Articulated Distance Fields for Ultra-Fast Tracking of Hands Interacting

JONATHAN TAYLOR*, VLADIMIR TANKOVICH*, DANHANG TANG*, CEM KESKIN*, DAVID KIM, PHILIP DAVIDSON, ADARSH KOWDLE, and SHAHRAM IZADI, perceptiveIO

Fig. 1. *Our high speed tracker accurately tracks one and two handed interactions from both head mounted (ego-centric) and front facing depth camera configurations.*

The state of the art in articulated hand tracking has been greatly advanced by hybrid methods that fit a generative hand model to depth data, leveraging both temporally and discriminatively predicted starting poses. In this paradigm, the generative model is used to define an energy function and a local iterative optimization is performed from these starting poses in order to find a "good local minimum" (i.e. a local minimum close to the true pose). Performing this optimization quickly is key to exploring more starting poses, performing more iterations and, crucially, exploiting high frame rates that ensure that temporally predicted starting poses are in the basin of convergence of a good local minimum. At the same time, a detailed and accurate generative model tends to deepen the good local minima and widen their basins of convergence. Recent work, however, has largely had to trade-off such a detailed hand model with one that facilitates such rapid optimization. We present a new implicit model of hand geometry that mostly avoids this compromise and leverage it to build an ultra-fast hybrid hand tracking system. Specifically, we construct an articulated signed distance function that, for any pose, yields a closed form calculation of both the distance to the detailed surface geometry and the necessary derivatives to perform gradient based optimization. There is no need to introduce or update any explicit "correspondences" yielding a simple algorithm that maps well to parallel hardware such as GPUs. As a result, our system can run at extremely high frame rates (e.g. up to 1000fps). Furthermore, we demonstrate how to detect, segment and optimize for two strongly interacting hands, recovering complex interactions at extremely high framerates. In the absence of publicly available datasets of sufficiently high frame rate, we leverage a multiview capture system to create a new 180fps dataset of one and two hands interacting together or with objects.

CCS Concepts: • **Computing methodologies** → *Volumetric models*;

Additional Key Words and Phrases: tracking, volumetric deformation

ACM Reference Format:
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, 4, Article 244 (November 2017), 12 pages. https://doi.org/10.1145/3130800.3130853

1 INTRODUCTION

Fully articulated hand tracking holds the potential to become a first class input mechanism. In order for this promise to be fulfilled, however, the bar for robustness and accuracy must continue to be increased and latency reduced. Further, tracking must not fail when a hand starts interacting with another hand, an object, the body or the scene. In this paper, we extend the state of the art on hand tracking by creating an implicit detailed model of hand geometry, that can be rapidly fit to data using gradient based methods, in order to improve robustness and accuracy while reducing latency. Further, we jointly optimize for two hands strongly interacting and improve upon current hand segmentation techniques to allow the hand to be comfortably close to the body.

The problem that we seek to solve is to estimate the pose θ (i.e. the joint angles and global orientation) of a hand from a depth image $\mathcal{I}$. Classical approaches typically build a generative model and fit this to the image using local optimization from the pose output from the previous frame, and are thus prone to getting trapped in local minima. In contrast, discriminative models typically break the

*Authors equally contributed to this work.

Authors' address: Jonathan Taylor; Vladimir Tankovich; Danhang Tang; Cem Keskin; David Kim; Philip Davidson; Adarsh Kowdle; Shahram Izadi, perceptiveIO.

0730-0301/2017/11-ART244
https://doi.org/10.1145/3130800.3130853

dependency on the previous frame by learning a mapping directly from image to pose, but struggle to generalize to images not represented in the training data. Recently, researchers have elucidated the benefits of combining generative and discriminative methods into a hybrid approach that continuously attempts to "reinitialize" the generative tracker from a discriminative prediction allowing recovery from tracking failures.

