPU	500	pose subspace	skeleton
[Ge et al., 2016]	regression of heat-maps with CNN	none	skeleton	GPU	70	pose subspace	skeleton
[Sinha et al., 2016]	classification of joint angles with CNN	none	skeleton	CPU	32	none	skeleton
[Taylor et al., 2016]	regression of joint angles with RDF	gradient descent	Loop subdivision surface model	CPU	> 30	pose, limits, temporal, background, fingertips, collisions	surface
[Zhou et al., 2016]	regression of joint angles and locations with CNN	none	skeleton	GPU	125	joint limits	skeleton
[Dibra et al., 2017]	regression of joint angles with CNN	generation of model surface with CNN	point cloud	GPU	285	collisions, joint limits	surface
[Guo et al., 2017]	regression of joint locations with CNN	none	inconsistent joints	GPU	3000	none	joint positions
[Madadi et al., 2017]	regression of joint locations with CNN	none	inconsistent joints	GPU	50	co-planar finger joints, sensor visual hull	joints positions
[Mueller et al., 2017]	regression of joint locations with CNN	none	skeleton	CPU	50	joint limits, temporal smoothness	skeleton
[Oberweger and Lepetit, 2017]	regression of joint locations with CNN	none	skeleton	GPU	30	pose subspace	skeleton

[Taylor et al., 2017]	regression of joint locations with RDF	gradient descent	articulated SDF model	GPU	1000	pose prior, joint limits	surface
[Wan et al., 2017]	regression joint angles with VAE	generation of model surface with GAN	skeleton and point cloud	CPU	90	none	surface

Table 1.1 – Comparative summary of hand tracking methods

Partial Ranking of Accuracy. Table 1.2 contains ranking of the described methods on the following benchmarks:

- *NUY dataset* - introduced by [Tompson et al., 2014];
- *NYU dataset, Subject 1* - introduced by [Tompson et al., 2014];
- *ICVL dataset* - introduced by [Tang et al., 2014];
- *MSRA dataset* - introduced by [Sun et al., 2015];
- *Dexter dataset* - introduced by [Sridhar et al., 2013];
- *FinterPaint dataset* - introduced by [Sharp et al., 2015];
- *Handy dataset* - introduced by [Tkach et al., 2016]

The ranking of the methods on the above benchmarks is obtained from the following sources:

A - [Oberweger and Lepetit, 2017], Table 1;

B - [Oberweger and Lepetit, 2017], Figure 5;

C - [Oberweger and Lepetit, 2017], Table 2;

D - [Oberweger and Lepetit, 2017], Table 3;

E - [Taylor et al., 2017], Figure 12;

F - [Taylor et al., 2016], Figure 10;

G - [Taylor et al., 2017], Figure 15;

H - [Oberweger and Lepetit, 2017], Figure 6;

I - [Tang et al., 2015], Figure 6;

J - [Sridhar et al., 2015], Figure 4;

K - [Tkach et al., 2017], Figure 9;

L - [Dibra et al., 2017], Figure 8;

M - [Neverova et al., 2017], Figure 8;

N - [Tkach et al., 2017], Figure 8;

The reference to the source in the header of the column, for example NUY $^{\mathbf{A}}$, means that the ranking of the all methods shown in the column was inferred from the source **A**. The reference to the source in the table cell, for example $1^{\mathbf{H}}$, means that the ranking of the corresponding method does not come from the source listed in the column header, but was inferred from the source **H**. The interval of the ranks instead of a single number, for example [10 - 11], refers to the fact that the exact rank is unclear and is somewhere in the interval.

Paper	NUY [A]	NUY Subject 1 [B]	ICVL [C]	MSRA [D]	Dexter [E]	Finger Paint [F]	Handy [G]
[Oikonomidis et al., 2011]							
[Keskin et al., 2012]			[11 – 12][I]				
[Melax et al., 2013]			[11 – 12][I]				
[Xu and Cheng, 2013]	3						
[Oikonomidis et al., 2014]							
[Qian et al., 2014]							
[Schroder et al., 2014]							
[Tang et al., 2014]			10		5[J]		
[Tompson et al., 2014]	11	7					
[Fleishman et al., 2015]							
[Li et al., 2015]							
[Oberweger et al., 2015a]	10		8				
[Oberweger et al., 2015b]	7	4					
[Poier et al., 2015]							
[Sharp et al., 2015]						2	5
[Sridhar et al., 2015]					4		
[Sun et al., 2015]			6	4			
[Tagliasacchi et al., 2015]		5			2		4
[Tang et al., 2015]		6	1[H]				
[Ge et al., 2016]				3			
[Sinha et al., 2016]	[10 – 11][M]						
[Taylor et al., 2016]		[2 – 3]			1	1	3
[Tkach et al., 2016]							1
[Wan et al., 2016]	5		4				
[Zhou et al., 2016]	9		9				
[Fourure et al., 2017]	8		5				
[Dibra et al., 2017]	[3 – 6][L]		[4 – 7][L]				
[Guo et al., 2017]	2		3[H]				
[Madadi et al., 2017]	6						
[Mueller et al., 2017]							
[Neverova et al., 2017]	4						
[Oberweger and Lepetit, 2017]	1	1	2[H]	1			
[Taylor et al., 2017]					3		2
[Tkach et al., 2017]		[2 – 3][N]					1[K]
[Wan et al., 2017]			7	2			

Table 1.2 – Comparative performance of hand tracking methods

2 Robust Articulated-ICP for Real-Time Hand Tracking

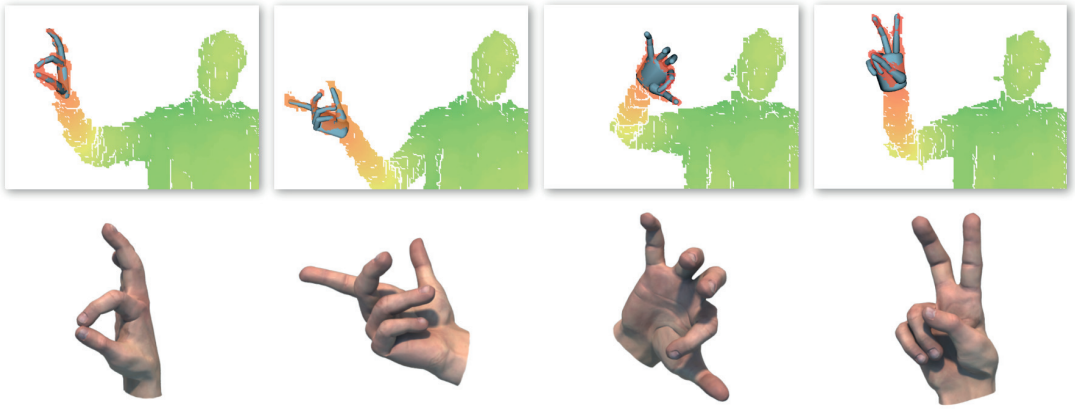


Figure 2.1 – Our system tracks the motion of hands while remaining robust to fast motion, sensor imperfections and self-occlusions.

This chapter is based on the following publication:

> TAGLIASACCHI A., SCHROEDER M., TKACH A., BOUAZIZ S., BOTSCH M., PAULY M.: Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum(Proc. of the Symposium on Geometry Processing) (2015)*.

The above publication also appears in the thesis of a co-author of the paper, Matthias Schroeder.

Abstract

We present a robust method for capturing articulated hand motions in real-time using a single depth camera. Our system is based on a real-time 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-

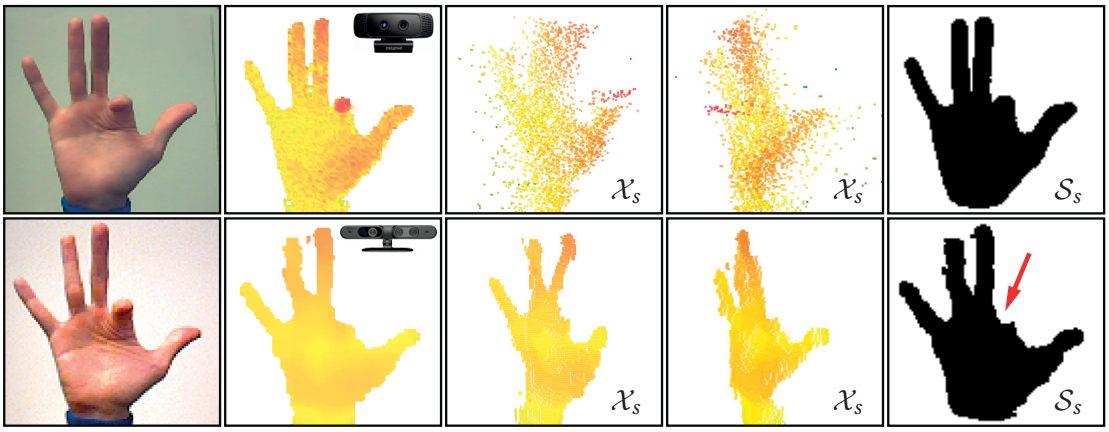


Figure 2.2 – The two different sensors used in our experiments provide data with substantially different characteristics. Top: Intel's Creative Interactive Gesture camera (time of flight) provides a complete silhouette image $\mathcal{S}_s$, but low quality depth measurements, resulting in severe noise in the point cloud $\mathcal{X}_s$. Bottom: Point clouds acquired by the PrimeSense camera (structured light) are much smoother, but the silhouette image can contain significant gaps.

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. We present a principled way of integrating such priors into our registration optimization to enable robust tracking without severely restricting the freedom of motion. A core technical contribution is a new method for computing tracking correspondences that directly models occlusions typical of single-camera setups.

2.1 Introduction

In this chapter we introduce a system for *real-time* hand tracking suitable for personal desktop environments. Our *non-invasive* setup using a single commodity RGBD sensor does not require the user to wear a glove or markers. Such single-camera acquisition is particularly advantageous as it is cheap, does not require any sensor calibration, and does not impede user movements.

Accurate hand tracking with a non-invasive sensing device in real-time is a challenging scientific problem. Human hands are highly articulated and therefore require models with sufficiently many degrees of freedom to adequately describe the corresponding motion space. Hand motion is often fast and exhibits intricate geometric configurations with complex contact patterns among fingers.

With a single-camera RGBD setup, we are faced with incomplete data due to self-occlusions and high noise levels (see Figure 2.2).

Yet the simplicity of the hardware and the ease of deployment make this setup the most promising for consumer applications as evidenced by the recent proliferation of new consumer-level

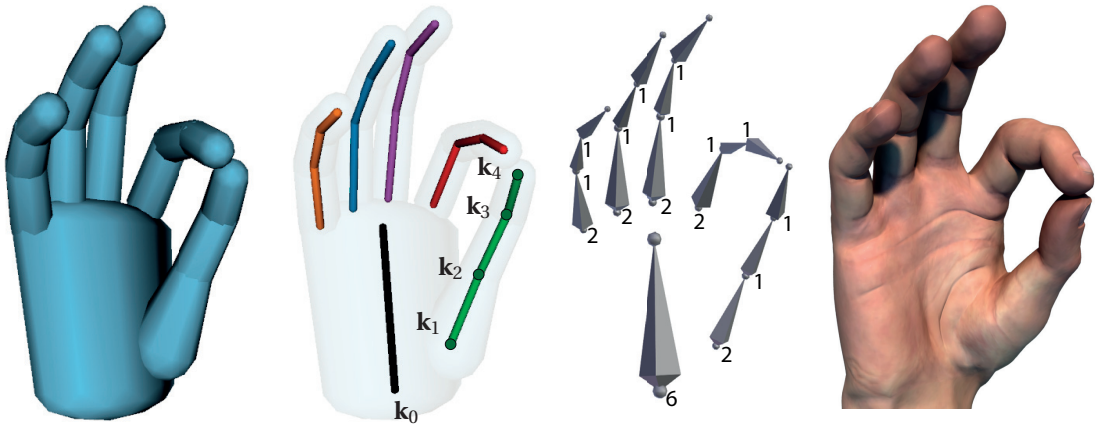


Figure 2.3 – A visualization of the template hand model with the number and location of degrees of freedom of our optimization. From left to right: The cylinder model used for tracking, the skeleton, the BVH skeleton exported to Maya to drive the rendering, the rendered hand model.

sensors.

To cope with the limited amount of available information, we employ an articulated template model as a geometric prior for shape completion and topology control. Our model does not only encode geometry, but also serves as a domain to represent information about *plausible* hand poses and motions. This statistical information, built by analyzing a database of annotated poses, is directly embedded into the optimization, which allows accurate tracking with a high number of degrees of freedom even in challenging scenarios.

Contributions

We present a complete system for real-time hand tracking using a single commodity RBGD input sensor. Our core technical contributions are:

- a novel articulated registration algorithm that efficiently integrates data and regularization priors into a *unified* real-time solver; see Section 2.3 and Appendix 2.7.6,
- a combined 2D/3D registration method to align the 3D hand model to the acquired depth map and extracted silhouette image; see Section 2.3.1,
- a new way of computing data-to-model correspondences that accounts for occlusions and significantly improves the robustness of the tracking; see Section 2.3.1,
- a new regularization strategy that combines a statistical pose-space prior with kinematic and temporal priors to simultaneously ensure the inferred hand poses are plausible and aid the algorithm in recovering from loss-of-tracking; see Section 2.3.2,

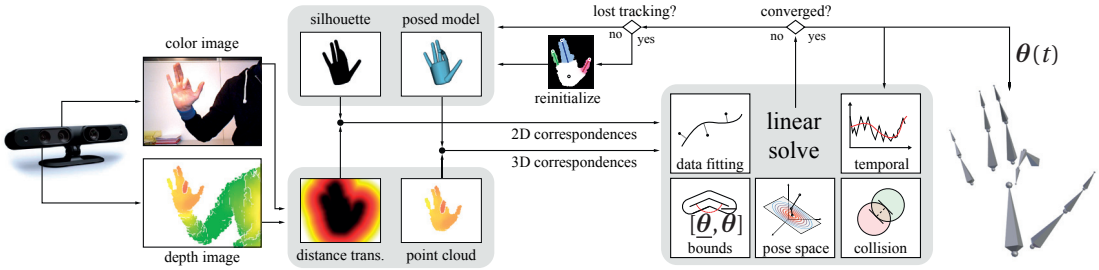


Figure 2.4 – Overview of our algorithm. For each acquired frame we extract a 3D point cloud of the hand and the 2D distance transform of its silhouette. From these we compute point correspondences to align a cylinder model of the hand to best match the data. This registration is performed in an ICP-like optimization that incorporates a number of regularizing priors to ensure accurate and robust tracking.

- exposing an interesting relationship between the well known point-to-plane registration energy and Gauss-Newton; see Appendix 2.7.4.

Another important contribution is that we fully disclose our source code[1]. To the best of our knowledge, no other freely available implementation is available, and we believe that publishing our code will not only ensure reproducibility of our results, but also facilitate future research in this domain.

Note that there is a widespread belief [Wei et al., 2012, Zhang et al., 2014, Qian et al., 2014] that ICP-like techniques are too local and prone to local minima to successfully deal with fast articulated motion. One of our contributions is to show this commonly held belief should be re-considered. We demonstrate that a regularized geometric registration approach in the spirit of ICP can achieve outstanding performance. We believe this will significantly impact future research in this domain, as it will allow further development of registration techniques for real-time tracking, in contraposition to commonly employed techniques from the vision community like discriminative [Tompson et al., 2014] and PSO [Qian et al., 2014] methods.

Our regularized geometric registration achieves robust, highly articulated hand tracking at up to 60 frames per second (fps). We quantitatively and qualitatively compare the performance of our algorithm to recent appearance-based and model-based techniques (see Section 2.5). These comparisons show a significant improvement in accuracy and robustness compared to the current state-of-the-art.

2.2 Overview

Robust hand tracking with a commodity depth sensor is highly challenging due to self-occlusion, low quality/density of sensor data and the high degree of articulation of the human hand. We address these issues by proposing a regularized articulated ICP-like optimization

[1] https://github.com/OpenGP/htrack

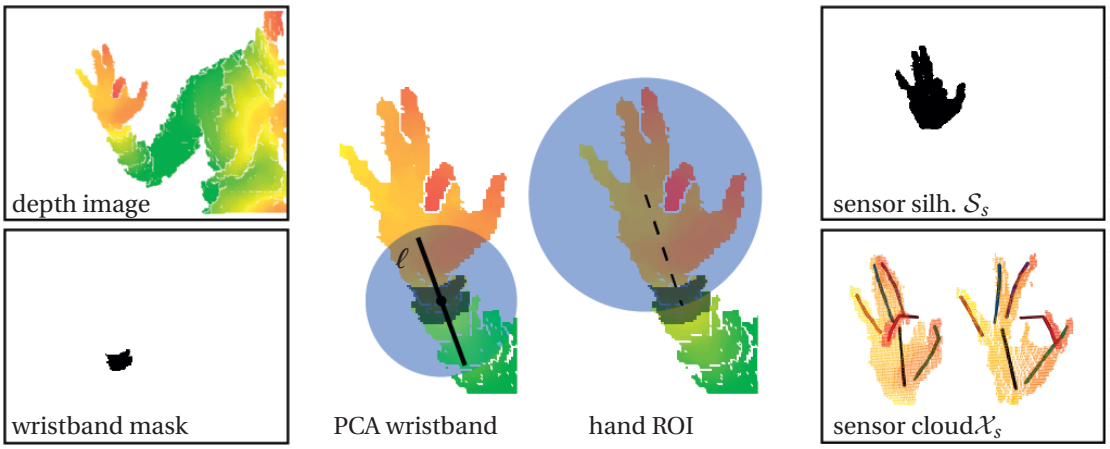


Figure 2.5 – We first identify the wristband mask by color segmentation, then compute the 3D orientation of the forearm as the PCA axis of points in its proximity. Offsetting a 3D sphere from the wristband center allows isolating the region of interest. The obtained silhouette image and sensor point clouds are shown on the right.

that carefully balances data fitting with suitable priors (Figure 2.4). Our data fitting performs a joint *2D-3D* optimization. The 3D alignment ensures that every point measured by the sensor is sufficiently close to the tracked model $\mathcal{M}$. Simultaneously, as we cannot create such constraints for occluded parts of the hand, we integrate a 2D registration that pushes the tracked model to lie within the sensor visual hull. A carefully chosen set of priors regularizes the solution to ensure the recovered pose is plausible.

Acquisition device

Our system processes raw data acquired at 60 fps from a single RGBD sensor. Figure 2.2 illustrates this data for the *PrimeSense Carmine 1.09* structured light sensor as well as the *Creative Gesture Camera* time-of-flight sensor. From the raw data our algorithm extracts a 2D *silhouette image* $\mathcal{S}_s$ and a 3D *point cloud* $\mathcal{X}_s$. The two sensors exhibit different types of imperfections. The precision of depth measurements in the PrimeSense camera is significantly higher. However, substantial holes often occur at grazing angles, e.g. note the gap in the data where we would expect to see the index finger. Conversely, the Creative Gesture Camera provides an accurate and gap-free silhouette image, but suffers from high noise in the depth measurements, therefore resulting in very noisy point clouds. Our algorithm is designed to handle both types of imperfections. This is achieved by formulating an optimization that *jointly* considers silhouette and point cloud, balancing their contribution in a way that conforms to the quality of sensor data.

Tracking model

Our algorithm registers a template hand model to the sensor data. Similar to other techniques [Oikonomidis et al., 2011, Schroder et al., 2014], we employ a simple (sphere capped) cylinder model as a geometric template; see Figure 4.10. We optimize for 26 degrees of freedom, 6 for global rotation and translation and 20 for articulation. Like in [Melax et al., 2013], the model can be quickly adjusted to the user by specifying global scale, palm size and finger lengths. In most scenarios, it is sufficient to perform a simple uniform scaling of the model. Such a coarse geometry is sufficient for hand tracking, as the signal-to-noise ratio for commercially available RGBD sensors is low for samples on the fingers when compared to the size of a finger. Furthermore, the computation of closest-point correspondences can be performed in closed form and in parallel, which is essential for real-time performance. The hand's palm region may be better approximated by geometries other than a cylinder, but we found using only cylinder primitives to work well for tracking in terms of accuracy and efficiency. Furthermore, it simplified the implementation as the same correspondence computation routine can be used for all primitives in the model. While the geometry of the model used for tracking remains coarse, our algorithm computes joint angles (including rigid transformation) in the widespread *BVH* motion sequence format; these can be used to drive a high-resolution skinned hand rig as illustrated in Figure 4.10-d.

Preprocessing

The silhouette image $\mathcal{S}_s$ is not directly available from the sensor and needs to be computed. This labeling can be obtained by extracting the sensor color image and performing a skin color segmentation [Oikonomidis et al., 2012, Schroder et al., 2014], or can be obtained directly from depth images by performing a classification with random decision forests [Tompson et al., 2014]. Another possibility is to exploit a full-body tracking algorithm [Shotton et al., 2011] and segment the hand according to the wrist position. For gestural tracking, where the hand is typically the closest object to the sensor [Qian et al., 2014], a black wristband can be used to simplify segmentation by creating a gap in the depth image. Similarly to this method, in our system the user wears a *colored wristband*. We first identify the position of the wristband in the scene by color segmentation, then retrieve the 3D points in the proximity of the wristband and compute the principal axis. This axis, in conjunction with the wristband centroid, is then used to segment the hand point cloud. Any depth pixel within the hand point cloud is labelled as belonging to the silhouette image $\mathcal{S}_s$ as shown in Figure 2.5.

2.3 Optimization

In this section we derive the objective functions of our model-based optimization method and provide the rationales for our design choices. Let $\mathcal{F}$ be the sensor input data consisting of a 3D point cloud $\mathcal{X}_s$ and 2D silhouette $\mathcal{S}_s$ (see Figure 2.2). Given a 3D hand model $\mathcal{M}$ with joint parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_{26}\}$, we aim at recovering the pose $\boldsymbol{\theta}$ of the user's hand, matching

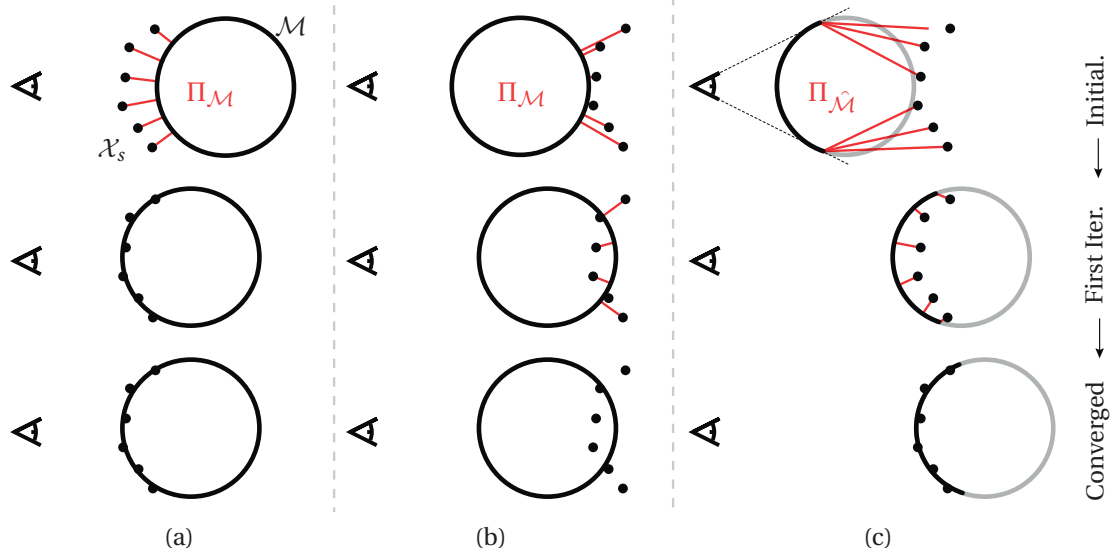


Figure 2.6 – Illustration of correspondences computations. The circles represent cross-sections of the fingers, the small black dots are samples of the depth map. (a) A configuration that can be handled by standard closest point correspondences. (b) Closest point correspondences to the back of the cylinder model can cause the registration to fall into a local minimum. Note that simply pruning correspondences with back-pointing normals would not solve this issue, as no constraints would remain to pull the finger towards the data. (c) This problem is resolved by taking visibility into account, and computing closest points only to the portion $\hat{\mathcal{M}}$ of $\mathcal{M}$ facing the camera.

the sensor input data $\mathcal{F}$. To achieve this goal, we solve the optimization problem

$$\min_{\boldsymbol{\theta}} \underbrace{E_{\text{3D}} + E_{\text{2D}} + E_{\text{wrist}}}_{\text{Fitting terms}} + \underbrace{E_{\text{pose}} + E_{\text{kin.}} + E_{\text{temporal}}}_{\text{Prior terms}}, \tag{2.1}$$

combining fitting terms that measure how well the hand parameters $\boldsymbol{\theta}$ represent the data frame $\mathcal{F}$, with prior terms that regularize the solution to ensure realistic hand poses. For brevity of notation we omit the arguments $\boldsymbol{\theta}, \mathcal{X}_s, \mathcal{S}_s$ of the energy terms. We first introduce the fitting terms and present our new solution to compute tracking correspondences. Then we discuss the prior terms and highlight their benefits in terms of tracking accuracy and robustness. More details on the implementation of the optimization algorithm will be given in Section 2.4 and the Section 2.7.

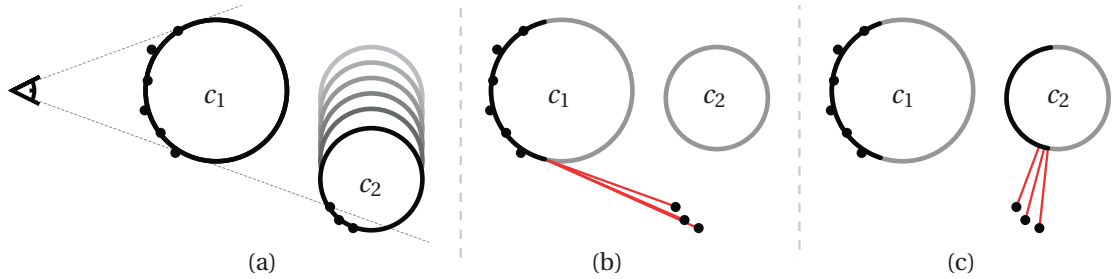


Figure 2.7 – Illustration of the impact of self-occlusion in correspondences computations. (a) The finger c_2 initially occluded by finger c_1 becomes visible, which causes new samples to appear. (b) Closest correspondences to the portion of the model visible from the camera do not generate any constraints that pull c_2 toward its data samples. This is the approach in [Wei et al., 2012], where these erroneous matches are then simply pruned. (c) Our method also considers front-facing portions of the model that are occluded, allowing the geometry to correctly register.

2.3.1 Fitting Energies

Point cloud alignment

The term E_{3D} models a 3D geometric registration in the spirit of ICP as

$$E_{3D} = \omega_1 \sum_{\mathbf{x} \in \mathcal{X}_s} \|\mathbf{x} - \Pi_{\mathcal{M}(\boldsymbol{\theta})}(\mathbf{x}, \boldsymbol{\theta})\|_2^1, \tag{2.2}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm, $\mathbf{x}$ represents a 3D point of $\mathcal{X}_s$, and $\Pi_{\mathcal{M}(\boldsymbol{\theta})}(\mathbf{x}, \boldsymbol{\theta})$ is the projection of $\mathbf{x}$ onto the hand model $\mathcal{M}$ with hand pose $\boldsymbol{\theta}$. Note that we compute a sum of absolute values of the registration residuals, not their squares. This corresponds to a mixed ℓ_2/ℓ_1 norm of the stacked vector of the residuals. For 3D registration such a sparsity-inducing norm has been shown to be more resilient to noisy point clouds containing a certain amount of outliers such as the ones produced by the Creative sensor (Figure 2.2). We refer to [Bouaziz et al., 2014] for more details.

3D correspondences

The 3D registration term involves computing the corresponding point $\mathbf{y} = \Pi_{\mathcal{M}(\boldsymbol{\theta})}(\mathbf{x}, \boldsymbol{\theta})$ on the cylinder model $\mathcal{M}$ for each sensor point $\mathbf{x} \in \mathcal{X}_s$. In contrast to standard closest point search, we define the correspondence $\mathbf{y}$ as the closest point on the *front-facing* part $\hat{\mathcal{M}}$ of $\mathcal{M}$. This includes parts of the model that are oriented towards the camera but occluded by other parts. In our experiments we learned that this seemingly simple extension proved absolutely essential to obtain high-quality tracking results. Only considering model points that are visible from the sensor viewpoint, i.e., matching to the rendered model, is not sufficient for handling occlusions or instances of disappearing and reappearing sensor data; see Figure 2.6

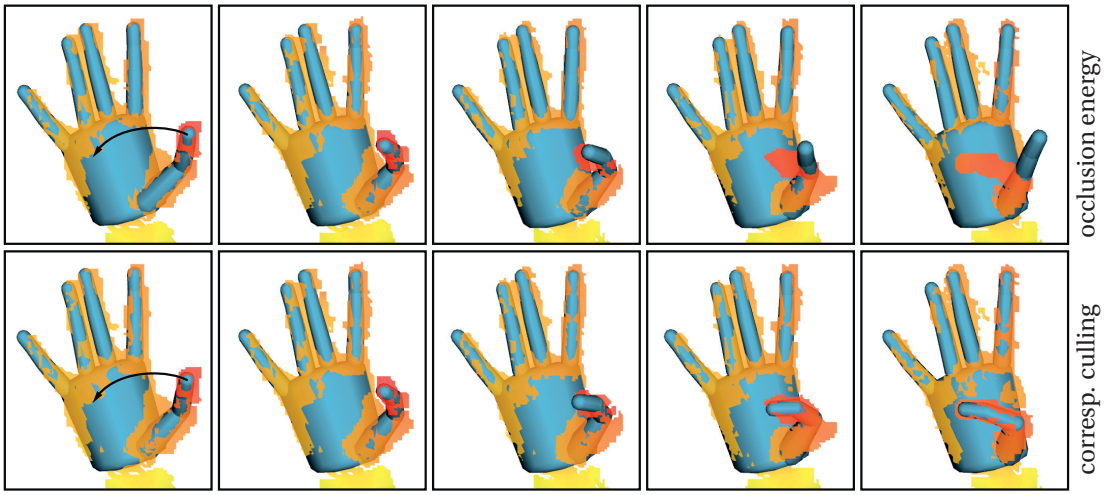


Figure 2.8 – Correspondence computations. The top row shows the strategy used in [Qian et al., 2014] adapted to our gradient-based framework according to the formulation given in [Wei et al., 2012]. The bottom row shows the improved accuracy of our new approach.

and Figure 2.7.

To calculate $\mathbf{y}$, we first compute the closest points $\mathbf{x}_{\mathcal{C}}$ of $\mathbf{x}$ to each cylinder $\mathcal{C} \in \mathcal{M}$. Recall that our hand model consists of sphere-capped cylinders so these closest points can be computed efficiently in closed form and in parallel for each $\mathbf{x} \in \mathcal{X}_s$. We then identify back-facing points using the dot product of the cylinder surface normal $\mathbf{n}$ at $\mathbf{x}_{\mathcal{C}}$ and the view ray vector $\mathbf{v}$. For efficiency reasons, we use a simplified orthographic camera model where the view rays are constant, i.e., $\mathbf{v} = [0\ 0\ 1]^T$. If a point on a cylinder is back-facing ($\mathbf{n}^T\mathbf{v} > 0$), we project $\mathbf{x}$ onto the cylinder's silhouette contour line from the camera perspective, whose normals are orthogonal to $\mathbf{v}$.

A different strategy to address visibility issues has been introduced in [Qian et al., 2014]. These methods propose an energy that penalizes areas of the model falling in front of the data, which is then optimized using particle swarms. This energy can be integrated into our optimization following the formulation in [Wei et al., 2012, Eq. 15]. However, such an energy is prone to create local minima in gradient-based optimization, as illustrated in Figure 2.8. Here the thumb has difficulty entering the palm region, as it must occlude palm samples before reaching its target configuration. Our correspondence search avoids such problems. Furthermore, note how [Qian et al., 2014] follows a *hypothesize-and-test* paradigm where visibility constraints in the form of *ray-casting* are easy to include. As discussed in [Ganapathi et al., 2012], such constraints are much more difficult to include in iterative optimization techniques like ours. However, our front-facing correspondences computation provides a simple and elegant way to deal with such shortcomings.

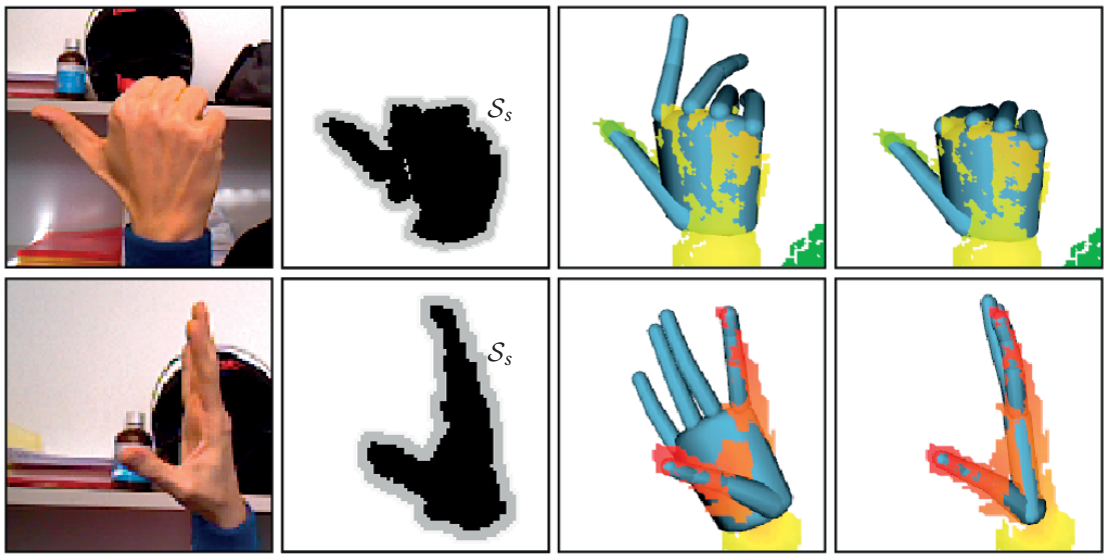


Figure 2.9 – Our 2D silhouette registration energy is essential to avoid tracking errors for occluded parts of the hand. When no depth data is available for certain parts of the model, a plausible pose is inferred by ensuring that the model is contained within the sensor silhouette image $\mathcal{S}_s$.

Silhouette alignment

The 3D alignment energy E_{3D} robustly measures the distance between every point in the 3D point cloud $\mathcal{X}_s$ to the tracked model $\mathcal{M}$. However, as hands are highly articulated, significant self-occlusions are common during tracking. Such self-occlusions are challenging, because occluded parts will not be constrained when only using a 3D alignment energy. For this reason, we use a 2D silhouette term E_{2D} that models the alignment of the 2D silhouette of our rendered hand model with the 2D silhouette extracted from the sensor data as

$$E_{2D} = \omega_2 \sum_{\mathbf{p} \in \mathcal{S}_r} \|\mathbf{p} - \Pi_{\mathcal{S}_s}(\mathbf{p}, \boldsymbol{\theta})\|_2^2, \tag{2.3}$$

where $\mathbf{p}$ is a 2D point of the *rendered* silhouette $\mathcal{S}_r$, and $\Pi_{\mathcal{S}_s}(\mathbf{p}, \boldsymbol{\theta})$ denotes the projection of $\mathbf{p}$ onto the *sensor* silhouette $\mathcal{S}_s$. Figure 2.9 shows why the silhouette term is crucial to avoid erroneous poses when parts of the model are occluded. Without the silhouette energy, the occluded fingers can mistakenly move to wrong locations, since they are not constrained by any samples in the depth map.

