Real-time Pose and Shape Reconstruction of Two Interacting Hands With a Single Depth Camera

FRANZISKA MUELLER, MPI Informatics, Saarland Informatics Campus
MICAH DAVIS, Universidad Rey Juan Carlos
FLORIAN BERNARD and OLEKSANDR SOTNYCHENKO, MPI Informatics, Saarland Informatics Campus
MICKEAL VERSCHOOR, MIGUEL A. OTADUY, and DAN CASAS, Universidad Rey Juan Carlos
CHRISTIAN THEOBALT, MPI Informatics, Saarland Informatics Campus

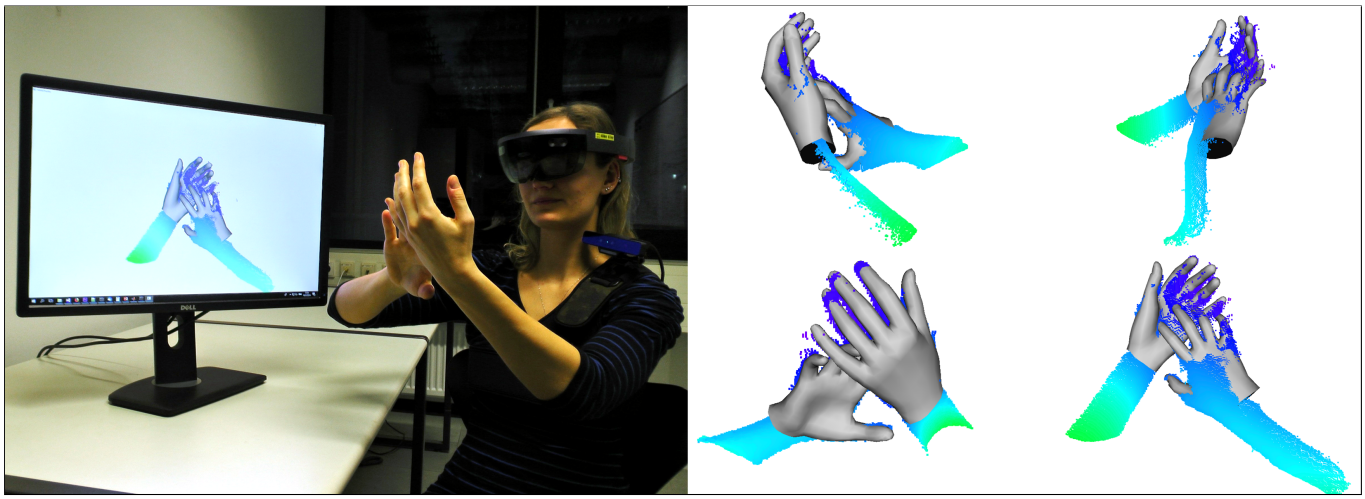


Fig. 1. We present a method that estimates the pose and shape of two interacting hands in real time from a single depth camera. On the left we show an AR setup with a shoulder-mounted depth camera. On the right we show the depth data and the estimated 3D hand pose and shape from four different views.

We present a novel method for real-time pose and shape reconstruction of two strongly interacting hands. Our approach is the first two-hand tracking solution that combines an extensive list of favorable properties, namely it is marker-less, uses a single consumer-level depth camera, runs in real time, handles inter- and intra-hand collisions, and automatically adjusts to the user's hand shape. In order to achieve this, we embed a recent parametric hand pose and shape model and a dense correspondence predictor based on a deep neural network into a suitable energy minimization framework. For training the correspondence prediction network, we synthesize a two-hand dataset based on physical simulations that includes both hand pose and shape annotations while at the same time avoiding inter-hand penetrations.

To achieve real-time rates, we phrase the model fitting in terms of a nonlinear least-squares problem so that the energy can be optimized based on a highly efficient GPU-based Gauss-Newton optimizer. We show state-of-the-art results in scenes that exceed the complexity level demonstrated by previous work, including tight two-hand grasps, significant inter-hand occlusions, and gesture interaction.[1]

CCS Concepts: • **Computing methodologies** → **Tracking**; *Computer vision*; Neural networks.

Additional Key Words and Phrases: hand tracking, hand pose estimation, two hands, depth camera, computer vision

ACM Reference Format:
Franziska Mueller, Micah Davis, Florian Bernard, Oleksandr Sotnychenko, Mickeal Verschoor, Miguel A. Otaduy, Dan Casas, and Christian Theobalt. 2019. Real-time Pose and Shape Reconstruction of Two Interacting Hands With a Single Depth Camera. *ACM Trans. Graph.* 38, 4, Article 49 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3322958

1 INTRODUCTION

The marker-less estimation of hand poses is a challenging problem that has received a lot of attention in the vision and graphics communities. The relevance of the problem is owed to the fact that hand pose recognition plays an important role in many application

Authors' addresses: Franziska Mueller, MPI Informatics, Saarland Informatics Campus, frmueller@mpi-inf.mpg.de; Micah Davis, Universidad Rey Juan Carlos, davis.micah@urjc.es; Florian Bernard, fbernard@mpi-inf.mpg.de; Oleksandr Sotnychenko, MPI Informatics, Saarland Informatics Campus, osotnych@mpi-inf.mpg.de; Mickeal Verschoor, mickeal.verschoor@urjc.es; Miguel A. Otaduy, miguel.otaduy@urjc.es; Dan Casas, Universidad Rey Juan Carlos, dan.casas@urjc.es; Christian Theobalt, MPI Informatics, Saarland Informatics Campus, theobalt@mpi-inf.mpg.de.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2019/7-ART49 $15.00
https://doi.org/10.1145/3306346.3322958

[1] project website: https://handtracker.mpi-inf.mpg.de/projects/TwoHands/

areas such as human-computer interaction [Kim et al. 2012], augmented and virtual reality (AR/VR) [Höll et al. 2018], sign language recognition [Koller et al. 2016], as well as body language recognition relevant for psychology. Depending on the particular application, additional requirements are frequently imposed on the method, such as performing hand tracking in real time, or dynamically adapting the tracking to person-specific hand shapes for increased accuracy. Ideally, reconstruction should be possible with a simple hardware setup and therefore methods with a single color or depth camera are widely researched. Existing marker-less methods for hand pose estimation typically rely on either RGB [Cai et al. 2018; Mueller et al. 2018; Zimmermann and Brox 2017], depth images [Sridhar et al. 2015; Supančič et al. 2018; Taylor et al. 2017; Yuan et al. 2018], or a combination of both [Oikonomidis et al. 2011a; Rogez et al. 2014]. The major part of existing methods considers the problem of processing a single hand only [Oberweger et al. 2015; Qian et al. 2014; Ye and Kim 2018]. Some of them are even able to handle object interactions [Mueller et al. 2017; Sridhar et al. 2016; Tzionas et al. 2016], which is especially challenging due to potential occlusions.

As humans naturally use both their hands during daily routine tasks, many applications require to track both hands simultaneously (see Fig. 1), rather than tracking a single hand in isolation. While there are a few existing works that consider the problem of tracking two hands at the same time, they are limited in at least one of the following points: (i) they only work for rather simple interaction scenarios (e.g. no tight two-hand grasps, significant inter-hand occlusions, or gesture interaction), (ii) they are computationally expensive and not real-time capable, (iii) they do not handle collisions between the hands, (iv) they use a person-specific hand model that does not automatically adapt to unseen hand shapes, or (v) they heavily rely on custom-built dedicated hardware. In contrast to existing methods, our approach can handle two hands in interaction while not having any of the limitations (i)-(v), see Table 1.

We present for the first time a marker-less method that can track two hands with complex interactions in real time with a single depth camera, while at the same time being able to estimate the person's hand shape. From a technical point of view, this is achieved thanks to a novel learned dense surface correspondence predictor that is combined with a recent parametric hand model [Romero et al. 2017]. These two components are combined in an energy minimization framework to find the pose and shape parameters of both hands in a given depth image. Inspired by the recent success of deep learning approaches, especially for image-based prediction tasks [Alp Güler et al. 2018; Alp Guler et al. 2017; Badrinarayanan et al. 2015; Zhang et al. 2017], we employ a correspondence regressor based on deep neural networks. Compared to ICP-like local optimization approaches, using such a global correspondence predictor is advantageous, as it is less prone to the failures caused by wrong initialization and can easily recover even from severe tracking errors (see our supplementary video). Since it is not feasible to obtain reliable dense correspondence annotations in real data, we create a synthetic dataset of interacting hands to train the correspondence regressor. Here, it is crucial to obtain *natural* interactions between the hands, which implies that simply rendering a model of the left and of the right hand (in different poses) into the same view is not sufficient. Instead, we make use of an extension of the motion capture-driven physical simulation [Verschoor et al. 2018] that leads to faithful, collision-free, and physically plausible simulated hand-hand interactions.

Table 1. Our method is the first to combine several desirable properties.

	[Oikon. 2012]	[Tzionas 2016]	[Tkach 2017]	[Taylor 2017]	Ours
Interacting Hands	✓	✓	✗	✓	✓
Shape Estimation	✗	✗	✓	✗	✓
Real Time	✗	✗	✓	✓	✓
Commodity Sensor	✓	✓	✓	✗	✓
Collision Avoidance	✓	✓	✓	✗	✓

The main contributions of our approach are summarized as follows:

- For the first time we present a method that can track two interacting hands in real time with a single depth camera, while at the same time being able to estimate the hand shape and taking collisions into account.
- Moreover, our approach is the first one that leverages physical simulations for creating a two-hand tracking dataset that includes both pose and dense shape annotations while at the same time avoiding inter-hand penetrations.
- Contrary to existing methods, our approach is more robust and reliable in involved hand-hand interaction settings.