In this framework, the generative model is fit to the image data by defining an appropriate energy function that can be locally optimized in the hopes of finding a "good local minimum" (i.e. a local minimum that corresponds to the correct hand pose). The accuracy of the implicit or explicit surface model and the details of the energy formulation determine the availability of such a good local minimum and the size of its basin of convergence. On the other hand, these factors also influence how much time is required to descend such a basin of convergence. As a consequence, the number of starting points one can explore, the number of iterations one can perform and the frame rate one can run at ultimately decides whether the system manages to land in the basin of convergence of a good local minimum or whether the system has "lost track". It is then crucial to choose a model and energy formulation that are both accurate, as to ensure good local minima exist with lower energies than all other poor local minima, and that can be rapidly optimized, as to ensure that the basin of convergence of one of these good local minima can be fully descended. In this paper, we present such a formulation that delivers the robustness of hand tracking at kilohertz speeds without the loss of accuracy that typically comes with using a less detailed model.

To both motivate and differentiate our approach, we first *re-explore* a simplified and somewhat abstract paradigm for fitting a surface $\mathcal{S}(\theta) \subseteq \mathbb{R}^3$ parameterized by a vector of pose parameters $\theta \in \mathbb{R}^J$ to a set of 3D data points $\{x_n\}_{n=1}^N \subseteq \mathbb{R}^3$. The goal is to decrease the squared distance $D(x, \theta)$ from each data point $x \in \mathbb{R}^3$ to the surface by minimizing the energy

$$E_{\text{data}}(\theta) = \sum_{n=1}^{N} D(x_n, \theta)^2 = \sum_{n=1}^{N} \min_{y \in \mathcal{S}(\theta)} \|x_n - y\|^2 \,. \tag{1}$$

Without making any assumptions on the form of $\mathcal{S}(\theta)$, it is not obvious how to effectively perform the inner minimization, let alone optimize the entire energy function.

Most approaches *"pull the min out"* of the sum, as e.g. described in [Taylor et al. 2014], by introducing a set of correspondences $U = \{u_n\}_{n=1}^N \subseteq \Omega$ and a map $S : \Omega \times \mathbb{R}^J \to \mathbb{R}^3$ to the posed surface so that[1]

$$E_{\text{data}}(\theta) = \sum_{n=1}^{N} \min_{y \in \mathcal{S}(\theta)} \|x_n - y\|^2 \tag{2}$$

$$= \sum_{n=1}^{N} \min_{u \in \Omega} \|x_n - S(u, \theta)\|^2 \tag{3}$$

$$= \min_{U} \sum_{n=1}^{N} \|x_n - S(u_n, \theta)\|^2 \,. \tag{4}$$

One can then focus on minimizing the "lifted" energy $\hat{E}_{\text{data}}(\theta, U) = \sum_{n=1}^{N} \|x_n - S(u_n, \theta)\|^2$, after noticing that it strictly bounds the original energy (i.e. $E_{\text{data}}(\theta) \leq \hat{E}_{\text{data}}(\theta, U)$ for any U).

For some models, closed form or fast methods are available to solve for the correspondences U while holding the pose θ fixed making alternation strategies such as ICP appealing [Qian et al. 2014; Tagliasacchi et al. 2015; Tkach et al. 2016]. Such strategies, however, admit the possibility of convergence problems caused by having to take many small axis aligned steps to descend a long non axis aligned energy valley.

To avoid these problems, [Cashman and Fitzgibbon 2013; Khamis et al. 2015; Taylor et al. 2016, 2014] smoothly parameterize $S(u, \theta)$ using a detailed but smooth subdivision surface so that Levenberg optimization can be jointly performed over all $J + 2N$ parameters in θ and U simultaneously. Although, the sparsity in the Gauss-Newton approximation of the Hessian allows one to avoid the cubic complexity one would normally expect to see in solving for a parameter update, exploiting this sparsity on highly parallelizable hardware seems problematic (e.g. there would be many kernel calls with complex dependencies and data shuffling on the GPU).