2D correspondences

In Equation 2.3, we compute the silhouette image $\mathcal{S}_r$ by first rendering the hand model $\mathcal{M}$ from the viewpoint of the sensor, caching the bone identifier and the 3D location associated with

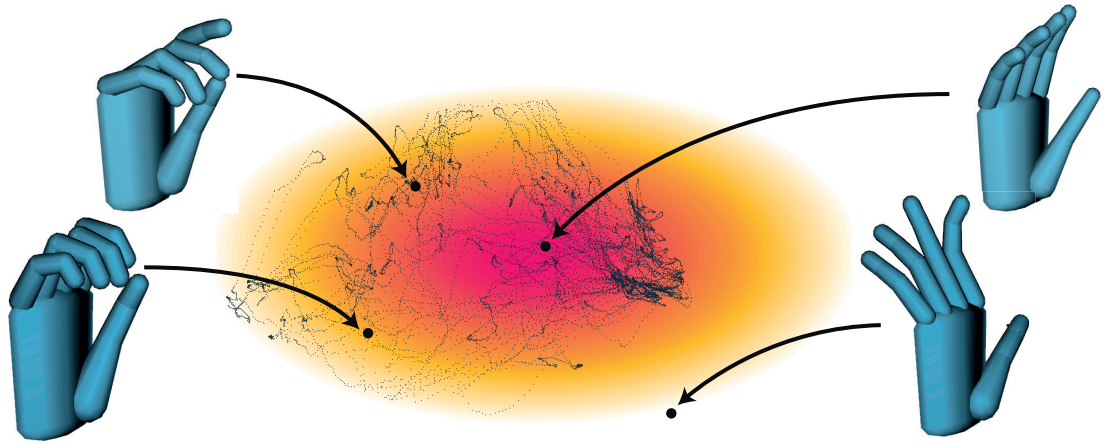


Figure 2.10 – An illustration of the PCA pose-space used to regularize the optimization. Black dots denote the samples of the data base. High likelihood poses are located nearby the mean of the latent space (dark red). The eigenvalues of the PCA define the metric in the low-dimensional space, skewing it in certain directions. Poses that, according to this metric, are far from the mean are likely to be unnatural and will be penalized in the optimization.

each pixel in a texture. The projection function $\Pi_{\mathcal{S}_s}(\mathbf{p},\boldsymbol{\theta})$ to compute the closest corresponding point to the sensor silhouette is evaluated efficiently using the 2D distance transform of $\mathcal{S}_s$. We use the linear time algorithm of [Felzenszwalb and Huttenlocher, 2012] and store at every pixel the index to the closest correspondence.

Wrist alignment

The inclusion of the forearm for hand tracking has been shown beneficial in [Melax et al., 2013]. Our wrist alignment energy encodes a much simplified notion of the forearm in the optimization that enforces the wrist joint to be located along its axis.

$$E_{\text{wrist}} = \omega_3 \|\Pi_{2D}(\mathbf{k}_0(\boldsymbol{\theta})) - \Pi_\ell(\mathbf{k}_0(\boldsymbol{\theta}))\|_2^2, \tag{2.4}$$

Minimizing this energy helps preventing the hand from erroneously rotating/flipping during tracking; an occurrence of this can be observed at Video1 [04:03][2]. Here $\mathbf{k}_0$ is the 3D position of the wrist joint, and ℓ is the 2D line extracted by PCA of the 3D points associated with the wristband; see Figure 2.5. Note that Π_{2D} causes residuals to be minimized in screen-space, therefore the optimization of this energy will be analogous to the one of Equation 2.3. We optimize in screen space because, due to occlusion, we are only able to observe half of the wrist and this causes its axis to be shifted toward the camera.

[2] Please find the accompanying Video1 at http://lgg.epfl.ch/publications/2015/Htrack_ICP/new_video.mp4 .

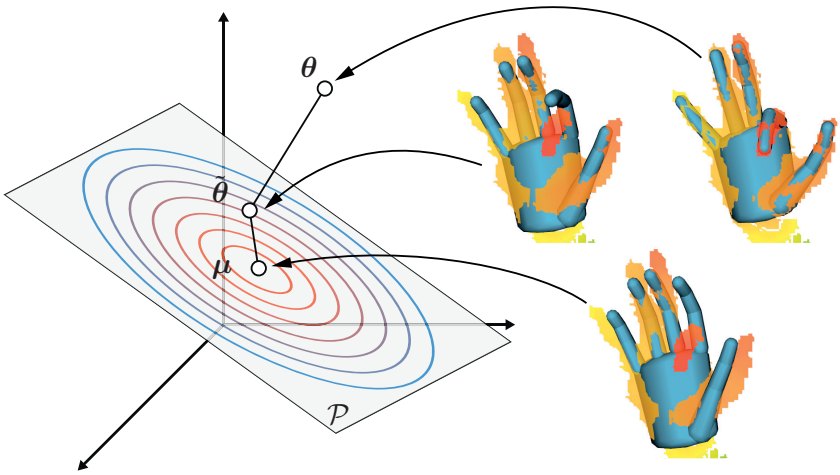


Figure 2.11 – An illustration of the energies involved in our pose-space prior. For illustration purposes the full dimensional parameter vector $\boldsymbol{\theta} \in \mathbb{R}^3$, while latent space variable $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^2$. The PCA optimization in [Schroder et al., 2014] constrains the pose parameters $\boldsymbol{\theta}$ to lie on the subspace $\mathcal{P}$. Conversely, we penalize the distance of our pose from $\mathcal{P}$ (Equation 2.5); simultaneously, we ensure our pose remains likely by preventing it from diverging from the mean of the distribution (Equation 2.6).

2.3.2 Prior Energies

Minimizing the fitting energies alone easily leads to unrealistic or unlikely hand poses, due to the deficiencies in the input data caused by noise, occlusions, or motion blur. We therefore regularize the registration with data-driven, kinematic, and temporal priors to ensure that the recovered hand poses are plausible. Each of these terms plays a fundamental role in the stability of our tracking algorithm, as we illustrate below.

Pose Space Prior (data-driven)

The complex and highly coupled articulation of human hands is difficult to model directly with geometric or physical constraints. Instead, we use a publicly available database of recorded hand poses [Schroder et al., 2014] to create a data-driven prior E_{pose} that encodes this coupling. We construct a low-dimensional subspace of plausible poses by performing dimensionality reduction using PCA (see Figure 2.10). We enforce the hand parameters $\boldsymbol{\theta}$ to lie close to this low-dimensional linear subspace using a data term $E_{\text{pose}} = E_{\text{projection}} + E_{\text{mean}}$. To define the data term, we introduce auxiliary variables $\tilde{\boldsymbol{\theta}}$, i.e, the PCA weights, representing the (not necessarily orthogonal) projection of the hand pose $\boldsymbol{\theta}$ onto the subspace; see Figure 2.11. The projection energy measures the distance between the hand parameters and the linear subspace as

$$E_{\text{projection}} = \omega_4 \|(\boldsymbol{\theta} - \boldsymbol{\mu}) - \Pi_{\mathcal{P}} \tilde{\boldsymbol{\theta}}\|_2^2, \tag{2.5}$$

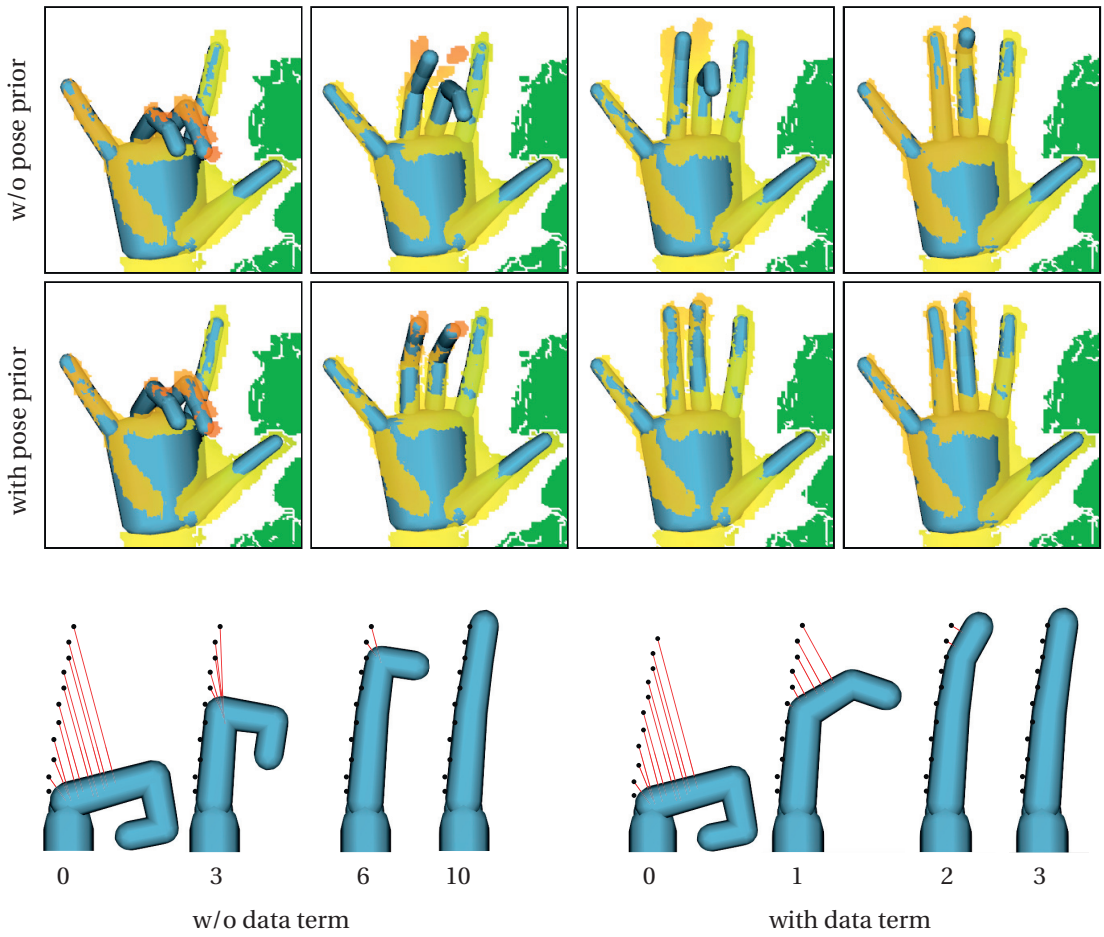
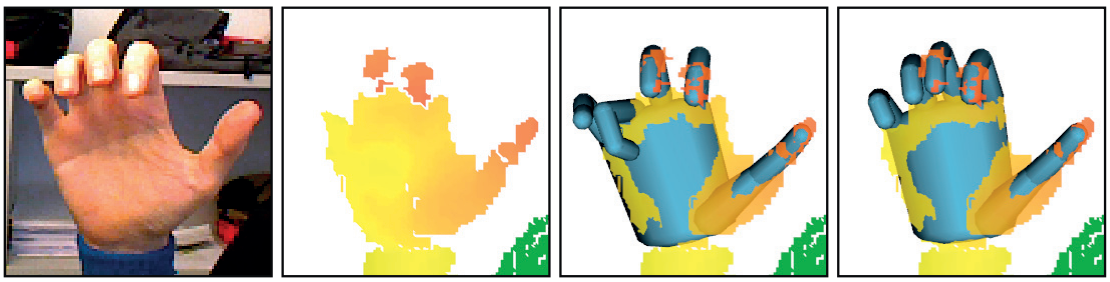


Figure 2.12 – Beyond favoring natural poses, the data prior term also positively affects convergence speed. Top: With the same number of iterations, only with activated data term does the model fully register to the scan. The illustration below shows how the same final state requires significantly fewer iterations with the data term.

where $\boldsymbol{\mu}$ is the PCA mean. The matrix $\Pi_{\mathcal{P}}$, i.e., the PCA basis, reconstructs the hand posture from the low-dimensional space. To avoid unlikely hand poses in the subspace, we regularize the PCA weights $\tilde{\boldsymbol{\theta}}$ using an energy

$$E_{\text{mean}} = \omega_5 \|\boldsymbol{\Sigma}\tilde{\boldsymbol{\theta}}\|_2^2. \tag{2.6}$$

$\boldsymbol{\Sigma}$ is a diagonal matrix containing the inverse of the standard deviation of the PCA basis. Our tracking optimization is modified to consider the pose space by introducing the auxiliary variable $\tilde{\boldsymbol{\theta}}$ and then jointly minimizing over $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{\theta}}$. The difference between our approach and optimizing directly in the subspace is further discussed in Section 2.7. Note how the regularization energy in Equation 2.6 helps the tracking system recover from tracking failures. When no sensor constraints are imposed on the model, the optimization will attempt to push the pose towards the mean – a statistically likely pose from which tracking recovery is highly



depth image | without PCA | with PCA

Figure 2.13 – Our pose-space regularization using a PCA prior ensures that a meaningful pose is recovered even when significant holes occur in the input data.

effective.

Figure 2.13 illustrates how the PCA data prior improves tracking by avoiding unlikely poses, in particular when the input data is incomplete. We found that even when data coverage is sufficient to recover the correct pose, the data term improves the convergence of the optimization as illustrated in Figure 2.12. Figure 2.14 shows how our regularized projective PCA formulation outperforms the direct subspace optimization proposed in previous work.

Kinematic Prior

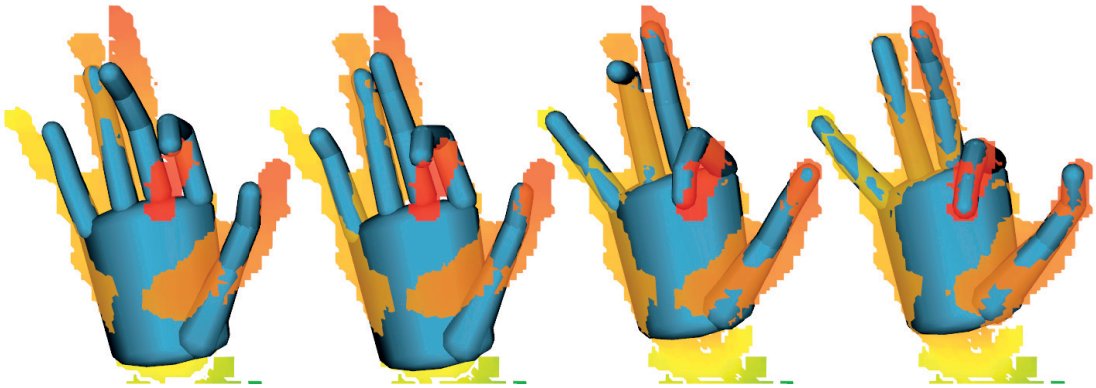
The PCA data term is a computationally efficient way of approximating the space of plausible hand poses. However, the PCA model alone cannot guarantee that the recovered pose is realistic. In particular, since the PCA is symmetric around the mean, fingers bending backwards beyond the physically realistic joint angle limits are not penalized by the data prior. Similarly, the PCA model is not descriptive enough to avoid self-intersections of fingers. These two aspects are addressed by the kinematic prior $E_{\text{kinematic}} = E_{\text{collision}} + E_{\text{bounds}}$. Under the simplifying assumption of a cylinder model, we can define an energy $E_{\text{collision}}$ that accounts for the inter-penetration between each pair of (sphere-capped) cylinders:

$$E_{\text{collision}} = \omega_6 \sum_{\{i,j\}} \chi(i,j)(d(\mathbf{c}_i, \mathbf{c}_j) - r)^2, \tag{2.7}$$

where the function $d(\cdot,\cdot)$ measures the Euclidean distance between the cylinders axes $\mathbf{c}_i$ and $\mathbf{c}_j$, and r is the sum of the cylinder radii. $\chi(i,j)$ is an indicator function that evaluates to one if the cylinders i and j are colliding, and to zero otherwise. The top row of Figure 2.15 shows how this term avoids interpenetrations of the fingers.

To prevent the hand from reaching an impossible posture by overbending the joints, we limit the joint angles of the hand model:

$$E_{\text{bounds}} = \omega_7 \sum_{\theta_i \in \boldsymbol{\theta}} \underline{\chi}(i)(\theta_i - \underline{\theta}_i)^2 + \overline{\chi}(i)(\theta_i - \overline{\theta}_i)^2, \tag{2.8}$$



89%, #PCA=4 $\omega_4 = 10^8$ | 96%, #PCA=6 $\omega_4 = 10^8$ | 99%, #PCA=9 $\omega_4 = 10^8$ | 79%, #PCA=2 $\omega_4 = 10^2$

Figure 2.14 – Optimizing directly in the PCA subspace [Schroder et al., 2014] can lead to inferior registration accuracy. We replicate this behavior by setting ω_4 in Equation 2.5 to a large value. Even when increasing the number of PCA bases to cover 99% of the variance in the database, the model remains too stiff to conform well to the input. Our approach is able to recover the correct hand pose by optimizing for projection distances even with a very limited number of bases (right).

where each hand joint is associated with conservative bounds $\left[\underline{\theta}_i, \overline{\theta}_i\right]$. For the bounds, we use the values experimentally determined by [Chan and Dubey, 1995]. $\underline{\chi}(i)$ and $\overline{\chi}(i)$ are indicator functions. $\underline{\chi}(i)$ evaluates to one if $\theta_i < \underline{\theta}_i$, and to zero otherwise. $\overline{\chi}(i)$ is equal to one if $\theta_i > \overline{\theta}_i$, and zero otherwise. The bottom row of Figure 2.15 illustrates the effect of the joint angle bounds.

Temporal Prior

A common problem in particular with appearance-based methods are small-scale temporal oscillations that cause the tracked hand to jitter. A standard way to enforce temporal smoothness is to penalize the change of model parameters $\boldsymbol{\theta}$ through time, for example, by penalizing a quadratic energy accounting for velocity $\|\dot{\boldsymbol{\theta}}\|^2$ and acceleration $\|\ddot{\boldsymbol{\theta}}\|^2$ [Wei et al., 2012]. However, if we consider a perturbation of the same magnitude, it would have a much greater effect if applied at the root, e.g., global rotation, than if applied to an element further down the kinematic tree, e.g., the last phalanx of a finger. Therefore, we propose a solution that measures the velocity and acceleration of a set of points attached to the kinematic chain. We consider the motion of vertices $\mathbf{k}$ of the kinematic chain $\mathcal{K}$ (Figure 4.10) and build an energy penalizing the velocity and acceleration of these points:

$$E_{\text{temporal}} = \omega_8 \sum_{\mathbf{k}_i \in \mathcal{K}} \|\dot{\mathbf{k}}(\boldsymbol{\theta})\|_2^2 + \omega_9 \sum_{\mathbf{k}_i \in \mathcal{K}} \|\ddot{\mathbf{k}}(\boldsymbol{\theta})\|_2^2. \tag{2.9}$$

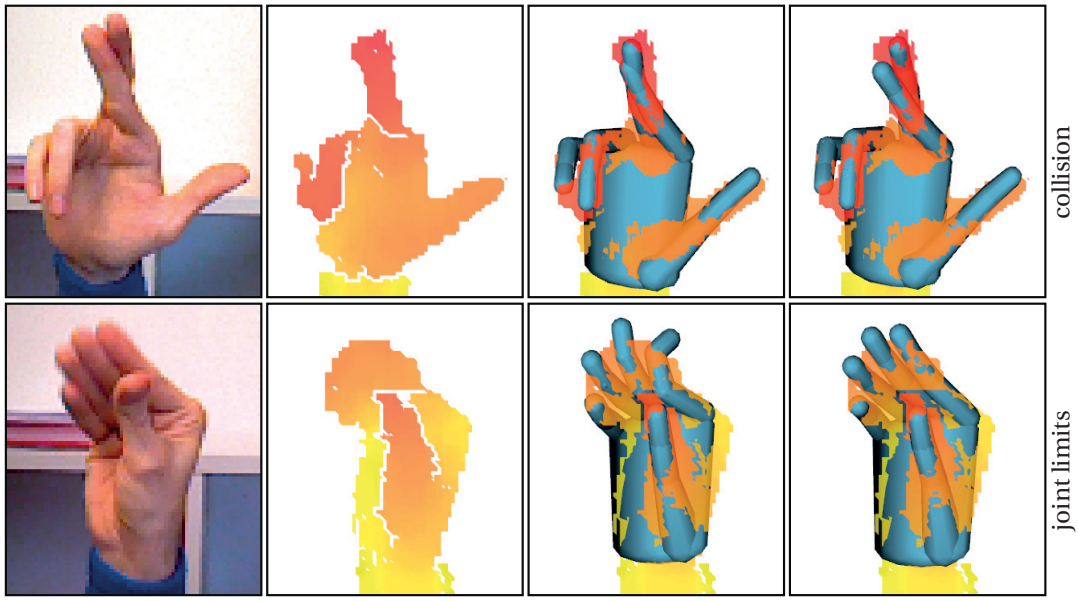


Figure 2.15 – Kinematic priors augment the data prior to account for inconsistencies in the pose space. The collision term avoids self-collisions (top row), while the term for joint angle bounds avoids overbending of the finger joints.

Figure 2.16 illustrates how the temporal prior reduces jitter and improves the overall robustness of the tracking; see also Video1 [01:20].

2.4 Implementation

In this section we provide more details on the implementation of our optimization algorithm. The derivation of the necessary gradients and Jacobians is given in the Section 2.7.

Optimization

The optimization of the tracking energy of Equation 2.1 over the pose θ is performed by solving the non-linear least squares problem with a Levenberg-Marquardt approach. The assumption is that a current estimate of θ is known from which we then compute an update. More specifically, the high acquisition speed of the sensing device allows us to employ the optimized parameters from the previous time frame as the starting estimate. We then iteratively approximate the energy terms using Taylor expansion and solve a linear system to get the update $\delta\theta$ at each iteration (see appendix). As our algorithm achieves 60 fps tracking, the previously reconstructed pose is of sufficiently high quality allowing our solver to converge within seven iterations.

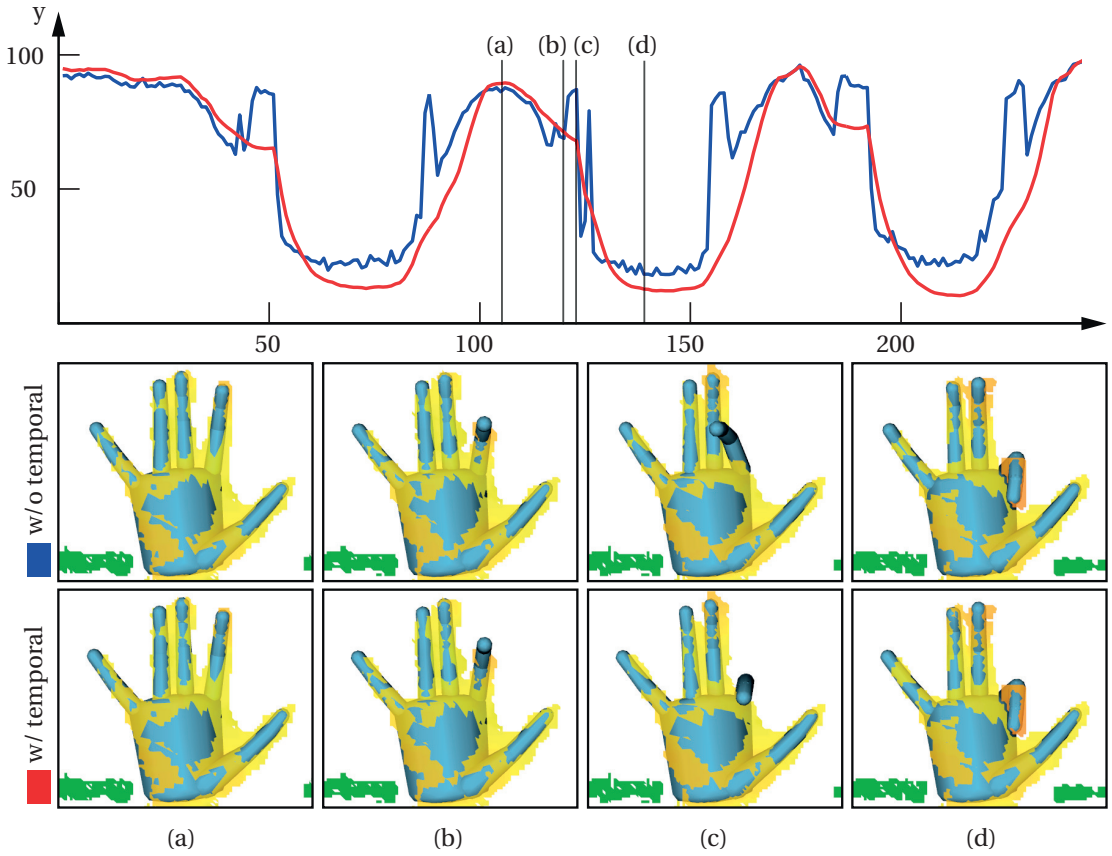


Figure 2.16 – The effect of the temporal prior. The graph shows the trajectory of the y-coordinate of the fingertip over time as the index finger is bend up and down repeatedly. The temporal prior reduces jitter, but also helps avoiding tracking artifacts that arise when fragments of data pop in and out of view.

Initialization

As a user enters the scene our method is initialized by the fingertip detection and fitting from [Qian et al., 2014]. Other appearance-based methods could be used for initialization as well [Tompson et al., 2014]. We also re-initialize the tracking in case a severe tracking failure is detected using the method of [Wei et al., 2012]. Such re-initialization occurs rarely (e.g. less than 0.5% of the frames in the sequence of Figure 2.22).

Rigid bias

To improve the convergence of our solver in case of fast motion, we first perform the optimization in Equation 2.1 for the rigid motion only by optimizing for the root of the kinematic chain. As shown in Figure 2.17, optimizing first for the rigid motion prior to the full pose estimation leads to improved robustness of the tracking.

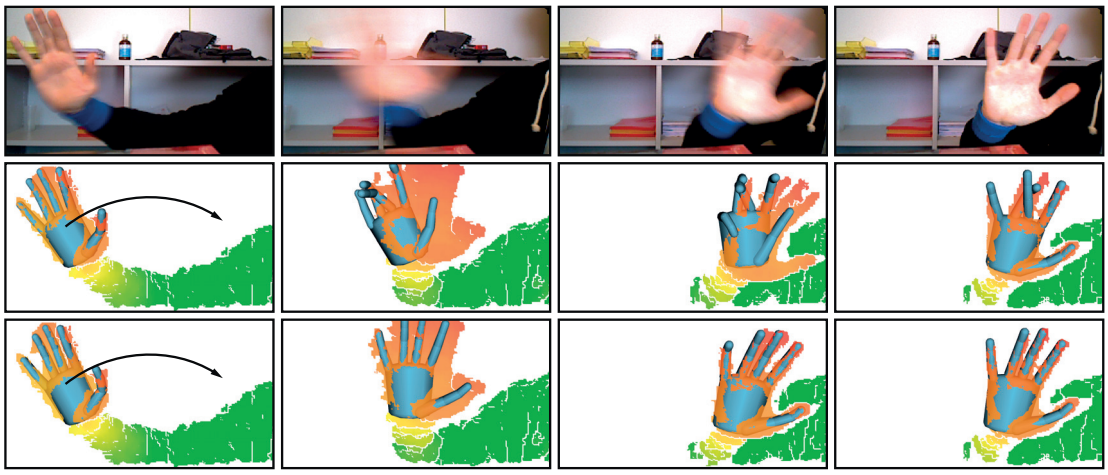


Figure 2.17 – During fast motion, optimizing directly for a fully articulated hand can lead to incorrect correspondences and cause loss of tracking (middle row). By compensating for the rigid motion ahead of solving for joint angles, our system can better capture fast movements (bottom row).

Parameters

For all our results we fix our parameters to $\omega_1 = \omega_2 = \omega_5 = 1$, $\omega_4 = 10^3$, $\omega_3 = \omega_6 = \omega_7 = 10^8$, $\omega_8 = \omega_9 = 3$. We determined these weights empirically by re-tracking multiple sequences with different sets of parameters. Our system was tested on an Intel Core i7 4GHz with NVIDIA GTX980 GPU running Ubuntu 12.02 . To run on a 60Hz RGBD device such as the PrimeSense Carmine 1.09 or the Creative Gesture Camera, we perform 1 rigid iteration and 7 full iterations, at 1.5ms per iteration. We perform closed form closest point correspondences and Jacobian computation for the fitting energies on the GPU. The number of iterations can be easily adapted to run on the new Intel RealSense 3D Camera (F200) at 120Hz or at even higher frame rates on future devices.

2.5 Evaluation

We refer to Video1 [06:20]to best appreciate the real-time tracking performance of our method. Here we analyze its performance by providing a comparison to several state-of-the art solutions.

Dexter-1 Dataset [SRS*14]

Figure 2.18 shows a quantitative comparison with several existing methods on a publicly available data set acquired at 25 Hz. As the graph illustrates, our solution clearly outperforms the method of [Tang et al., 2014] that uses regression forest classifiers in an appearance-

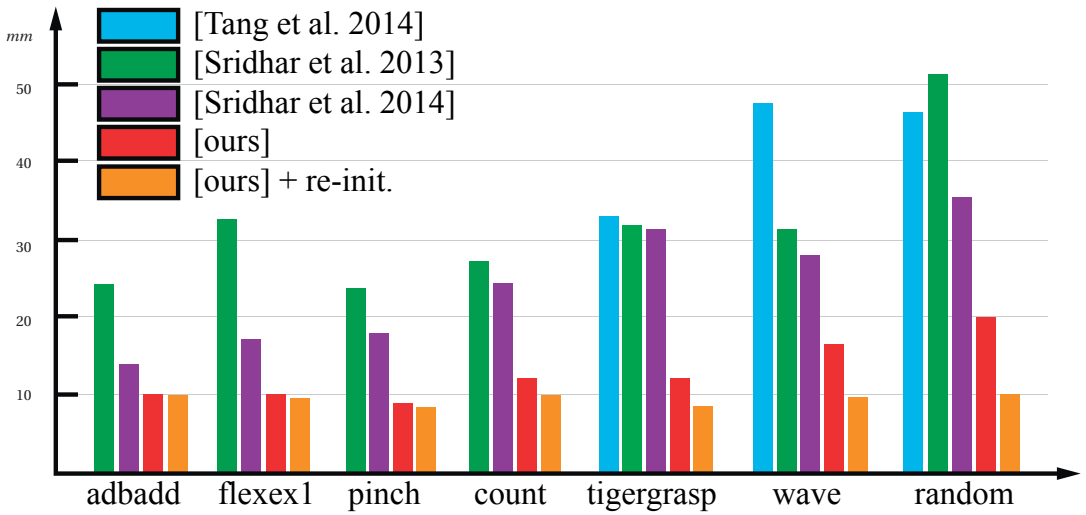


Figure 2.18 – We quantitatively evaluate our algorithm on the **Dexter-1** dataset from [Sridhar et al., 2013]. The measurements report the root mean square errors of fingertip placements. The acquisition setup consists of several calibrated video cameras and a single depth camera. For our results and the method of [Tang et al., 2014], only the depth image is used for tracking, while the algorithms of Sridhar and colleagues also use the video streams. The blue, green, and purple bars are reproduced from [Sridhar et al., 2014]. For our algorithm we report results without (red) and with (orange) reinitialization.

based approach to estimate hand poses. We also significantly improve upon the gradient-based optimization methods of [Sridhar et al., 2013, Sridhar et al., 2014] that, in addition to the depth information, use RGB data from five additional video cameras. As the dataset is acquired at 25 Hz, the performance of our algorithm (red) is suboptimal. In particular, in a single frame fingers are occasionally displaced by 2 to 3 times their radii, thus corrupting ICP correspondences. By re-initializing with finger detection as in [Qian et al., 2014] our performance considerably improves, as shown in the figure.

Subspace ICP [SMRB14]

Figure 2.19 shows a comparison to the model-based approach of [Schroder et al., 2014]. The recorded sequences were directly processed by the authors and employed to pose our cylinder model for ease of comparison. As the figure illustrates, our method clearly outperforms this previous work. A key difference is that they optimize directly in a PCA subspace, which tends to over-constrain the solution, while we introduce a PCA data term as a regularizer, which preserves the full expressiveness of the tracking model. In addition, we introduce collision handling, apply robust norms for automatic outlier detection, and employ a more advanced correspondence search that handles self-occlusions. In combination, these factors lead to substantial improvements in tracking accuracy and robustness without compromising computational efficiency.

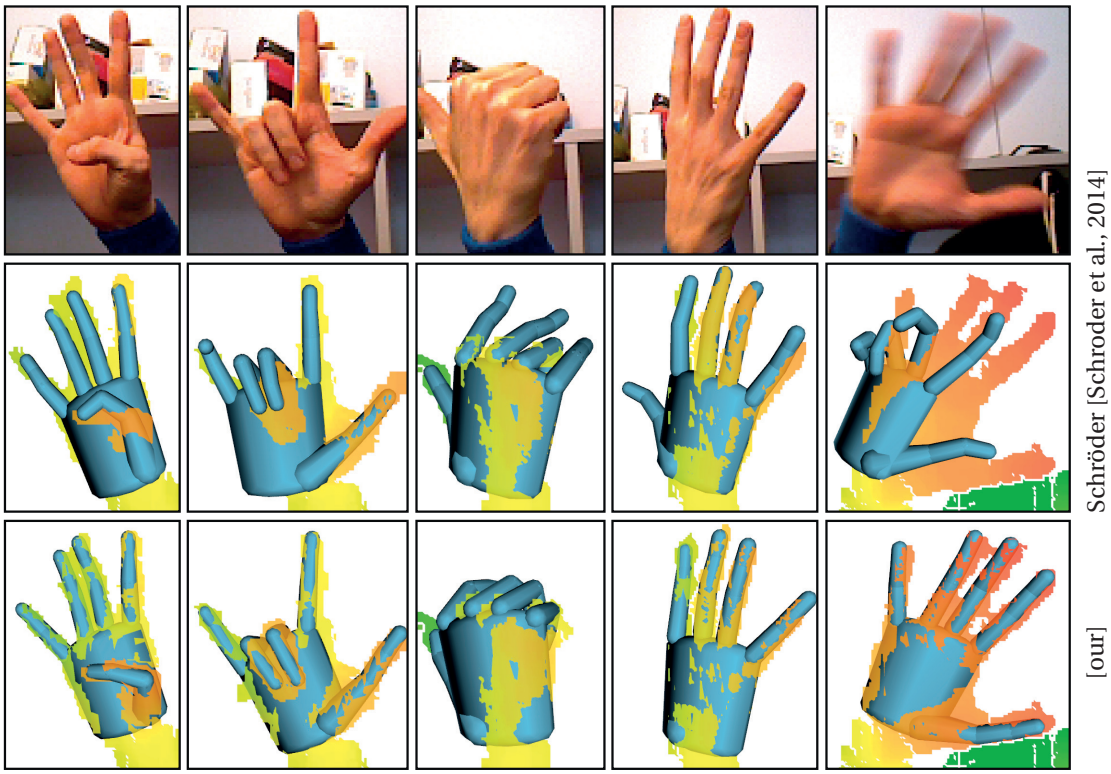


Figure 2.19 – A few comparison frames illustrating the difference in performance of our method compared to [Schroder et al., 2014] (results provided by the authors of that paper). From left to right we can observe problems related to: correspondences to the back of the model, lack of silhouette energy (3 times) and loss of tracking due to fast motion.