2 RELATED WORK

Hand pose estimation is an actively researched topic that has a wide range of applications, for example in human–computer interaction or activity recognition. While many methods reconstruct the motion of a single hand in isolation, comparably few existing approaches can work with multiple interacting hands, or estimate the hand shape. In the following, we discuss related works that tackle one of these problems.

Capturing a Single Hand: Although multi-camera setups [Oikonomidis et al. 2011b; Sridhar et al. 2013] are advantageous in terms of tracking quality, e.g. less ambiguity under occlusions, they are infeasible for many applications due to their inflexibility and cumbersome setup. Hence, the majority of recent hand pose estimation methods either uses a single RGB or depth camera. These methods can generally be split into three categories: generative, discriminative, and hybrid algorithms. Generative methods fit a model to image evidence by optimizing an objective function [Melax et al. 2013; Tagliasacchi et al. 2015; Tkach et al. 2016]. While they have the advantage that they generalize well to unseen poses, a downside is that generally they are sensitive to the initialization and thus may not recover from tracking failures. At the other end of the spectrum are discriminative methods that use machine learning techniques to estimate the hand pose with (usually) a single prediction [Choi et al. 2015; Rogez et al. 2014; Tang et al. 2014; Wan et al. 2016]. Despite being dependent on their training corpus, they

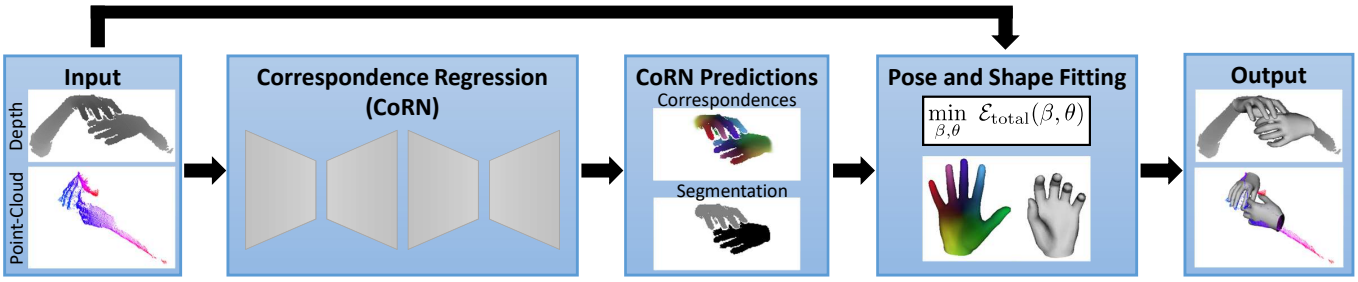


Fig. 2. Overview of our two-hand pose and shape estimation pipeline. Given only a depth image as input, our dense correspondence regression network (CoRN) computes a left/right segmentation and a vertex-to-pixel map. To obtain the hand shape estimation and pose tracking we use this data in an energy minimization framework, where a parametric hand pose and shape model is fit so that it best explains the input data.

do not require initialization at test time and hence recover quickly from failures. The recent success of deep neural networks has led to many works that regress hand pose from depth images [Baek et al. 2018; Ge et al. 2018; Oberweger et al. 2015; Tompson et al. 2014; Wan et al. 2017] or even from more underconstrained monocular RGB input [Cai et al. 2018; Mueller et al. 2018; Simon et al. 2017; Spurr et al. 2018; Zimmermann and Brox 2017]. Hybrid methods [Sridhar et al. 2015; Tang et al. 2015] combine generative and discriminative techniques, for example to get robust initialization for model fitting. A more detailed overview of depth-based approaches for single-hand pose estimation is provided by Yuan et al. [2018] and Supančič et al. [2018].

Hand Shape Models: There exist various hand models, i.e. models of hand geometry and pose, that are used for pose estimation by generative and hybrid methods, ranging from a set of geometric primitives [Oikonomidis et al. 2011a; Qian et al. 2014] to surface models like meshes [Sharp et al. 2015]. Such models are usually personalized to individual users and are obtained manually, e.g. by laser scans or simple scaling of a base model. Only few methods estimate a detailed hand shape from depth images automatically. Khamis et al. [2015] build a shape model of a hand mesh from sets of depth images acquired from different users. A method for efficiently fitting this model to a sequence of a new actor was subsequently presented by Tan et al. [2016]. Tkach et al. [2017] jointly optimize pose and shape of a sphere mesh online, and accumulate shape information over time to minimize uncertainty. In contrast, Remelli et al. [2017] fit a sphere mesh directly to the whole image set by multi-stage calibration with local anisotropic scalings. Recently, Romero et al. [2017] proposed a low-dimensional parametric model for hand pose and shape which was obtained from 1000 high-resolution 3D scans of 31 subjects in a wide variety of hand poses.

Capturing Two Hands: Reconstructing two hands jointly introduces profound new challenges, such as the inherent segmentation problem and more severe occlusions. Some methods try to overcome these challenges by using marker gloves [Han et al. 2018] or multi-view setups [Ballan et al. 2012]. Other approaches tackle the problem from a single depth camera to achieve more flexibility and practical usability. An analysis-by-synthesis approach is employed by Oikonomidis et al. [2012] who minimize the discrepancy of a rendered depth image and the input using particle swarm optimization. Kyriazis and Argyros [2014] apply an ensemble of independent trackers, where the per-object trackers broadcast their state to resolve collisions. Tzionas et al. [2016] use discriminatively detected salient points and a collision term based on distance fields to obtain an intersection-free model fit. Nevertheless, the aforementioned single-camera methods do not achieve real-time rates, and operate at 0.2 to 4 frames per second. There exist some methods that track two hands in real time, albeit without being able to deal with close hand-hand interactions. Taylor et al. [2016] jointly optimize pose and correspondences of a subdivision surface model but the method fails when the hands come close together, making it unusable for capturing any hand-hand interaction. Taylor et al. [2017] employ machine learning techniques for hand segmentation and palm orientation initialization, and subsequently fit an articulated distance function. They use a custom-built high frame-rate depth camera to minimize the motion between frames, thus being able to fit the model with very few optimizer steps. However, they do not resolve collisions and they do not estimate hand shape, so that they require a given model for every user. While they show some examples of hand-hand interactions, they do not show very close and elaborate interactions, e.g. with tight grasps.

In contrast to previous two-hand tracking solutions, our approach (i) runs in real time with a commodity camera, (ii) is marker-less, (iii) uses a single (depth) camera only, (iv) handles hand collisions, and (v) automatically adjusts to the user's hand shape.

3 OVERVIEW

In Fig. 2 we provide an overview of the pipeline for performing real-time hand pose and shape reconstruction of two interacting hands from a single depth sensor. First, we train a neural network that regresses dense correspondences between the hand model and a depth image that depicts two (possibly interacting) hands. In order to disambiguate between pixels that belong to the left hand, and pixels that belong to the right hand, our dense correspondence map also encodes the segmentation of the left and right hand. For obtaining realistic training data of hand interactions, we make use of motion capture-driven physical simulation to generate (synthetic) depth images along with ground-truth correspondence maps. This data is additionally augmented with real depth data that is used for training the segmentation channel of the correspondence map.

The so-obtained correspondence maps are then used to initialize an energy minimization framework, where we fit a parametric hand model to the given depth data. During fitting we make use of statistical pose and shape regularizers to avoid implausible configurations, a temporal smoothness regularizer, as well as a collision regularizer in order to avoid interpenetration between both hands and within each hand.

In order to achieve real-time performance, we phrase the energy minimization step in terms of a nonlinear least-squares formulation, and make use of a highly efficient ad-hoc data-parallel GPU implementation based on a Gauss-Newton optimizer.

In the remainder of this section we describe the hand model that we use for the tracking (Sec. 3.1). Subsequently, we provide a detailed explanation of the dense correspondence regression including the data generation (Sec. 4), followed by a description of the pose and shape estimation (Sec. 5).

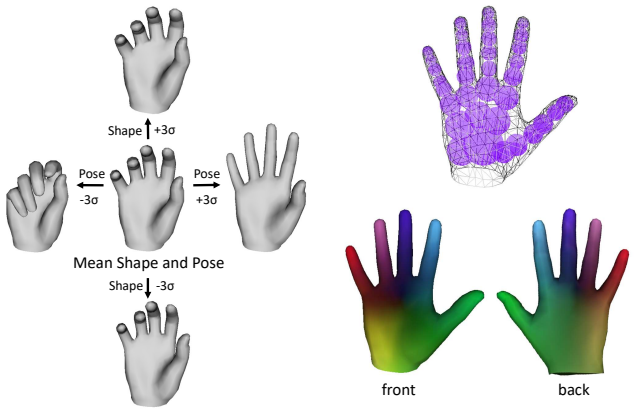


Fig. 3. Illustration of MANO hand model (left) that is augmented with our collision proxies (top right), as well as the correspondence color-encoding (bottom right). Notice that front and back color assignments differ in saturation, especially in the palm area.