If instead one is able to *"leave the min in"* (1) and directly calculate $D(x, \theta) = \min_{y \in \mathcal{S}(\theta)} \|x - y\|$ and its derivatives with respect to the pose θ, a small dense non-linear optimization[2] of pose can be performed. In this way the convergence problems that alternation strategies might yield can still be avoided without introducing the complexities of performing a large sparse non-linear joint optimization of pose and parameterized correspondences. For a rigid object, [Fitzgibbon 2003] shows that for any rotation $R \in SO(3)$ and translation $t \in R^3$, $D(x, R, t) = D(R^{-1}(x - t); I, 0)$ and thus a single single dense distance transform of the object in the base pose $I, 0$ can be precomputed and interpolated to provide closed form access to the required distances and derivatives. In [Schmidt et al. 2014], this is extended by decomposing an articulated object into a set of C rigid parts and setting $D(x, \{R_c, t_c\}_{c=1}^C) = \min_c D_c(x, R_c, t_c)$, but artifacts and creases will arise at articulation points where these distance functions interact. Similarly, [Qian et al. 2014; Tagliasacchi et al. 2015; Tkach et al. 2016] decompose explicit surface models into a set of C primitives such as spheres or cylinders where $D_c(x, \theta)$ and its derivatives are easy to compute. This again arises in bumps at primitive intersections (i.e. a discontinuity of normals but not surface geometry), and a large number of such primitives need to be used in order to increase detail and decrease the intensity of the bumps in the model.

The primary technical contribution of this paper is to show how to create a detailed and largely smooth implicit model of an articulated surface so that one can *"leave the min in"* (1) as to enable its fast optimization and ultra-fast surface registration. Specifically, we show how to volumetrically deform a single dense signed distance field (SDF) using a skinned tetrahedral mesh (see Fig. 2). The SDF

[1] Note that θ, Ω and $S(u; \theta)$ have been left intentionally abstract. They could respectively be, for example, a vector of blend shape coefficients, the set of vertex indices of a corresponding blend shape model and a function to evaluate the 3D position of an indexed vertex using the blend shape coefficients.

[2] Note that the sparsity of a non-linear optimization refers to the sparsity of the Gauss-Newton Hessian approximations typical non-linear optimizers, such as the Levenberg algorithm, employ.

smoothly captures geometric detail using tricubic interpolation of a dense grid, while the skinned tetrahedral mesh allows this detail to be transformed into different poses. Although the tetrahedral mesh warp can introduce creases (e.g. discontinuous first derivatives), these only occur at articulation points and the severity of these bumps can be addressed by densifying the mesh in these areas. In contrast to [Schmidt et al. 2014] where a strict piecewise rigid assumption is made, which may be quite unnatural for a human hand, we can apply arbitrary mesh skinning techniques to deform a *single* SDF. Indeed, we believe that the decoupling of the deformation function (through the skinned tetrahedral mesh) and the representation of detail (through the SDF) is a key feature of this representation allowing future work to potentially "fuse" static geometry, such as a watch or a ring, into the SDF.

More crucially though, this representation allows us to use a detailed model of the hand while enabling a highly parallelizable algorithm amenable to execution on modern day GPUs. Leveraging recent work on efficient high frame rate active-stereo depth estimation [Fanello et al. 2017a,b] we use depth sensors that are capable of producing 1280×1024 depth maps at a maximum of 210fps. Although, we only have access to depth cameras that can run at a maximum of 210fps and ground truth sequences captured at 180fps our tracker can run at 1000fps on precaptured data using an NVIDIA Titan X. Despite the likely validity of the pose from the previous frame in such scenarios, we also leverage a coarse reinitializer to maintain robustness to the occasional tracking failures.