Convex body solver [MKO13]

We compare to this algorithm by employing the precompiled binaries from the Intel Perceptual Computing SDK. We modifed the demo application to save the recorded depth/color frames to disk while tracking. We then re-tracked this data from scratch using our technique. As illustrated in Video1 [05:20], as well as Figure 2.20, our method offers a substantial increase in tracking robustness compared to [Melax et al., 2013]. This can be attributed to any of the improvements we presented, but it is difficult to quantitatively identify the causes, because no control on tracking parameters nor source code is given. Their approach computes closest correspondences to the entire model, therefore not explicitly handling occlusion. The authors also proposed a technique to ensure that the model is fully contained in the 3D convex hull of the data. Note that in camera space, this amounts to constraints similar to the ones enforced by our 2D registration (Equation 2.3), except that the distance transform would be computed from the 2D convex hull of the silhouette image. Figure 2.20 (Frame 448) illustrates how our 2D registration better constrains feasible solutions. While in [Melax et al., 2013] correlation between fingers is manually introduced as a *grasping bias*, our optimization is data driven and encodes correlation in a more principled way. As illustrated in Figure 2.20 and Video1 [05:20],

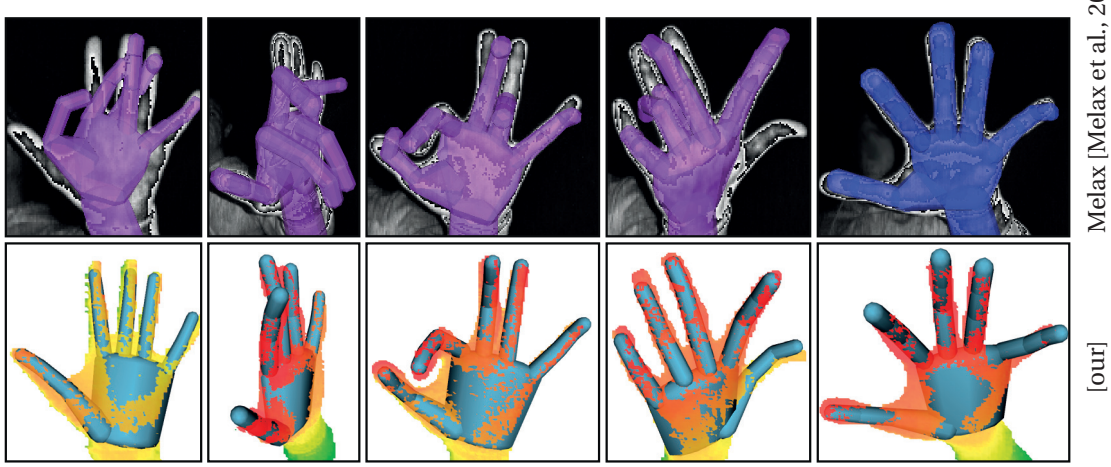


Figure 2.20 – Comparison to the method of [Melax et al., 2013]. The full sequence can be seen in the Video1 [05:20]. We highlight a few frames that are not resolved correctly by this method, but that can be handled successfully with our solution. The last frame shows the better geometric approximation quality of the convex body model used in [Melax et al., 2013] compared to our simpler cylinder model.

this approach often loses tracking during complex motion. However, it is sometimes capable of recovering by sampling and then evaluating a reduced set of poses, with an approach that is similar in spirit to [Oikonomidis et al., 2011]. One advantage of their method is the higher geometric fidelity of their convex bodies hand model compared to our cylinder model. Furthermore, our evaluation demonstrated how their more precise representation of the hand's Thenar eminence, as well as the thumb articulation, can result in more natural fitting in these regions.

Convolutional Networks [TSLP14]

Figure 2.22 shows a quantitative comparison with the appearance-based method of [Tompson et al., 2014] on a dataset provided by the authors of that paper. Overall, the tracking quality is comparable, with a somewhat lower average error for our method. However, our solution avoids many of the high-error peaks of [Tompson et al., 2014] where tracking is lost completely. An additional advantage of our approach in comparison to any of the existing appearance-based methods is that we can handle more complex interactions of two hands, since such configurations are not part of the training data sets of existing methods; see Figure 2.21.

Limitations

Single-camera depth acquisition yields incomplete data and as such the pose reconstruction problem is inherently ill-posed. Tracking errors can occur in certain situations as explained above when insufficient data is acquired due to occlusions or fast motion. Similarly, the reso-

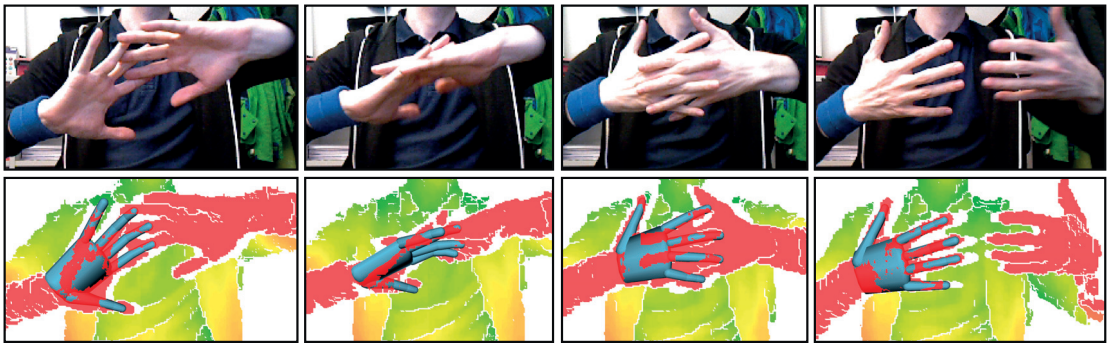
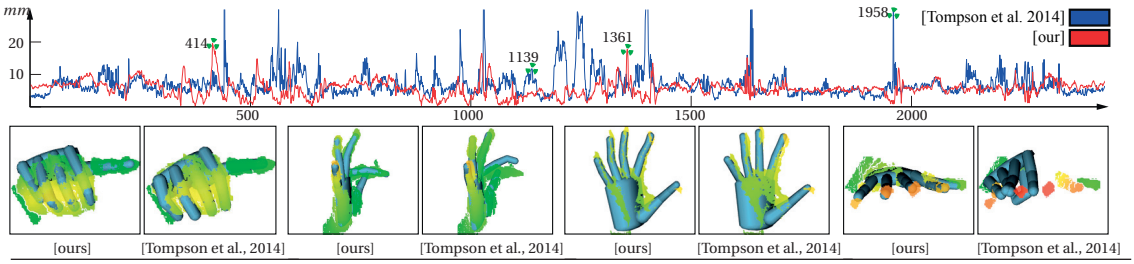


Figure 2.21 – Developing robust model-based tracking is essential to enable tracking of hands interacting with each other or with other objects in the environment. Here we illustrate that for our method tracking accuracy is not significantly affected even though we are not modeling the second hand. Note that such motion cannot be tracked successfully by appearance-based methods such as [Tompson et al., 2014].



Frame 414 Frame 1139 Frame 1361 Frame 1958

Figure 2.22 – Quantitative comparison to [Tompson et al., 2014]. The graph shows the average root mean square tracking error w.r.t. ground truth across 2440 frames. Some frames where the accuracy of the two methods differs significantly are highlighted in the bottom row.

lution of the sensor limits tracking accuracy. As shown in Figure 2.23, when geometric features become indiscriminate, our registration approach fails. Integrating color and shading information could potentially address this issue [de La Gorce et al., 2011]. While our current system requires the user to wear a wristband for detection and stabilization, this could be replaced by automatic hand labeling, e.g. using random decision forest classifiers as in [Tompson et al., 2014].

Our cylinder model proved adequate for the data quality of current commodity sensors, but is overall limited in geometric accuracy, and hence might not scale with increasing sensor resolution. Also, in our current implementation the model needs to be manually adapted to the user through simple scaling operations. Without such adaptation, tracking accuracy degrades as shown in Figure 2.24. This user-specific adaption could be automated [Taylor et al., 2014] and potentially even performed simultaneously with the real-time tracking as recently proposed for face tracking [Bouaziz et al., 2013].

The PCA model used in the prior energy is an efficient, but rather simplistic representation of

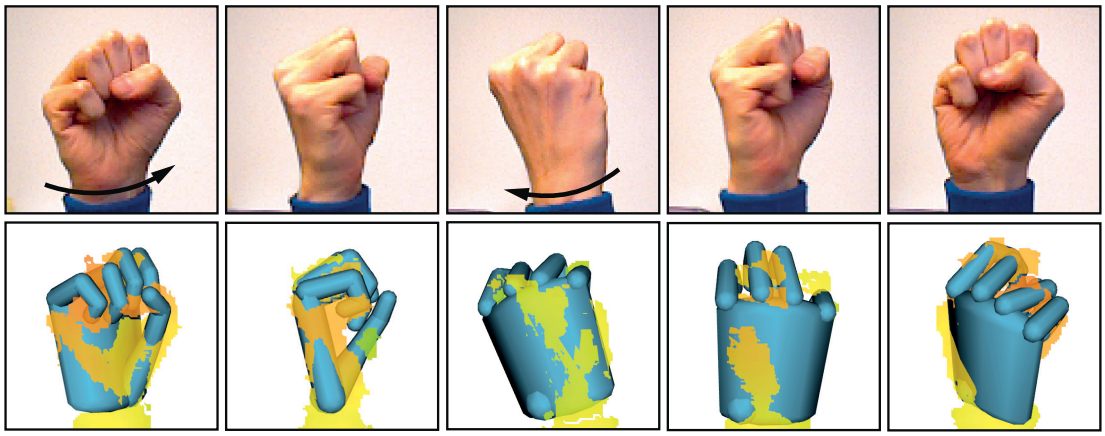


Figure 2.23 – Our algorithm relies on the presence of salient geometric features in the depth map. Challenging sequences like a rotating fist lack such features when acquired with current commodity depth sensors, which can result in loss of tracking.

the pose space. We currently do not consider the temporal order in which the hand poses of the database have been acquired, which could potentially be exploited for more sophisticated temporal priors.

2.6 Conclusions

We have introduced a new model-based approach to real-time hand tracking using a single low-cost depth camera. This simple acquisition setup maximizes ease of deployment, but poses significant challenges for robust tracking. Our analysis revealed that a major source of error when tracking articulated hands are erroneous correspondences between the hand model and the acquired data, mainly caused by outliers, holes, or data popping in and out during acquisition. We demonstrate that these problems can be resolved by our new formulation of correspondence search. In combination with suitable 2D/3D registration energies and data-driven priors, this leads to a robust and efficient hand tracking algorithm that outperforms existing model- and appearance-based solutions.

By fully disclosing our source code and data we ensure that our method and results are reproducible, as well as facilitate future research and product development.

We are investigating a technique for efficient automatic personalization of the tracking model to the user, in order to facilitate a more seamless usage of our system across different subjects. Other examples of future efforts are robust two-hand tracking with object interactions, combinations of hand tracking with full body tracking, and integrating our hand tracking solution to new interfaces and real-time applications.

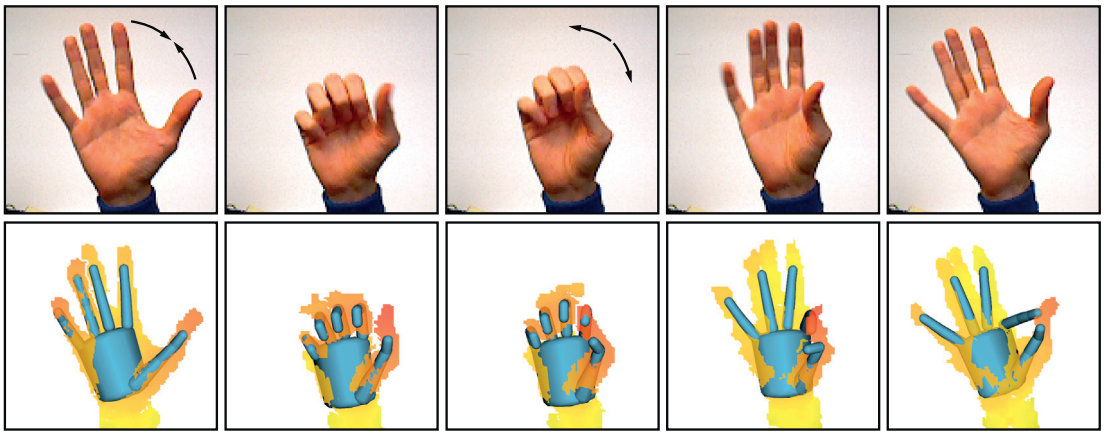


Figure 2.24 – When tracking with an uncalibrated model, tracking correspondences can map to data belonging to erroneous portions of the model. In the figure, the index finger remains attached to samples associated with the thumb.

2.7 Implementation Details

2.7.1 Projective v.s. subspace PCA

In Equation 2.6, minimizing E_{pose} over $\tilde{\boldsymbol{\theta}}$ has a closed form solution:

$$\tilde{\boldsymbol{\theta}} = (\omega_5 \boldsymbol{\Sigma}^2 + \omega_4 \mathbf{I})^{-1} (\omega_4 \Pi_{\mathcal{P}}^T (\boldsymbol{\theta} - \boldsymbol{\mu})).$$

We can therefore rewrite our data-driven energy only as a function of $\boldsymbol{\theta}$ as

$$E_{\text{pose}} = \omega_4 \| (\boldsymbol{\theta} - \boldsymbol{\mu}) - \Pi_{\mathcal{P}} \mathbf{M} \Pi_{\mathcal{P}}^T (\boldsymbol{\theta} - \boldsymbol{\mu}) \|_2^2,$$

where $\mathbf{M} = \omega_4 (\omega_5 \boldsymbol{\Sigma}^2 + \omega_4 \mathbf{I})^{-1}$. Our formulation does not only allow the solution to stay close to the pose space, but also penalizes unlikely poses replacing the conventional orthogonal projection matrix $\Pi_{\mathcal{P}} \Pi_{\mathcal{P}}^T$ by a matrix $\Pi_{\mathcal{P}} \mathbf{M} \Pi_{\mathcal{P}}^T$ taking into account the PCA standard deviation. Note that when $\omega_5 = 0$ we retrieve the orthogonal projection $\Pi_{\mathcal{P}} \Pi_{\mathcal{P}}^T$.

2.7.2 Jacobians

Perspective projection Jacobian

The Jacobian of the perspective projection is a $[2 \times 3]$ matrix depending from the focal length of the camera $\mathbf{f} = [f_x, f_y]$ and the 3D position $\mathbf{x}$ at which it is evaluated [Bouaziz et al., 2014]:

$$\mathbf{J}_{\text{persp}}(\mathbf{x}) = \begin{bmatrix} f_x/\mathbf{x}_z & 0 & -\mathbf{x}_x f_x/\mathbf{x}_z^2 \\ 0 & f_y/\mathbf{x}_z & -\mathbf{x}_y f_y/\mathbf{x}_z^2 \end{bmatrix}$$

Skeleton Jacobian

The skeleton Jacobian $\mathbf{J}_{\text{skel}}(\mathbf{x})$ is a $[3 \times 26]$ matrix. For each constraint, the bone identifier $b = id(\mathbf{x})$ associated to each 3D point $\mathbf{x}$ determines the affected portion of the kinematic chain. That is, it identifies the non-zero columns of $\mathbf{J}_{\text{skel}}(\mathbf{x})$. As discussed in [Buss, 2004], the i-th column of $\mathbf{J}_{\text{skel}}(\mathbf{x})$ contains the linearization of i-th joint about $\mathbf{x}$.

2.7.3 Approximation using linearized function.

To approximate the following energies, we approximate $E = \|\mathbf{f}(\mathbf{x})\|_2^2$ by linearizing $\mathbf{f}(\mathbf{x})$ as

$$\mathbf{f}(\mathbf{x}+\delta\mathbf{x})|_{\mathbf{x}} \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x}.$$

The approximation is then expressed as

$$\bar{E} = \|\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x}\|_2^2. \tag{2.10}$$

Joint bounds

The joint bounds energy can be written as

$$\bar{E}_{\text{bound}} = \omega_7 \sum_{\theta_i \in \boldsymbol{\theta}} \overline{\chi}(i)(\delta\boldsymbol{\theta}_i + \boldsymbol{\theta}_i - \overline{\boldsymbol{\theta}}_i)^2 + \underline{\chi}(i)(\delta\boldsymbol{\theta}_i + \boldsymbol{\theta}_i - \underline{\boldsymbol{\theta}}_i)^2$$

Temporal coherence

To compute the velocity $\dot{\mathbf{k}}(\boldsymbol{\theta})$ and the acceleration $\ddot{\mathbf{k}}(\boldsymbol{\theta})$ of a point $\mathbf{k}$ attached to the kinematic chain, we use finite differences. The linearization of the temporal energy becomes

$$\begin{aligned}\bar{E}_{\text{temporal}} = \omega_8 \sum_{\mathbf{k}\in\mathcal{K}} \|\mathbf{J}_{\text{skel}}(\mathbf{k})\delta\boldsymbol{\theta} + (\mathbf{k} - \mathbf{k}_{t-1})\|_2^2 \\ + \omega_9 \sum_{\mathbf{k}\in\mathcal{K}} \|\mathbf{J}_{\text{skel}}(\mathbf{k})\delta\boldsymbol{\theta} + (\mathbf{k} - 2\mathbf{k}_{t-1} + \mathbf{k}_{t-2})\|_2^2,\end{aligned}$$

where $\mathbf{k}_{t-1}$ and $\mathbf{k}_{t-2}$ are the position of such points from the two previously optimized frames.

Data-driven (PCA)

The data-driven projection energy can be rewritten as

$$\bar{E}_{\text{pose}} = \omega_4 \left\| (\mathbf{I} - \Pi_{\mathcal{P}}\mathbf{M}\Pi_{\mathcal{P}}^T)(\delta\boldsymbol{\theta} + \boldsymbol{\theta} - \boldsymbol{\mu}) \right\|_2^2.$$

2.7.4 Approximation using Linearized ℓ_2 Distance.

To approximate the following energies, we first reformulate the quadratic form $E = \|\mathbf{f}(\mathbf{x})\|_2^2$ as $E = (\|\mathbf{f}(\mathbf{x})\|_2)^2$. We then linearize the ℓ_2 norm $\|\mathbf{f}(\mathbf{x})\|_2$ as

$$\|\mathbf{f}(\mathbf{x}+\delta\mathbf{x})\|_2|_{\mathbf{x}} \approx \|\mathbf{f}(\mathbf{x})\|_2 + \frac{\mathbf{f}(\mathbf{x})^T}{\|\mathbf{f}(\mathbf{x})\|_2}\mathbf{J}(\mathbf{x})\delta\mathbf{x}.$$

The approximation is then expressed as

$$\bar{E} = \left(\|\mathbf{f}(\mathbf{x})\|_2 + \frac{\mathbf{f}(\mathbf{x})^T}{\|\mathbf{f}(\mathbf{x})\|_2}\mathbf{J}(\mathbf{x})\delta\mathbf{x}\right)^2.$$

When the energy is of the form $E = \|\mathbf{x} - \Pi(\mathbf{x})\|_2^2$ where $\Pi(\mathbf{x})$ is a projection operator, Bouaziz et al. [Bouaziz et al., 2012] showed that $\mathbf{f}(\mathbf{x})^T\mathbf{J}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T$. In this case, the approximate energy can be simplified as

$$\bar{E} = \left(\|\mathbf{f}(\mathbf{x})\|_2 + \frac{\mathbf{f}(\mathbf{x})^T}{\|\mathbf{f}(\mathbf{x})\|_2}\delta\mathbf{x}\right)^2.$$

Contrary to the approximation in Equation 2.10, the Jacobian of the projection function does not need to be known. This formulation is useful as the approximation in the equation above only needs to evaluate the projection function and therefore allows to use arbitrarily complex projection functions.

Point cloud alignment

We linearize the point cloud alignment energy as

$$\bar{E}_{3D} = \omega_1 \sum_{\mathbf{x}\in\mathcal{X}_s} \omega_{re}(\mathbf{n}^T(\mathbf{J}_{skel}(\mathbf{y})\delta\boldsymbol{\theta} + d))^2,$$

where $\mathbf{y} = \Pi_{\mathcal{M}}(\mathbf{x}, \boldsymbol{\theta})$ is the closest point from $\mathbf{x}$ on the hand model $\mathcal{M}$ with hand pose $\boldsymbol{\theta}$. $\mathbf{n}$ is the surface normal at $\mathbf{y}$, and $\mathbf{d} = (\mathbf{y} - \mathbf{x})$. As we minimize the ℓ_2 norm we use a weight $\omega_{re} = 1/\|\mathbf{d}\|_2$ in an iteratively re-weighted least squares fashion.

Silhouette alignment

The silhouette energy is expressed in screen space, and therefore employs the perspective projection Jacobian $\mathbf{J}_{persp}(\mathbf{x})$, where $\mathbf{x}$ is the 3D location of a rendered silhouette point $\mathbf{p}$. Similarly to the point cloud alignment the linearization can be expressed as

$$\bar{E}_{2D} = \omega_2 \sum_{\mathbf{p}\in\mathcal{S}_r} (\mathbf{n}^T(\mathbf{J}_{persp}(\mathbf{x})\mathbf{J}_{skel}(\mathbf{x})\delta\boldsymbol{\theta} + d))^2,$$

where $\mathbf{d} = (\mathbf{p} - \mathbf{q})$ with $\mathbf{q} = \Pi_{\mathcal{S}_s}(\mathbf{p}, \boldsymbol{\theta})$, and $\mathbf{n}$ is the 2D normal at the sensor silhouette location $\mathbf{q}$.

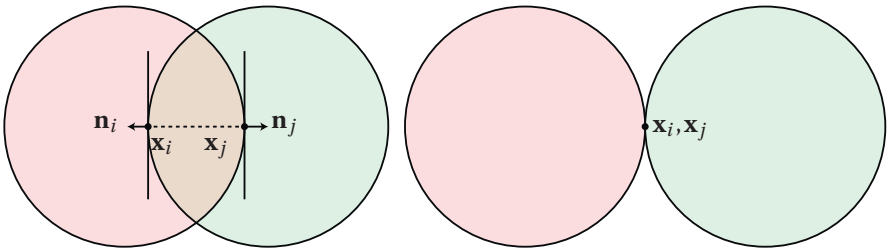


Figure 2.25 – (left) Collision constraints definition, deepest penetration points marked as $\mathbf{x}_i, \mathbf{x}_j$. (right) When the collision energy is minimized in isolation the penetration points are co-located.

Collision

Figure 2.25 illustrates the necessary notation with a 2D example, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the end-points of the shortest segment between the two cylinders axes. The linearized energy is defined as

$$\bar{E}_{\text{coll.}} = \omega_6 \sum_{\{i,j\}} \chi(i,j) \left(\mathbf{n}_i^T \left((\mathbf{J}_{\text{skel}}(\mathbf{x}_i) - \mathbf{J}_{\text{skel}}(\mathbf{x}_j))\delta\boldsymbol{\theta} + d\right)\right)^2$$

where $\mathbf{n}_i$ is the surface normal at $\mathbf{x}_i$ (as shown in Figure 2.25), and $\mathbf{d} = (\mathbf{x}_i - \mathbf{x}_j)$.

2.7.5 Non-Linear Least Squares Optimization

To solve our optimization problem we use a Levenberg-Marquardt approach. We iteratively solve Equation 2.1 using the approximate energies described in Section 2.7.2 through Section 2.7.4 leading to a damped least squares minimization

$$\min_{\delta\boldsymbol{\theta}} \bar{E}_{\text{3D}} + \bar{E}_{\text{2D}} + \bar{E}_{\text{wrist}} + \bar{E}_{\text{pose}} + \bar{E}_{\text{kin.}} + \bar{E}_{\text{temp.}} + \bar{E}_{\text{damp}},$$

and update our hand pose using the update $\boldsymbol{\theta} = \boldsymbol{\theta} + \delta\boldsymbol{\theta}$. Note that since our energies are written in the form:

$$\Sigma_i \bar{E}_i = \Sigma_i \|\mathbf{J}_i \delta\boldsymbol{\theta} - \mathbf{e}_i\|_2^2,$$

our solve can be re-written as

$$\delta\boldsymbol{\theta} = \left(\Sigma_i \mathbf{J}_i^T \mathbf{J}_i\right)^{-1} \left(\Sigma_i \mathbf{J}_i^T \mathbf{e}_i\right) = 0. \tag{2.11}$$

To stabilize the optimization, we introduce a damping energy $\bar{E}_{\text{damp}} = \lambda \|\delta\boldsymbol{\theta}\|_2^2$, where $\lambda = 100$.

2.7.6 CPU/GPU Optimization

Our technique elegantly de-couples the components of our optimization on CPU and GPU. With regards to Figure 2.4 only large-scale and trivially parallelizable tasks, like the computation of constraints associated with 2D/3D ICP correspondences are performed on GPU, while all others run efficiently on a *single* CPU thread. In particular, the inversion in Equation 2.11 is performed on CPU by Cholesky factorization (Eigen3). As the final solve is performed on CPU,

we designed our optimization to minimize memory transfers between CPU/GPU. First of all, note that although at each iteration we need to render an image of the cylinder model, the texture is already located on the GPU buffers. Furthermore, although the large ($\approx 20k \times 26$) Jacobian matrices associated with E_{3D} and E_{2D} are assembled on the GPU, a CuBLAS kernel is used to compute the much smaller (26×26, 26×1) matrices $\mathbf{J}_i^T \mathbf{J}_i$ and $\mathbf{J}_i^T \mathbf{e}_i$. Only these need to be transferred back to CPU for each solver iteration.

3 Sphere-Meshes for Real-Time Hand Modeling and Tracking

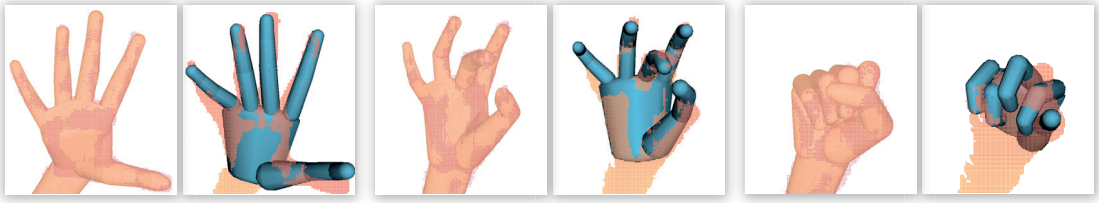


Figure 3.1 – Three side-by-side comparisons of tracking performance from the HANDY/TEASER sequence. Our model allows us to obtain much higher tracking quality. Tracking at a finer scale is instrumental to prevent tracking failure. The whole sequence can be seen in Video2 [03:53].

This chapter is based on the following publication:

> TKACH A., PAULY M., TAGLIASACCHI A.: Sphere-meshes for real-time hand modeling and tracking. *In ACM Trans. Graph. (Proc. SIGGRAPH Asia) (2016)*.

Abstract

Modern systems for real-time hand tracking rely on a combination of discriminative and generative approaches to robustly recover hand poses. Generative approaches require the specification of a geometric model. In this chapter we propose the use of sphere-meshes as a novel geometric representation for real-time generative hand tracking. How tightly this model fits a specific user heavily affects tracking precision. We derive an optimization to non-rigidly deform a template model to fit the user data in a number of poses. This optimization jointly captures the user's static and dynamic hand geometry, thus facilitating high-precision registration. At the same time, the limited number of primitives in the tracking template allows us to retain excellent computational performance. We confirm this by embedding our model in an open source real-time registration algorithm to obtain a tracker steadily running at 60Hz. We demonstrate the effectiveness of our solution by qualitatively and quantitatively

evaluating tracking precision on a variety of complex motions. We show that the improved tracking accuracy at high frame-rate enables stable tracking of extended and complex motion sequences without the need for per-frame re-initialization. To enable further research in the area of high-precision hand tracking, we publicly release our source code and evaluation datasets.

3.1 Introduction

The main goal of this chapter is to explore novel tracking models that strike an optimal balance between accuracy and performance.

More specifically, we propose a geometric model that more accurately captures the user's hand geometry, while retaining the ability to answer registration queries in closed form with very high efficiency. In Figure 3.2 and Video2 [03:53] [1] we illustrate the importance of employing a tracking template that strikes this delicate balance.

Implicit vs. explicit templates

In modern digital production the de-facto standard is to represent objects by a surface mesh of their boundary (e.g. triangle or quad meshes). Fast rendering and easy direct manipulation make *explicit* surface representation attractive for many applications. However, unlike *implicit* models [Bloomenthal et al., 1997], explicit representations cannot efficiently answer queries such as the distance from a point to the object's boundary, or whether a point lies inside/outside the model [Botsch et al., 2010, Ch.1]. In tracking applications these queries play a fundamental role, as the optimization attempts to find configurations where the average

[1] Please find the accompanying Video2 at http://lgg.epfl.ch/publications/2016/HModel/video.mp4.

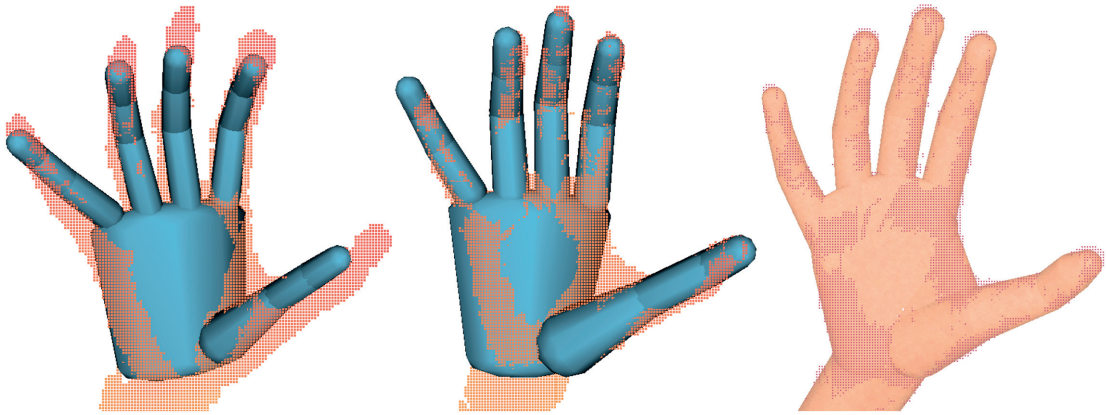


Figure 3.2 – (left) Tracking when the model from [Tagliasacchi et al., 2015] is used without proper coarse scale calibration. (middle) A roughly manually calibrated model can help increasing the fitting fidelity, but tuning becomes increasingly difficult with more degrees of freedom. (right) The proposed automatically calibrated model.

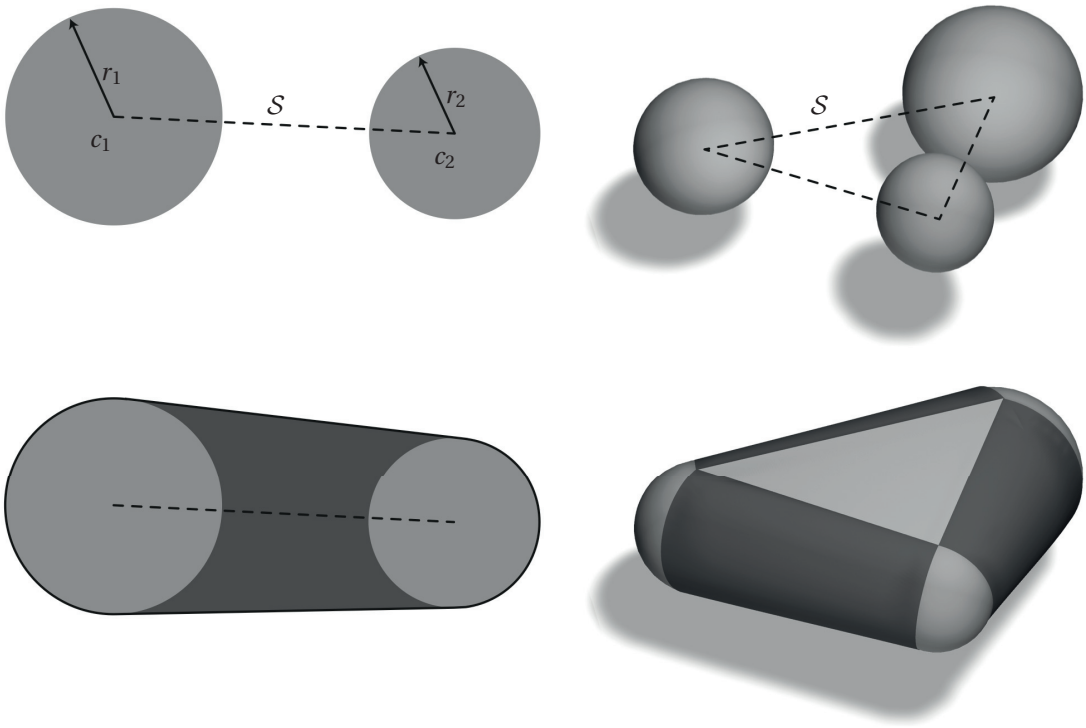


Figure 3.3 – The sphere-mesh skeleton $\mathcal{S}$ identifies sphere positions and radii. The surface of the object is obtained as the convex-hull of the spheres on the vertices of the skeleton. Sphere-meshes can be rendered through GPU ray-tracing, or by meshing the zero-crossing of their implicit function; see Equation 3.1.

distance from model to data is minimized. Similarly, a tracker should prevent the model from assuming implausible configurations, for example by preventing self-intersections as measured by inside/outside predicates. For all these reasons, and as demonstrated by compelling results in rigid [Newcombe et al., 2011] and non-rigid [Newcombe et al., 2015] reconstruction, implicit models are highly suitable for registration applications.

To address the challenges of *real-time* registration, we propose to employ a *hybrid* model that combines the advantages of explicit and implicit representations.