3.1 Hand Model

As 3D hand representation, we employ the recently introduced MANO model [Romero et al. 2017], which is a low-dimensional parametric hand surface model that captures hand shape variation as well as hand pose variation, see Fig. 3 (left). It was built from about 1000 3D hand scans of 31 persons in wide range of different hand poses. The hand surface is represented by a 3D mesh with vertices $\mathcal{V}$, where $N_V := |\mathcal{V}| = 778$. The MANO model defines a function $\mathbf{v} : \mathbb{R}^{N_S} \times \mathbb{R}^{N_P} \rightarrow \mathbb{R}^{3N_V}$, that computes the 3D positions of all of the mesh's N_V vertices, given a shape parameter vector $\beta \in \mathbb{R}^{N_S}$ and pose parameter vector $\theta \in \mathbb{R}^{N_P}$, with $N_S = 10$ and $N_P = 51$. We use the notation $\mathbf{v}_i(\beta, \theta) \in \mathbb{R}^3$ to denote the 3D position of the i-th vertex. Parameters β and θ are coefficients of a low-dimensional pose and shape subspace that was obtained by PCA on the training data. As such, the MANO model naturally allows for a statistical regularization by simply imposing that the parameters are close to zero, which corresponds to a Tikhonov regularizer.

Note that since we are tracking two hands that can move independently, we use independent hand models of the left and right hand, which are denoted by $\mathcal{V}_{\text{left}}$ and $\mathcal{V}_{\text{right}}$ with vertices $\mathbf{v}_{\text{left}}(\beta_{\text{left}}, \theta_{\text{left}})$ and $\mathbf{v}_{\text{right}}(\beta_{\text{right}}, \theta_{\text{right}})$, respectively. For notational convenience, we stack the parameters of the left and right hand so that we have $\beta = (\beta_{\text{left}}, \beta_{\text{right}})$ and $\theta = (\theta_{\text{left}}, \theta_{\text{right}})$, and we use $\mathcal{V}$ with $N_v := |\mathcal{V}| = 2{\cdot}778$ to denote the combined vertices of the left and the right hand.

To resolve interpenetrations at high computational efficiency, we add collision proxies to the hand model. We follow the approach of Sridhar et al. [2015], who approximate the volumetric extent of the hand with a set of spheres that are modeled with 3D Gaussians. Using this formulation, interpenetrations can then be avoided by penalizing the overlap of the Gaussians during pose optimization. Note that the overlap between Gaussians is differentiable and can be computed in closed form—in contrast to naïve binary collision checking. We combine the Gaussians with the existing MANO model by rigging their positions to the hand joints and coupling their standard deviations to pairs of manually selected vertices. By doing this, we ensure that the position and size of the Gaussians vary in accordance with the pose and shape parameters β and θ. For each hand we add 35 3D Gaussians, which leads to a total number of $N_C = 70$ for the combined two-hands model. A visualization of the isosurface at 1 standard deviation of the Gaussians is shown in Fig. 3 (top right). Next, we describe our correspondence regressor that is eventually coupled with the two-hands model in order to perform pose and shape reconstruction.

4 DENSE CORRESPONDENCE REGRESSION

Let $\mathcal{I}$ be the input depth image of pixel-dimension h by w defined over the image domain Ω. Our aim is to learn a vertex-to-pixel correspondence map $c : \mathcal{V} \rightarrow \bar{\Omega}$ that assigns to each vertex of the model $\mathcal{V}$ a corresponding pixel of $\mathcal{I}$ in the image domain Ω. In order to allow the possibility to not assign an image pixel to a vertex (i.e. a vertex currently not visible), we extend the set Ω to also include $\emptyset$, which is denoted by $\bar{\Omega}$.

4.1 Dense Correspondence Encoding

To obtain the vertex-to-pixel correspondence map c we make use of a pixel-to-color mapping $\mathcal{N} : \Omega \rightarrow [0, 1]^4$ that assigns to each image pixel a 4-channel color value that encodes the correspondence. Here, the first 3 channels correspond to the dense correspondence on the hand surface (with the colors as shown in Fig. 3 bottom right) and the last channel encodes the segmentation label (0: left hand, 0.5: right hand, 1: non-hand). Due to this correspondence encoding in image space, it is more convenient to learn the pixel-to-color mapping $\mathcal{N}$, compared to directly learning the vertex-to-pixel mapping c. We emphasize that the function $\mathcal{N}$ is defined over the entire input image domain and thus is able to predict color-encoded correspondence values for any pixel. As such, it contains correspondence information for *both hands simultaneously* and thus implicitly learns to handle interactions between the left and the right hand. Please refer to Section 4.2 on how we generate the training data and Section 4.3 for details on how we learn $\mathcal{N}$.

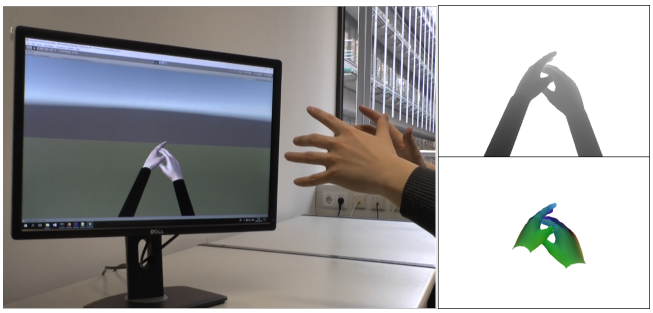


Fig. 4. We generate our synthetic dataset by tracking two hands, separated by a safety distance, which are used to control in real-time a physically-based simulation of two interacting hands in the virtual scenario (left). We output the depth map (top right) and dense surface annotations (bottom right) of the resulting simulation.

In order to associate the hand *model vertices* in $\mathcal{V}$ to image pixels in Ω based on the function $\mathcal{N}$, we also define a vertex-to-color mapping $\mathcal{M} : \mathcal{V} \rightarrow [0,1]^4$, similar to recent dense regression works [Huang et al. 2016; Taylor et al. 2012; Wei et al. 2016]. Note that the output of $\mathcal{M}$ for symmetric vertices in the left and right hand model only differs in the last component (0: left hand, 0.5: right hand), whereas the first three components encode the position on the hand surface as visualized in Fig. 3 (bottom right). Hence, the correspondences between vertices and pixels can be determined based on the similarity of the colors obtained by the mappings $\mathcal{N}$ and $\mathcal{M}$. In order to assign a color value $\mathcal{M}(i)$ to the i-th model vertex, we first use multi-dimensional scaling [Bronstein et al. 2006] for embedding the hand model into a three-dimensional space that (approximately) preserves geodesic distances. Subsequently, we map an HSV color space cylinder onto the embedded hand mesh such that different hues are mapped onto different fingers, cf. Fig. 3 (bottom right). As we later demonstrate (Fig. 6), the proposed geodesic HSV embedding leads to improved results compared to a naïve coloring (by mapping the RGB cube onto the original mesh, which is equivalent to regressing 3D vertex positions in the canonical pose).

Obtaining vertex-to-pixel mappings from the color encodings: To obtain a vertex-to-pixel correspondence map c for a given depth image $\mathcal{I}$, we first map the image pixels to colors using $\mathcal{N}$ (a function over the image domain). Subsequently, we compare the per-pixel colors obtained through $\mathcal{N}$ with the fixed per-vertex colors $\mathcal{M}$, from which the vertex-to-pixel maps are constructed by a thresholded nearest-neighbor strategy. For all $i \in \mathcal{V}$ we define

$$\hat{c}(i) = \underset{j \in \Omega}{\operatorname{argmin}} \ \|\mathcal{N}(j) - \mathcal{M}(i)\|_2 \,, \text{ and} \tag{1}$$

$$c(i) = \begin{cases} \hat{c}(i) & \text{if } \|\mathcal{N}(\hat{c}(i)) - \mathcal{M}(i)\|_2 < \eta, \\ \emptyset & \text{otherwise}\,. \end{cases} \tag{2}$$

If the closest predicted color for some vertex i is larger than the empirically chosen threshold η=0.04, this vertex is likely to be invisible in the input depth image.

4.2 Data Generation

In the following, we describe how we obtain suitable data to train the correspondence regression network.

Synthetic Data from Mocap-Driven Hand Simulation. To overcome the challenge of generating training data with dense correspondences under complex hand-hand interactions we leverage a physics-based hand simulator, similar in spirit to [Zhao et al. 2013]. To this end, we drive the simulation using skeletal hand motion capture (mocap) data [LeapMotion 2016] to maximize natural hand motion. We tackle the issue that existing hand mocap solutions cannot robustly deal with close and complex hand-hand interactions by letting the actor move both hands at a safety distance from each other. This safety distance is subtracted in the simulation to produce closely interacting hand motions. By running the hand simulation in real time, the actor receives immediate visual feedback and is thus able to simulate natural interactions. Fig. 4 depicts a live session of this data generation step.