In addition, we demonstrate two handed interaction, by first detecting and segmenting the left and right hands and formulating an appropriate energy function that can recover from slight left/right hand mis-segmentations. As shown in the supplementary video, we demonstrate both one and two handed tracking, recovering complex interactions, at high frame rates and low latencies. Finally, we capture a new dataset that is significantly higher frame rate than what is publicly available, contains difficult two handed interactions and hand with object interactions.[3]

2 RELATED WORK

2.1 Hand Detection

As hands are very complex articulated objects, it is common to rely either on simple color based heuristics or existing skeleton trackers for hand detection. Color based detectors typically look for skin colored objects in the scene [de La Gorce et al. 2011; Oikonomidis et al. 2011; Sridhar et al. 2013] or a wristband [Tagliasacchi et al. 2015; Tkach et al. 2016], and typically assume a single hand in the scene as there is no mechanism to distinguish between left and right hands. Depth based detectors in [Sharp et al. 2015; Tang et al. 2015; Taylor et al. 2016] depend on Kinect's skeleton tracker to estimate a rough location for potentially multiple hands. Such trackers themselves often rely on Randomized Decision Forests (RDF) for body part classification [Shotton et al. 2011], and this method is also used in [Tompson et al. 2014] for binary per-pixel classification into hand and background classes. For our much more complex setting that involves strongly interacting hands, we propose a Convolutional Neural Network (CNN) based approach and compare our method to RDFs with novel features. As this is essentially a complex semantic segmentation task with very limited computational budget, we rely on a kind of Fully Convolutional Network (FCN) [Long et al. 2014] that is suitable for fast per-pixel classification.

[3]Note that despite the inclusion of object interactions in the dataset, our algorithm is not designed to work with object interactions unless a perfect segmentation can be provided.

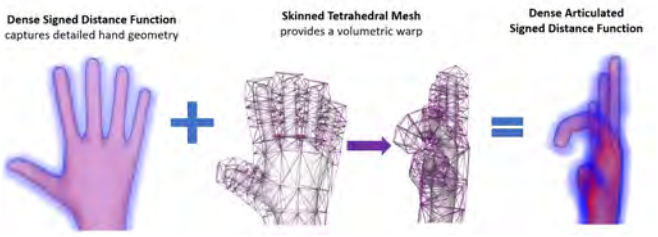

Fig. 2. *We use a skinned tetrahedral mesh (center) to warp a precomputed signed distance field (left) into a new pose (right) to create an articulated signed distance field. In the volumetric visualizations of the signed distance functions (left and right), voxels are shaded red or blue to indicate whether they contain negative (i.e. inside) or positive signed distances (i.e. outside).*

2.2 Learning Based Hand Pose Estimation

Most methods train a machine learning method such as an RDF [Keskin et al. 2012; Sun et al. 2015; Tang et al. 2016, 2015], Convolutional Neural Network (CNN) [Ge et al. 2016; Mueller et al. 2017; Oberweger et al. 2015a,b; Wan et al. 2017; Ye et al. 2016] or their combination [Tompson et al. 2014] to produce a mapping from input images to hand pose. These methods typically do not depend on the previous frame, are fast and generally bypass incorporating prior knowledge of hand kinematics and/or surface geometry. A refinement step [Oberweger et al. 2015a] or inverse kinematics [Tompson et al. 2014] can follow the initial estimation phase but incorporation of prior knowledge of hand kinematics and surface geometry can be avoided. Recent work in [Sinha et al. 2016] employs a joint matrix factorization and completion method to determine hand pose.

2.3 Hybrid Hand Pose Estimation

Despite considerable advances in learning based hand pose estimation, systems that employ generative models of explicit hand kinematics and surface geometry and fit these models to depth data using local optimization have produced the most compelling systems. Even then, these methods are typically "hybrid" methods that employ a discriminative predictor to "reinitialize" the system when it loses track. Generally this is done by proposing new starting poses, augmenting the pose from the previous frame, that can be refined for a final selection by an energy or score function [Melax et al. 2013]. Ultimately though, the line between generative and discriminative models continues to blur as more prior knowledge of hand kinematics [Zhou et al. 2016] and surface geometry get incorporated into learning based methods.