Hybrid sphere-mesh templates

The model we propose in this chapter is a variant of a convolution surface [Bloomenthal and Shoemake, 1991]. Its fundamental building blocks are illustrated in Figure 3.3. The surface is defined as the zero iso-level of the scalar function

$$\phi(\mathbf{x}) = \min_{\mathbf{c}\in\mathcal{S}} \mathcal{B}(\mathbf{x}|\mathbf{c}, r(\mathbf{c})) \tag{3.1}$$

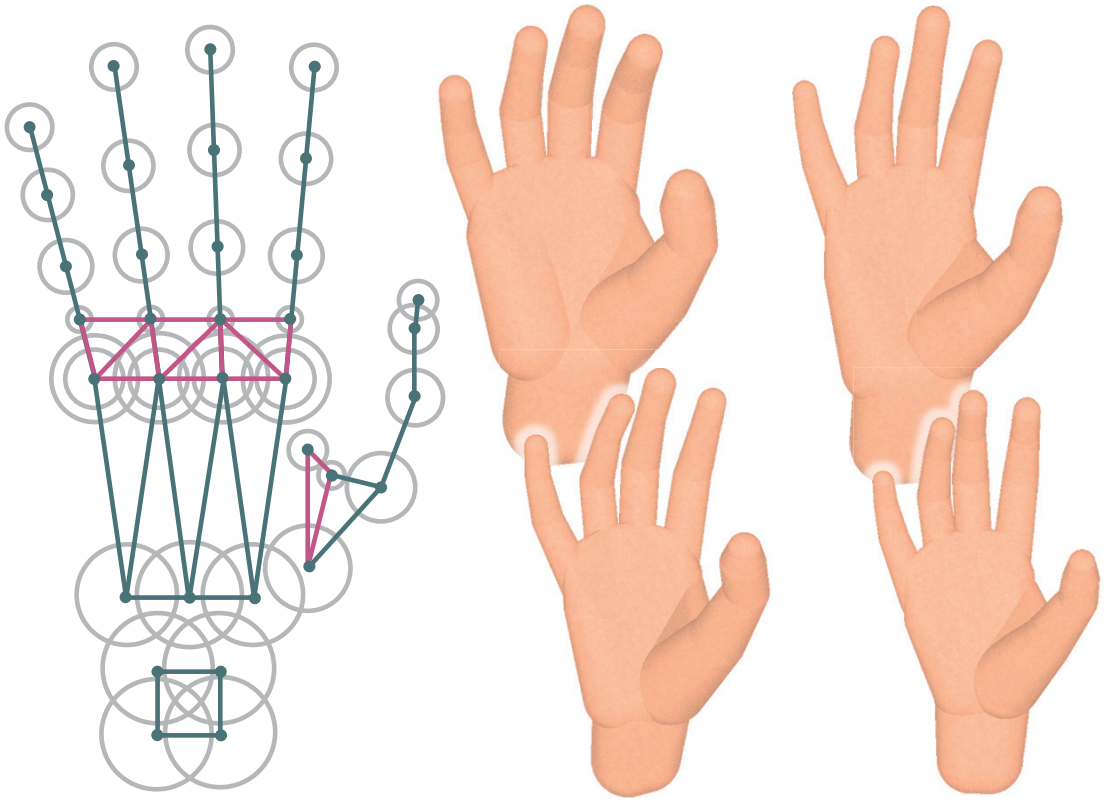


Figure 3.4 – (left) The skeleton $\mathcal{S}$ parametrizes the sphere-mesh through vertex positions and radii. In our template, articulated components are shown in dark green while flexible components in purple. (right) Calibration instantiates our template by adjusting the skeletal vertex positions and radii.

where $\mathcal{S}$ is a skeletal control mesh (a segment or a triangle in the simple examples of Figure 3.3), and $\mathcal{B}$ is the implicit function of a sphere given its center $\mathbf{c}$ and radius r:

$$\mathcal{B}(\mathbf{x}|\mathbf{c},r) = \|\mathbf{x}-\mathbf{c}\|^2 - r^2 \tag{3.2}$$

The sphere centers $\mathbf{c}$ span the skeleton $\mathcal{S}$, while the radii are a function of the position $\mathbf{c}$ within an element, linearly interpolated from values $r_* = r(\mathbf{c}_*)$ specified on the skeletal mesh vertices $\mathbf{c}_*$. This is indeed a *hybrid* model, as Equation 3.1 defines an implicit surface $\mathcal{M} = \{\mathbf{x} \in \mathbb{R}^n | \phi(\mathbf{x}) = 0\}$, while the underlying skeleton $\mathcal{S}$ is an explicit representation (i.e. a simplicial complex). We generalize this construct to devise a model suitable to represent a human hand; see Figure 3.4. Distances to $\mathcal{M}$ can conveniently be computed by querying distances to the piecewise linear elements of $\mathcal{S}$; see Figure 3.7.

Tracking and calibration with sphere-meshes

Our novel tracking model has two significant advantages. (1) Distance queries to $\mathcal{M}$ can be executed by measuring the distance to the skeletal structure $\mathcal{S}$. The number of elements in $\mathcal{S}$ is significantly smaller (30 in our model) than the number of polygons in a typical triangular mesh surface representation [Thiery et al., 2013]. Therefore, distance queries can be performed efficiently using a brute force approach, which leads to a simple algorithm that is trivially parallelizable. (2) The parameterization of our hand model is compact, as we can generate a family of models by simply adjusting *positions* and *radii* of the control skeleton vertices $\mathbf{c}_* \in \mathcal{S}$. This allows adapting the model to the hand geometry of a specific user.

Contributions

The core contribution of this chapter is to demonstrate that sphere-meshes provide superior hand tracking performance for single-view depth sensors. We introduce an optimization approach that allows adapting our tracking model to different human hands with a high level of accuracy. The improved geometric fidelity compared to existing representations leads to quantifiable reductions in registration error and allows accurate tracking even for intricate hand poses and complex motion sequences that previous methods have difficulties with. At the same time, due to a very compact model representation and closed-form correspondence queries, our generative model retains high computational performance, leading to sustained tracking at 60Hz.

Overview

The remainder of the chapter is structured as follows: We survey related work in Section 3.2. In Section 3.3 we describe our generative real-time hand tracking technique, which details how our novel formulation enables efficient correspondence computation. Section 3.4 explains how we build our template model from 3D scans acquired either through multi-view stereo or from depth maps. In Section 3.5 we analyze the performance of our model for real-time tracking and provide comparisons to the state-of-the-art. We conclude in Section 3.6 with a discussion of current limitations and ideas for future work.

3.2 Related Work

Generative tracking models

The capsule model originally proposed by [Rehg and Kanade, 1994] has been adopted by a number of researchers [Oikonomidis et al., 2011, Schroder et al., 2014, Fleishman et al., 2015, Tagliasacchi et al., 2015]; see Figure 3.5(a). Such a coarse representation is suitable to the task given the low signal-to-noise ratio in modern depth sensors, while its simplicity enables the efficient closed-form computation of alignment queries. Cylinders can also

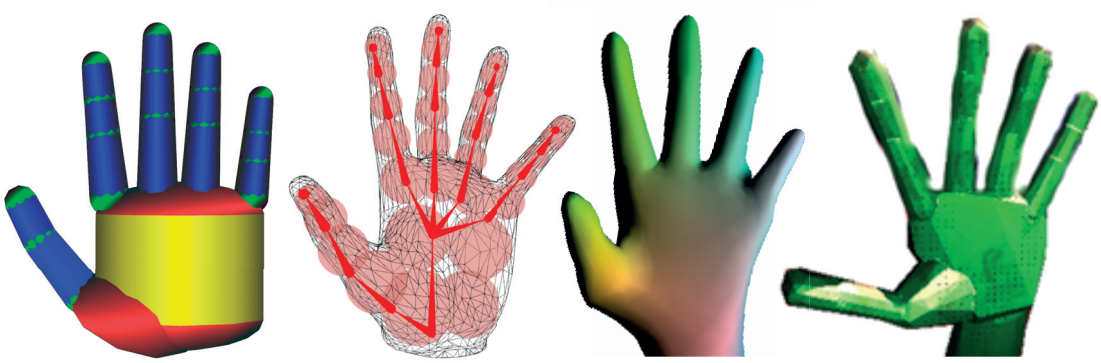


Figure 3.5 – Several tracking templates employed by recent generative (or hybrid) real-time hand-tracking methods. Images courtesy of (a) [Oikonomidis et al., 2011], (b) [Sridhar et al., 2013], (c) [Taylor et al., 2016], and (d) [Melax et al., 2013].

be approximated by a small set of disconnected spheres [Qian et al., 2014], but this rough approximation is only sufficient for coarse-scale tracking. An alternative to cylinders and spheres is the use of isotropic [Sridhar et al., 2013, Sridhar et al., 2015], as well as anisotropic Gaussians [Sridhar et al., 2014]; see Figure 3.5(b). The use of surface meshes, while widespread in other domains (e.g. face tracking [Bouaziz et al., 2013] or offline registration [Loper and Black, 2014]), has been limited to the visualization of tracking performance through skinned model animations [Tompson et al., 2014, Schroder et al., 2014]. Sharp et al. [Sharp et al., 2015] employed mesh models for tracking in a render-and-compare framework, while the very recent work of [Taylor et al., 2016] presents the first attempt towards a continuous registration framework for tracking hands with triangular meshes; see Figure 3.5(c). Other variants of tracking models include the union of convex bodies from [Melax et al., 2013], a convolutional neural network capable of directly synthesizing hand depth images [Oberweger et al., 2015b], and some initial attempts at tracking with implicit templates [Plankers and Fua, 2003]. Our sphere-mesh model offers accuracy comparable to triangle meshes used in recent hand trackers, while retaining a compact representation for efficient correspondence queries and effective user adaptation.

Template calibration

Albrecht et al. [Albrecht et al., 2003] pioneered the creation of a realistic hand model (i.e. bones and muscles) by aligning a template mesh to data acquired by a laser-scanned plaster cast. Rhee et al. [Rhee et al., 2006] use a simpler setup consisting of a single color image to identify approximate joint locations by localizing skin creases, and adapt a mesh template to conform to its silhouette. While these methods focus on a static template, in [de La Gorce et al., 2011] a model is roughly adapted to the user through simple bone scaling to produce the first *animatable* template. Calibration of a cylinder model through particle swarm has been investigated in [Makris and Argyros, 2015]. Mesh calibration techniques were proposed in [Taylor et al., 2014] and extended in [Khamis et al., 2015], which introduces compact and

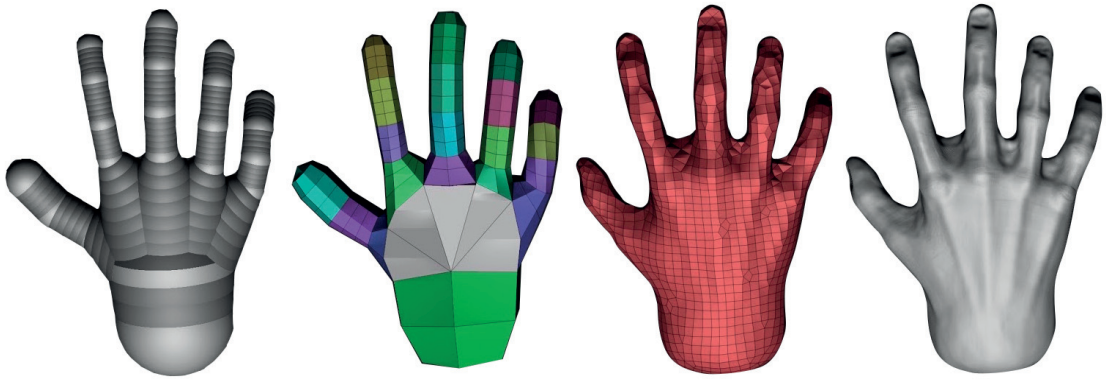


Figure 3.6 – (a) An artist creates a hand model by first sketching its topological structure as a union of spheres (ZSphere). (b) The model is then converted into a volumetric representation and meshed (Unified Skinning) to be further refined (c,d).

linear shape-spaces of human hand geometry. The method in [Taylor et al., 2014] shares some similarities with our work, where the model is adjusted to *jointly* fit a set of depth frames, but with a fundamental difference in the way in which geometry is represented. Our sphere-mesh model is *naturally compact*, leading to straightforward calibration and tracking algorithms.

Implicit modeling

Implicit sculpting tools have recently become a viable alternative to mesh or spline-based approaches for modeling complex geometries. This paradigm lies at the basis of the success of the *PixoLogic ZBrush* product line. For articulated geometry, it is often convenient to first create a coarse geometric structure analogous to the one described in Equation 3.1, a process that *PixoLogic* has re-branded as *ZSphere* modeling; see Figure 3.6. Editing the radii and centers of the sphere-mesh offers a *natural* way of editing the model, making it easy for both humans and algorithms to calibrate. Note that any geometric model can be approximated, to any desired precision, as a union of spheres [Tagliasacchi et al., 2016]. However, by considering spheres that are linearly interpolated across edges, we can heavily reduce the required number of primitives. Following this principle, [Thiery et al., 2013] recently investigated a method to automatically generate Sphere Meshes provided a (static) input model. Extending this work, [Thiery et al., 2016] proposed a method to fit a model to a sequence of dynamic meshes. While seemingly related, our calibration optimization is solving a fundamentally different problem, as in our technique a template is fixed and provided in input.

3.3 Tracking

Given a calibrated hand model $\mathcal{M}$, our real-time tracking algorithm optimizes the 28 degrees of freedom $\boldsymbol{\theta}$ (i.e. joint angles) so that our hand model matches the sensor input data; the generation of a calibrated model $\mathcal{M}$ for a user is detailed in Section 3.4. Directly extending the

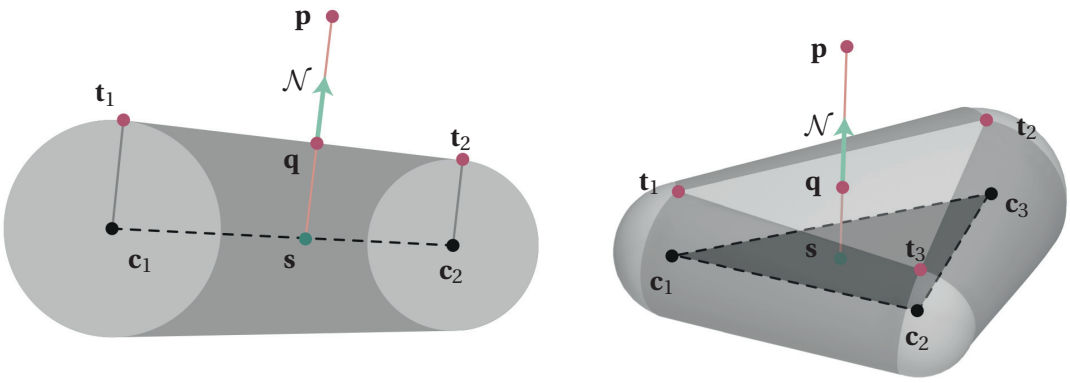


Figure 3.7 – The computation of closest point correspondences on pill (left) and wedge (right) elements can be performed by tracing a ray along the normal of the line (resp. plane) tangent to the circles (resp. spheres).

open source *htrack* framework of [Tagliasacchi et al., 2015], we write our tracking optimization in Gauss-Newton/Levenberg-Marquardt form:

$$\boldsymbol{\theta}_t = \underset{\boldsymbol{\theta}}{\arg\min} \sum_{\mathcal{T} \in \mathcal{T}_{\text{track}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathcal{D}_t, \boldsymbol{\theta}, \boldsymbol{\theta}_{t-1}) \tag{3.3}$$

where fitting energies are combined with a number of priors to regularize the solution and ensure the estimation of plausible poses.

The energy terms $\mathcal{T}_{\text{track}}$ in our optimization are:

d2m	each data point is explained by the model
m2d	the model lies in the sensor visual-hull
pose	hand poses sample a low-dimensional manifold
limits	joint limits must be respected
collision	fingers cannot interpenetrate
temporal	the hand is moving smoothly in time

We limit our discussion to the computational elements that need to be adapted to support sphere-meshes, while referring the reader to [Tagliasacchi et al., 2015] for other details.

Hausdorff distance

The similarity of two geometric models can be measured by the symmetric Hausdorff distance $\varphi_{X \leftrightarrow Y}$:

$$\begin{aligned} \varphi_{X \to Y} &= \max_{x \in X} \left[\min_{y \in Y} \varphi(x, y) \right] \\ \varphi_{Y \to X} &= \max_{y \in Y} \left[\min_{x \in X} \varphi(x, y) \right] \\ \varphi_{X \leftrightarrow Y} &= \max\{d_{X \to Y}, \varphi_{Y \to X}\} \end{aligned}$$

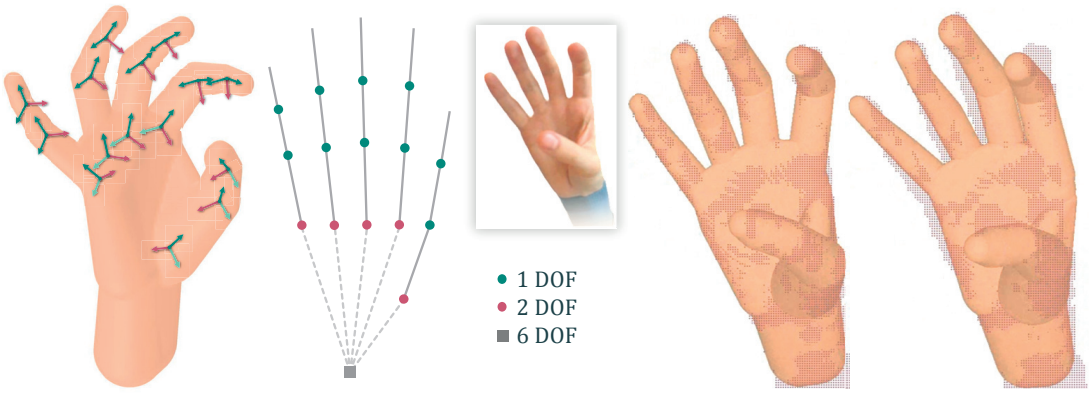


Figure 3.8 – (a) A visualization of the posed kinematic frames $\bar{T}_*$. (b) The kinematic chain and number of degrees of freedom for posing our tracking model. Tracking quality with (c) optimal and (d) non-optimal kinematic transformation frames.

We therefore interpret our terms E_{d2m} and E_{m2d} as approximations to the asymmetric Hausdorff distances $\varphi_{X\to Y}$ and $\varphi_{Y\to X}$, where the difficult to differentiate *max* operators are replaced by arithmetic means, and a robust ℓ_1 distance is used [Tagliasacchi and Li, 2016].

Data → Model

The first asymmetric distance minimizes the average closest point projection of each point $\mathbf{p}$ in the depth frame $\mathcal{D}$:

$$E_{d2m} = |\mathcal{D}|^{-1} \sum_{\mathbf{p}\in\mathcal{D}} \|\mathbf{p} - \Pi_{\mathcal{M}(\boldsymbol{\Theta})}(\mathbf{p})\|_2^1 \tag{3.4}$$

Adapting this energy, as well as its derivatives, to sphere-meshes requires the specification of the projection operator $\Pi_{\mathcal{M}}$ that is described in Section 3.3.1.

Model → Data

The second asymmetric distance considers how our monocular acquisition system does not have a complete view of the model. While the 3D location is unknown, we can penalize the model from lying outside the sensor's *visual hull*:

$$E_{m2d} = |\mathcal{M}(\boldsymbol{\Theta})|^{-1} \int_{\mathbf{x}\in\mathcal{M}(\boldsymbol{\Theta})} \|\mathbf{x} - \Pi_{\mathcal{D}}(\mathbf{x})\|_2^1 \tag{3.5}$$

In the equation above, the integral is discretized as a sum over the set of pixels obtained through rasterization; see Section 3.3.2. The rasterization renders the model to the image plane using the intrinsic and extrinsic parameters of the sensor's depth camera.

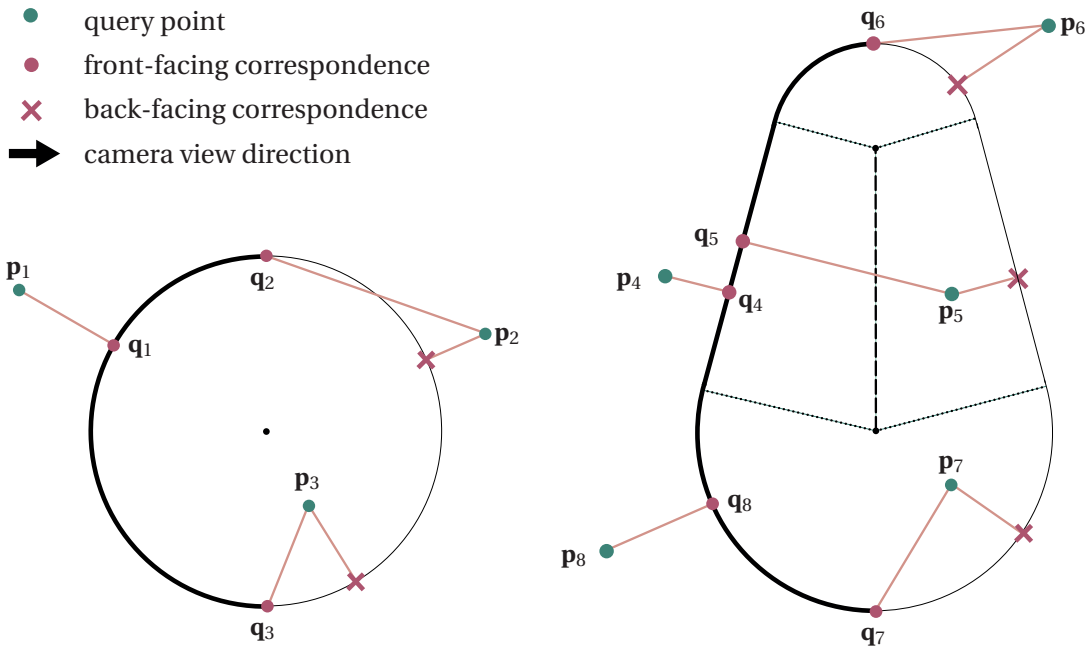


Figure 3.9 – In monocular acquisition only the front-facing part of the model should be registered to the data. Here the camera is observing (left to right) two elements and the occluded parts of the model are marked. Correspondences whose normals point away from the camera are discarded, and replaced by the closest amongst silhouette correspondences or front-facing portions of wedges.

3.3.1 Correspondences

Our correspondence search leverages the structure of Equation 3.1, by decomposing the surface into several elementary elements $\mathcal{E}^e$, where e indexes the 30 elements of our template; see Video2 [00:58]. As illustrated in Figure 3.7, elements are classified into *pill* and *wedge* implicit primitives, with an associated implicit functions ϕ_e. Given a point $\mathbf{p}$ in space, the implicit function of the whole surface can be written by evaluating the expression:

$$\phi_{\mathcal{M}}(\mathbf{p}) = \underset{e=1\ldots E}{\arg\min}\, \phi_e(\mathbf{p}) \tag{3.6}$$

Given a query point $\mathbf{p}$, we can first compute the closest-points $\mathbf{q}_e = \Pi_{\mathcal{E}^e}(\mathbf{p})$ to each element independently; within this set, the closest-point projection to the full model $\mathbf{q} = \Pi_{\mathcal{M}}(\mathbf{p})$ is the one with the smallest associated implicit function value $\phi_e(\mathbf{p})$. In a tracking session with an average of 2500 points/frame the computation of closest-point correspondences takes 250 μs/iteration. We now describe in detail how the projection is evaluated on each element in closed form.

Pill correspondences: $\mathbf{q} = \Pi_{\mathbf{pill}}(\mathbf{p})$

A pill is defined by two spheres $\mathcal{B}_1(\mathbf{c}_1, r_1)$ and $\mathcal{B}_2(\mathbf{c}_2, r_2)$. By construction the closest point correspondence lies on the plane passing through the triplet $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{p}\}$, thus allowing us to solve the problem in 2D; see Figure 3.7-(left). We compute the intersection point $\mathbf{s}$ of the ray $\mathbf{r}(t) = \mathbf{p} + t\mathbf{n}$ with the segment $\overline{\mathbf{c}_1\mathbf{c}_2}$ and parametrize its location in barycentric coordinates as $\mathbf{s} = \alpha\mathbf{c}_1 + (1-\alpha)\mathbf{c}_2$. If $\alpha \in [0,1]$, our closest point correspondence is given by $\mathbf{q} = \Pi_{\mathcal{L}}(\mathbf{p})$, that is, the intersection of $\overline{\mathbf{c}_1\mathbf{c}_2}$ and $\mathbf{r}(t)$. If $\alpha < 0$ or $\alpha > 1$, then the closest point will be $\mathbf{q} = \Pi_{\mathcal{B}_1}(\mathbf{p})$ or $\mathbf{q} = \Pi_{\mathcal{B}_2}(\mathbf{p})$, respectively.

Wedge correspondences: $\mathbf{q} = \Pi_{\mathbf{wedge}}(\mathbf{p})$

A wedge is defined by three spheres $\mathcal{B}_i = \{\mathbf{c}_i, r_i\}$. Figure 3.3 illustrates how a wedge element can be decomposed in three parts: *spherical*, *conical*, and *planar* elements, associated with vertices, edges, and faces of the sphere-mesh skeleton. For the planar element $\mathcal{P}(\mathbf{t}_1, \mathbf{n})$ with normal $\mathbf{n}$ and tangent $\mathbf{t}_1$ to $\mathcal{B}_1$ we compute the skewed projection $\mathbf{s}$ by finding the intersection of the ray $\mathbf{r}(t) = \mathbf{p} + t\mathbf{n}$ with the triangle $\mathcal{T}$ formed by $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$. According to the position of $\mathbf{s}$ we have two possible solutions: If $\mathbf{s}$ lies inside the triangle $\mathcal{T}$, then our footpoint is $\mathbf{q} = \Pi_{\mathcal{P}}(\mathbf{p})$. Otherwise, we use the barycentric coordinates of $\mathbf{s}$ in $\mathcal{T}$ to identify the closest pill element and compute $\mathbf{q} = \Pi_{\text{pill}}(\mathbf{p})$.

Monocular Correspondences

In monocular acquisition (i.e. single sensor), an oracle registration algorithm aligns the portion of the model that is *visible* from the sensor viewpoint to the available data. Hence, when computing ICP's closest-point correspondences, only the portion of the model currently visible by the camera should be considered [Tagliasacchi et al., 2015]. Given the camera direction $\mathbf{v}$, we can test whether the retrieved footpoint $\mathbf{q}$ is back-facing by testing the sign of $\mathbf{v}\cdot\mathcal{N}_{\mathcal{M}}(\mathbf{q})$, where the second term is the object's normal at $\mathbf{q}$. As illustrated in 2D in Figure 3.9, whenever this test fails, there are additional candidates for closest point that must be checked: (1) the closest-point on the silhouette of the model (e.g. $\mathbf{p}_{2,3,6,7}$), and (2) the front facing planar portions of elements (e.g. $\mathbf{p}_5$). These additional correspondences for the query point are computed, and the one closest to $\mathbf{p}$ becomes our front-facing footpoint $\mathbf{q}$. The additional computational cost caused by front-facing correspondences with an average of 2500 points/frame is 100 μs/iteration.

Silhouette computation

The *object-space silhouette* $\partial\mathcal{M}$ is a (3D) curve separating front-facing from back-facing portions of a shape [Olson and Zhang, 2006, Sec.1]. To simplify the silhouette computation we approximate the perspective camera of the sensor with an orthographic one. We then offset all elements on the 2D camera plane, and perform a cross-section with this plane:

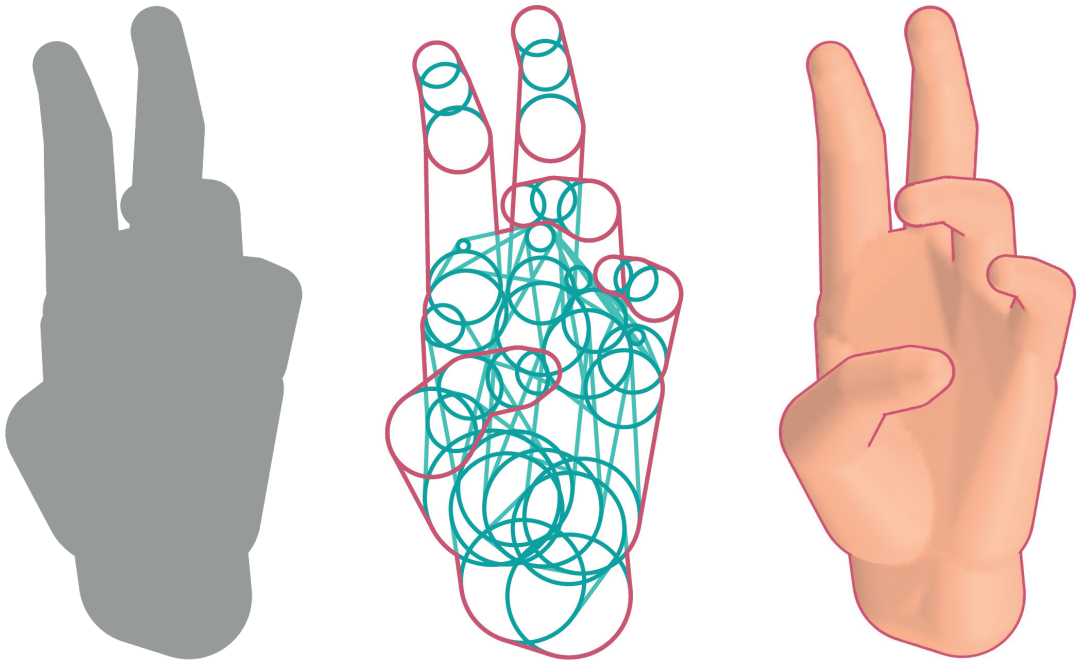


Figure 3.10 – The image-space silhouette of the model computed by projecting the model in the camera plane (left). The 2D object-space silhouette curves are computed separately for palm and fingers and then composited back together (center). The 3D object-space silhouette (pink) is re-projected in 3D (right).

spheres are replaced with circles and planes/cylinders with segments; see Figure 3.10-(left). We then compute an *arrangement*, splitting curves whenever intersection or tangency occurs; see Figure 3.10-(center). We traverse this graph, starting from a point that is guaranteed to be on the outline (e.g. a point on the bounding box). The traversal selects the next element as the one whose tangent forms the smallest counter-clockwise angle thus identifying the silhouette. Once the 2D silhouette has been computed, it can be re-projected to 3D; see Figure 3.10-(right). Note the process described above would compute the *image-space silhouette* of our model. Therefore, we apply the process to palm and fingers separately, and merge them in a second phase. The merge process simply checks whether vertices $v \in \partial\mathcal{M}$ are contained within the model, which means it discards those where $\phi_{\mathcal{M}}(v) < 0$. In our experiments the average computation of the silhouette on the CPU takes 150 μs/iteration.

3.3.2 Rendering

Rendering the sphere-meshes in real time is not only employed for visual verification of tracking performance; e.g. Figure 3.2. The real-time tracking algorithm reviewed above performs a 2D registration in the image plane that requires the computation of an (image-space) silhouette. There are two alternatives for rendering a sphere-mesh model like the one shown in Figure 3.4. One possibility is to explicitly extract the surface of individual elements

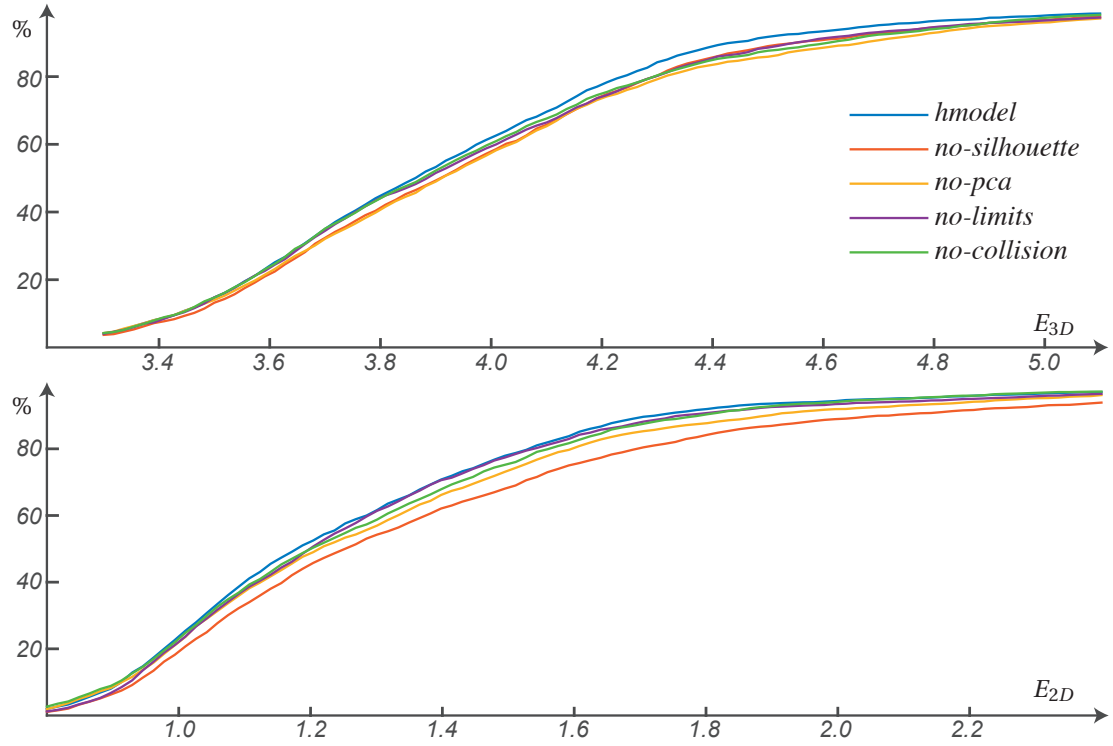


Figure 3.11 – Each plot visualizes on the y axis the portion of frames with a mean error metric below the value reported on the x axis. We employ the HANDY/TEASER sequence for this purpose. Curves closer to the top-left quadrant indicate better performance.

by computing the convex hull of pairs or triplets of spheres; see Figure 3.3. While this process would be suitable in applications where the model is fixed, it is hardly appropriate in our scenario where we want to calibrate the model to the user. Therefore, similarly to [Thiery et al., 2016], we ray-trace the model on the GPU. We render a unit fullscreen quad and in the fragment shader use the camera intrinsics to compute the camera ray $\mathbf{r}(\mathbf{x})$ associated with each pixel $\mathbf{x}$. Each ray is intersected with each element of our model, and the closest intersection point is retained. Tests are performed with the planar, conical, and spherical primitives that compose each element. Rendering at a resolution of 320×240 pixels provides the best trade-off between accuracy and performance, leading to a total rendering time of ≈ 3ms for visualization and $\approx 500\mu$s/iteration for the evaluation of E_{m2d}.

3.4 Calibration

Our calibration procedure adapts our template model to a specific user from a set of N 3D measurements $\{\mathcal{D}_1 \dots \mathcal{D}_N\}$ of the user's hand in different poses. Multiple measurements are necessary, as it is not possible to understand the kinematic behavior by analyzing static geometry, and the redundancy of information improves fitting precision. Further, in monocular acquisition this redundancy is essential, as single-view data is highly incomplete, making the

problem ill-posed. In our research we have experimented with datasets $\{\mathcal{D}_n\}$ acquired via multi-view stereo (e.g. *Agisoft Photoscan*), as well as a single RGBD sensor. Our calibration formulation can be employed for both acquisition modalities. Dynamic reconstruction frameworks such as [Newcombe et al., 2015] or [Innmann et al., 2016] could also be used to generate a dynamic template mesh over which sphere-mesh decimation could be executed [Thiery et al., 2016]. However, as no public implementation is currently available, it is currently unclear how well these methods would cope with loop-closure for features as small as human fingers.