We extended the work of Verschoor et al. [2018] by enabling simultaneous two hand simulation as well as inter-hand collision detection. The hand simulator consists of an articulated skeleton surrounded by a low-resolution finite-element soft tissue model. The hands of the actor are tracked using Leap Motion [2016], and the mocap skeletal configuration is linked through viscoelastic springs (a.k.a. PD controller) to the articulated skeleton of the hand simulator. In this way, the hand closely follows the mocap input during free motion, yet it reacts to contact. The hand simulator resolves inter-hand collisions using a penalty-based frictional contact model, which provides smooth soft tissue interactions at minimal computational cost. We have observed that the soft tissue layer is particularly helpful at allowing smooth and natural motions in highly constrained situations such as interlocking fingers. As the hands are commanded by the mocap input, their motion is inherently free of intra-hand collisions. While inter-hand interaction may produce finger motions that lead to intra-hand collisions, we found those to be negligible for the training purposes of this step. We thus avoided self-collision handling to maintain real-time interaction at all times.

In practice, in this data generation step, we output a depth image for each simulated frame as well as the corresponding rendered image of the hand meshes colored with the mapping $\mathcal{M}$. Additionally, we postprocess the generated depth images to mimic typical structured-light sensor noise at depth discontinuities. Using the above procedure, we recorded 5 users and synthesized 80,000 images in total.

Real Data with Segmentation Annotation: When only trained with synthetic data, neural networks tend to overfit and hence may not generalize well to real test data. To overcome this, we integrate real depth camera footage of hands into our so far synthetically generated training set. Since it is infeasible to obtain dense correspondence annotations on real data, we restrict the annotation on real data to the left/right hand segmentation task. As body paint [Soliman et al. 2018; Tompson et al. 2014] has less influence on the observed hand shape, in contrast to colored gloves [Taylor et al. 2017], we use body paint to obtain reliable annotations by color segmentation in the RGB image provided by the depth camera. In

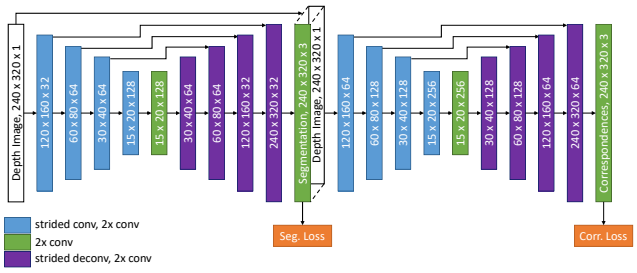


Fig. 5. Our correspondence regression network (CoRN) consists of two stacked encoder-decoder networks. The output sizes of the layer blocks are specified as height × width × number of feature channels. In addition, the colors of the layer blocks indicate which operations are performed (best viewed in color).

total, we captured 3 users (1 female, 2 male) with varying hand shapes (width: 8–10cm, length: 17–20.5cm). We recorded ≈ 3,000 images per subject and viewpoint (shoulder-mounted camera and frontal camera), resulting in a total number of 19,926 images.

4.3 Neural Network Regressor

Based on the mixed real and synthetic training data described in Section 4.2, we train a neural network that learns the pixel-to-color mapping $\mathcal{N}$, as depicted in Fig. 5. Inspired by recent architectures used for per-pixel predictions [Newell et al. 2016; Ronneberger et al. 2015], our network comprises two stacked encoder-decoder processing blocks. The first block is trained to learn the segmentation task, i.e. it outputs per-class probability maps in the original input resolution for the three possible classes {left, right, non-hand}. These class probability maps are concatenated with the input depth image $\mathcal{I}$ and fed into the second encoder-decoder to regress the 3-channel per-pixel hand surface correspondence information. The final mapping $\mathcal{N} : \Omega \rightarrow [0,1]^4$ is then obtained by concatenating the correspondence output with the label of the most likely class for each pixel. Note that we scale the class labels to also match the range $[0,1]$ by setting left = 0, right = 0.5, and non-hand = 1. Both our encoder-decoder subnetworks share the same architecture. We downsample the resolution using convolutions with stride 2 and upsample with the symmetric operation, deconvolutions with stride 2. Note that every convolution and deconvolution is followed by batch normalization and rectified linear unit (ReLU) layers. In addition, we use skip connections to preserve spatially localized information and enhance gradient backpropagation. Since the second subnetwork needs to learn a harder task, we double the number of features in all layers. The segmentation loss is formulated as the softmax cross entropy, a standard classification loss. For the correspondence loss, we use the squared Euclidean distance as commonly used in regression tasks. We train the complete network end-to-end, with mixed data batches containing both synthetic and real samples in one training iteration. For the latter, only the segmentation loss is active.

5 POSE AND SHAPE ESTIMATION

The pose and shape of the hands present in image $\mathcal{I}$ are estimated by fitting the hand surface model (Sec. 3.1) to the depth image data. To this end, we first extract the foreground point-cloud $\{\mathbf{d}_j \in \mathbb{R}^3\}_{j=1}^{N_I}$ in the depth image $\mathcal{I}$, along with the respective point-cloud normals $\{\mathbf{n}_j \in \mathbb{R}^3\}_{j=1}^{N_I}$ obtained by Sobel filtering. Based on the assumption that the hands and arms are the objects closest to the camera, the foreground is extracted using a simple depth-based thresholding strategy, where N_I denotes the total number of foreground pixels (of both hands together). Subsequently, this point-cloud data is used in conjunction with the learned vertex-to-pixel correspondence map c within an optimization framework. By minimizing a suitable nonlinear least-squares energy, which we will define next, the hand model parameters that best explain the point-cloud data are determined.

The total energy function for both the left and the right hand is defined as

$$\mathcal{E}_{\text{total}}(\beta,\theta) = \mathcal{E}_{\text{data}}(\beta,\theta) + \mathcal{E}_{\text{reg}}(\beta,\theta)\,, \tag{3}$$

where β are the shape parameters and θ are the hand pose parameters, as described in Sec. 3.1.

5.1 Data Term

The data term $\mathcal{E}_{\text{data}}$ measures for a given parameter tuple (β,θ) how well the hand model explains the depth image $\mathcal{I}$, and the term $\mathcal{E}_{\text{reg}}$ is a regularizer that accounts for temporal smoothness, plausible hand shapes and poses, as well as avoiding interpenetrations within and between the hands. We define the data term based on a combination of a point-to-point and a point-to-plane term as

$$\mathcal{E}_{\text{data}}(\beta,\theta) = \omega_{\text{point}} E_{\text{point}}(\beta,\theta) + \omega_{\text{plane}} E_{\text{plane}}(\beta,\theta)\,, \tag{4}$$

where we use $\omega_{\odot}$ to denote the relative weights of the terms.

Point-to-point: Let γ_i be the visibility indicator for the i-th vertex, which is defined to be 1 if $c(i) \neq \emptyset$, and 0 otherwise. The point-to-point energy measures the distances between all visible model vertices $\mathbf{v}_i(\beta,\theta)$ and the *corresponding* 3D point at pixel $c(i)$, and is defined as

$$E_{\text{point}}(\beta,\theta) = \sum_{i=1}^{N_{\mathcal{V}}} \gamma_i ||\mathbf{v}_i(\beta,\theta) - \mathbf{d}_{c(i)}||_2^2\,. \tag{5}$$

Point-to-plane: The point-to-plane energy is used to penalize the deviation from the model vertices $\mathbf{v}_i(\beta,\theta)$ and the point-cloud surface tangent, and is defined as

$$E_{\text{plane}}(\beta,\theta) = \sum_{i=1}^{N_{\mathcal{V}}} \gamma_i \left\langle \mathbf{v}_i(\beta,\theta) - \mathbf{d}_{c(i)}, \mathbf{n}_{c(i)} \right\rangle^2\,. \tag{6}$$

5.2 Regularizer

Our regularizer $\mathcal{E}_{\text{reg}}$ comprises statistical pose and shape regularization terms, a temporal smoothness term, as well as a collision term. We define it as

$$\mathcal{E}_{\text{reg}}(\beta,\theta) = \omega_{\text{shape}} E_{\text{shape}}(\beta) + \omega_{\text{pose}} E_{\text{pose}}(\theta) \tag{7}$$

$$\omega_{\text{temp}} E_{\text{temp}}(\beta,\theta) + \omega_{\text{coll}} E_{\text{coll}}(\beta,\theta)\,. \tag{8}$$

Statistical Regularizers: As explained in Sec. 3.1, the MANO model is parameterized in terms of a low-dimensional linear subspace obtained via PCA. Hence, in order to impose a plausible pose and shape at each captured frame, we use the Tikhonov regularizers

$$E_{\text{shape}}(\beta) = ||\beta||_2^2 \quad \text{and} \quad E_{\text{pose}}(\theta) = ||\theta||_2^2 \,. \tag{9}$$

Temporal Regularizer: In order to achieve temporal smoothness, we use a zero velocity prior on the shape parameters β and the pose parameters θ, i.e. we define

$$E_{\text{temp}}(\beta, \theta) = ||\beta^{(t)} - \beta^{(t-1)}||_2^2 + ||\theta^{(t)} - \theta^{(t-1)}||_2^2 \,. \tag{10}$$

Collision Regularizer: In order to avoid interpenetration between individual hands, as well as interpenetrations between the left and the right hand, we use a collision energy term. As described in Sec. 3.1, we place spherical collision proxies inside each hand mesh, and then penalize overlaps between these collision proxies. Mathematically, we phrase this based on the overlap of (isotropic) Gaussians [Sridhar et al. 2015], which results in soft collision proxies defined as smooth occupancy functions. The energy reads

$$E_{\text{coll}}(\beta, \theta) = \sum_{p=1}^{N_C} \sum_{q=p+1}^{N_C} \int_{\mathbb{R}^3} G_p(x; \beta, \theta) \cdot G_q(x; \beta, \theta)\, dx \,. \tag{11}$$

Here, G_p, G_q denote the Gaussian collision proxies whose mean and standard deviation depend on both the shape parameters β as well as on the pose parameters θ.