Kinematics

The rest-pose geometry of our model is fully specified by two matrices specifying the set of sphere positions $\bar{\mathbf{C}}$ and the set of radii $\bar{\mathbf{r}}$. The geometry is then posed through the application of kinematic chain transformations; see Figure 3.8a. Given a point $\bar{\mathbf{p}}$ on the model $\mathcal{M}$ at rest pose, its 3D position after posing can be computed by evaluating the expression:

$$\mathbf{p} = \left[\Pi_{k \in K(\bar{\mathbf{p}})} \bar{\mathbf{T}}_k \mathbf{T}_k \bar{\mathbf{T}}_k^{-1}\right] \bar{\mathbf{p}} \tag{3.7}$$

where $\mathbf{T}_*$ are the *pose* transformations parameterized by $\boldsymbol{\theta}$ and Π left multiplies matrices by recursively traversing the kinematic chain K of point $\bar{\mathbf{p}}$ towards the root [Buss, 2004]. Each node k of the kinematic chain is associated with an orthogonal frame $\bar{\mathbf{T}}_k$ according to which local transformations are specified. In most tracking systems, the frames $\bar{\mathbf{T}}_*$ are manually set by a 3D modeling artist and kept fixed across users. However, incorrectly specified kinematic frames can be highly detrimental to tracking quality; see Figure 3.8(c,d) and Video2 [02:12]. Therefore, in our formulation, the kinematic structure (i.e. the matrices $\bar{\mathbf{T}}_*$) is directly optimized from acquired data.

Formulation

Let $\boldsymbol{\theta}_n$ be the *pose* parameters optimally aligning the rest-pose template to the data frame $\mathcal{D}_n$, and $\bar{\boldsymbol{\delta}}$ be the *posture* parameters representing the transformations $\bar{\mathbf{T}}_*$ via Euler angles.

For notational brevity, we also define $\boldsymbol{\Theta}_n = [\boldsymbol{\theta}_n, \bar{\boldsymbol{\delta}}, \bar{\mathbf{C}}, \bar{\mathbf{r}}]$. Our calibration optimization can then be written as:

$$\underset{\{\boldsymbol{\Theta}_n\}}{\arg\min} \sum_{n=1}^{N} \sum_{\mathcal{T} \in \mathcal{T}_{\text{calib}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathcal{D}_n, \boldsymbol{\Theta}_n) \tag{3.8}$$

We employ a set of energies $\mathcal{T}_{\text{calib}}$ to account for different requirements. On one hand we want a model that is a good fit to the data; on the other, we seek a non-degenerate sphere-mesh template that has been piecewise-rigidly posed. The following calibration energies $\mathcal{T}_{\text{calib}}$ encode these requirements:

d2m data to model distance

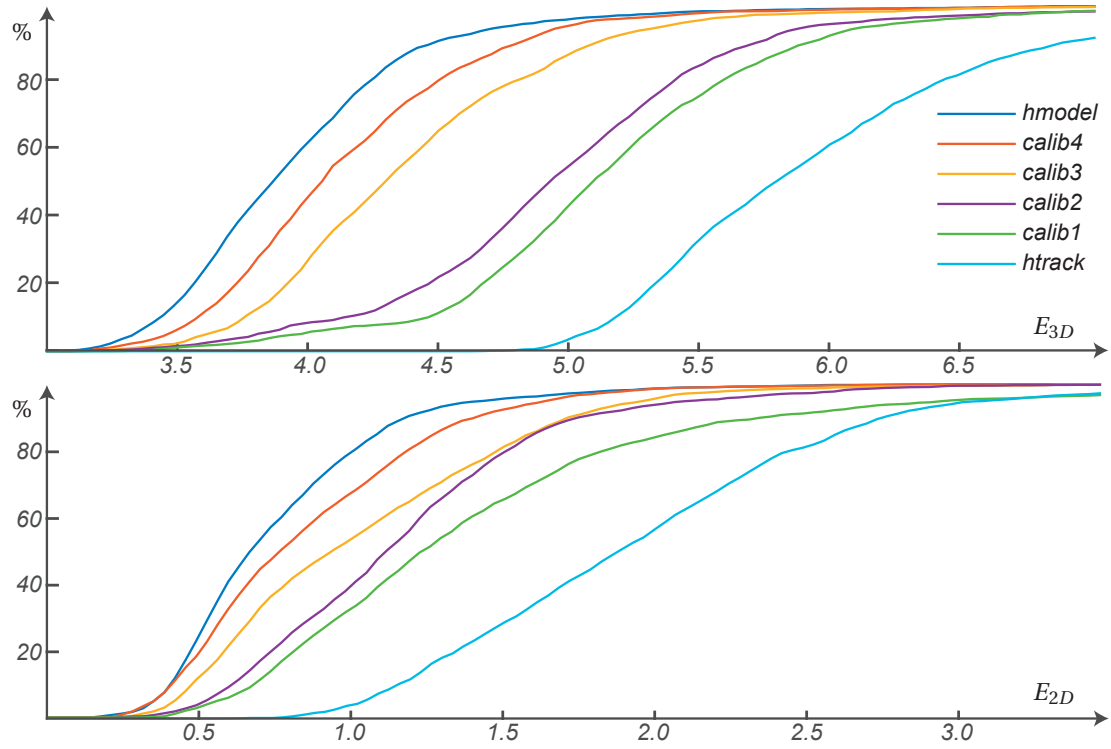


Figure 3.12 – Calibrating progressively improves the 2D/3D tracking metrics, showing a remarkable improvement in tracking fidelity from [Tagliasacchi et al., 2015] to [Proposed Method].

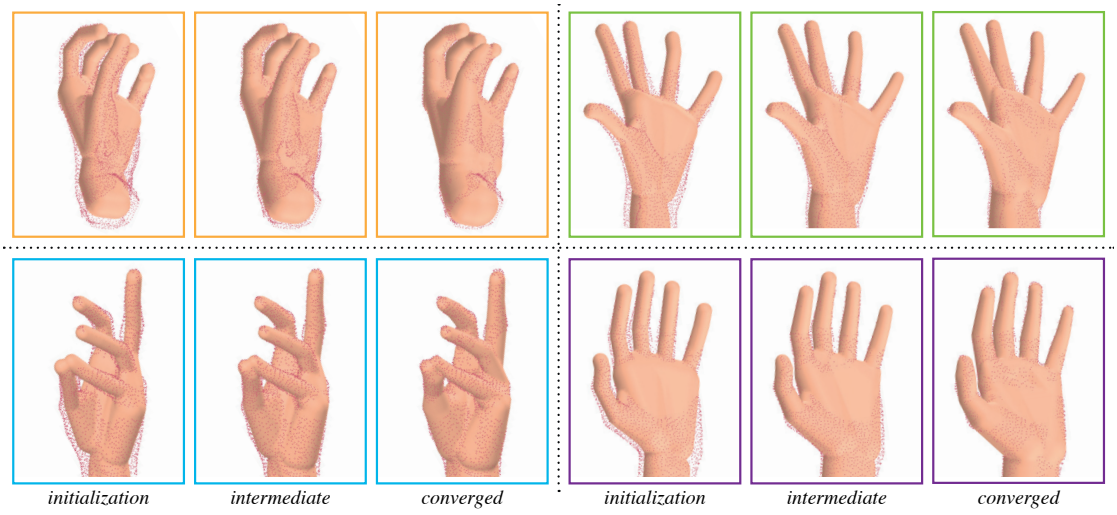


Figure 3.13 – A visualization of a few iterations of our calibration optimization procedure; see Video2 [01:30]. Each quadrant displays a data frame $\mathcal{D}_n$, $n = 1 \ldots 4$. Within each quadrant we show three iterations of the optimization. The model being calibrated here is the one employed for real-time tracking in Video2 [02:57].

m2d model to data distance
rigid elements are posed rigidly

valid elements should not degenerate

To make this calibration more approachable numerically, we rewrite Equation 3.8 as an alternating optimization problem:

$$\underset{\{\mathbf{C}_n\},\bar{\mathbf{C}},\bar{\mathbf{r}}}{\arg\min} \quad \sum_{n=1}^{N}\sum_{\mathcal{T}\in\mathcal{T}_{\text{calib}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathcal{D}_n, \mathbf{C}_n, \bar{\mathbf{C}}, \bar{\mathbf{r}}) \tag{3.9}$$

$$\underset{\{\boldsymbol{\theta}_n\},\bar{\boldsymbol{\delta}}}{\arg\min} \quad \sum_{n=1}^{N}\sum_{\mathcal{T}\in\mathcal{T}_{\text{calib}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathbf{C}_n, \boldsymbol{\Theta}_n) \tag{3.10}$$

Our first step adjusts rest-pose sphere centers $\bar{\mathbf{C}}$ and radii $\bar{\mathbf{r}}$, by allowing the model to fit to the data without any kinematic constraint beyond rigidity, and returning as a side product a set of *per-frame* posed centers $\{\mathbf{C}_n\}$. Our second step takes the set $\{\mathbf{C}_n\}$ and projects it onto the manifold of kinematically plausible template deformations. This results in the optimization of the rotational components of rest-pose transformations $\bar{\mathbf{T}}_*$, as their translational components are simply derived from $\bar{\mathbf{C}}$.

Optimization

The energies above are non-linear and non-convex, but can be optimized offline, as real-time tracking only necessitates a pre-calibrated model. For this reason, we conveniently employ the $lsqnonlin$ Matlab routine, which requires the gradients of our energies as well as an initialization point. The initialization of $\bar{\mathbf{C}}$ is performed automatically by anisotropically scaling the vertices of a generic template to roughly fit the rest pose. The initial transformation frame rotations $\bar{\boldsymbol{\delta}}$ are retrieved from the default template, while $\{\boldsymbol{\theta}_n\}$ are obtained by either aligning the scaled template to depth images, or by executing inverse kinematics on a few manually selected keypoints (multi-view stereo). Our (unoptimized) Matlab script calibrates the model within a few minutes for all our examples.

3.4.1 Energies

Our fitting energies are analogous to the ones used in tracking. They approximate the symmetric Hausdorff distance, but they are evaluated on a *collection* of N frames:

$$E_{d2m} = \sum_{n=1}^{N} |\mathcal{D}_n|^{-1} \sum_{\mathbf{p}\in\mathcal{D}_n} \|\mathbf{p} - \Pi_{\mathcal{M}(\boldsymbol{\Theta}_n)}(\mathbf{p})\|_2^1 \tag{3.11}$$

$$E_{m2d} = \sum_{n=1}^{N} |\mathcal{M}(\boldsymbol{\Theta}_n)|^{-1} \sum_{\mathbf{x}\in\mathcal{M}(\boldsymbol{\Theta}_n)} \|\mathbf{x} - \Pi_{\mathcal{D}_n}(\mathbf{x})\|_2^1 \tag{3.12}$$

Note that the projection operator $\Pi_{\mathcal{D}_n}$ changes according to the type of input data. If a multi-view acquisition system is used to acquire a complete point cloud, then the projection operator fetches the closest point to $\mathbf{p}$ in the point cloud of frame $\mathcal{D}_n$. If $\mathcal{D}_n$ is acquired through monocular acquisition, then $\Pi_{\mathcal{D}_n}$ computes the 2D projection to the image-space silhouette

of the model.

Rigidity

It is essential to estimate a single user template that, once articulated, *jointly* fits the set of data frames $\{\mathcal{D}_n\}$. For this purpose we require each posed model to be a piecewise-rigid articulation of our rest pose. This can be achieved by constraining each segment $\{(\mathbf{c}_{n,i}, \mathbf{c}_{n,j}) \mid ij \in \mathcal{S}\}$ of $\mathbf{C}_n$ to have the same length as the corresponding segment $(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_j)$ of the rest pose configuration $\bar{\mathbf{C}}$:

$$E_{\text{rigid}} = \sum_{ij \in \mathcal{S}} (\|\mathbf{c}_{n,i} - \mathbf{c}_{n,j}\| - \|\bar{\mathbf{c}}_i - \bar{\mathbf{c}}_j\|)^2 \tag{3.13}$$

Note that only a subset of the edges of our control skeleton, as illustrated in Figure 3.4, are required to satisfy this rigidity condition.

Validity

The calibration optimization should avoid producing degenerate configurations in our *rest pose* template $\bar{\mathbf{C}}$. For example, a pill degenerates into a sphere when one of its balls is fully contained within the volume of the other. Analogously, a wedge can degenerate into a pill or a sphere. We monitor validity by an indicator function $\chi(\bar{\mathcal{B}}_i)$ that evaluates to one if $\bar{\mathcal{B}}_i$ is degenerate and zero otherwise. We make a conservative choice and use $\chi(\bar{\mathcal{B}}_i)$, which verifies whether $\bar{\mathbf{c}}_i$ is inside $\bar{\mathcal{E}} \setminus \bar{\mathcal{B}}_i$, the element obtained by removing a vertex, as well as all its adjacent edges, from $\bar{\mathcal{E}}$. This leads to the following conditional penalty function:

$$E_{\text{valid}} = \sum_{\bar{\mathcal{E}} \in \bar{\mathbf{C}}} \sum_{\bar{\mathcal{B}}_i \in \bar{\mathcal{E}}} \chi(\bar{\mathcal{B}}_i) \|\bar{\mathbf{c}}_i - \Pi_{\bar{\mathcal{E}} \setminus \bar{\mathcal{B}}_i}(\bar{\mathbf{c}}_i)\|_2^2 \tag{3.14}$$

3.5 Results

We evaluate our technique on a variety of sequences across a number of users, and performe qualitative as well as quantitative comparisons of our method to the state-of-the-art [Qian et al., 2014, Sridhar et al., 2015, Tagliasacchi et al., 2015, Sharp et al., 2015, Taylor et al., 2016]. We also propose new algorithm-agnostic metrics tailored to high-precision tracking evaluation, and introduce the HANDY dataset.

Template Calibration

The calibration of our model to a collection of 3D data frames is illustrated in Figure 3.13; note that the same model is rigidly articulated to fit to multiple poses. While for this user we build a model from multi-view stereo data (omni-directional, complete), it is important to notice that the use of multiple frames in different poses is a necessity. Only in this situation

can the centers $\{\mathbf{C}_n\}$ be jointly adjusted to create an articulated model that consensually fits the whole dataset. We refer the reader to Video2 [01:30] for a visualization of our iterative calibration procedure. The calibration from RGBD datasets can be seen at Video2 [01:45], and the resulting models are illustrated in Figure 3.4.

Kinematic Calibration

The importance of adjusting kinematic chain transformations is shown in Figure 3.8, as well as the first images pair in Figure 4.1. With incorrect transformations, joint limits and the articulation restrictions of the kinematic chain can prevent the model from being posed correctly; see a dramatization in Video2 [02:12]. In our experiments we discovered it was crucial to identify the *typical* kinematic chain structure using the dataset in Figure 3.13; user-specific calibration optimization used these transformations as initialization.

Comparison metrics

Taylor and colleagues [Taylor et al., 2016] have recently reported how state-of-the-art hand tracking algorithms have reached human precision in determining the location of key features (e.g. fingertips and wrist position). Therefore, publicly available datasets like [Tompson et al., 2014] and [Sridhar et al., 2013], often relying on human labeling of data, are now unsuitable to quantitatively evaluate the quality of high-precision tracking. We propose two easy-to-compute metrics to evaluate the quality of generative tracking algorithms. A core element that makes these metrics appealing is that, much like key feature positions, they are completely *algorithm agnostic*: they can be evaluated as far as a depth map of the tracking model can be synthesized. This is essential, as it will enable the research community to validate and compare results through quantitative analysis. We achieve this goal by expressing these metrics exclusively as a function of the acquired depth image $\mathcal{D}_n$ and of the depth image $\mathcal{R}_n$ of the rendered model. Below we drop the subscript n for notational brevity and only consider points $\mathbf{p}$ within the RoI. The data-to-model metric is:

$$E_{3\mathrm{D}} = |\mathcal{D}|^{-1} \sum_{\mathbf{p}\in\mathcal{D}} \|\mathbf{p} - \Pi_{\mathcal{R}}(\mathbf{p})\|_2^1 \tag{3.15}$$

Differently from before, $\Pi_{\mathcal{R}}$ computes the (kd-tree accelerated) closest point correspondence to the rendered model *point cloud*, rather than to the model itself. The model-to-data metric is:

$$E_{2\mathrm{D}} = |\mathcal{R} \setminus \partial\mathcal{D}|^{-1} \sum_{\mathbf{x}\in\mathcal{R}\setminus\partial\mathcal{D}} \|\mathbf{x} - \Pi_{\partial\mathcal{D}}(\mathbf{x})\|_2^1 \tag{3.16}$$

Each summation term above can be evaluated efficiently by pre-computing the 2D Euclidean distance transform of the RoI's (image-space) silhouette $\partial\mathcal{D}$ [Tagliasacchi et al., 2015], where the transform evaluates to zero for a pixel inside the silhouette. Another algorithm agnostic

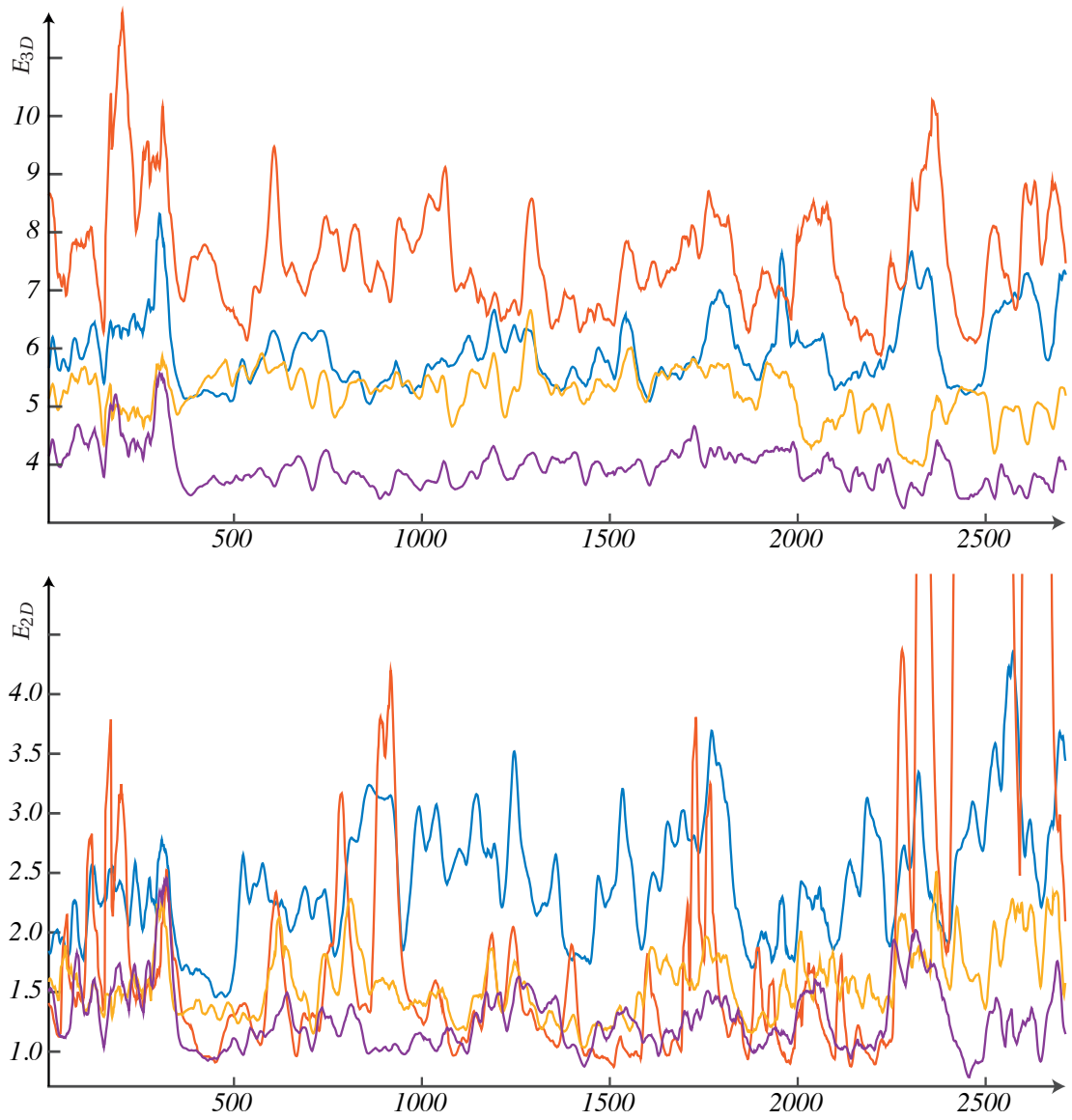


Figure 3.14 – [Tkach et al. 2016] is quantitatively compared over time to [Tagliasacchi et al. 2015], [Sharp et al. 2015] and [Taylor et al. 2016] on the HANDY/TEASER sequence.

metric is the *golden energy* from Sharp et al. [Sharp et al., 2015], but this distance does not encode a monocular Hausdorff like ours do.

Handy dataset

We create the new HANDY tracking dataset for the evaluation of high-precision generative tracking algorithms. Our dataset contains ≈ 30k depth and color images recorded with an *Intel RealSense* SR300 sensor. The dataset is designed to cover the entire range of motions that has been surveyed in recent techniques. As detailed in Figure 3.16, we identified three main axes

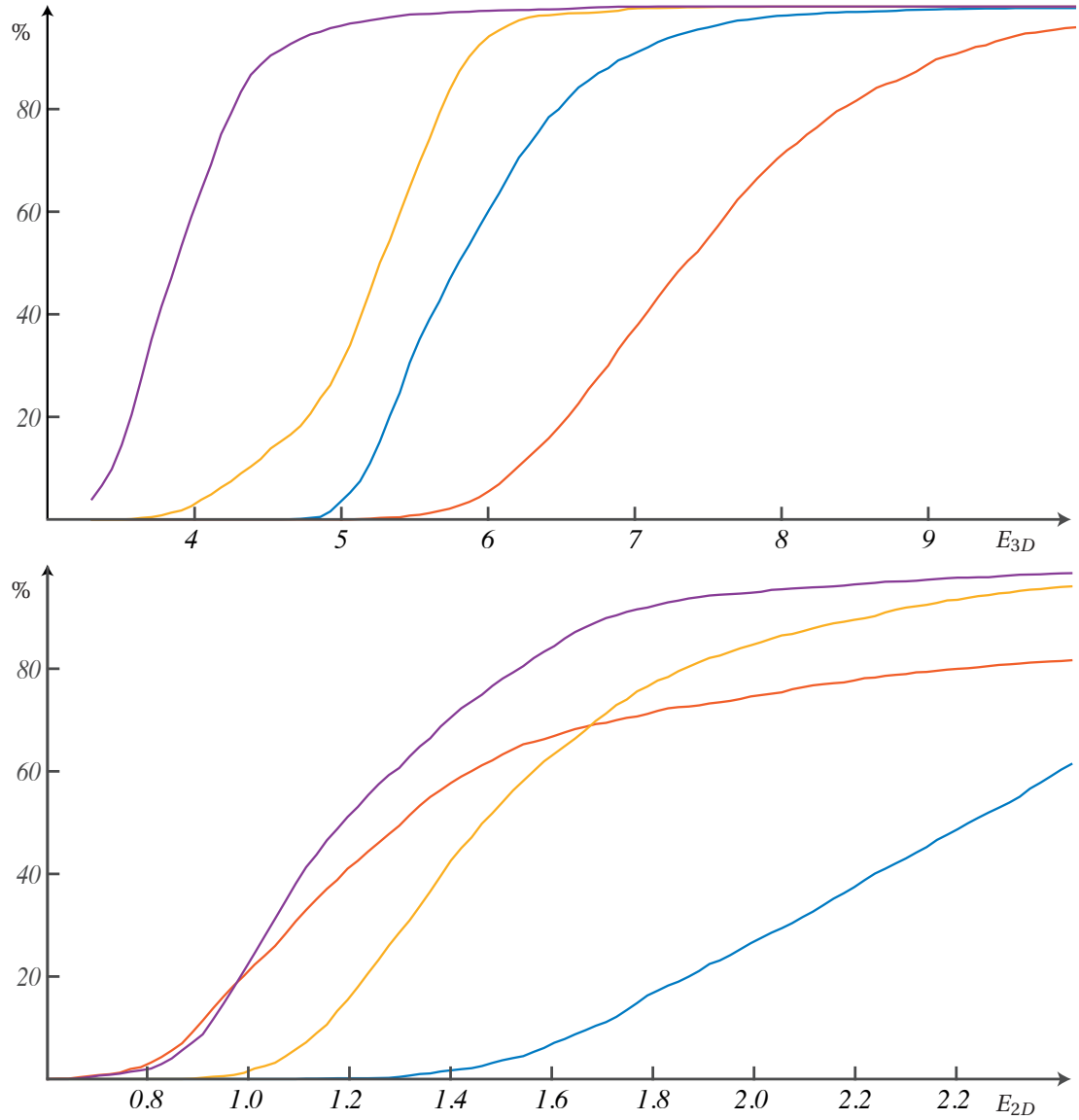


Figure 3.15 – Aggregated errors are reported for the tracking sequences in Figure 3.14. These aggregated measures reveal a significant improvement in tracking precision; see legend in Figure 3.14.

of complexity in the hand tracking literature, and devised the TEASER dataset to thoroughly sample this space; see Video2 [02:51].

Further, to enable qualitative comparisons to motions from state-of-the-art papers we also devised an additional set of sequences:

Video2 [04:53] – TAYL1 rigid and clenching
Video2 [05:40] – SRID1 fingers extension
Video2 [05:56] – SRID2 fingers contact

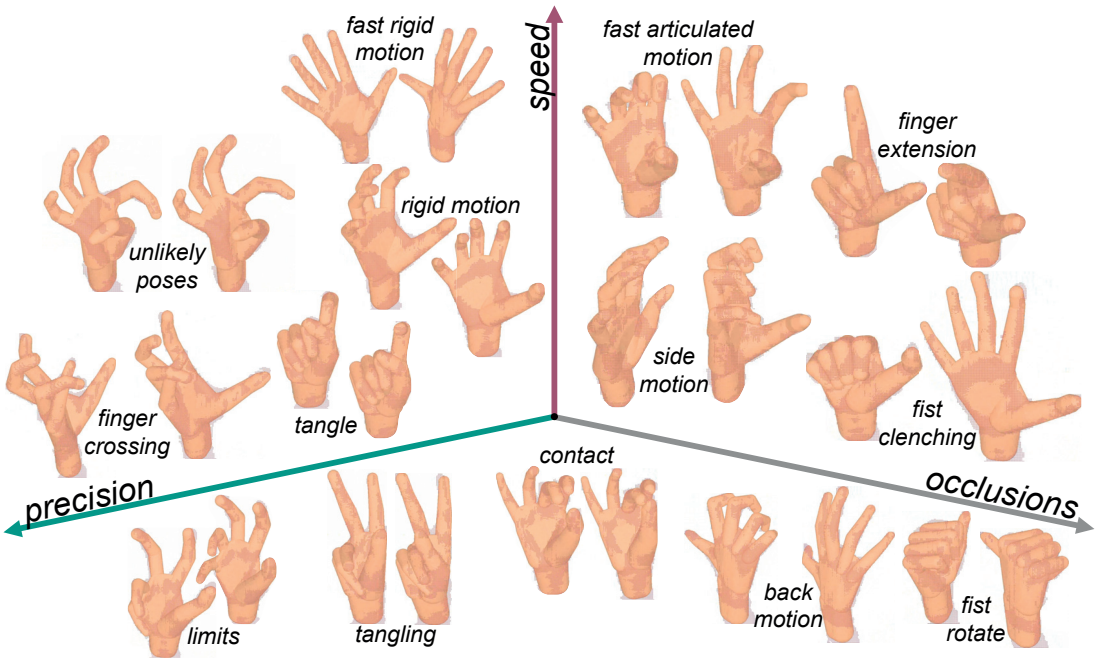


Figure 3.16 – Our dataset contains a wide range of motions. We identify three main axes of complexity by analyzing recent hand-tracking papers; see Video2 [02:51]. The distance to the origin indicates the level of tracking difficulty.

Video2 [06:15] – SRID3	crossing fingers
Video2 [06:29] – SRID4	pinching
Video2 [07:33] – SHAR1	fast and complex
Video2 [08:07] – SHAR2	fast rigid
Video2 [08:37] – SHAR3	rotating fist

The sequences marked as *tayl**, *srid**, and *shar** are respectively designed to emulate the motions in [Taylor et al., 2016], [Sridhar et al., 2015] and [Sharp et al., 2015]. We do not devise sequences for [Qian et al., 2014] and [Tompson et al., 2014], as the previous datasets already covered the motion space.

Self evaluation

In Figure 3.11, we adopt the self-evaluation visualization proposed by [Taylor et al., 2016]. We study the changes in algorithm performance as we disable the tracking energy terms in Equation 3.3 on the HANDY/TEASER sequence – in all tests, the *d2m* term is never disabled, as otherwise immediate loss of tracking occurs. Not surprisingly, we identify the *m2d* and *pose* terms to be the ones dominating tracking performance. Similarly to [Taylor et al., 2016], while the contribution of other terms is small, we found that it still yields a visually noticeable improvement.

Quantitative comparison

Our algorithm has been tested with the *Intel RealSense* SR300 (QVGA@60Hz). We have tailored the method of [Tagliasacchi et al., 2015] to support this sensor to enable quantitative comparisons. In Figure 3.14, our two metrics are plotted per-frame as multiple tracking algorithms are executed on the HANDY/TEASER sequence, while Figure 3.15 reports aggregated errors; see Video2 [03:53]. It is important to note our metrics are designed to evaluate fitting precision; the method of [Sharp et al., 2015] still achieves good tracking robustness on the test sequences, but the lack of user calibration heavily biases this metric. Aggregated performance comparisons are also reported in Figure 3.17 for each sequence in the HANDY dataset; see Video2 [04:50]. These metrics reveal a consistent and significant increase in performance. Figure 3.12 quantitatively illustrates the tracking benefits of template calibration.

Qualitative comparison

We employ the HANDY sequences to perform a qualitative comparison to [Qian et al., 2014, Sridhar et al., 2015, Sharp et al., 2015, Taylor et al., 2016]. As it can be observed in Video2 [04:50], our calibrated tracker is capable to replicate any of the motions benchmarked by state-of-the-art techniques with excellent accuracy.

Further comparisons

Given a sufficiently rich annotated data sample, it is generally possible to adapt a discriminative tracker to a different sensor from what it was originally designed for. However, for generative algorithms the task requires some parameter tweaking, a challenging task to achieve without direct access to each sensor variant. For these reasons, comparisons to datasets developed on different sensors like DEXTER [Sridhar et al., 2013] or FINGERPAINT [Sharp et al., 2015] would be misleading. Most importantly, these datasets were acquired at 30Hz, while our generative algorithm is specifically designed to execute at 60Hz. To enable a fair comparison, we would require the per-frame re-initializer employed by the authors, but no source code for these algorithms is available.

3.6 Discussion

Our analysis demonstrates how sphere-meshes tracking templates are particularly well suited for real-time hand-tracking. Our calibration and tracking algorithms are simple to implement, efficient to optimize for, and allow for the geometry to be represented with high fidelity. While the calibration algorithm is currently implemented in Matlab, we are confident real-time performance can be achieved with a simple C++ port of our code. The recently proposed system of Taylor and colleagues [Taylor et al., 2016], has also demonstrated excellent tracking quality. Their formulation employs a triangular mesh model and optimizes it in a semi-

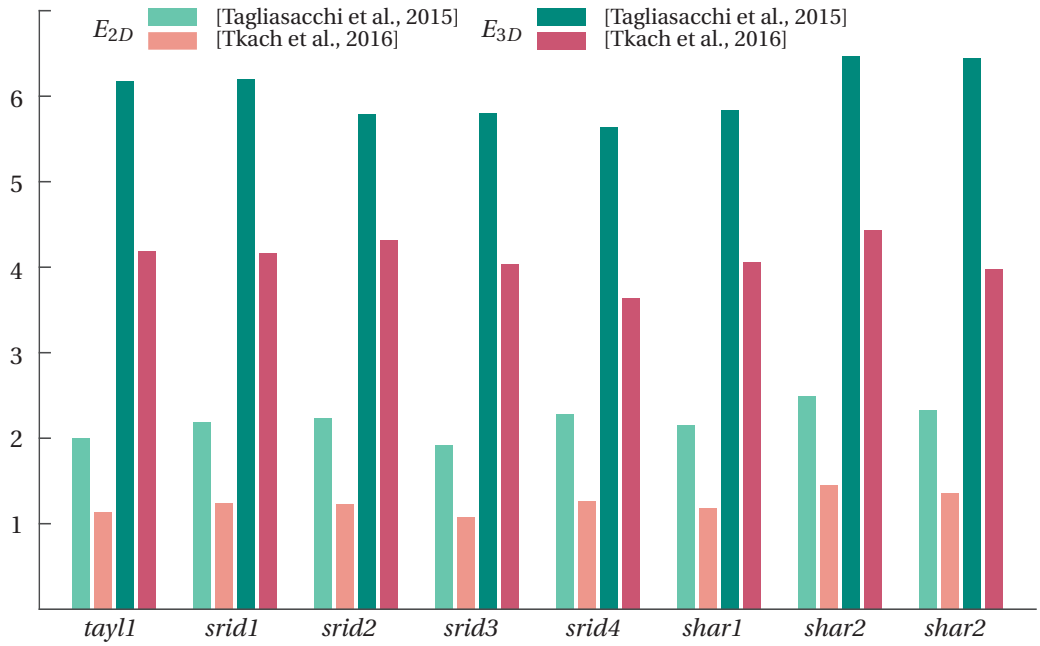


Figure 3.17 – Average 2D/3D tracking performance metrics of the proposed method compared to [Tagliasacchi et al., 2015]. In the additional material we report error plots through time for the aggregated data above.

continuous fashion. However, as their model is articulated through linear blend skinning, joint collapse artifacts can occur. Conversely, our model is volumetric and naturally overcomes this shortcoming; see Video2 [01:18]. Although it is difficult to predict whether surface or volumetric models will eventually prevail, we believe the simplicity of our representation will lead to extremely performant articulated tracking algorithms.

Generative tracker

In this chapter we demonstrated a generative algorithm that yields unprecedented levels of robustness to tracking failure. We would like to stress that our real-time tracking algorithm is (almost) *purely* generative: a discriminative technique [Qian et al., 2014] is only employed in the first frame for tracking initialization. We believe our robustness is due to the quality of the calibrated model, and to the ability to optimize at a constant 60Hz rate. Discriminative algorithms could still be necessary to compensate for situations where the hand re-appears from complete occlusions, but their role in real-time tracking will diminish as RGBD sensors will start offering imaging at frequencies above $60Hz$. To highlight our high frame-rate dependancy, in Video2 [11:02] we analyze the performance on the tracker with varying frame-rates (60Hz, 30Hz, 15Hz and 7.5Hz) while the additional material reports the corresponding tracking metrics. In Video2 [10:30] we further investigate tracking failures that include long phases of total occlusion; note how in these scenarios the mean-pose (Probabilistic PCA) regularizer

described in [Tagliasacchi et al., 2015, Eq.6] helps tracking recovery.