5.3 Optimization

We have phrased the energy $\mathcal{E}_{\text{total}}$ in terms of a nonlinear least-squares formulation, so that it is amenable to be optimized based on the Gauss-Newton algorithm. All derivatives of the residuals can be computed analytically, so that we can efficiently compute all entries of the Jacobian on the GPU with high accuracy. More details on the GPU implementation can be found in the Appendix.

Note that although in principal it would be sufficient to optimize for the shape parameter once per actor and then keep it fixed throughout the sequence, we perform the shape optimization in each frame of the sequence. This has the advantage that a poorly chosen frame for estimating the shape parameter does not have a negative impact on the tracking of subsequent frames. We have empirically found that the hand shape is robust and does not significantly change throughout a given sequence.

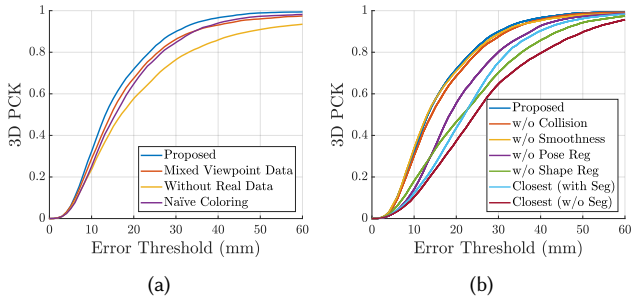
6 EVALUATION

In this section we thoroughly evaluate our proposed two-hand tracking approach. In Sec. 6.1 we present additional implementation details. Subsequently, in Sec. 6.2 we perform an ablation study, followed by a comparison to state-of-the-art tracking methods in Sec. 6.3. Eventually, in Sec. 6.4 we provide additional results that demonstrate the ability of our method to adapt to user-specific hand shapes.

6.1 Implementation

Our implementation runs on two GPUs of type NVIDIA GTX 1080 Ti. One GPU runs the correspondence regression network CoRN, as well as the per-vertex correspondence matching for frame $t+1$, while the other GPU runs the model optimization for frame t. Overall, we achieve 30 fps using an implementation based on C++, CUDA, and the Tensorflow library. We have used a depth camera Intel RealSense SR300 for our real-time results and evaluation. In Sec. 6.3 we also demonstrate results when using a publicly available dataset that was captured with a different sensor.

Unless stated otherwise, for training CoRN we always use synthetic and real images (cf. Sec. 4.2) rendered and recorded from a *frontal* view-point. We emphasize that it is reasonable to use view-specific correspondence regressors as for a given application it is usually known from which view-point the hands are to be tracked.



(a) (b)

Fig. 6. Results of our ablation study. (a) shows different configurations regarding the correspondence regressor (CoRN). (b) shows configurations regarding the optimizer.

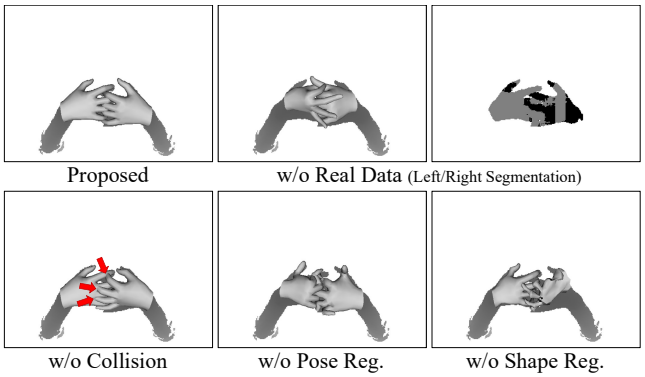


Fig. 7. Qualitative examples from our ablation study.

6.2 Ablation Study

We have conducted a detailed ablation study, where we analyze the effects of the individual components of the proposed approach. For these evaluations we use the dataset provided by Tzionas et al. [2016], which comes with annotations of the joint positions on the depth images. In Fig. 6 we show quantitative results of our analysis for a range of different configurations. To this end, we use the *percentage of correct keypoints* (PCK) as measure, where the horizontal axis shows the error, and the vertical axis indicates the percentage of points that fall within this error. To compute the PCK, we consider the same set of keypoints as Tzionas et al. [2016]. Notice that despite using Tzionas et al.'s dataset, Fig. 6 does not show their

results because they do not provide 3D PCK values. Qualitative results of our ablation study are shown in Fig. 7.

Correspondence Regression Network. In the Fig. 6a we show four settings of different configurations for training the correspondence regressor (CoRN):

(1) The proposed CoRN network as explained in Sec. 4 (blue line, "Proposed").
(2) The CoRN network but trained based on data from two viewpoints, egocentric as well as frontal (orange line, "Mixed Viewpoint Data").
(3) The CoRN network that is trained only with synthetic data, i.e. we do not use real data as described in Sec. 4.2 in order to train the segmentation sub-network (yellow line, "Without Real Data").
(4) Instead of using our proposed geodesic HSV embedding as color-encoding for the correspondences (cf. Fig. 3), we use a naïve color-encoding by mapping the original mesh onto the RGB cube (purple line, "Naïve Coloring").

It can be seen that the proposed training setting outperforms all other settings.

Pose and Shape Estimation. In Fig. 6b we show different optimizer configurations. We evaluate five versions of the energy:

(1) The complete energy $\mathcal{E}_{\text{total}}$ that includes all terms (blue line, "Proposed").
(2) The energy without the collision term E_{coll} (orange line, "w/o Collision").
(3) The energy without the temporal smoothness term E_{temp} (yellow line, "w/o Smoothness").
(4) The energy without the pose regularizer E_{pose} (purple line, "w/o Pose Reg").
(5) The energy without the pose regularizer E_{shape} (green line, "w/o Shape Reg").

In addition, to demonstrate the importance of CoRN, we compare to two configurations using closest point correspondences instead:

(1) Finding the vertex correspondence as the closest input point that was classified with the same handedness (light blue line, "Closest (with Seg)").
(2) Finding the vertex correspondence as the closest input point in the whole point cloud (dark red line, "Closest (w/o Seg)").

Note that we initialized the hand models manually as close as possible in the first frame to enable a fair comparison. We emphasize that this is not necessary with CoRN.

We can observe that the complete energy performs best, compared to leaving individual terms out. Moreover, we have found that removing the pose regularizer or the shape regularizer worsens the outcome significantly more compared to dropping the collision or the smoothness terms when looking at the PCK. We point out that the smoothness term removes temporal jitter that is only marginally reflected by the numbers. Similarly, while removing the collision term does not affect the PCK significantly, in the supplementary video we demonstrate that this severely worsens the results. Using naïve closest points instead of predicted CoRN correspondences results in significantly higher errors, this holds for both versions, with and without segmentation information. Additionally, in Figure 7 we show qualitative examples from our ablation study that further validate that each term of the complete energy formulation is essential to obtain high quality tracking of hand-hand interaction.

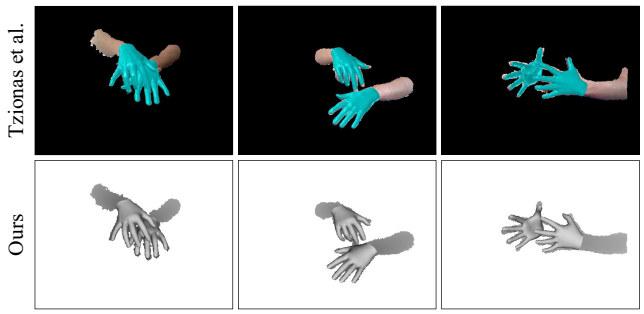


Fig. 8. Qualitative comparison with [Tzionas et al. 2016]. Our method achieves results with comparable visual quality while running multiple orders of magnitude faster.

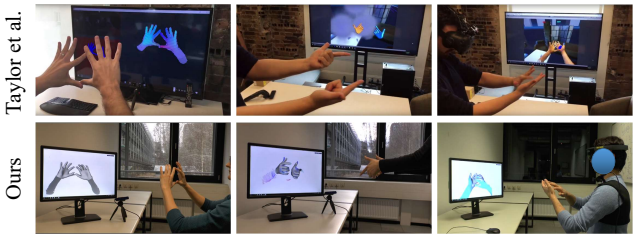


Fig. 9. Qualitative comparison with [Taylor et al. 2017]. Our method is able to track two hands in similar poses while at the same time reconstructing shape automatically and avoiding collisions.

Independence of Initialization: In the supplemental video we also show results where our hand tracker is able to recover from severe errors that occur when the hand motion is extremely fast, so that the depth image becomes blurry. In this scenario, as soon as the hand moves with a normal speed again, the tracker is able to recover and provide an accurate tracking. Note that this is in contrast to local optimization approaches (e.g. based on an ICP-like procedure for pose and shape fitting) that cannot recover from bad results due to severe non-convexity of the energy landscape.

6.3 Comparison to the State of the Art

Next, we compare our method with state-of-the-art methods.

Comparison to Tzionas et al. [2016]. In Table 2 we present results of our quantitative comparison to the work of Tzionas et al. [2016]. The evaluation is based on their two-hand dataset that comes with joint annotations. As shown, the relative 2D pixel error is very small in both methods. While it is slightly higher with our approach, we emphasize that we achieve a 150× speed-up and do not require a user-specific hand model. Furthermore, in Fig. 8 we qualitatively show that the precision error difference does not result in any noticeable visual quality gap. Moreover, we point out that the finger

Table 2. We compare our method to the method by Tzionas et al. [2016] on their provided dataset. We show the average and standard deviation of the 2D pixel error (relative to the diagonal image dimension), as well as the per-frame runtime. Note that the pixel errors of both methods are very small, and that our method is 150× faster. Moreover, our approach automatically adjusts to the user-specific hand shape, whereas Tzionas et al. require a 3D scanned hand model.

	2D Error	Runtime	Shape Estimation
Ours	1.35±0.28 %	33ms	✓
Tzionas et al.	0.63±0.12 %	4960ms	✗

tip detection method of Tzionas et al. [2016] is ad-hoc trained for their specific camera, whereas our correspondence regressor has never seen data from the depth sensor used in this comparison.

Comparison to Leap Motion [2016]. In the supplementary video we also compare our method qualitatively with the skeletal tracking results using the commercial solution [LeapMotion 2016]. As shown, while Leap Motion successfully tracks two hands when they are spatially separated by a significant offset, it struggles and fails for complex hand-hand interactions. In contrast, our approach is able to not only successfully track challenging hand-hand interactions, but also estimate the 3D hand shape.

Other Methods. Since the authors of [Taylor et al. 2017] did not release their dataset, we were not able to directly compare with their results. Nevertheless, in Fig. 9 and in the supplementary video we show tracking results on similar scenes, as well as some settings that are arguably more challenging than theirs.

6.4 More Results

In this section we present additional results on hand shape adaption as well as additional qualitative results.

Hand Shape Adaptation. Here, we investigate the adaptation to user-specific hand shapes. In Fig. 10 we show the obtained hand shape when running our method for four different persons with varying hand shapes. It can be seen that our method is able to adjust the geometry of the hand model to the users' hand shapes.

Due to the severe difficulty in obtaining reliable 3D ground truth data and disentangling shape and pose parameters, we cannot quantitatively evaluate shape directly. Instead, we additionally evaluate the consistency of the estimated bone lengths on the sequences of Tzionas et al. [2016]. The average standard deviation is 0.6 mm, which indicates that our shape estimation is stable over time.

Qualitative Results. In Fig. 11 we present qualitative results of our pose and shape estimation method. In the first two rows we show frames for an egocentric view-point, where CoRN was also trained for this setting, whereas the remaining rows show frames for a frontal view-point. It can be seen that in a wide range of complex hand-hand interactions our method robustly estimates the hand pose and shape. CoRN is an essential part of our method and is able to accurately predict segmentation and dense correspondences for a variety of inputs (see Fig. 12). However, wrong predictions may lead to errors in the final tracking result as demonstrated in Fig. 13.

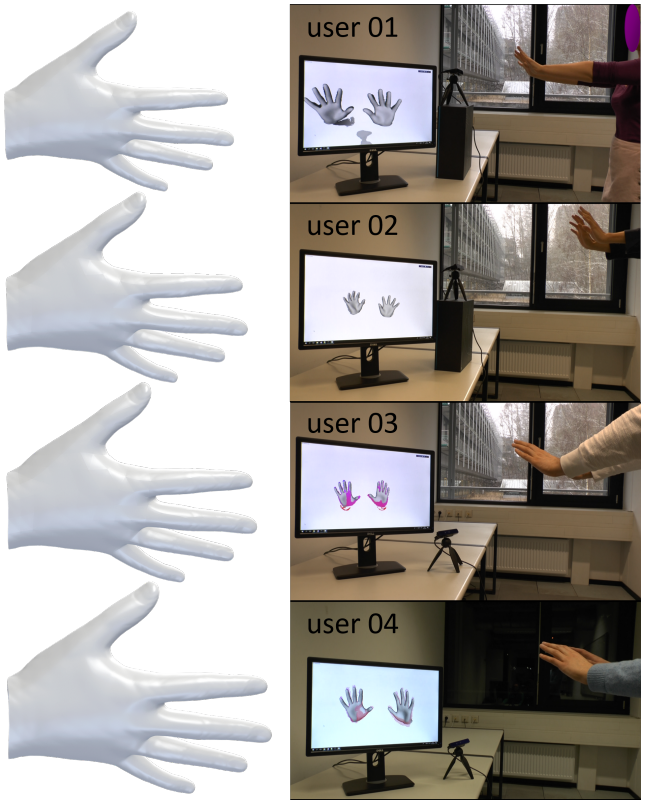


Fig. 10. We present the 3D hand models (left) that we obtained from fitting our model to different users with varying hand shape. From top to bottom we show small to large hand shapes. Note that we show all four hand shapes on the left in the same pose in order to allow for a direct comparison.

7 LIMITATIONS AND DISCUSSION

Although in overall we have demonstrated compelling results for the estimation of hand pose and shape in real-time, there are several points that leave room for further improvements. In terms of computational cost, currently our setup depends on two high-end GPUs, one for the regression network and one for the optimizer. In order to achieve a computationally more light-weight processing pipeline, one could consider lighter neural network architectures, such as CNNs tailored towards mobile platforms (e.g. [Howard et al. 2017]). While our approach better handles complex hand-hand interactions compared to previous real-time approaches, in very challenging situations our method may still struggle. For example, this may happen when the user performs extremely fast hand motions that lead to a severely blurred depth image, or when one of the hands is mostly occluded. In the latter case, temporal jitter may occur due to the insufficient information in the depth image. This could be mitigated by a more involved temporal smoothness term, e.g. stronger smoothing when occlusions are happening, or a temporal network architecture for correspondence prediction. Also, our current temporal smoothness prior may cause a delay in the tracking for large inter-frame motion. To further improve the quality of the results, in the future one can use more elaborate strategies for

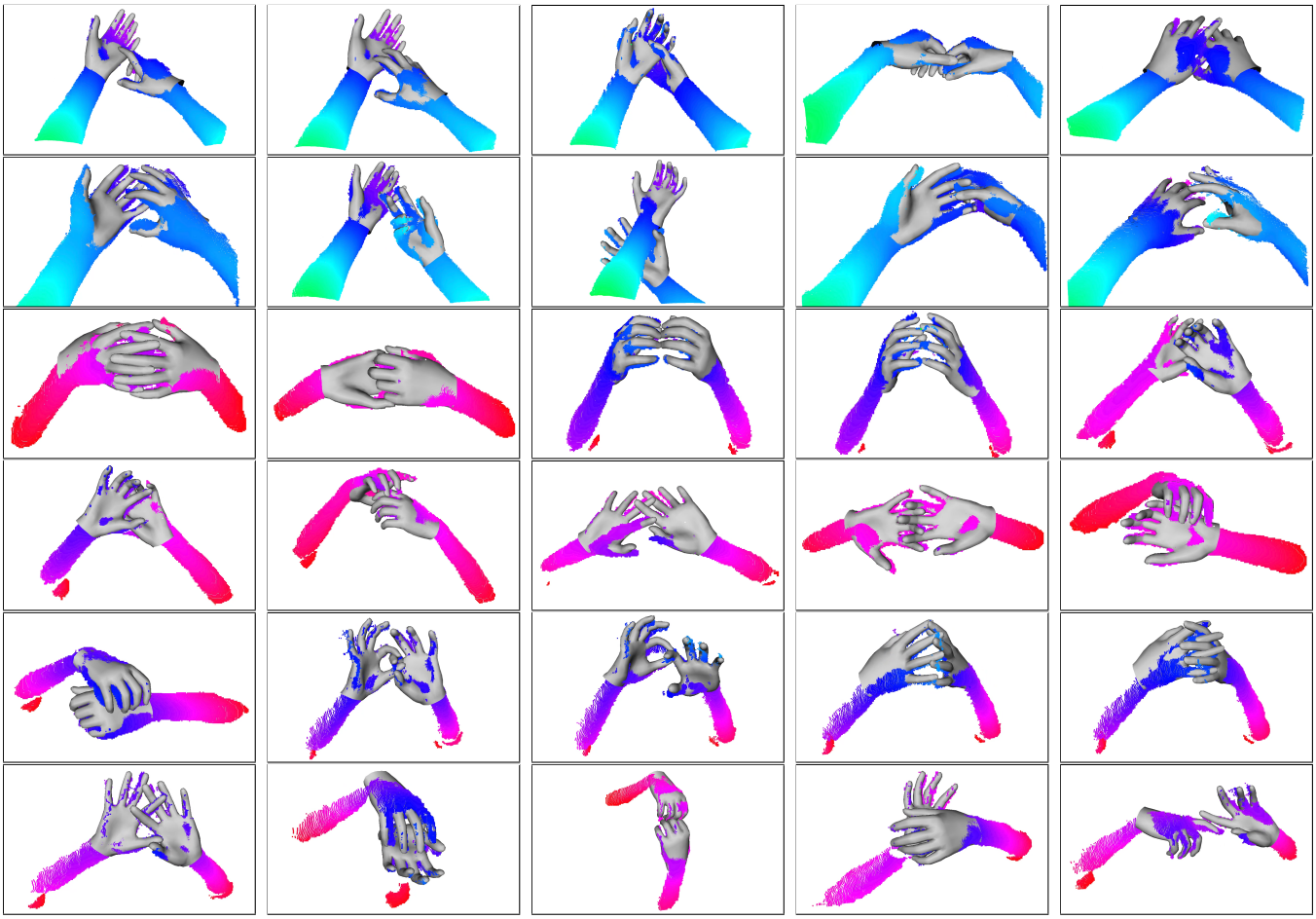


Fig. 11. We show qualitative results for the proposed method. Note that the different colors of the depth image are due to different absolute depth values.