Downsampling

Although the *Intel RealSense* sensor is a short range camera, in this work we have downsampled the depth image to QVGA format with a median filter, giving an average of 2500 pixels/frame; this is approximatively the number of samples found on a hand in long-range cameras. The recent work of [Taylor et al., 2016] reports a total of 192 pixels/frame, therefore enabling CPU optimization without significant loss of tracking precision. Inspired by this work, we have experimented with further downsampling and reached analogous conclusions. However, the computational bottleneck of the *htrack* system lies in the overhead caused by render/compute context switching. While this is currently an issue, it is possible to optimize the *m2d* energies without rasterizing the model at each iteration. Instead, similarly to [Qian et al., 2014], we could compute screen-space coordinates of sphere centers, and then construct our *m2d* registration energies on this subset.

Reproducibility

The weights of the energy terms used in tracking and calibration optimizations have been identified by manually tweaking the runtime until our tracker reached the desired performance level. The parameters of our system are $\tau_{d2m} = 1$, $\tau_{m2d} = .5$, $\tau_{rigid} = .3$, $\tau_{valid} = 1e2$, $\tau_{pose} = 1e4$, $\tau_{limits} = 1e7$ and $\tau_{collision} = 1e3$. We use 7 iterations for the tracking LM optimization, while $lsqnonlin$ automatically terminates in 5-15 iterations. Source code and datasets are available at: *http://github.com/OpenGP/hmodel*.

3.7 Conclusion

In this chapter we have introduced the use of sphere-meshes as a novel geometric representation for articulated tracking. We have demonstrated how this representation yields excellent results for real-time registration of articulated geometry, and presented a calibration algorithm to estimate a per-user tracking template. We have validated our results by demonstrating qualitative as well as quantitative improvements over the state-of-the-art. Our volumetric model can be thought of as a generalization of the spherical models presented in [Sridhar et al., 2015, Qian et al., 2014], and the cylinder models of [Oikonomidis et al., 2011, Tagliasacchi et al., 2015]. It is also related to the convex body model from [Melax et al., 2013], with the core advantage that its control skeleton compactly parameterizes its geometry. Our calibration optimization is related to the works in [Taylor et al., 2014, Khamis et al., 2015, Tan et al., 2016], with a fundamental difference: the innate simplicity of sphere-meshes substantially simplifies the algorithmic complexity of calibration and tracking algorithms. This considered, we believe that with the use of *compute shaders*, articulated tracking on the GPU can become as effortless and efficient as simple mesh rasterization.

Limitations and future work

The topology of our template has been defined in a manual trial-and-error process. A more suitable topology could be estimated by optimization, possibly even adapting the topology for specific users; For example, the work in [Thiery et al., 2016] could be extended to space-time point clouds. Similarly, one could think of a variant of [Newcombe et al., 2015] where sphere-meshes are instantiated on-the-fly. The use of more advanced re-initialization techniques than [Qian et al., 2014], like [Krupka et al., 2014] or [Oberweger et al., 2015b], would be beneficial. Further, we believe an interesting venue for future work is how to elegantly integrate per-frame estimates into generative trackers. Model calibration is currently done in pre-processing. For certain consumer applications, it would be desirable to calibrate the model online during tracking, as recently proposed for face tracking systems [Bouaziz et al., 2013]. Our sphere-mesh models are a first approximation to the implicit functions lying at the core of the recently proposed geometric skinning techniques [Vaillant et al., 2013, Vaillant et al., 2014]. Therefore, we believe the calibration of *sphere-meshes* to be the first step towards photorealistic real-time hand modeling and tracking.

4 Online Generative Model Personalization for Hand Tracking

Figure 4.1 – Our adaptive hand tracking algorithm optimizes for a tracking model on the fly, leading to progressive improvements in tracking accuracy over time. **Above**: hand surface color-coded to visualize the spatially-varying confidence of the estimated geometry. Insets: color-coded *cumulative* certainty. Notice how in the last frame all parameters are certain. **Below**: histograms visualize the certainty of each degree of freedom, that is, the diagonal entries of the inverse of the covariance estimate from: (a) data in the current frame Σ^*, or (b) the information $\hat{\Sigma}$ accumulated through time by our system.

This chapter is based on the following publication:

> TKACH A., TAGLIASACCHI A., REMELLI E., PAULY M., FITZGIBBON A.: Online generative model personalization for hand tracking. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia). 2017.*

Abstract

We present a new algorithm for real-time hand tracking on commodity depth-sensing devices. Our method does not require a user-specific calibration session, but rather learns the geometry as the user performs live in front of the camera, thus enabling seamless virtual interaction

at the consumer level. The key novelty in our approach is an online optimization algorithm that jointly estimates pose and shape in each frame, and determines the uncertainty in such estimates. This knowledge allows the algorithm to integrate per-frame estimates over time, and build a personalized geometric model of the captured user. Our approach can easily be integrated in state-of-the-art continuous generative motion tracking software. We provide a detailed evaluation that shows how our approach achieves accurate motion tracking for real-time applications, while significantly simplifying the workflow of accurate hand performance capture. We also provide quantitative evaluation datasets at http://lgg.epfl.ch/publications/2017/HOnline/index.php

4.1 Introduction

Tracking templates and personalization

Since depth imagery provides incomplete 3D data of the tracked object, generative trackers attempt to register a geometric *template*, also referred to as a *tracking model*, to 3D data so to minimize alignement residuals. The more accurately a model fits the observed user, the better tracking accuracy can be achieved [Tkach et al., 2016, Taylor et al., 2016]. The process of accurately generating a *user-specific tracking model* from input data is referred to in the literature as *calibration* or *personalization*.

Calibrating a template from a set of static poses is a standard component in the workflow of facial performance capture [Weise et al., 2011, Cao et al., 2015], and the work of [Taylor et al., 2014] pioneered it within the realm of hand tracking. However, current methods such as [Taylor et al., 2016] and [Tkach et al., 2016] suffer a major drawback: the template must be created during a controlled calibration stage, where the hand is scanned in several static poses (i.e. *offline*). While appropriate for professional use, a calibration session is a severe drawback for seamless deployment in consumer-level applications. Therefore, inspired by recent efforts in facial performance capture that calibrate templates while tracking [Li et al., 2013, Bouaziz et al., 2013], in this chapter we propose a pipeline for *online* model calibration. The approach we present has been tailored to monocular acquisition, where we tackle the significant technical challenges created by missing data due to self-occlusions.

Contributions

Our core contribution is a principled way to integrate per-frame information into an online real-time pose/shape tracking algorithm: one that estimates the hand's *pose*, while simultaneously refining its *shape*. That is, as more of the user's hand and articulation is observed during tracking, the more the tracking template is progressively adapted to match the performer, which in turns results in more accurate motion tracking. From a single frame only a subset of the shape degrees of freedom can be estimated, for example, it is difficult to estimate the length of a phalanx when observing a straight finger. Our technique automatically estimates

the *confidence* in per-frame parameter computations, and leverages this information to build a tracking model that selectively *accumulates* confident parameter estimates over time. Assuming a reasonable performance by the user, our system typically constructs a fully calibrated model within a few seconds, while simultaneously tracking the user in real time. Perhaps more importantly, however, if the user is "unreasonable", holding his/her hand in an ambiguous pose (e.g. fingers unbent), the system maintains its shape uncertainty until a constraining pose is adopted.

The key technical component of our solution is a recent tool from control theory – the Levenberg-Marquardt Kalman Filter (LMKF) of [Skoglund et al., 2015]. Although it has long been known [Bell and Cathey, 1993, Bellaire et al., 1995] that there are strong links between Levenberg-style algorithms and the Kalman filter, and that Kalman filters are useful to maintain uncertainty in visual tracking and SLAM [Strasdat et al., 2012], only recently have the advantages of both views been combined. This chapter shows, in both qualitative and quantitative performance evaluations, that the LMKF enables practically useful online calibration. Overall, our solution yields a fully automatic, real-time hand tracking system that is well-suited for consumer applications.

4.2 Related Work

Model personalization is a core ingredient in generative motion tracking. Due to the large number of hand self-occlusions, the low signal-to-noise ratio in current depth sensors, a globally unconstrained pose, and the similar appearance of fingers make the personalization of a hand model a harder problem than face or body model calibration; see [Supancic et al., 2015].

Offline model calibration

[Albrecht et al., 2003] pioneered the construction of realistic (skin, bone and muscles) personalized models. They proposed a pipeline for the registration of a 3D mesh model to RGB data manually pre-processed by the user. Reducing the amount of manual interaction required from the user, [Rhee et al., 2006] showed how skin creases and silhouette images can also be used to guide the registration of a model to color imagery. [Taylor et al., 2014] introduced a more automatic pipeline, generating personalized hand models from input depth sequences where the user rotates his hand while articulating fingers. More closely related to ours is the work by [Tan et al., 2016]. They show how to robustly personalize a hand model to an individual user from a set of depth measurements using a trained shape basis such as the one proposed by [Khamis et al., 2015]. The calibration pipeline, although robust and efficient, is not fully automated as the user needs to *manually* pick the set of frames over which the calibration optimization is performed. In facial calibration, [Weise et al., 2011] asked users to assume a set of standard facial expressions to match standard poses in the Facial Action Coding System (FACS) of [Ekman and Friesen, 1977]. Inspired by these approaches, [Taylor

et al., 2016] recently proposed an analogous offline hand calibration method, but the question "which is the set of *optimal* hand poses that allows to properly capture the hand's geometry?" has yet to be addressed. Hence, none of the above methods is suitable or easily adaptable to the kind of consumer-level applications that we target.

Online model calibration

In [de La Gorce et al., 2011], the authors introduced a (non real-time) model-based approach for hand tracking from a monocular RGB video sequence. Hand pose, texture and lighting are dynamically estimated, while shape is determined by optimizing over the first frame only. Recently [Makris and Argyros, 2015] proposed a model-based approach to jointly solve the pose tracking and shape estimation problem from depth measurements in an online framework. They solve for the cylindrical geometry of a hand through render-and-compare evaluations over a set of frames with particle swarm optimization (PSO). Their pipeline runs in real-time (30fps), but lacks the degree of robustness and accuracy desirable for consumer level applications, and does not address uncertainty. More sophisticated approaches to information agglomeration such as the ones for face tracking/modeling by [Bouaziz et al., 2013], [Li et al., 2013] and [Thies et al., 2015], where shape estimation is performed over the whole set of frames, allow to obtain more accurate results, while guaranteeing real-time performances. [Thies et al., 2015] jointly optimize face identity and expression during calibration stage and keep identity fixed during tracking. The work of [Zou and Tan, 2013], although in a different applicative domain, is also related to ours, as they solve for SLAM by considering uncertainties when aggregating information through time. [Gu et al., 2017] propose a holistic approach for aggregating per-frame measurements. They demonstrate how an LSTM layer in a CNN allows to maintain an online estimate that surpasses the performance of a more standard Kalman filter approach.

Online algorithms offer other key advantages compared to offline methods: (1) the ability to offer immediate *feedback* to the user on the quality of the result [Izadi et al., 2011], (2) the potential to dynamically adapt to transparently *hot-swap* users [Bouaziz et al., 2013], and (3) reduced storage and computational resources, as information is integrated frame-by-frame, in a *streaming* fashion.

4.3 Online model calibration

We now describe our *joint* calibration and tracking algorithm, which combines the Levenberg-style optimization of previous hand trackers with the uncertainty maintenance framework of Kalman filtering. Previous hand tracking work has made use of temporal smoothing priors to propagate *pose* information from previous frames, without the use of filtering. However this approach cannot be used for *shape* because it is so weakly constrained in any given frame, and because its temporal prior is so strong, as shape parameters are *persistent* over time: we observe the same user performing in front of the camera for thousands of frames. However,

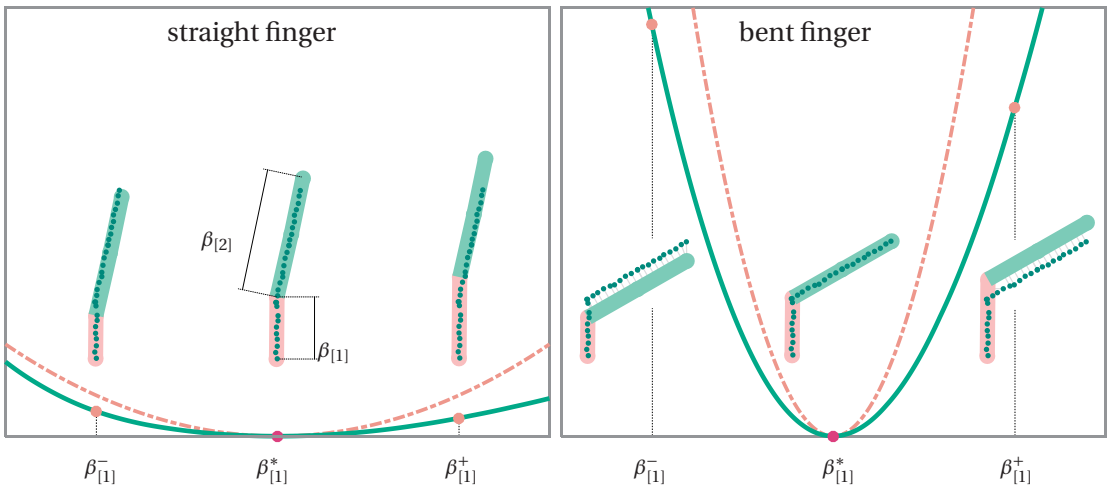


Figure 4.2 – **(Per-frame regression)** We abstract the hand shape/pose estimation problem from a single frame into the one of a simpler 2D stick-figure. Note, however, that this illustration is not hand-crafted, but is derived from numerical optimization executed on these simplified datasets. When the finger is straight (left), it is difficult to estimate the length of individual phalanges as the optimization problem is ill-posed. With a bent finger (right) the problem is better conditioned. We analyze the landscape of the registration energy $E(\beta_{[1]})$, and observe how estimation uncertainty relates to the width of the local minima valley. This uncertainty, the posterior distribution of shape parameters after computing their estimate from the data in the current frame, can be estimated through a quadratic approximation $\tilde{E}(\beta_{[1]})$, derived from the Hessians of the registration energies.

sufficient information to estimate certain shape parameters is simply not available in certain frames. For example, by observing a straight finger like the one in Figure 4.2-(left), it is difficult to estimate the length of a phalanx. Therefore, knowledge must be gathered from a *collection* of frames capturing the user in different poses.

As we illustrate in Figure 4.2 and Figure 4.4, the confidence in regressed *shape* parameters is conditional on the *pose* of the current frame. Rather than manually picking a few frames in different poses as in [Taylor et al., 2016], we show how propagation of not just the shape estimate, but also its uncertainty allows reliable calibration even if the initial poses fail entirely to constrain some shape dimensions. Additionally, the temporal priors of previous work are easily incorporated in the LMKF formulation.

Input data and shape model

The input data are a sequence of depth frames $\mathcal{D}_n$, which are segmented via a wristband [Tagliasacchi et al., 2015] to produce a point cloud $d_n \subset \mathbb{R}^3$. The pose vector in frame n is θ_n, and our shape model $\mathcal{M}(\theta;\beta)$ is the sphere mesh of [Tkach et al., 2016]. Shape is encoded via scalar length parameters β instead of sphere positions; see Figure 4.10 and [Remelli et al., 2017].

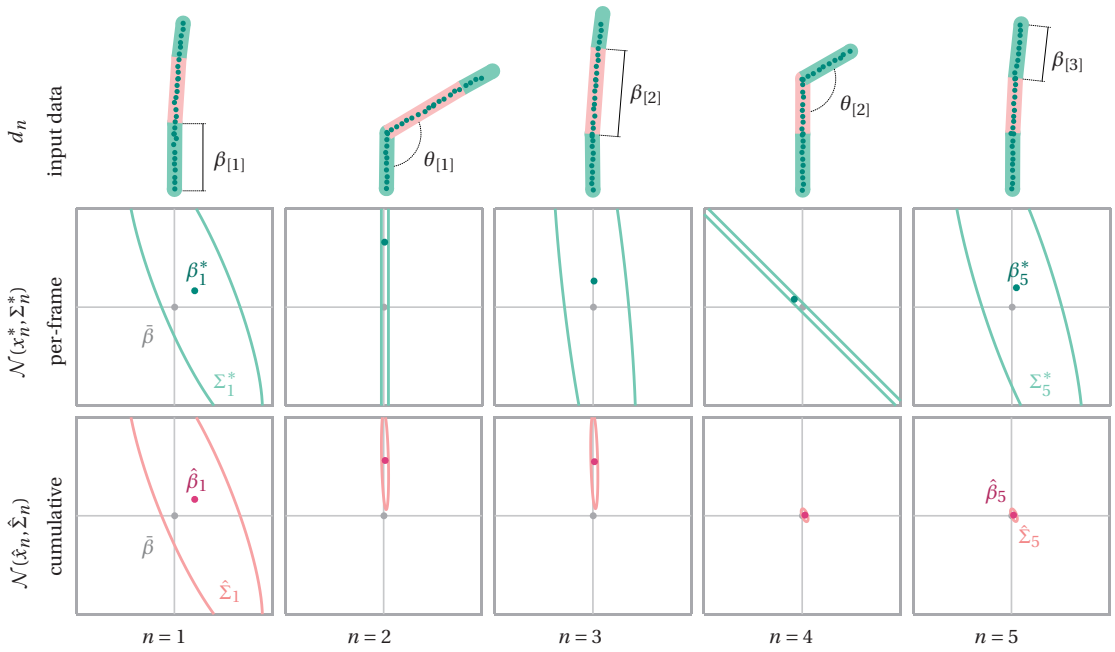


Figure 4.3 – **(Cumulative regression)** We visualize several temporally sorted frames of input data d_n, the uncertainty ellipsoid Σ_n^* estimated by per-frame regression, and the *online* uncertainty estimate $\hat{\Sigma}_n$. For illustration purposes, we only display the two-dimensional ellipsoids representing the covariance of $\beta_{[1]}$ and $\beta_{[2]}$. Although $\Sigma_1^* = \Sigma_5^*$, observe how $\hat{\Sigma}_1 \succ \hat{\Sigma}_5$: in the last frame we have a confident estimate as the information from frames $_{2:4}$ has been integrated. Further, notice how even though the parameter $\beta_{[2]}$ was not observed directly in any of the presented frames, its value was inferred from the highly-certain measurements $(\beta_{[1]})_{n=2}$ and $(\beta_{[1]} + \beta_{[2]})_{n=4}$.

Estimation

Let $x_n = [\theta_n; \beta_n]$ denote the model *state*: the vector of coalesced pose and shape parameters at frame n. Our goal in tracking is to produce the best estimate $\hat{x}_n$, at frame n, of the state x_n, given all the data seen previously, $d_1, ..., d_n$. Additionally, we want to estimate not just the state, but the parameters of the *probability distribution* over the state $p(x_n | d_{1..n})$. Thus, if we write

$$p(x_n | d_{1..n}) \approx \mathcal{N}(x_n \mid \hat{x}_n, \hat{\Sigma}_n), \tag{4.1}$$

we are saying that x_n approximately follows a normal distribution with mean $\hat{x}_n$ and covariance $\hat{\Sigma}_n$. When we display a tracked hand to the user, we will most likely just draw the hand with pose and shape parameters $\hat{x}_n$, which sometimes leads to $\hat{x}_n$ being called "the estimate of x_n", but it is more correctly "the estimate of the mean of the distribution $p(x_n)$", and similarly with $\hat{\Sigma}_n$.

It is generally computationally intractable to estimate the parameters conditioned on all the previous history $d_{1..n}$ at every frame (although in Section 4.3.4 we compute some related

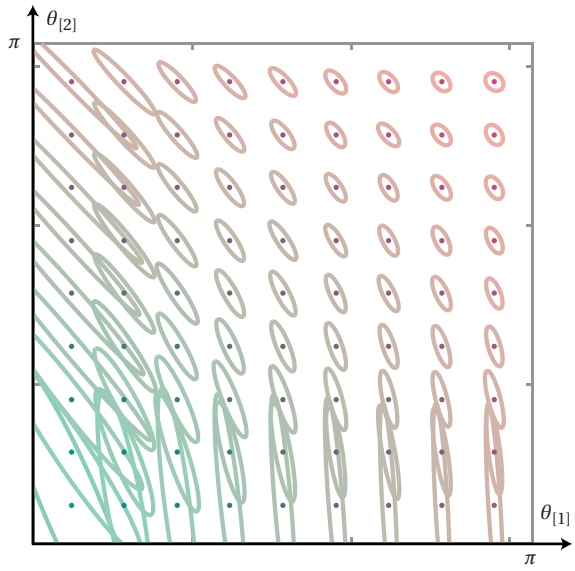


Figure 4.4 – A visualization of the covariance estimate for phalanx lengths $\{\beta_{[1]}, \beta_{[2]}\}$ as we vary phalanx bend angles $\{\theta_{[1]}, \theta_{[2]}\}$. A confident measurement of $\beta_{[1]}$ is only available when $\theta_{[1]}$ is bent, while a confident measurement of $\beta_{[1]} + \beta_{[2]}$ is available when $\theta_{[2]}$ is bent. The covariance ellipsoids are centered at the corresponding $\{\theta_{[1]}, \theta_{[2]}\}$ location.

$$\hat{x}_n = \arg\max_{x_n} \underbrace{\log\left(p(x_n^*|x_n)\, p(x_n|\hat{x}_{n-1})\right)}_{L(x_n)}$$

$$p(x_n^*|x_n) = \exp\left(-\tfrac{1}{2}(x_n^* - x_n)^T \Sigma_n^{*-1}(x_n^* - x_n)\right)$$

$$p(x_n|\hat{x}_{n-1}) = \exp\left(-\tfrac{1}{2}(x_n - \hat{x}_{n-1})^T \hat{\Sigma}_{n-1}^{-1}(x_n - \hat{x}_{n-1})\right)$$

$$\hat{\Sigma}_n^{-1} = \frac{\partial^2 L}{\partial x_n^2}\Big|_{\hat{x}_n} \approx \begin{bmatrix} \sqrt{\Sigma_n^{*-1}} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix}^T \begin{bmatrix} \sqrt{\Sigma_n^{*-1}} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix} = \Sigma_n^{*-1} + \hat{\Sigma}_{n-1}^{-1}$$

Table 1 – *Split* cumulative regression – Kalman Filter (KF)

quantities as a baseline), so the estimation is typically expressed in terms of a *per-frame* term $p(x_n|d_n)$, which describes the components due only to information in frame d_n and *cumulative* term $p(x_n|d_1, ..., d_{n-1})$. Different approximations for this term lead to different methods, denoted *split cumulative* and *joint cumulative* below.

4.3.1 Per-frame estimate – $p(x_n|d_n)$

The distribution $p(x_n|d_n)$ is, by Bayes' rule, proportional to the product of a data term and a prior $p(d_n|x_n)p(x_n)$, which is naturally related to the traditional energy formulations by identifying the negative log likelihood with the energy. Consider the energy:

$$E(x_n) = \sum_{\tau \in \mathcal{T}} E_\tau(d_n, x_n) \tag{4.2}$$

Where the terms $\mathcal{T}$ ensure that:

d2m	data points are explained by the model
m2d	model lies in the sensor visual-hull
smooth	recorded sequence is smooth
pose-prior	calibrated hand pose is likely
shape-prior	calibrated hand shape is likely
pose-valid	semantics: collisions and joint limits
shape-valid	semantics: finger order and connectivity

The energy terms in the objective function are detailed in [Tkach et al., 2016] and [Tagliasacchi et al., 2015], with the exception of *shape-prior* and *shape-valid* that are discussed in Section 4.3.5. Given E as above, we can write

$$p(x_n|d_n) \propto \exp(-E(x_n)), \tag{4.3}$$

but to perform propagation, we will need a more compact form, for example a Gaussian approximation. A natural choice is the *Laplace approximation*: a Gaussian with its mean at the mode of (4.3) (see Appendix 4.6.1) and covariance chosen to match a second-order expansion of E about that mode. The mode computation is the standard energy minimization

$$x_n^* = \underset{x_n}{\arg\min}\, E(x_n) \tag{4.4}$$

which can be solved by nonlinear optimization given an initialization x_n^0 (obtained from a discriminative method or from the solution of the previous time-step), and indeed this is the same minimization performed by current state-of-the-art hand trackers. The covariance matrix Σ_n^* of the Laplace approximation is the inverse of the Hessian of E, and as we are using a Gauss-Newton solver, $E(x)$ is of the form $\|d_n - F(x_n)\|^2$, so we may make the G-N approximation of the Hessian in terms of the Jacobian of $\bar{F}(x_n) = d_n - F(x_n)$, yielding

$$\Sigma_n^* = \left(\frac{\partial \bar{F}(x^*)}{\partial x}^\top \frac{\partial \bar{F}(x^*)}{\partial x}\right)^{-1}. \tag{4.5}$$

Thus, after processing the information in a frame d_n, the sought-after quadratic approximation of posterior distribution of model parameters is

$$\tilde{p}(x_n|d_n) \approx \mathcal{N}\left(x_n^*, \Sigma_n^*\right), \tag{4.6}$$

so the "per-frame" posterior parameters are $\hat{x}_n = x_n^*, \hat{\Sigma}_n = \Sigma_n^*$.

4.3.2 Split cumulative estimate – $p(x_n|d_{1..n})$

The per-frame distribution in Section 4.3.1 encodes the uncertainty in pose and shape solely due to the data in frame n. To aggregate information from previous frames, we would like a

$$\hat{x}_n = \arg\max_{x_n} \underbrace{\log\left(p(d_n|x_n)\, p(x_n|\hat{x}_{n-1})\right)}_{L(x_n)}$$

$$p(d_n|x_n) = \exp\left(-\tfrac{1}{2}(d_n - F(x_n))^T (d_n - F(x_n))\right)$$

$$p(x_n|\hat{x}_{n-1}) = \exp\left(-\tfrac{1}{2}(x_n - \hat{x}_{n-1})^T \hat{\Sigma}_{n-1}^{-1} (x_n - \hat{x}_{n-1})\right)$$

$$\hat{\Sigma}_n^{-1} = \frac{\partial^2 L}{\partial x_n^2}\Big|_{\hat{x}_n} \approx \begin{bmatrix} -\frac{\partial F(\hat{x}_n)}{\partial x_n} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix}^T \begin{bmatrix} -\frac{\partial F(\hat{x}_n)}{\partial x_n} \\ \sqrt{\hat{\Sigma}_{n-1}^{-1}} \end{bmatrix} = \Sigma_n^{-1} + \hat{\Sigma}_{n-1}^{-1}$$

Table 2 – *Joint* cumulative regression – Iterated Extended KF (IEKF)

Figure 4.5 – We evaluate our real-time calibration framework on twelve different subjects. For each user we show a frame in a (more-or-less) rest pose configuration, as well as a different pose selected from the recorded sequence. These results are better appreciated by watching our Video3 [04:03].

simple form of the distribution $p(x_n|d_{1..n})$, for example a Gaussian:

$$p(x_n|d_{1..n}) \approx \mathcal{N}(\hat{x}_n, \hat{\Sigma}_n) \tag{4.7}$$

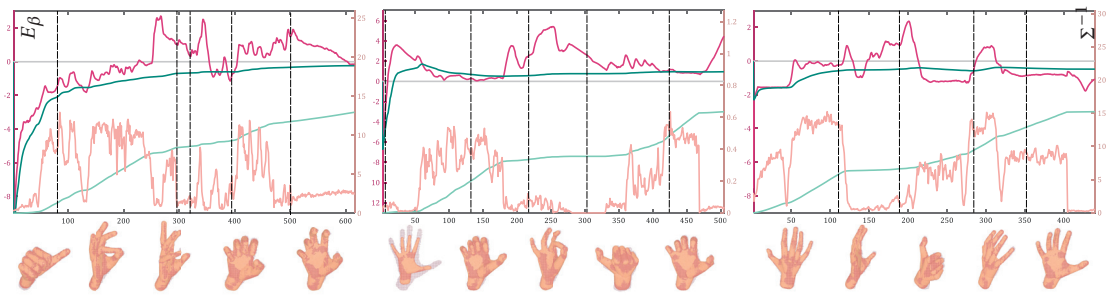
Then, given values of the parameters $\hat{x}_{n-1}, \hat{\Sigma}_{n-1}$ at the previous timestep, we must update them to incorporate the new information in frame n. This leads to the following pair of inductive update equations:

$$\mathcal{N}(x_n|\hat{x}_1, \hat{\Sigma}_1) = \mathcal{N}(x_n|x_1^*, \Sigma_1^*) \tag{4.8}$$

$$\mathcal{N}(x_n|\hat{x}_n, \hat{\Sigma}_n) = \mathcal{N}(x_n|\hat{x}_{n-1}, \hat{\Sigma}_{n-1})\mathcal{N}(x_n|x_n^*, \Sigma_n^*) \tag{4.9}$$

By applying the product of Gaussians rule [Petersen et al., 2008], we obtain update equations for $\hat{x}_n$ and $\hat{\Sigma}_n$:

$$\begin{aligned} \hat{x}_n &= \Sigma_n^*(\hat{\Sigma}_{n-1} + \Sigma_n^*)^{-1}\hat{x}_{n-1} + \hat{\Sigma}_{n-1}(\hat{\Sigma}_{n-1} + \Sigma_n^*)^{-1}x_n^* \\ \hat{\Sigma}_n &= \hat{\Sigma}_{n-1}(\hat{\Sigma}_{n-1} + \Sigma_n^*)^{-1}\Sigma_n^* = \left(\Sigma_n^{*-1} + \hat{\Sigma}_{n-1}^{-1}\right)^{-1} \end{aligned} \tag{4.10}$$



ring attachment-y – $\beta_{[25]}$ middle middle-length – $\beta_{[7]}$ middle attachment-z – $\beta_{[23]}$

Figure 4.6 – We illustrate the formulation in Section 4.3 on the calibration of four different degrees-of-freedom on the `Handy/Teaser` dataset. (top) Frames are indexed by n and we display: ground-truth value $\bar{\beta}$, per-frame estimate β_n^* and cumulative estimate $\hat{\beta}_n$; the scale for these quantities is relative to $\bar{\beta}$, and shown on left-hand side of each axis. In the same plot, we also visualize the *inverse* of the diagonal entry of the covariance matrices (i.e. certainty) for per-frame Σ_n^* and cumulative $\hat{\Sigma}_n$ estimates; the scale for these quantities is on the right-hand side of each plot. (bottom) We also display the pose corresponding to a selection of frames in the sequence (dashed); please note the correlation between pose and uncertainty.

In Appendix 4.6.2, we shown how Equation 4.10 is equivalent to the Kalman Filter (KF) update equations in Table 1, with measurement x_n^*, and measurement noise covariance Σ_n^*. This optimization, which we refer to as *split cumulative* is arguably the simplest way of achieving an online parameter regression: by treating the results of the per-frame solve $\mathcal{N}(x_n^*, \Sigma_n^*)$ as the measurements in a KF.

4.3.3 Joint cumulative estimate – $p(x_n|d_{1..n})$

The optimization in Table 1 does not provide any information about the current estimate of the parameters $\hat{x}_n$ to the independent solve described in Section 4.3.1. This could be problematic, as in this case Equation 4.2 does not leverage any temporal information aside from initialization, while relying on a sufficiently good initialization to compute $\mathcal{N}(x_n^*, \Sigma_n^*)$. We propose to coalesce the cumulative and per-frame optimization resulting in the *joint* cumulative regression scheme in Table 2. The optimization in Table 2 can be expressed in least-squares form, and embedded in Equation 4.2 through the term:

$$E_{\text{iekf}} = \|\hat{\Sigma}_{n-1}^{-1/2}(x_n - \hat{x}_{n-1})\|_2^2 \tag{4.11}$$

In Appendix 4.6.3 we link this update to LM [Skoglund et al., 2015], demonstrating that optimizing the objective in Table 2 with a Levenberg Marquardt method is equivalent to an Iterated Extended Kalman Filter (IEKF) with measurement update d_n. In a practical setting, this observation creates a very simple way to encode IEKF-like behavior within existing LM optimization codebases.

4.3.4 Joint multiframe (batch/offline) estimate

While we focus on an online/streaming algorithm, we also describe an offline baseline calibration procedure – inspired by the work of [Taylor et al., 2014] – where multiple frames in the input sequence are simultaneously considered.

Offline-Hard

To achieve this, Equation 4.2 is modified to consider N frames, each with its own pose parameters θ_n, but with the same underlying shape β, resulting in what we refer to as *Offline-Hard* calibration:

$$\underset{\beta,\{\theta_n\}}{\arg\min} \sum_{n=1}^{N} \sum_{\tau \in \mathcal{T}} E_\tau(d_n, [\theta_n, \beta]) \tag{4.12}$$

Such optimization is initialized with a single β^0, and in our experiments, we noticed how this resulted in reduced convergence performance and a propensity for the optimization to fall into local minima.

Offline-Soft

Therefore, we introduce the *Offline-Soft* calibration, where the constraint that a single β should be optimized is enforced through a soft penalty:

$$\underset{\beta,\{\theta_n\}}{\arg\min} \sum_{n=1}^{N} \sum_{\tau \in \mathcal{T}} E_\tau(d_n, [\theta_n, \beta_n]) + \omega_\beta \sum_{n=1}^{N} \|\beta_n - \beta\|^2 \tag{4.13}$$

The initializations β_n^0 are derived from the per-frame optimization of Equation 4.2, while the penalty weight is set to a large value ($\omega_\beta = 10e4$). The advantage of Offline-Soft over Offline-Hard can be clearly observed in Figure 4.8, where the former achieves a performance comparable to the one of the (overfitting) per-frame optimization. Finally, note that in practice we do not consider every frame as this large problem would not fit into memory, but instead we sub-sample at a $\approx 1/20$ rate, the same subsampling is used for Kalman *measurement* updates in our online solution to avoid a bias in the comparisons.

4.3.5 Shape regularizers

Hand shape variation can be explained in a low dimensional space whose fundamental degrees of freedom include variation like uniform scale, palm width, and finger thickness [Khamis et al., 2015]. We follow the ideas presented in [Remelli et al., 2017], and build a latent-space encoding hand shape variability through *anisotropic scaling*. By setting $\omega_{\text{shape-space}} \ll 1$, this prior acts as a soft regularizer and does not prevent the algorithm from computing a tight fit:

$$E_{\text{shape-space}} = \|\beta - (\bar{\beta} \circ \mathcal{I}\tilde{\beta})\|^2 \tag{4.14}$$

where $\tilde{\beta} \in \mathbb{R}^3$ is a latent vector encoding relative changes in hand *height, width* and sphere *thickness* with respect to the default template $\bar{\beta}$, while $\mathcal{I}$ is a matrix mapping latent DOFs to the corresponding full-dimensional DOFs $\beta_{[i]}$; see Figure 4.10. Since our shape-prior has a small weight, unfeasible hand-shape configurations are still possible, such as a finger floating in mid-air, or when the natural order of fingers {index, middle, ring, pinky} has been compromised. We overcome this problem by a set of quadratic *barrier* constraints that are conditionally enabled in the optimization when unfeasible configurations are detected (encoded via $\chi_c(\beta) \in \{0,1\}$):

$$E_{\text{shape-valid}} = \sum_{c=1}^{C} \chi_c(\beta) \| \langle \beta, \kappa \rangle \|_2^2 \tag{4.15}$$

For example to avoid middle and index fingers from permuting, one such constraint is written in the following form, and $\chi_0(\beta) = 1$ only when an invalid configuration is detected:

$$\chi_0(\beta) \| \beta_{\text{idx-base-x}} - \beta_{\text{idx-base-rad}} - \beta_{\text{mid-base-x}} + \beta_{\text{mid-base-rad}} \|_2^2$$

4.4 Evaluation

To evaluate the technical validity of our approach we verify its effectiveness by applying it to a new dataset acquired through commodity depth cameras (Section 4.4.1); corroborate the formulation of our optimization on a synthetic 3D dataset (Section 4.4.2); analyze its robustness through randomly perturbing the algorithm initialization (Section 4.4.3); and attest how our method achieves state-of-the-art performance on publicly available datasets (Section 4.4.4 and Section 4.4.5).