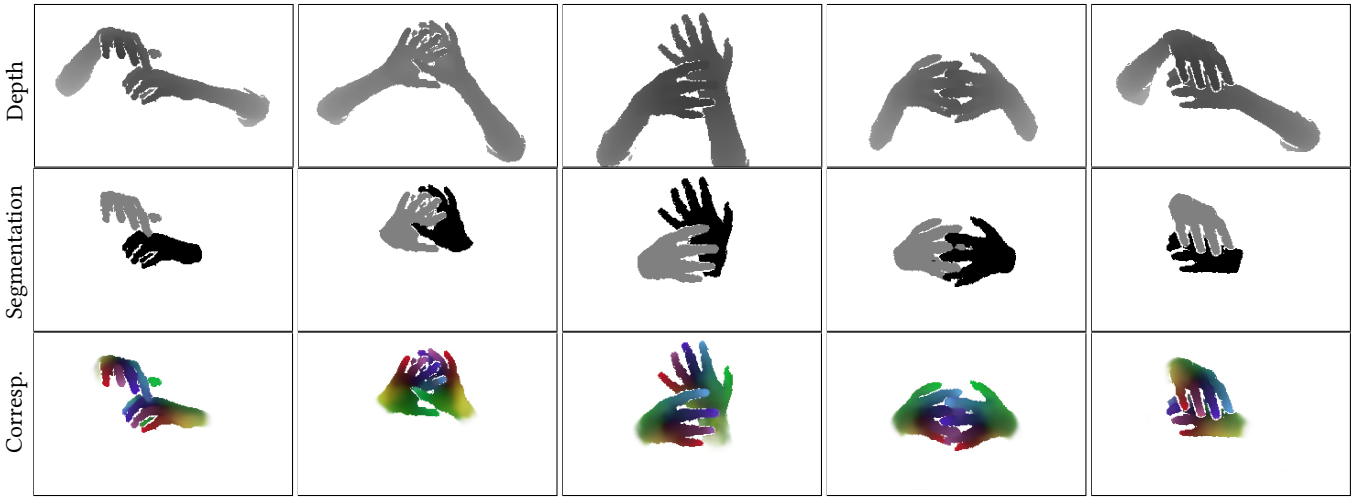


Fig. 12. Given a depth image (top) as input, our CoRN produces accurate segmentation (middle) and dense correspondences (bottom).

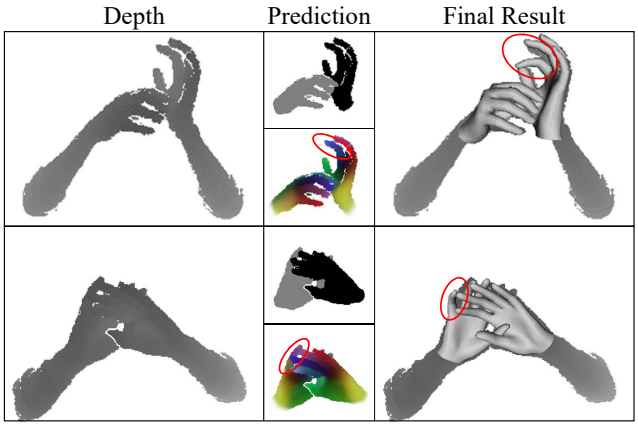


Fig. 13. Erroneous CoRN predictions, e.g. wrongly classified fingers, negatively impact the final tracking result (see Fig. 3 for the reference coloring).

finding correspondences, e.g. by using matching methods that are more advanced than nearest-neighbor search, or by incorporating confidence estimates in the correspondence predictor. Although our data generation scheme has proven successful for training CoRN, some generated images might not be completely realistic. This is due to the LeapMotion tracker's limitations and the hence mandatory distance between the two real hands. In future work, our proposed method could drive the simulation, and the data could be iteratively refined by bootstrapping. While our approach is the only real-time approach that can adjust to user-specific hand shapes, the obtained hand shapes are not as detailed as high-quality laser scans. On the one hand, this is because the MANO model [Romero et al. 2017] is rather coarse with its 778 vertices per hand, and on the other hand the depth image is generally of lower resolution compared to laser scans. One relevant direction for future works is to deal with two hands that manipulate an object. Particular challenges are that one additionally needs to separate the object from the hands, as well as being able to cope with more severe occlusions due to the object. Another point that we leave for future work is to also integrate a physics simulation step directly into the tracker, so that at run-time one can immediately take fine-scale collisions into account. Currently, slight intersections may still happen due to our computationally efficient but coarse collision proxies.

8 CONCLUSION

We have presented a method for real-time pose and shape reconstruction of two interacting hands. The main features that distinguishes our work from previous two-hand tracking approaches is that it combines a wide range of favorable properties, namely it is markerless, relies on a single depth camera, handles collisions, runs in real time with a commodity camera, and adjusts to user-specific hand shapes. This is achieved by combining a neural network for the prediction of correspondences with an energy minimization framework that optimizes for hand pose and shape parameters. For training the correspondence regression network, we have leveraged a physics-based simulation for generating (annotated) synthetic training data that contains physically plausible interactions between two hands. Due to a highly efficient GPU-based implementation of the energy minimization based on a Gauss-Newton optimizer, the approach is real-time capable. We have experimentally shown that our approach achieves results that are qualitatively similar and quantitatively close to the two-hand tracking solution by Tzionas et al. [2016], while at the same time being two orders of magnitude faster. Moreover, we demonstrated that qualitatively our method can handle more complex hand-hand interactions compared to recent state-of-the-art hand trackers.

ACKNOWLEDGMENTS

The authors would like to thank all participants of the live recordings. The work was supported by the ERC Consolidator Grants *4DRepLy* (770784) and *TouchDesign* (772738). Dan Casas was supported by a Marie Curie Individual Fellowship (707326).

REFERENCES

Riza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. DensePose: Dense Human Pose Estimation in the Wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Riza Alp Guler, George Trigeorgis, Epameinondas Antonakos, Patrick Snape, Stefanos Zafeiriou, and Iasonas Kokkinos. 2017. DenseReg: Fully Convolutional Dense Shape Regression In-The-Wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2015. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561* (2015).

Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. 2018. Augmented Skeleton Space Transfer for Depth-Based Hand Pose Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Luca Ballan, Aparna Taneja, Juergen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion Capture of Hands in Action using Discriminative Salient Points. In *European Conference on Computer Vision (ECCV)*.

Michael M Bronstein, Alexander M Bronstein, Ron Kimmel, and Irad Yavneh. 2006. Multigrid multidimensional scaling. *Numerical linear algebra with applications* 13, 2-3 (2006), 149–171.

Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. 2018. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *European Conference on Computer Vision*. Springer, Cham, 1–17.

Chiho Choi, Ayan Sinha, Joon Hee Choi, Sujin Jang, and Karthik Ramani. 2015. A collaborative filtering approach to real-time hand pose estimation. In *Proceedings of the IEEE international conference on computer vision*. 2336–2344.

Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. 2018. Hand PointNet: 3D Hand Pose Estimation Using Point Sets. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D Twigg, and Kenrick Kin. 2018. Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 166.

Markus Höll, Markus Oberweger, Clemens Arth, and Vincent Lepetit. 2018. Efficient Physics-Based Implementation for Realistic Hand-Object Interaction in Virtual Reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces*.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861* (2017).

Chun-Hao Huang, Benjamin Allain, Jean-Sébastien Franco, Nassir Navab, Slobodan Ilic, and Edmond Boyer. 2016. Volumetric 3d tracking by detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3862–3870.

Sameh Khamis, Jonathan Taylor, Jamie Shotton, Cem Keskin, Shahram Izadi, and Andrew Fitzgibbon. 2015. Learning an efficient model of hand shape variation from depth images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2540–2548.

David Kim, Otmar Hilliges, Shahram Izadi, Alex D Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 167–176.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Oscar Koller, O Zargaran, Hermann Ney, and Richard Bowden. 2016. Deep sign: hybrid CNN-HMM for continuous sign language recognition. In *Proceedings of the British*

Machine Vision Conference 2016.

Nikolaos Kyriazis and Antonis Argyros. 2014. Scalable 3d tracking of multiple interacting objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3430–3437.

LeapMotion. 2016. https://developer.leapmotion.com/orion.

Stan Melax, Leonid Keselman, and Sterling Orsten. 2013. Dynamics based 3D skeletal hand tracking. In *Proceedings of Graphics Interface 2013*. Canadian Information Processing Society, 63–70.

Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2018. GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 11. http://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/

Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2017. Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. In *International Conference on Computer Vision (ICCV)*.

Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*. Springer, 483–499.

Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. 2015. Training a feedback loop for hand pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*. 3316–3324.

Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. 2011a. Efficient model-based 3D tracking of hand articulations using Kinect.. In *BMVC*, Vol. 1. 3.

Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. 2011b. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2088–2095.

Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. 2012. Tracking the articulated motion of two strongly interacting hands. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 1862–1869.

Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. 2014. Realtime and Robust Hand Tracking from Depth. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1106–1113.

Edoardo Remelli, Anastasia Tkach, Andrea Tagliasacchi, and Mark Pauly. 2017. Low-Dimensionality Calibration Through Local Anisotropic Scaling for Robust Hand Model Personalization. In *The IEEE International Conference on Computer Vision (ICCV)*.

Grégory Rogez, Maryam Khademi, JS Supančič III, Jose Maria Martinez Montiel, and Deva Ramanan. 2014. 3D hand pose detection in egocentric RGB-D images. In *Workshop at the European Conference on Computer Vision*. Springer, 356–371.

Javier Romero, Dimitrios Tzionas, and Michael J. Black. 2017. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Trans. Graph.* 36, 6, Article 245 (Nov. 2017), 17 pages. https://doi.org/10.1145/3130800.3130883

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. 2015. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. ACM, 3633–3642.

Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mohamed Soliman, Franziska Mueller, Lena Hegemann, Joan Sol Roo, Christian Theobalt, and Jürgen Steimle. 2018. FingerInput: Capturing Expressive Single-Hand Thumb-to-Finger Microgestures. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. ACM, 177–187.

Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. 2018. Cross-Modal Deep Variational Hand Pose Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. 2015. Fast and Robust Hand Tracking Using Detection-Guided Optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 9. http://handtracker.mpi-inf.mpg.de/projects/FastHandTracker/

Srinath Sridhar, Franziska Mueller, Michael Zollhöefer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. 2016. Real-time Joint Tracking of a Hand Manipulating an Object from RGB-D Input. In *European Conference on Computer Vision (ECCV)*. 17. http://handtracker.mpi-inf.mpg.de/projects/RealtimeHO/

Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. 2013. Interactive markerless articulated hand motion tracking using RGB and depth data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2456–2463.

Srinath Sridhar, Helge Rhodin, Hans-Peter Seidel, Antti Oulasvirta, and Christian Theobalt. 2014. Real-time Hand Tracking Using a Sum of Anisotropic Gaussians Model. In *Proceedings of the International Conference on 3D Vision (3DV)*.

James Steven Supančič, Grégory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. 2018. Depth-Based Hand Pose Estimation: Methods, Data, and Challenges. *International Journal of Computer Vision* 126, 11 (01 Nov 2018), 1180–1198. https://doi.org/10.1007/s11263-018-1081-7

Andrea Tagliasacchi, Matthias Schroeder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. 2015. Robust Articulated-ICP for Real-Time Hand Tracking. *Computer Graphics Forum (Symposium on Geometry Processing)* 34, 5 (2015).

David Joseph Tan, Thomas Cashman, Jonathan Taylor, Andrew Fitzgibbon, Daniel Tarlow, Sameh Khamis, Shahram Izadi, and Jamie Shotton. 2016. Fits Like a Glove: Rapid and Reliable Hand Shape Personalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5610–5619.

Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. 2014. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3786–3793.

Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, and Jamie Shotton. 2015. Opening the Black Box: Hierarchical Sampling Optimization for Estimating Human Hand Pose. In *Proc. ICCV*.

Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. 2016. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 143.

Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. 2012. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 103–110.

Jonathan Taylor, Vladimir Tankovich, Danhang Tang, Cem Keskin, David Kim, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. 2017. Articulated Distance Fields for Ultra-fast Tracking of Hands Interacting. *ACM Trans. Graph.* 36, 6, Article 244 (Nov. 2017), 12 pages. https://doi.org/10.1145/3130800.3130853

Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. 2016. Sphere-meshes for real-time hand modeling and tracking. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 222.

Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. 2017. Online Generative Model Personalization for Hand Tracking. *ACM Trans. Graph.* 36, 6, Article 243 (Nov. 2017), 11 pages. https://doi.org/10.1145/3130800.3130830

Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. 2014. Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. *ACM Transactions on Graphics* 33 (August 2014).

Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. 2016. Capturing Hands in Action using Discriminative Salient Points and Physics Simulation. *International Journal of Computer Vision (IJCV)* (2016). http://files.is.tue.mpg.de/dtzionas/Hand-Object-Capture

Mickeal Verschoor, Daniel Lobo, and Miguel A Otaduy. 2018. Soft Hand Simulation for Smooth and Robust Natural Interaction. In *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 183–190.

Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. 2017. Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 680–689.

Chengde Wan, Angela Yao, and Luc Van Gool. 2016. Hand pose estimation from local surface normals. In *European conference on computer vision*. Springer, 554–569.

Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. 2016. Dense Human Body Correspondences Using Convolutional Networks. In *Computer Vision and Pattern Recognition (CVPR)*.

Qi Ye and Tae-Kyun Kim. 2018. Occlusion-aware Hand Pose Estimation Using Hierarchical Mixture Density Network. In *The European Conference on Computer Vision (ECCV)*.

Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, and Tae-Kyun Kim. 2018. Depth-Based 3D Hand Pose Estimation: From Current Achievements to Future Goals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing* 26, 7 (2017), 3142–3155.

Wenping Zhao, Jianjie Zhang, Jianyuan Min, and Jinxiang Chai. 2013. Robust Realtime Physics-based Motion Control for Human Grasping. *ACM Trans. Graph.* 32, 6, Article 207 (Nov. 2013), 12 pages. https://doi.org/10.1145/2508363.2508412

Christian Zimmermann and Thomas Brox. 2017. Learning to Estimate 3D Hand Pose from Single RGB Images.. In *International Conference on Computer Vision (ICCV)*.

A NEURAL NETWORK TRAINING DETAILS

All our networks were trained in Tensorflow using the Adam [Kingma and Ba 2014] optimizer with the default parameter settings. We trained for 450,000 iterations using a batch size of 8. Synthetic and real images were sampled with 50% probability each. With a training time of approximately 20 seconds for 100 iterations, the total training process took 25 hours on an Nvidia Tesla V100 GPU.

We scale the depth values to meters and subtract the mean value of all valid depth pixels. Furthermore, we apply the following image augmentations to our training data, where all augmentation parameters are sampled from a uniform random distribution:

- rotation augmentation with rotation angle $\in [-90, 90]$ degrees,
- translation augmentation in the image plane with offset $\in [-0.25, 0.25] \cdot$ image size, as well as
- scale augmentation with possibly changing aspect ratio in the range of $[1.0, 2.0]$.

Note that all these augmentations are applied on-the-fly while training, i.e. the sampled augmentations for a training sample differ for each epoch, effectively increasing the training set size. In addition to these on-the-fly augmentations, we also mirror all images (and apply the respective procedure to the annotations), which however is performed offline.

B GPU IMPLEMENTATION DETAILS

For our Gauss Newton optimization steps, we compute the non-constant entries of the Jacobian matrix $J \in \mathbb{R}^{8871 \times 122}$ and the residuals $f \in \mathbb{R}^{8871}$ using CUDA kernels on the GPU. We make sure that all threads in the same block compute derivatives for the same energy term. Subsequently, we compute the matrix-matrix and matrix-vector products $J^\top J$ and $J^\top f$ using an efficient implementation in shared memory. For solving the linear system $J^\top J \cdot \delta = J^\top f$, we copy $J^\top J \in \mathbb{R}^{122 \times 122}$ and $J^\top f \in \mathbb{R}^{122}$ to the CPU and employ the preconditioned conjugate gradient (PCG) solver of the Eigen library to obtain the parameter update δ.

C COLLISION ENERGY

The 3D Gaussian collision proxies are coupled with the hand model s.t. the mean $\boldsymbol{\mu}$ depends on the pose and shape parameters β, θ, whereas the standard deviation σ only depends on the shape β. As described by [Sridhar et al. 2014], an integral over a product of two isotropic Gaussians $G_p(x; \mu_p, \sigma_p)$ and $G_q(x; \mu_q, \sigma_q)$ of dimension d is given as:

$$\int_{\mathbb{R}^3} G_p(x) \cdot G_q(x)\, dx = \frac{(2\pi)^{\frac{3}{2}} (\sigma_p^2 \sigma_q^2)^{\frac{3}{2}}}{(\sigma_p^2 + \sigma_q^2)^{\frac{3}{2}}} \exp\left(- \frac{||\boldsymbol{\mu}_p - \boldsymbol{\mu}_q||_2^2}{2(\sigma_p^2 + \sigma_q^2)} \right) \quad (12)$$

This term is differentiable with respect to $\boldsymbol{\mu}$ and σ. Furthermore, the derivatives $\frac{\partial \boldsymbol{\mu}}{\partial \beta}$, $\frac{\partial \boldsymbol{\mu}}{\partial \theta}$, $\frac{\partial \sigma}{\partial \beta}$ can be derived from the hand model. Please note that we do not use the derivative $\frac{\partial E_{\text{coll}}(\beta, \theta)}{\partial \beta}$ in the optimization since this encourages shrinking of the hand models when they are interacting. Instead, the shape β is optimized using all other energy terms and the Gaussian parameters are updated according to β in every optimizer iteration.