4.4.1 Calibration dataset: `Handy/GuessWho?` – Figure 4.5

We stress-tested our system by qualitatively evaluating our calibration technique with data acquired from *twelve* different users performing in front of an Intel RealSense SR300 camera (a consumer-level time-of-flight depth sensor). Snapshots of the twelve calibration sequences are reported in Figure 4.5. While ground truth information is not available, these datasets will enable comparisons to our method through the use of empirical metrics; e.g. E_{d2m} and E_{m2d} [Tkach et al., 2016], or the golden-energy [Taylor et al., 2016].

4.4.2 Synthetic dataset: formulation analysis – Figure 4.6

For synthetic data the ground truth shape parameters $\bar{\beta}$ are readily available, and the sphere-mesh model $\mathcal{M}(\theta, \beta)$ is animated in time with the $\bar{\theta}_n$ parameters of the complex motions in the `Handy` dataset [Tkach et al., 2016].

The following metric measures average ground-truth residuals (in millimeters):

$$E_\beta = \frac{1}{|M|} \sum_{m \in M} \left| \beta_{[m]} - \bar{\beta}_{[m]} \right| \tag{4.16}$$

For ground-truth comparisons, analogously to [Taylor et al., 2016], we selected a subset of shape parameters in Figure 4.10: $M = \{0{:}16, 19, 22, 25, 28, 45{:}74\}$. This is necessary as sphere-centres on the palm can move without affecting the tracking energy – a null-space of our optimization. The tracking algorithm is initialized in the first frame by $\bar{\theta}_0$. In Figure 4.6, we report an experiment analogous to that of Figure 4.3 but on a full 3D sequence. Consider Figure 4.6b, where we report the runtime estimates for the length of the middle-finger's middle-phalanx; the subscript [7] will henceforth be implied. Congruously to the discussion in Section 4.3, the phalanx length estimates computed in frames where the finger is bent are given a large weight. Per-frame estimates β_n^* can often oscillate away from the ground truth, but these incorrect estimates are associated with a small weight. Our algorithm estimates these per-frame uncertainties Σ_n^*, and accordingly updates the online cumulative estimates $\hat{\beta}_n$ and $\hat{\Sigma}_n$.

4.4.3 Synthetic dataset: robustness – Figure 4.7

We evaluate the *robustness* of the algorithm by analyzing its convergence properties as we vary the magnitude of perturbations. We provide two experiments: *synthetic* and *real*. The real dataset consists of depth maps $\{\mathcal{D}_n\}$ measured by an Intel Realsense SR300 sensor, where we track motion with the multi-view stereo (MVS) calibrated model from [Tkach et al., 2016] to estimate a sequence of pose parameters $\{\theta_n\}$; the shape parameters $\bar{\beta}$ of this user are known with good confidence thanks to the MVS data. In the synthetic dataset, depth images $\{\mathcal{D}_n\}$ are generated by animating the sphere-mesh model $\mathcal{M}(\theta_n, \bar{\beta})$ and then rasterizing the model as in Section 4.4.2. To achieve this, random initializations for the user-personalized models are drawn from the Gaussian distribution $\mathcal{N}(\bar{\beta}, \sigma)$. A few examples of such perturbations for $\sigma = .4$ are visualized in Figure 4.7. In our experiments, we draw fifteen random samples per each value of σ, and compute mean and standard deviation of the measured ground truth residuals E_β. As each sample requires the re-tracking of the entire sequence (≈ 20 seconds) with a new initialization, the two plots in Figure 4.7 amount to roughly four hours of footage. For this reason, in Video3 [03:16] [1] we only display a few examples of calibration and apply a random perturbation every few seconds. Notice that although we still have a non-zero average residual of $\approx$ 1mm, the video shows how the model is an excellent fit to the synthetic data. In both experiments, Offline-Hard performs worse than Offline-Soft for the reasons discussed in Section 4.3.4. With the large ($\sigma = .4$) perturbations Offline-Soft still had some troubles converging to the correct results, as per-frame pose initializations were too significantly wrong; in this regard, we believe discriminative pose re-initializers such as [Oberweger et al., 2015a] could be helpful to increase the performance of both offline calibration algorithms.

[1] Please find the accompanying Video3 at http://lgg.epfl.ch/publications/2017/HOnline/video.mp4.

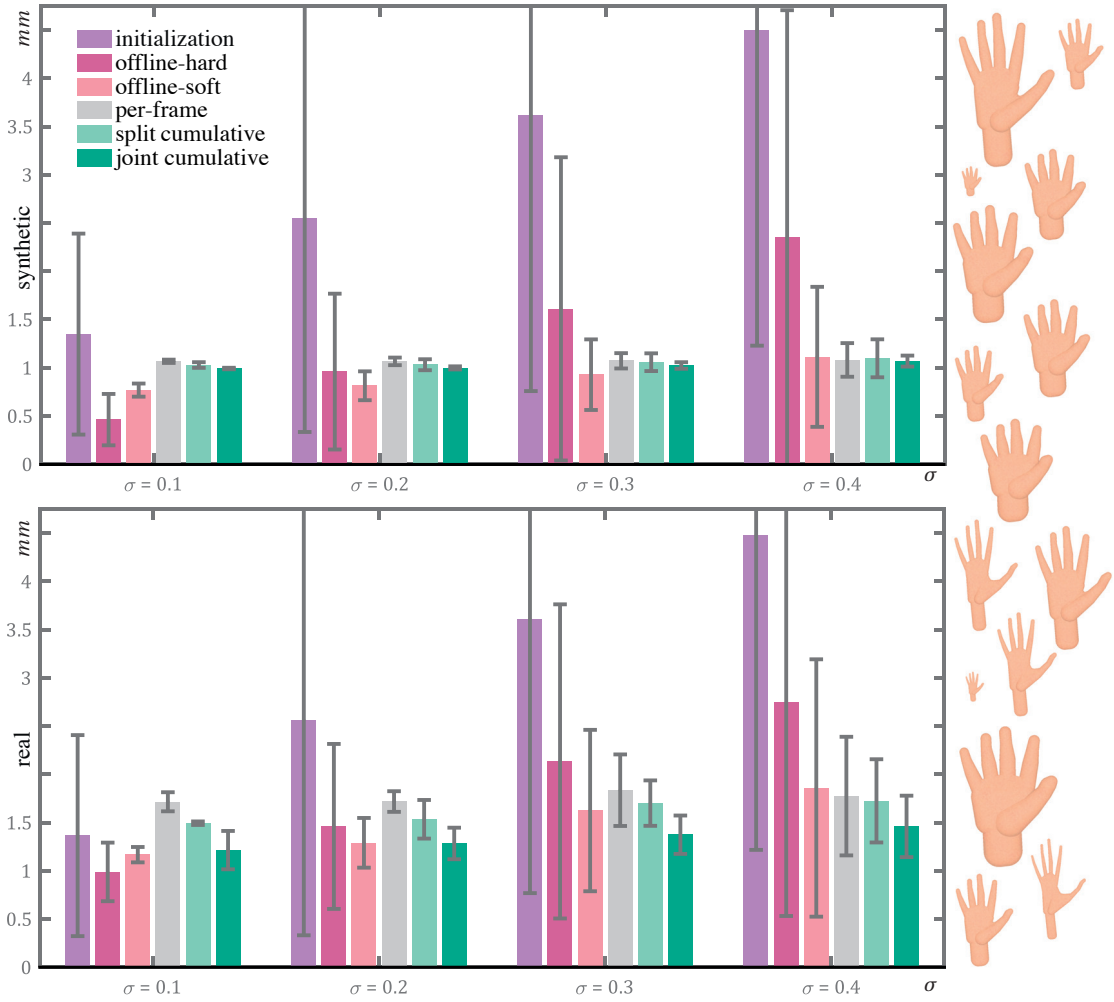


Figure 4.7 – Mean and standard deviation for ground-truth calibration residuals as we vary the algorithm's initialization with a random perturbation of standard deviation σ. We evaluate the residuals on (top) synthetic depth maps, as well as (bottom) on the raw depth maps. (right) the exemplars drawn from the $\sigma = 0.4$ perturbation used in the evaluation.

Technically we could not display the per-frame algorithm performance in Figure 4.7, since it does not provide a single final estimate of shape parameters. To do this, we employ the model parameters it estimated in the last frame of each sequence. In the last frame the error is low, as each frame is initialized with the values from the previous frame; see Video3 [03:41]. Note how per-frame calibration performs excellently, even outperforming Offline-Hard. This is because, thanks to our carefully designed shape priors, per-frame calibration is quite robust; see Video3 [03:41]. This is essential in cumulative calibration, as the true value of a parameter can be recovered only if accurate measurements are available in at least some poses. The per-frame algorithm should also not be mistaken for tracking algorithm (where shape parameters are fixed) which is twice more efficient (calibration executes at 30 Hz, while tracking executes at 60 Hz) and, in general, much more robust.

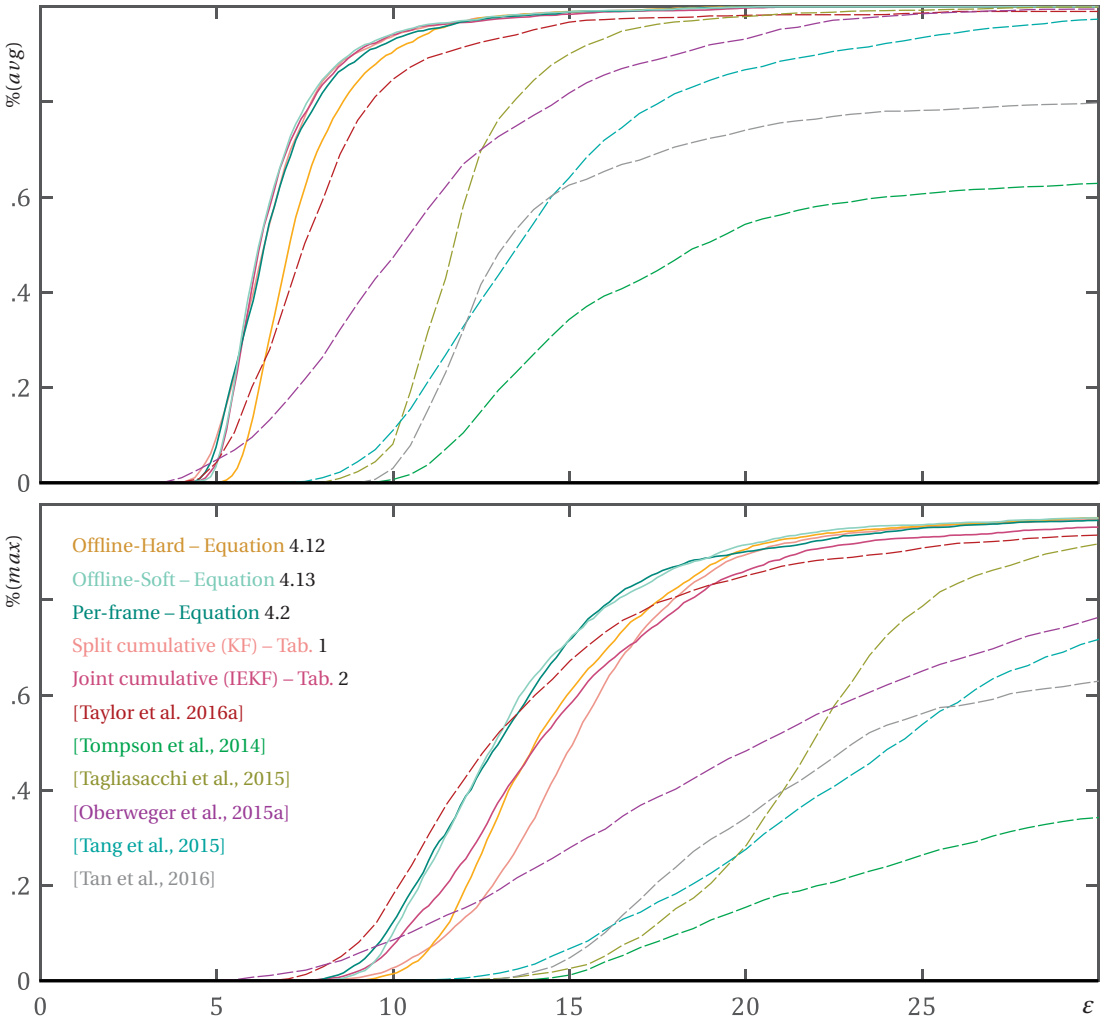


Figure 4.8 – Evaluation on the NYU dataset from [Tompson et al., 2014] reporting the percentage of frames with average (top) and max (bottom) ground-truth marker-error less than ε.

It is difficult to differentiate the split vs. joint cumulative variants in the synthetic dataset, as calibration converges very effectively when it can rely on precise measurements. Overall, on the sensed dataset our *joint cumulative* calibration performs the best. Our split variant performs very well when per-frame consistently provides an accurate solution (e.g. on the synthetic sequences). Nonetheless, we noticed that how with more challenging motions, the joint-cumulative can aid the per-frame solver by providing a temporal regularization. This is beneficial when dealing with an uncooperative user, or to perform calibrations in sequences that were not specifically designed for this task (e.g. fast motion and long-term occlusions).

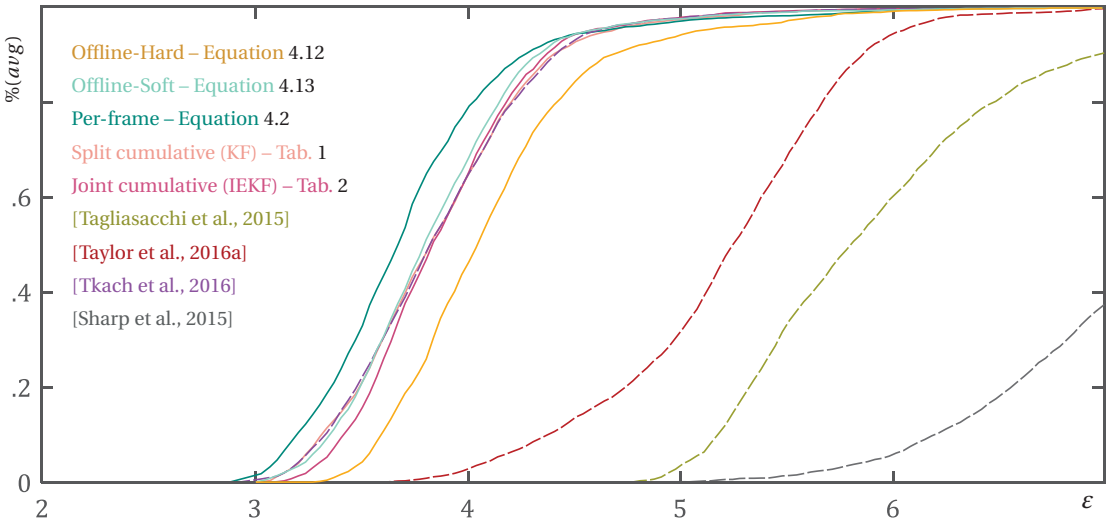


Figure 4.9 – Evaluation on the `Handy/Teaser` dataset, reporting the percentage of frames with an average E_{d2m} data-to-model energy below ε.

4.4.4 Marker-based evaluation on `NYU` dataset – Figure 4.8

Although several marker-based datasets are available, such as [Qian et al., 2014], [Sharp et al., 2015] and [Yuan et al., 2017], state-of-the-art *generative* methods have focused on the `NYU` [Tompson et al., 2014] and `Handy` [Tkach et al., 2016] datasets for quantitative evaluation. On the `NYU` dataset, to properly compare to [Taylor et al., 2016], we evaluate the metrics on the first 2440 frames (user #1), and consider only markers on finger joints. This dataset allow us to compare our method (and its variants) to a number of other algorithms including: the PSO tracker by [Sharp et al., 2015], the calibration methods by [Khamis et al., 2015] and [Tan et al., 2016], the subdivision tracker of [Taylor et al., 2016], the cylindroid tracker by [Tagliasacchi et al., 2015], the sphere-mesh tracker by [Tkach et al., 2016], the Gaussian tracker of [Sridhar et al., 2015], and discriminative methods such as those of [Tompson et al., 2014], [Tang et al., 2015] and [Oberweger et al., 2015a]. Our online algorithm achieves very competitive tracking performance while being the *first* capable of calibrating the user-personalized tracking model *online*, rather than in an offline calibration session like [Taylor et al., 2016]. Notice how the best performance is achieved by either: (1) the per-frame optimization, where per-frame *overfitting* takes place, or (2) by offline calibration techniques such as Offline-Soft or [Taylor et al., 2016]. This is expected, as offline algorithms jointly consider all available information, while online/streaming algorithms can only integrate information one frame at a time.

4.4.5 Dense evaluation on the `Handy` dataset – Figure 4.9

Another way to evaluate the quality of tracking/calibration is to compare the depth map $\mathcal{D}_n$ (i.e. sensor point cloud) to the tracking model depth map $\bar{\mathcal{D}}(\theta,\beta)$ (i.e. model point cloud); see [Tkach et al., 2016]. The model depth map is obtained by rendering $\mathcal{M}(\theta,\beta)$ with the

same intrinsic/extrinsic parameters of the sensor. The following metric measures the average magnitude of data-to-model ICP correspondences:

$$E^n_{\text{d2m}} = \frac{1}{|d_n|} \sum_{\mathbf{p}_j \in d_n} \|\mathbf{p}_j - \Pi_{\bar{\mathcal{D}}(\theta,\beta)}(\mathbf{p}_j)\|_2 \tag{4.17}$$

where Π is an operator computing the closest-point projection of points in the sensor's point cloud, onto the point-cloud associated with the synthetic depth-map $\mathcal{D}(\theta,\beta)$. This metric is *dense*, as it computes residual of an entire geometry model rather than just a sparse set of markers. If $E_{\text{d2m}} \approx 0$ (up to sensor noise) in every frame, then the personalized model is a seemingly perfect *dynamic replica* of the user's hand. The `Handy` dataset from [Tkach et al., 2016] enables these type of comparisons and includes rendered depth maps for [Tagliasacchi et al., 2015], [Sharp et al., 2015], as well as the state-of-the-art method of [Taylor et al., 2016]. Further, note how this dataset considers a range of motion substantially more complex than the one in the `NYU` dataset. Like in earlier comparisons, the per-frame technique performs best as it overfits to the data, by generating a collection of β_n instead of a single tuple β. Our techniques calibrate a model with performance comparable to that of [Tkach et al., 2016], where a high-quality MVS point cloud with manual annotations was used for calibration.

4.5 Conclusions

From an application point of view, our approach significantly improves on the usability of real-time hand tracking, as it requires neither controlled calibration scans nor offline processing prior to tracking. This allows easy deployment in consumer-level applications. From a technical point of view, we introduce a principled approach to online integration of shape information of user-specific hand geometry. By leveraging uncertainty estimates derived from the optimization objective function, we automatically determine how informative each input frame is for improving the estimates of the different unknown model parameters. Our approach is general and can be applied to different types of calibration, e.g., for full body tracking. More broadly, we envisage applications to other difficult types of model estimation problems, where unreliable data needs to be accumulated and integrated into a consistent representation.

Limitations and future works

The intrinsic limitation of our online approach as well its offline counterparts is reliance on reasonable tracking quality during calibration. If tracking fails, the model quality is compromised as shown in the Video3 [07:18]. Currently, our optimization relies on heavy parallelization and high-end GPU hardware – we use a 4GHz i7 equipped with an NVIDIA GTX 1080Ti. In future we want to reduce computational overhead to facilitate deployment on mobile devices.

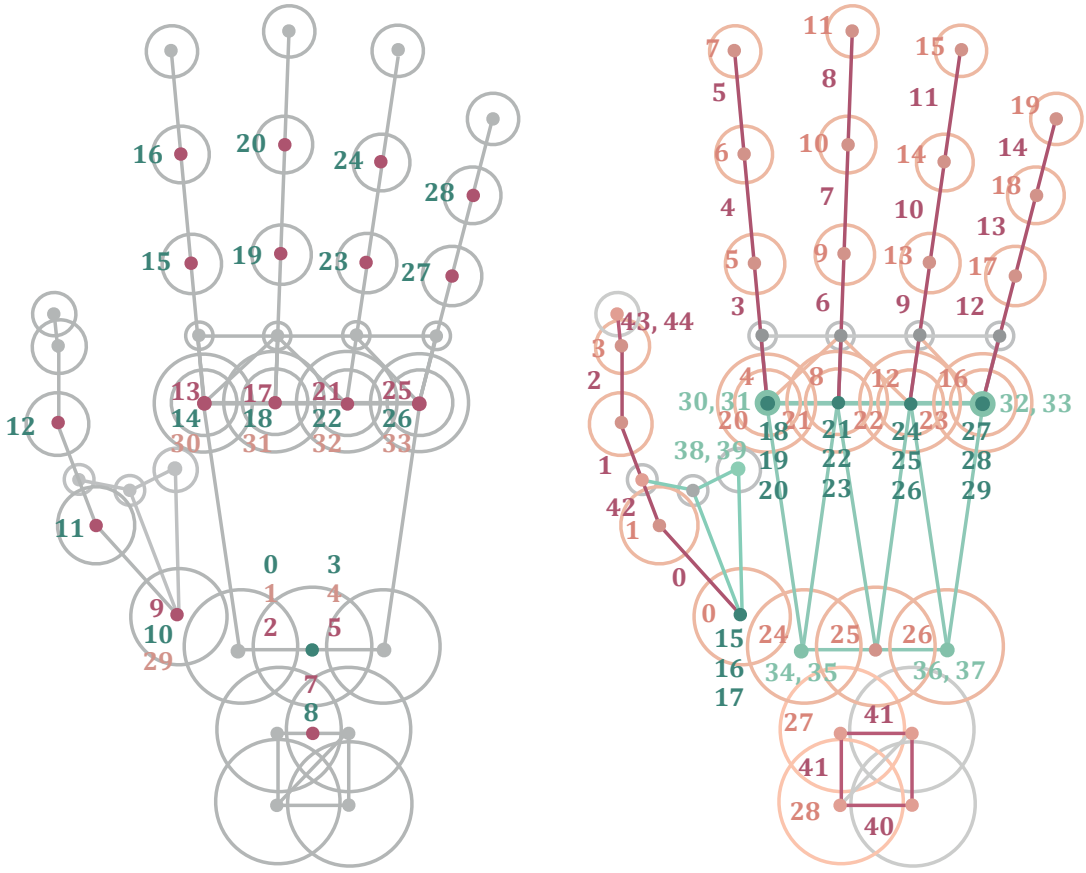


Figure 4.10 – The degrees of freedom of our optimization, where use a cartesian right-handed coordinate frame for translational DOFs. For pose parameters, global translation is represented by $\theta_i || i \in [0,1,2]$ and rotation by $\theta_i || i \in [3,4,5]$. We then color code DOFs according to whether they represent flexion, twist, and abduction. For shape parameters, we color code DOFs for lengths, radii, 3DOFs vertices (x,y,z), 2DOFs vertices (x,y), and passive DOFs (linearly dependent).

4.6 Implementation Details

Kalman Filter (KF)

Following the notation in [Welch and Bishop, 1995], let us denote the latent state of a discrete-time controlled process as $x_n \in \mathbb{R}^N$, a generic measurement as $z_n \in \mathbb{R}^M$ and let us consider the following linear stochastic difference equations

$$x_n = Ax_{n-1} + w_{n-1} \tag{4.18}$$

$$z_n = Jx_n + v_n \tag{4.19}$$

Time	Measurement
$\hat{x}_n^0 = \hat{x}_{n-1}$	$K_n = P_n^0 J^T (J P_n^0 J^T + R)^{-1}$
$P_n^0 = P_{n-1} + Q$	$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - J\hat{x}_n^0)$
	$P_n = (I - K_n J) P_n^0$

Table 3 – Kalman Filter update equations (with $A = I$).

where w is a normally distributed process noise $p(w) \sim \mathcal{N}(0, Q)$, and v is a normally distributed measurement noise $p(v) \sim \mathcal{N}(0, R)$. The matrix A provides a linear estimate for state updates, while J maps the state x_n to the measurement z_n. Given a generic frame n, let us define an initial (*a priori*) state estimate $\hat{x}_n^0$, together with an improved (*a posteriori*) state estimate $\hat{x}_n$ accounting for the measurement z_n. We can then define *a priori* and *a posteriori* estimate error covariances as

$$P_n^0 = \mathbb{E}[(x_n - \hat{x}_n^0)^T (x_n - \hat{x}_n^0)] \tag{4.20}$$

$$P_n = \mathbb{E}[(x_n - \hat{x}_n)^T (x_n - \hat{x}_n)]. \tag{4.21}$$

The Kalman Filter (KF) estimates the latent state x_n of a discrete control linear process by minimizing the *a posteriori* error covariance. In particular it estimates the process through a *predictor-corrector* approach: given a generic time n the filter first estimates the process state (*time update equation*) and then obtains feedback in the form of noisy measurements (*measurement update equation*). Let us now particularize the system above to our framework, where the latent state of our system corresponds to hand parameters and the measurement corresponds to the solution of Equation 4.2. An estimate of the current hand parameters is given by the one of the previous time-step up to Gaussian noise, that is $x_n = x_{n-1} + w_{n-1}$, while the noisy measurement corresponds to the state itself, meaning that $J = I$ (note that in order to highlight the similarities to other Kalman filter formulations we will maintain the notation J). Our discrete-time process can simply be written in the following form, resulting in the equations of Table 3; see [Welch and Bishop, 1995]:

$$x_n = x_{n-1} + w_{n-1} \tag{4.22}$$

$$z_n = J x_n + v_n \tag{4.23}$$

Time	Measurement
$\hat{x}_n^0 = \hat{x}_{n-1}$	$K_n = P_n^0 J_n^T (J_n P_n^0 J_n^T + R)^{-1}$
$P_n^0 = P_{n-1} + Q$	$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - F_n)$
	$P_n = (I - K_n J_n) P_n^0$

Table 4 – Extended Kalman Filter update equations (with linear $\tilde{F}$).

Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) extends the KF to the case in which the process to be estimated and/or the measurement relationship to the process are not linear:

$$x_n = \tilde{F}(x_{n-1}, w_{n-1}) \tag{4.24}$$

$$z_n = F(x_n, v_n) \tag{4.25}$$

where $\tilde{F}$ relates the current latent state x_n to the previous time step one x_{n-1} and F relates the current latent state x_n to measurement z_n. The EKF simply estimates the latent state of such system by means of linearization of process and measurement equations around the current estimate; see [Welch and Bishop, 1995] for a detailed overview. We can apply this framework to ours and, differently from the linear case, consider now the input depth map d_n as system measurement. The function $F(\cdot)$ therefore maps state x_n to measurement z_n by *applying* shape and pose parameters to the template hand model and computing the closest model points to sensor data points, while as discussed in the previous section $\tilde{F}(\cdot)$ is a simple identity mapping. We can write the non-linear process and measurement equations associated to our framework as:

$$x_n = x_{n-1} + w_{n-1} \tag{4.26}$$

$$z_n = F(x_n) + v_n \tag{4.27}$$

By defining $F_n = F(\hat{x}_n^0)$ and $J_{n[i,j]} = \partial F_{[i]} / \partial x_{[j]}(\hat{x}_n^0)$, the EKF update equations can be written as reported in Table 4; see [Welch and Bishop, 1995].

Iterated Extended Kalman Filter (IEKF)

The EKF performs well for systems with mildly nonlinear measurement functions, but if the measurement equation is strongly nonlinear the performance of the filter deteriorates; see [Havlík and Straka, 2015]. To address this problem, we can perform measurement updates in several steps, where in each one we linearize the measurement function F around the updated value iteration $\hat{x}_n^i$, leading to the Iterated Extended Kalman Filter (IEKF) formulation [Havlík and Straka, 2015]. The time update equation for IEKF is analogous to the one in Table 4, while the measurement update is reported in Table 5.

Extended Information Filters (EIF)

In order to ease the derivations of the upcoming section let us observe that the EKF measurement updates can also be rewritten in the equivalent *Extended Information Filter* form [Anderson and Moore, 1979]; see Table 6. We introduce this formulation in order to ease the upcoming derivations. Note that in order to do that we need to assume the measurement noise to be independent and identically distributed (i.i.d.) across samples, therefore $R = rI$ where $r \in \mathbb{R}^+$ and I is the identity matrix. Further, similarly to EKF, we can write the iterated version of an EIF, as reported in Table 7.

4.6.1 Laplace Approximation

To derive our uncertainties, we start by converting the data terms *d2m* and *m2d* of Equation 4.2 into probabilistic form:

$$p(d_n|x_n) = \exp\left(-\tfrac{1}{2}(d_n - F(x_n))^T(d_n - F(x_n))\right) \tag{4.28}$$

By temporarily omitting the frame index n for conveniency, our problem is rewritten as a maximum likelihood optimization:

$$x^* = \arg\max_x \log p(d|x) = \arg\max_x L(x) \tag{4.29}$$

We now perform a second-order Taylor expansion of the log-likelihood of the data $L(x)$ around the *optimal* solution x^*:

$$L(x) \approx \tilde{L}(x) = L(x^*) + \tfrac{\partial L(x^*)}{\partial x}\Delta x + \tfrac{1}{2}\Delta x^T \tfrac{\partial^2 L(x^*)}{\partial x^2}\Delta x + \text{h.o.t.} \tag{4.30}$$

where $\Delta x = x - x^*$, and let $\partial f(x^*)/\partial x$ indicate the partial derivative of $f(x)$ evaluated at x^*. We rewrite $\bar{F}(x_n) = d_n - F(x_n)$ for brevity. Note how the Jacobian and the Hessian are respectively

$$\begin{aligned}
&\text{for} \quad i = 1 \ldots i_{\max} \\
&\qquad F_n^i = F(\hat{x}_n^i) \Rightarrow J_{n[u,v]}^i = \partial F_{[u]}^i(\hat{x}_n^i)/\partial x_{[v]} \\
&\qquad K_n^i = P_n^0 J_n^{iT}(J_n^i P_n^0 J_n^{iT} + R)^{-1} \\
&\qquad \hat{x}_n^{i+1} = \hat{x}_n^0 + K_n^i(z_n - F_n^i - J_n^i(\hat{x}_n^0 - \hat{x}_n^i)) \\
&\text{end} \\
&\hat{x}_n = \hat{x}_n^i \\
&P_n = (I - K_n J_n^i) P_n^0
\end{aligned}$$

Table 5 – Iterated EKF measurement update equations.

Ex. Kalman Filter	Ex. Information Filter
	$H_n = (P_n)^{-1}$
$K_n = P_n^0 J_n^T (J_n P_n^0 J_n^T + R)^{-1}$	$H_n = H_n^0 + J_n^T R^{-1} J_n$
$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - F_n)$	$K_n = H_n{}^{-1} J_n^T R^{-1}$
$P_n = (I - K_n J_n) P_n^0$	$\hat{x}_n = \hat{x}_n^0 + K_n(z_n - F_n)$

Table 6 – Analogy of EKF and EIF.

for $i = 1...i_{\max}$

$$H_n^i = \tfrac{1}{r}(r H_n^0 + J_n^{iT} J_n^i)$$

$$K_n^i = \tfrac{r}{r} H_n^{i\,-1} J_n^{iT}$$

$$\hat{x}_n^{i+1} = \hat{x}_n^0 + K_n^i(z_n - F_n^i - J_n^i(\hat{x}_n^0 - \hat{x}_n^i))$$

end

$$\hat{x}_n = \hat{x}_n^i$$

Table 7 – Iterated EIF update equations.

zero and positive definite at our optimal point x^* (see [Nocedal and Wright, 2006, Sec. 10.2]):

$$\frac{\partial L(x^*)}{\partial x} = -\bar{F}(x^*)^T \frac{\partial \bar{F}(x^*)}{\partial x} = 0 \tag{4.31}$$

$$\frac{\partial^2 L(x^*)}{\partial x^2} \approx -\frac{\partial \bar{F}(x^*)}{\partial x}^T \frac{\partial \bar{F}(x^*)}{\partial x} \triangleq -\Sigma^{*\,-1} \prec 0 \tag{4.32}$$

From Equation 4.30, using $\tilde{p}(d|x) = \exp(\tilde{L}(x))$, we can then derive the *approximated* posterior distribution:

$$\tilde{p}(d|x) = \exp\left(-\tfrac{1}{2}(x - x^*)^T \Sigma^{*\,-1}(x - x^*)\right) = \mathcal{N}\left(x^*, \Sigma^*\right) \tag{4.33}$$

4.6.2 Derivation for Section 4.3.2

Let us consider the Kalman Filter measurement update equations introduced in Table 4, recalling to the reader that we are considering the case in which the measurement $z_n = x_n^*$ is in the same space of the estimated state $\hat{x}_n$, thus when J is the identity matrix.

$$\begin{aligned}
\hat{x}_n &= \hat{x}_n^0 + P_n^0(P_n^0 + R)^{-1}(z_n - \hat{x}_n^0) \\
&= (P_n^0 + R)(P_n^0 + R)^{-1}\hat{x}_n^0 + P_n^0(P_n^0 + R)^{-1}(z_n - \hat{x}_n^0) \\
&= R(P_n^0 + R)^{-1}\hat{x}_n^0 + P_n^0(P_n^0 + R)^{-1} z_n \\
P_n &= ((P_n^0 + R)(P_n^0 + R)^{-1} - P_n^0(P_n^0 + R)^{-1}) P_n^0 = R(P_n^0 + R)^{-1} P_n^0
\end{aligned}$$

Note how setting $z_n = x_n^*$, $P_n^0 = \hat{\Sigma}_{n-1}$ and $R = \Sigma_n^*$ the measurement update equations coincide with Equation 4.10 for product of two Gaussians, showing how the inter-frame regression algorithm is indeed equivalent to a KF.

4.6.3 Derivation for Section 4.3.3

Focusing on the optimization associated to Table 2, let us consider a generic Gauss-Newton iterative update which reads:

$$x_n^{i+1} = x_n^i - (\bar{J}_n^T \bar{J}_n)^{-1} \bar{J}_n^T \bar{F}_n \tag{4.34}$$

observing that

$$\bar{J}_n = \begin{bmatrix} -J_n^i \\ \hat{\Sigma}_{n-1}^{-1/2} \end{bmatrix} \quad \bar{F}_n = \begin{bmatrix} d_n - F_n^i \\ \hat{\Sigma}_{n-1}^{-1/2}(x_n^i - \hat{x}_{n-1}) \end{bmatrix} \tag{4.35}$$

we obtain what follows:

$$\begin{aligned} \bar{J}_n^T \bar{J}_n &= {J_n^i}^T J_n^i + \hat{\Sigma}_{n-1}^{-1} = A^{-1} \\ \bar{J}_n^T \bar{F}_n &= -{J_n^i}^T (z_n - F_n^i) + \hat{\Sigma}_{n-1}^{-1}(x_n^i - \hat{x}_{n-1}) \end{aligned}$$

where $F_n^i = F(x_n^i)$ and $J_n^i = \frac{\partial F}{\partial x_n}(x_n^i)$. Hence, expanding the matrix products in (4.34), we can write:

$$\begin{aligned} x_n^{i+1} &= x_n^i + \overbrace{A{J_n^i}^T (d_n - F_n^i)}^{B} - A\hat{\Sigma}_{n-1}^{-1}(x_n^i - \hat{x}_{n-1}) = \\ &= \hat{x}_{n-1} + B - A\hat{\Sigma}_{n-1}^{-1}(x_n^i - \hat{x}_{n-1}) + AA^{-1}(x_n^i - \hat{x}_{n-1}) = \\ &= \hat{x}_{n-1} + B + A(-\hat{\Sigma}_{n-1}^{-1} + {J_n^i}^T J_n^i + \hat{\Sigma}_{n-1}^{-1})(x_n^i - \hat{x}_{n-1}) = \\ &= \hat{x}_{n-1} + B + A{J_n^i}^T J_n^i (x_n^i - \hat{x}_{n-1}) = \\ &= \hat{x}_{n-1} + \underbrace{\left({J_n^i}^T J_n^i + \hat{\Sigma}_{n-1}^{-1}\right)^{-1} {J_n^i}^T}_{K_n^i} \left(d_n - F_n^i - J_n^i(\hat{x}_{n-1} - x_n^i)\right). \end{aligned}$$

Recalling the definition of the *a priori* estimate $x_n^0 = \hat{x}_{n-1}$, setting $H_n^0 = H_{n-1}$ and denoting $\hat{\Sigma}_{n-1}^{-1} = r H_{n-1}$ we can now see that such an iterative update is equivalent to the update of the IEIF from Table 7 for measurement $z_n = d_n$. Finally, under the assumption of the measurement noise to be i.i.d. across samples, we can conclude that the optimization from 7 is indeed equivalent to an IEKF.

5 Conclusion

5.1 Summary

In Chapter 2 we have presented a new model-based approach to real-time hand tracking using a single low-cost depth camera. This simple acquisition setup maximizes ease of deployment, but poses significant challenges for robust tracking. Our analysis revealed that a major source of error when tracking articulated hands are erroneous correspondences between the hand model and the acquired data, mainly caused by outliers and missing data. We demonstrate that these problems can be resolved by our new formulation of correspondence search. In combination with suitable 2D/3D registration energies and data-driven priors, this leads to a robust and efficient hand tracking algorithm that outperforms existing model- and appearance-based solutions. In our experiments we show that our system runs seamlessly for sensors capturing data at 60 Hz.

In Chapter 3 we have introduced the use of sphere-meshes as a novel geometric representation for articulated tracking. We have demonstrated how this representation yields excellent results for real-time registration of articulated geometry, and presented a calibration algorithm to estimate a per-user tracking template. We have validated our results by demonstrating qualitative as well as quantitative improvements over the state-of-the-art. Our calibration optimization is related to the works of [Taylor et al., 2014], [Khamis et al., 2015] and [Joseph Tan et al., 2016], with a fundamental difference: the innate simplicity of sphere-meshes substantially simplifies the algorithmic complexity of calibration and tracking algorithms. This considered, we believe that with the use of compute shaders, articulated tracking on the GPU can become as effortless and efficient as simple mesh rasterization. Our sphere-mesh models are first approximations to the implicit functions lying at the core of the recently proposed geometric skinning techniques [Vaillant et al., 2013], [Vaillant et al., 2014]. Therefore, we believe the calibration of sphere-meshes to be the first step towards photorealistic real-time hand modeling and tracking.

In Chapter 4 we have presented the first accurate online hand calibration system. From an application point of view, our approach significantly improves on the usability of real-time

hand tracking, as it requires neither controlled calibration scans nor offline processing prior to tracking. This allows easy deployment in consumer-level applications. From a technical point of view, we introduce a principled approach to online integration of shape information of user-specific hand geometry. By leveraging uncertainty estimates derived from the optimization objective function, we automatically determine how informative each input frame is for improving the estimates of the different unknown model parameters. Our approach is general and can be applied to different types of calibration, e.g., for full body tracking. More broadly, we envision applications to other difficult types of model estimation problems, where unreliable data needs to be accumulated and integrated into a consistent representation.

By fully disclosing our source code and data we ensure that our method and results are reproducible, as well as facilitate future research and product development.

5.2 Future Work

Discriminative Tracking

Our previous work was focused on the generative component of a hand tracking system. For re-initialization we used a simple algorithm, similar to the one proposed by [Qian et al., 2014]. Meanwhile, discriminative tracking quality and efficiency was greatly improved in recent works. The use of more advanced re-initialization techniques like [Oberweger and Lepetit, 2017] would benefit our system. Furthermore, we believe an interesting venue for future work is how to optimally integrate per-frame estimates into generative trackers. State-of-the-art discriminative algorithms regress a full hand pose; however, a generative component of a system might only require predicting the palm transformation to re-initialize, similar to [Taylor et al., 2017]. It is instructive to research what would be the most suitable form an input provided by discriminative algorithm to a model-based tracker.

Hand Calibration for CNN

In discriminative tracking a costly and time-consuming part is annotating the training data. Recently [Dibra et al., 2017] presented a system that eliminates the need for annotation. They use a differentiable rendering algorithm and predict hand pose by training a CNN to match a synthesized depth and an input depth. The accuracy of this system could be enhanced by regressing hand *shape* online in addition to hand *pose*. In previous work we developed a shape-parametrized sphere-meshes hand template which could be used for this purpose.

Hand Tracking from RGB

Currently our system is only using depth images as an input. In the future we could incorporate an RGB-based term in our optimization. An RGB input is complementary to depth data, since it does not contain holes and sensor noise. Moreover, it provides higher quality edges and finer

texture details. For example, [Weise et al., 2011] have proposed an optical flow constraints energy term. We could use a similar energy as a starting point of our experiments. The discriminative component of hand tracking system can also benefit from being augmented with RGB input. Recently the first hand-tracking approaches that require RGB-only input were presented [Simon et al., 2017], [Zimmermann and Brox, 2017]. These techniques could be further explored and combined with standard depth input.

Improving Efficiency

In future work we plan to reduce computational overhead to facilitate deployment on mobile devices. In our system the closest point correspondences are computed independently for each point, thus the algorithm has a parallelizable structure. The run time can be decreased in several ways: through the use of compute shaders that eliminate context switch time between CUDA and OpenGL; via on-chip implementation of the algorithm; or through higher downsampling factors of the input data (possibly with adaptive rates depending on the hand part).

Feedback for Hand Calibration

To obtain a complete personalized tracking model, the user needs to perform a suitable series of hand poses. As discussed above, if a finger is never bent, the estimates of the phalanx lengths will be unreliable. Currently, the system provides limited visual feedback to the user to guide the calibration. In the future, we aim to design a feedback system that provides visual indication of the most informative hand poses given the current model estimate. For example, one could create a dictionary containing a suitable pose for estimating each parameter with high certainty. During calibration the user is prompted to show hand pose corresponding to the lowest certainty parameter.

Tracking Hands and Body

Other interesting avenues for future work include adapting the method to other tracking scenarios, such as full body tracking. The first *offline* hands and body tracking system was recently introduced by [Romero et al., 2017]. We envision full-body avatars used for communication as a primary application of such methods, which necessitates interactive frame rates. Approaches like [Dou et al., 2017] are developed for high frame rates, however they struggle handling high deformation exhibited during hand motion.

Dynamic Fusion with Sphere Meshes

The topology of our sphere-meshes template has been defined in a manual trial-and-error process. A more suitable topology could be estimated by optimization, possibly even adapting

the topology for specific users; For example, the work in [Thiery et al., 2016] could be extended to space-time point clouds. Similarly, one could think of a variant of [Newcombe et al., 2015] where sphere-meshes are instantiated on-the-fly.

Two hands and Hand-object Interactions

Building upon the basis of high-precision generative hand tracking developed in our previous works, one could extend the system to enable two hands or hand-object interaction. The main component to be added is a discriminative system that labels the pixels of the input image as right hand, left hand, object or background. Another crucial modification would be adjusting the optimized energy to work robustly under heavy occlusion. Excellent works in two-hand and hand-object interaction were recently presented - [Taylor et al., 2017], [Mueller et al., 2017]. However, this is still a very new area with room for improvement in accuracy.

Photorealistic Hands Avatars

Photorealistic hands could be used as a part of body avatar for communication or for increasing immersion in virtual reality applications. Systems for creating photorealistic face avatars were already proposed, for example [Thies et al., 2016] and [Tewari et al., 2017]. In our work we have already developed a parametric model of hand pose and shape. For creating hand avatars similarly to [Thies et al., 2016] and [Tewari et al., 2017], we need to also create a parametric model of hand texture. Alternatively, the texture along with fine geometric details could accumulated from RGB input during the calibration sequence.

Natural User Interfaces

Integrating our hand tracking solution to natural user interfaces and testing its capabilities for interaction with virtual objects could drive further research directions. Through this kind of testing we could learn which components require improvement and which are not so crucial for the control applications.

Bibliography

[Albrecht et al., 2003] Albrecht, I., Haber, J., and Seidel, H.-P. (2003). Construction and animation of anatomically based human hand models. In *Proc. of the EG/SIGGRAPH Symposium on Computer Animation*.

[Anderson and Moore, 1979] Anderson, B. and Moore, J. (1979). *Optimal filtering*. Englewood Cliffs.

[Ballan et al., 2012] Ballan, L., Taneja, A., Gall, J., Van Gool, L., and Pollefeys, M. (2012). Motion capture of hands in action using discriminative salient points. In *Proc. of the European Conference on Computer Vision*.

[Bell and Cathey, 1993] Bell, B. M. and Cathey, F. W. (1993). The iterated kalman filter update as a gauss-newton method. In *IEEE Trans. on Automatic Control*.

[Bellaire et al., 1995] Bellaire, R. L., Kamen, E. W., and Zabin, S. M. (1995). New nonlinear iterated filter with applications to target tracking. In *SPIE Intl. Symposium on Optical Science, Engineering, and Instrumentation*.

[Bloomenthal et al., 1997] Bloomenthal, J., Bajaj, C., Blinn, J., Cani, M.-P., Rockwood, A., Wyvill, B., and Wyvill, G. (1997). *Introduction to Implicit Surfaces*. Morgan Kaufmann.

[Bloomenthal and Shoemake, 1991] Bloomenthal, J. and Shoemake, K. (1991). Convolution surfaces. In *Computer Graphics (Proc. SIGGRAPH)*.

[Botsch et al., 2010] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Lévy, B. (2010). *Polygon mesh processing*. CRC press.

[Bouaziz et al., 2012] Bouaziz, S., Deuss, M., Schwartzburg, Y., Weise, T., and Pauly, M. (2012). Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum (Proc. of the Symposium on Geometry Processing)*.

[Bouaziz et al., 2014] Bouaziz, S., Tagliasacchi, A., and Pauly, M. (2014). Dynamic 2D/3D registration. *Eurographics Tutorial*.

[Bouaziz et al., 2013] Bouaziz, S., Wang, Y., and Pauly, M. (2013). Online modeling for realtime facial animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*.

[Bullock et al., 2013] Bullock, I. M., Ma, R. R., and Dollar, A. M. (2013). A hand-centric classification of human and robot dexterous manipulation. *IEEE transactions on Haptics*, 6(2):129–144.

[Buss, 2004] Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. In *IEEE Journal of Robotics and Automation*.

[Cao et al., 2015] Cao, C., Bradley, D., Zhou, K., and Beeler, T. (2015). Real-time high-fidelity facial performance capture. In *ACM Trans. Graph. (Proc. SIGGRAPH)*.

[Chan and Dubey, 1995] Chan, T. F. and Dubey, R. V. (1995). A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. In *IEEE Trans. on Robotics and Automation*.

[de La Gorce et al., 2011] de La Gorce, M., Fleet, D. J., and Paragios, N. (2011). Model-based 3D hand pose estimation from monocular video. *Trans. on Pattern Analysis and Machine Intelligence*.

[Dibra et al., 2017] Dibra, E., Wolf, T., Öztireli, A. C., and Gross, M. H. (2017). How to refine 3d hand pose estimation from unlabelled depth data ? In *Fifth International Conference on 3D Vision (3DV), Qingdao, China, October 10-12, 2017*.

[Dipietro et al., 2008] Dipietro, L., Sabatini, A. M., and Dario, P. (2008). A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics*.

[Dollar, 2014] Dollar, A. M. (2014). Classifying human hand use and the activities of daily living. In *The Human Hand as an Inspiration for Robot Hand Development*, pages 201–216. Springer.

[Dou et al., 2017] Dou, M., Davidson, P., Fanello, S., Khamis, S., Kowdle, A., Rhemann, C., Tankovich, V., and Izadi, S. (2017). Motion2fusion: real-time volumetric performance capture. *ACM Trans. Graph.*, 36:1–16.

[Ekman and Friesen, 1977] Ekman, P. and Friesen, W. V. (1977). *Facial Action Coding System*. Consulting Psychologists Press, Stanford University, Palo Alto.

[Erol et al., 2007] Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. (2007). Vision-based hand pose estimation: A review. In *Computer Vision and Image Understanding*.

[Felzenszwalb and Huttenlocher, 2012] Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). Distance transforms of sampled functions. *Theory of Computing*.

[Fleishman et al., 2015] Fleishman, S., Kliger, M., Lerner, A., and Kutliroff, G. (2015). Icpik: Inverse kinematics based articulated-icp. In *Proc. of the IEEE CVPR Workshops (HANDS)*.

[Fourure et al., 2017] Fourure, D., Emonet, R., Fromont, E., Muselet, D., Neverova, N., Trémeau, A., and Wolf, C. (2017). Multi-task, multi-domain learning: application to semantic segmentation and pose regression. *Neurocomputing*, 251:68–80.

[Ganapathi et al., 2012] Ganapathi, V., Plagemann, C., Koller, D., and Thrun, S. (2012). Real-time human pose tracking from range data. In *Proc. of the European Conference on Computer Vision*, pages 738–751. Springer.

[Ge et al., 2016] Ge, L., Liang, H., Yuan, J., and Thalmann, D. (2016). Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3593–3601.

[Goman, 2009] Goman, C. K. (2009). *The Nonverbal Advantage: Secrets and Science of Body Language at Work: Easyread Super Large 20pt Edition*. ReadHowYouWant. com.

[Goman, 2011] Goman, C. K. (2011). The silent language of leaders. *How Body Language Can Help–Or Hurt-How You Lead*.

[Gu et al., 2017] Gu, J., Yang, X., De Mello, S., and Kautz, J. (2017). Dynamic facial analysis: From bayesian filtering to recurrent neural network. In *Computer Vision and Pattern Recognition*.

[Guo et al., 2017] Guo, H., Wang, G., Chen, X., Zhang, C., Qiao, F., and Yang, H. (2017). Region ensemble network: Improving convolutional network for hand pose estimation. *arXiv preprint arXiv:1702.02447*.

[Havlík and Straka, 2015] Havlík, J. and Straka, O. (2015). Performance evaluation of iterated extended kalman filter with variable step-length. In *Journal of Physics: Conf. Series*.

[Innmann et al., 2016] Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., and Stamminger, M. (2016). Volumedeform: Real-time volumetric non-rigid reconstruction. In *Proc. of the European Conference on Computer Vision*.

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al. (2011). Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. of the Symposium on User Interface Software and Technology*.

[Joseph Tan et al., 2016] Joseph Tan, D., Cashman, T., Taylor, J., Fitzgibbon, A., Tarlow, D., Khamis, S., Izadi, S., and Shotton, J. (2016). Fits like a glove: Rapid and reliable hand shape personalization. In *Computer Vision and Pattern Recognition*.

[Keskin et al., 2012] Keskin, C., Kıraç, F., Kara, Y. E., and Akarun, L. (2012). Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proc. of the European Conference on Computer Vision*.

[Khamis et al., 2015] Khamis, S., Taylor, J., Shotton, J., Keskin, C., Izadi, S., and Fitzgibbon, A. (2015). Learning an efficient model of hand shape variation from depth images.

[Krupka et al., 2014] Krupka, E., Vinnikov, A., Klein, B., Bar Hillel, A., Freedman, D., and Stachniak, S. (2014). Discriminative ferns ensemble for hand pose recognition. In *Computer Vision and Pattern Recognition*.

[Li et al., 2013] Li, H., Yu, J., Ye, Y., and Bregler, C. (2013). Realtime facial animation with on-the-fly correctives. In *ACM Trans. Graph. (Proc. SIGGRAPH)*.

[Li et al., 2015] Li, P., Ling, H., Li, X., and Liao, C. (2015). 3d hand pose estimation using randomized decision forest with segmentation index points. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 819–827.

[Loper and Black, 2014] Loper, M. M. and Black, M. J. (2014). OpenDR: An approximate differentiable renderer. In *Proc. of the European Conference on Computer Vision*.

[Madadi et al., 2017] Madadi, M., Escalera, S., Baro, X., and Gonzalez, J. (2017). End-to-end global to local cnn learning for hand pose recovery in depth data. *arXiv preprint arXiv:1705.09606*.

[Makris and Argyros, 2015] Makris, A. and Argyros, A. (2015). Model-based 3d hand tracking with online hand shape adaptation.

[Melax et al., 2013] Melax, S., Keselman, L., and Orsten, S. (2013). Dynamics based 3d skeletal hand tracking. In *Proc. of Graphics Interface*.

[Mueller et al., 2017] Mueller, F., Mehta, D., Sotnychenko, O., Sridhar, S., Casas, D., and Theobalt, C. (2017). Real-time hand tracking under occlusion from an egocentric rgb-d sensor. *arXiv preprint arXiv:1704.02201*.

[Neverova et al., 2017] Neverova, N., Wolf, C., Nebout, F., and Taylor, G. (2017). Hand pose estimation through semi-supervised and weakly-supervised learning. *arXiv preprint arXiv:1511.06728*.

[Newcombe et al., 2015] Newcombe, R. A., Fox, D., and Seitz, S. M. (2015). Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. *Computer Vision and Pattern Recognition*.

[Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer.

[Oberweger and Lepetit, 2017] Oberweger, M. and Lepetit, V. (2017). Deepprior++: Improving fast and accurate 3d hand pose estimation. In *ICCV workshop*, volume 840.

[Oberweger et al., 2015a] Oberweger, M., Wohlhart, P., and Lepetit, V. (2015a). Hands deep in deep learning for hand pose estimation. In *Proc. Computer Vision Winter Workshop*.

[Oberweger et al., 2015b] Oberweger, M., Wohlhart, P., and Lepetit, V. (2015b). Training a Feedback Loop for Hand Pose Estimation. In *International Conference on Computer Vision*.

[Oikonomidis et al., 2011] Oikonomidis, I., Kyriazis, N., and Argyros, A. A. (2011). Efficient model-based 3D tracking of hand articulation using kinect.

[Oikonomidis et al., 2012] Oikonomidis, I., Kyriazis, N., and Argyros, A. A. (2012). Tracking the articulated motion of two strongly interacting hands. In *Computer Vision and Pattern Recognition*. IEEE.

[Oikonomidis et al., 2014] Oikonomidis, I., Lourakis, M., Argyros, A., et al. (2014). Evolutionary quasi-random search for hand articulations tracking. In *Computer Vision and Pattern Recognition*.

[Olson and Zhang, 2006] Olson, M. and Zhang, H. (2006). Silhouette extraction in hough space. *Computer Graphics Forum (Proc. Eurographics)*.

[Petersen et al., 2008] Petersen, K. B., Pedersen, M. S., et al. (2008). *The matrix cookbook*. Technical University of Denmark.

[Plankers and Fua, 2003] Plankers, R. and Fua, P. (2003). Articulated soft objects for multi-view shape and motion capture. *Trans. on Pattern Analysis and Machine Intelligence*.

[Poier et al., 2015] Poier, G., Roditakis, K., Schulter, S., Michel, D., Bischof, H., and Argyros, A. A. (2015). Hybrid one-shot 3d hand pose estimation by exploiting uncertainties. *arXiv preprint arXiv:1510.08039*.

[Qian et al., 2014] Qian, C., Sun, X., Wei, Y., Tang, X., and Sun, J. (2014). Realtime and robust hand tracking from depth. In *Computer Vision and Pattern Recognition*.

[Rehg and Kanade, 1994] Rehg, J. M. and Kanade, T. (1994). Visual tracking of high dof articulated structures: An application to human hand tracking. In *Proc. of the European Conference on Computer Vision*.

[Remelli et al., 2017] Remelli, E., Tkach, A., Tagliassachi, A., and Pauly, M. (2017). Low-dimensionality calibration through local anisotropic for scaling for robust hand model personalization. In *International Conference on Computer Vision*.

[Rhee et al., 2006] Rhee, T., Neumann, U., and Lewis, J. P. (2006). Human hand modeling from surface anatomy. In *Proc. Symposium on Interactive 3D graphics and games*.

[Romero et al., 2017] Romero, J., Tzionas, D., and Black, M. (2017). Embodied hands: Modeling and capturing hands and bodies together. *ACM Trans. Graph.*

[Schroder et al., 2014] Schroder, M., Maycock, J., Ritter, H., and Botsch, M. (2014). Real-time hand tracking using synergistic inverse kinematics. In *International Conference on Robotics and Automation*.

[Sharp et al., 2015] Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Leichter, D. K. C. R. I., Wei, A. V. Y., Krupka, D. F. P. K. E., Fitzgibbon, A., and Izadi, S. (2015). Accurate, robust, and flexible real-time hand tracking. In *Proc. ACM Special Interest Group on Computer-Human Interaction (CHI)*.

[Shotton et al., 2011] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition*. IEEE.

[Simon et al., 2017] Simon, T., Joo, H., Matthews, I., and Sheikh, Y. (2017). Hand keypoint detection in single images using multiview bootstrapping. *arXiv preprint arXiv:1704.07809*.

[Sinha et al., 2016] Sinha, A., Choi, C., and Ramani, K. (2016). Deephand: Robust hand pose estimation by completing a matrix imputed with deep features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4150–4158.

[Skoglund et al., 2015] Skoglund, M. A., Hendeby, G., and Axehill, D. (2015). Extended kalman filter modifications based on an optimization view point. In *Intl. Conf. on Inf. Fusion*.

[Sridhar et al., 2015] Sridhar, S., Mueller, F., Oulasvirta, A., and Theobalt, C. (2015). Fast and robust hand tracking using detection-guided optimization. In *Computer Vision and Pattern Recognition*.

[Sridhar et al., 2013] Sridhar, S., Oulasvirta, A., and Theobalt, C. (2013). Interactive markerless articulated hand motion tracking using RGB and depth data. In *International Conference on Computer Vision*.

[Sridhar et al., 2014] Sridhar, S., Rhodin, H., Seidel, H.-P., Oulasvirta, A., and Theobalt, C. (2014). Real-time hand tracking using a sum of anisotropic gaussians model. In *Proc. International Conference on 3D Vision (3DV)*.

[Strasdat et al., 2012] Strasdat, H., Montiel, J. M., and Davison, A. J. (2012). Visual slam: why filter? In *Image and Vision Computing*.

[Sun et al., 2015] Sun, X., Wei, Y., Liang, S., Tang, X., and Sun, J. (2015). Cascaded hand pose regression. In *Computer Vision and Pattern Recognition*.

[Supancic et al., 2015] Supancic, J. S., Rogez, G., Yang, Y., Shotton, J., and Ramanan, D. (2015). Depth-based hand pose estimation: data, methods, and challenges. In *International Conference on Computer Vision*.

[Tagliasacchi et al., 2016] Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., and Telea, A. (2016). 3d skeletons: A state-of-the-art report. *Computer Graphics Forum (Proc. Eurographics)*.

[Tagliasacchi and Li, 2016] Tagliasacchi, A. and Li, H. (2016). Modern techniques and applications for real-time non-rigid registration. *Proc. SIGGRAPH Asia (Technical Courses)*.

[Tagliasacchi et al., 2015] Tagliasacchi, A., Schroeder, M., Tkach, A., Bouaziz, S., Botsch, M., and Pauly, M. (2015). Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum (Proc. of the Symposium on Geometry Processing)*.

[Tan et al., 2016] Tan, D. J., Cashman, T., Taylor, J., Fitzgibbon, A., Tarlow, D., Khamis, S., Izadi, S., and Shotton, J. (2016). Fits like a glove: Rapid and reliable hand shape personalization. In *Computer Vision and Pattern Recognition*.

[Tang et al., 2014] Tang, D., Chang, H. J., Tejani, A., and Kim, T.-K. (2014). Latent regression forest: Structured estimation of 3D articulated hand posture. In *Computer Vision and Pattern Recognition*. IEEE.

[Tang et al., 2015] Tang, D., Taylor, J., Kohli, P., Keskin, C., Kim, T.-K., and Shotton, J. (2015). Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *International Conference on Computer Vision*.

[Taylor et al., 2016] Taylor, J., Bordeaux, L., Cashman, T., Corish, B., Keskin, C., Sharp, T., Soto, E., Sweeney, D., Valentin, J., Luff, B., et al. (2016). Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. In *ACM Trans. Graph. (Proc. SIGGRAPH)*.

[Taylor et al., 2014] Taylor, J., Stebbing, R., Ramakrishna, V., Keskin, C., Shotton, J., Izadi, S., Hertzmann, A., and Fitzgibbon, A. (2014). User-specific hand modeling from monocular depth sequences. In *Computer Vision and Pattern Recognition*.

[Taylor et al., 2017] Taylor, J., Tankovich, V., Tang, D., Keskin, C., Kim, D., Davidson, P., Kowdle, A., and Izadi, S. (2017). Articulated distance fields for ultra-fast tracking of hands interacting. *ACM Trans. Graph.*, 36(6):244:1–244:12.

[Tewari et al., 2017] Tewari, A., Zollhöfer, M., Kim, H., Garrido, P., Bernard, F., Pérez, P., and Theobalt, C. (2017). Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. *arXiv preprint arXiv:1703.10580*.

[Thiery et al., 2013] Thiery, J.-M., Guy, E., and Boubekeur, T. (2013). Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Trans. Graph. (Proc. SIGGRAPH)*.

[Thiery et al., 2016] Thiery, J.-M., Guy, E., Boubekeur, T., and Eisemann, E. (2016). Animated mesh approximation with sphere-meshes. *ACM Trans. Graph.*

[Thies et al., 2015] Thies, J., Zollhöfer, M., Nießner, M., Valgaerts, L., Stamminger, M., and Theobalt, C. (2015). Real-time expression transfer for facial reenactment. *ACM Trans. Graph.*, 34(6):183–1.

[Thies et al., 2016] Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C., and Nießner, M. (2016). Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2387–2395.

[Tkach et al., 2016] Tkach, A., Pauly, M., and Tagliasacchi, A. (2016). Sphere-meshes for real-time hand modeling and tracking. In *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*.

[Tkach et al., 2017] Tkach, A., Tagliasacchi, A., Remelli, E., Pauly, M., and Fitzgibbon, A. (2017). Online generative model personalization for hand tracking. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*.

[Tompson et al., 2014] Tompson, J., Stein, M., Lecun, Y., and Perlin, K. (2014). Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Graph.*

[Vaillant et al., 2013] Vaillant, R., Barthe, L., Guennebaud, G., Cani, M.-P., Rohmer, D., Wyvill, B., Gourmel, O., and Paulin, M. (2013). Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph. (Proc. SIGGRAPH)*.

[Vaillant et al., 2014] Vaillant, R., Guennebaud, G., Barthe, L., Wyvill, B., and Cani, M.-P. (2014). Robust iso-surface tracking for interactive character skinning. *ACM Trans. Graph.*

[Wan et al., 2017] Wan, C., Probst, T., Van Gool, L., and Yao, A. (2017). Crossing nets: Dual generative models with a shared latent space for hand pose estimation. *arXiv preprint arXiv:1702.03431*.

[Wan et al., 2016] Wan, C., Yao, A., and Van Gool, L. (2016). Hand pose estimation from local surface normals. In *European Conference on Computer Vision*, pages 554–569. Springer.

[Wei et al., 2012] Wei, X., Zhang, P., and Chai, J. (2012). Accurate realtime full-body motion capture using a single depth camera. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*.

[Weise et al., 2011] Weise, T., Bouaziz, S., Li, H., and Pauly, M. (2011). Realtime performance-based facial animation. In *ACM Trans. Graph. (Proc. SIGGRAPH)*.

[Welch and Bishop, 1995] Welch, G. and Bishop, G. (1995). *An introduction to the Kalman filter*.

[Welch and Foxlin, 2002] Welch, G. and Foxlin, E. (2002). Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Comput. Graph. Appl.*

[Xu and Cheng, 2013] Xu, C. and Cheng, L. (2013). Efficient hand pose estimation from a single depth image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3456–3462.

[Yuan et al., 2017] Yuan, S., Ye, Q., Stenger, B., Jain, S., and Kim, T.-K. (2017). Big-hand2m benchmark: Hand pose dataset and state of the art analysis. In *arXiv preprint arXiv:1704.02612*.

[Zhang et al., 2014] Zhang, P., Siu, K., Zhang, J., Liu, C. K., and Chai, J. (2014). Leveraging depth cameras and wearable pressure sensors for full-body kinematics and dynamics capture. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*.

[Zhou et al., 2016] Zhou, X., Wan, Q., Zhang, W., Xue, X., and Wei, Y. (2016). Model-based deep hand pose estimation. *arXiv preprint arXiv:1606.06854*.

[Zimmermann and Brox, 2017] Zimmermann, C. and Brox, T. (2017). Learning to estimate 3d hand pose from single rgb images. *arXiv preprint arXiv:1705.01389*.

[Zou and Tan, 2013] Zou, D. and Tan, P. (2013). Coslam: Collaborative visual slam in dynamic environments. In *Trans. on Pattern Analysis and Machine Intelligence*.

Rue du Centre 164
1025 Saint-Sulpice, Switzerland
anastasia.l.tkach@gmail.com
+41 (774) 262 752

Anastasia Tkach

EDUCATION

09.14-current	**PhD in Computer Science (Computer Vision, Hand Tracking)** Swiss Federal Institute of Technology (EPFL), Switzerland
09.11-02.14	**Master in Computer Science** Swiss Federal Institute of Technology (EPFL), Switzerland, AGP 5.72/6.00
09.07-07.11	**Bachelor in Robotics** Bauman Moscow State Technical University (BMSTU), Russia, AGP 5.00/5.00

PHD THESIS - High Accuracy Real-Time Hand Tracking from Depth Sensor Data

- Advisors: Dr. Prof. Mark Pauly, Dr. Prof. Andrea Tagliasacchi
- Motivation: hand control of virtual or augmented reality devices
- Components of our real-time hand tracking system
  Numerical optimization (finding pose and shape parameters of hand model given the data);
  Bayesian Modelling (online update shape parameters);
  Classification and Regression (hand detection; tracking failure detection);
  Real-time implementation (C++; GPU/CUDA; OpenGL for rendering);
  Latent space embedding (learning prior distributions on hand pose and shape);

PUBLICATIONS

- A. Tkach*, A. Tagliasacchi*, E. Remelli, M. Pauli, A. Fitzgibbon. "Online Generative Model Personalization for Hand Tracking". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia), 2017.
- E. Remelli*, A. Tkach*, A. Tagliasacchi, M. Pauli. "Low-Dimensionality Calibration through Local Anisotropic Scaling for Robust Hand Model Personalization". ICCV, 2017.
- A. Tkach, M. Pauly, A. Tagliasacchi. "Sphere-Meshes for Real-Time Hand Modeling and Tracking". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia), 2016.
- A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, M. Pauly. "Robust Articulated-ICP for Real-Time Hand Tracking". Computer Graphics Forum (Proceedings of SGP), 2015.

SKILLS

- Programming C++, Python, Tensorflow, C#, MATLAB, Java
- Computer Vision, Numerical Optimization, Performance Capture, Machine Learning, Image Processing
- Data Structures and Algorithms

AWARDS

- Travel grant for HANDS workshop at CVPR 2016
- Best paper award at Symposium on Geometry Processing, Graz, 2015
- Computer Science School Fellowship, Swiss Federal Institute of Technology 2014-2015
- Google (Anita Borg) Scholarship finalist, June 2009

PATENTS

- A. Tkach, A. Tagliasacchi, M. Pauly, 2016, provisional patent "Convolution Models for Real-Time Hand Modeling and Tracking" (KS Ref. No. 2847-97161-01), filed 14.08.2016.

OPEN SOURCE PROJECTS

- https://github.com/OpenGP/htrack
- https://github.com/OpenGP/hmodel
- https://github.com/OpenGP/honline

INDUSTRIAL EXPERIENCE

07.17-09.17

Google Corporation, Zurich, Switzerland

Research intern, Machine Perception team, supervisor Dr. Luciano Sbaiz

- *Trained a deep network for video quality prediction.*
- *Compared seven model architectures used for prediction from video input using single type of input features.*
- *Compared fusion architectures for prediction from several different input features.*
- *Implemented a tool for interpreting model results and examining causes of false positives and negatives.*

07.16-09.16

Microsoft Research Cambridge, United Kingdom

Research intern, Machine Intelligence and Perception team, supervisor Dr. Prof. Andrew Fitzgibbon

- *Derived robust optimization procedure (in spirit of Kalman Filter) for calibrating hand model in real time to the user of hand tracking system.*

04.14-08.14

Google Corporation, Krakow, Poland

Software Engineer intern, Google TV team

- *Developed the system that determines main idea for an input group of movies and recommends movies having the same idea.*
- *Enabled navigation in movies space by adding and subtracting attributes.*
- *Implemented a parallelized, cloud-based prototype working on massive amounts of data.*
- *To our knowledge, this is the first system of such kind. The system was approved to be put in production.*

07.13-09.13

Microsoft Corporation, Redmond, USA

Software Development Engineer in Test intern, Common Language Runtime team, Visual Studio

- *Developed from the scratch an approach for predicting pass/failure of the tests on the current version of the software (accuracy 97.31%). The approach enabled breaking changes detection by order of magnitude faster.*
- *Implemented and validated with daily testing results a tool that is using the suggested approach.*
- *Prepared the tool for integration in test automation system of CLR team.*

02.13-07.13

Sony Research, Stuttgart, Germany

Research intern on "Personalization and Recommendation" project

- *Implemented a recommendation system based on Matrix Completion using a recently published technique - Accelerated Proximal Gradient algorithm (APG).*
- *Adapted APG algorithm designed to work with ratings of the products for working only with a list of previous purchases, which does not require that customer explicitly rates a product.*
- *Improved the performance of recommendation system by 9.3 percent w.r.t previous approaches.*

MASTER THESIS - Photorealistic Face Synthesis

- *Developed a pipeline for generating high resolution and complete facial texture given a geometry model and several low-resolution photos of an arbitrary face.*
- *Extended a state-of-art super-resolution algorithm by adding several stages of image quality enhancement. Tailored the algorithm to facial texture input. Implemented exemplar-based texture generation for the face regions missing from the input photos. Introduced adaptive exemplar sizes depending on missing area diameter thus speeding up the algorithm.*

INVITED TALKS

- ICCV 2017, PeopleCap: capturing and modeling human bodies, faces and hands

REVIEWED PAPERS FOR

- Computer Graphics Forum 2016
- ICCV workshop 2017
- Eurographics 2018
- Graphics Interface 2018