Conceptually, the most holistic generative optimization strategy is to simply render the generative model and compare to the image observations [de La Gorce et al. 2011; Oikonomidis et al. 2011; Sharp et al. 2015; Tan et al. 2016]. In practice, this results in an energy that

is difficult to optimize and thus proxy formulations, such as a the "sum of Gaussians" model [Sridhar et al. 2013], that are easier to optimize are employed.

The point cloud registration energy (1) explored in the introduction is perhaps the most popular such proxy.[4] Methods that "pull the min out" of (1) and perform alternation over correspondences and pose, leveraging cheap optimal correspondences calculations are [Melax et al. 2013; Tagliasacchi et al. 2015; Tkach et al. 2016]. Other work that "pulls the min out" but instead performs joint non-linear optimization of correspondences and pose simultaneously [Cashman and Fitzgibbon 2013; Khamis et al. 2015; Taylor et al. 2016, 2014]. Work that "leaves the min in" include [Fitzgibbon 2003] for rigid models and [Schmidt et al. 2014] for articulated structures. Although the introduction alludes to the relative pros and cons of "pull/leave the min out/in" strategies, it is difficult to draw strong conclusions from hand tracking systems that contain so many other confounding factors. The work of [Hong and Fitzgibbon 2015], however, performs a controlled quantitative analysis of similar strategies in the context of matrix factorization.

The value of a detailed and accurate generative model was shown for monocular hand tracking in [de La Gorce et al. 2011] and for multiview in [Ballan et al. 2012]. Indeed, recent work has demonstrated the importance of such shape adaptation [Khamis et al. 2015; Remelli et al. 2017; Tan et al. 2016; Taylor et al. 2014; Tkach et al. 2017].

An important instance of exploiting an SDF for 3D surface alignment is monocular depth based reconstruction [Izadi et al. 2011]. But when this has been extended to the deformable case [Dou et al. 2016, 2015; Innmann et al. 2016; Newcombe et al. 2015], the representation is switched for alignment by first extracting a surface mesh from the SDF.

The most relevant work to ours is that of [Schmidt et al. 2014] who create an articulated distance function by taking the min over a set of rigidly moving and kinematically consistent signed distance functions (see Sec. 1).

While most of these methods can be easily extended to track two hands in isolation, few published approaches allow for interaction between the hands [Oikonomidis et al. 2012; Tzionas et al. 2016].

3 METHOD

We parameterize the pose $\theta \in \mathbb{R}^J$ of the hand using the standard four joint articulations of each of the five fingers, two degrees of freedom at the wrist and six degrees of freedom for global orientation. We over-parameterize 3-DOF global rotation using a quaternion so that θ is $J = 29$ dimensional. In the following, we show how to fit this model, formulated as an articulated SDF to the depth data.

3.1 Hand Detection and Segmentation

Given a depth image $\mathcal{I}$, either a CNN or a RDF (see below) is used to produce probability maps $P^{\text{left}} \in [0,1]^{W \times H}$, $P^{\text{right}} \in [0,1]^{W \times H}$ and $P^{\text{bg}} \in [0,1]^{W \times H}$ that encodes for each pixel, the probability of that pixel belonging to the left hand, the right hand and the background respectively. To detect the right hand we temporarily

[4]Note that many other terms are typically added to (1) to regularize under constrained poses, enforce joint limits, etc.

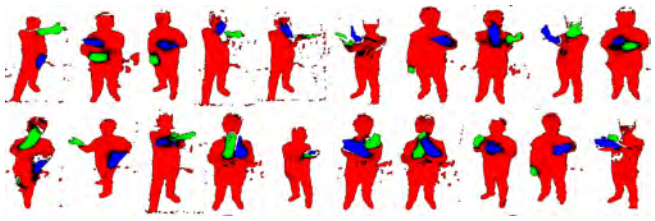

Fig. 3. *Some examples of left (blue), right (green) and background (red) segmentation using our CNN. Black pixels are not sufficiently confident in their decisions.*

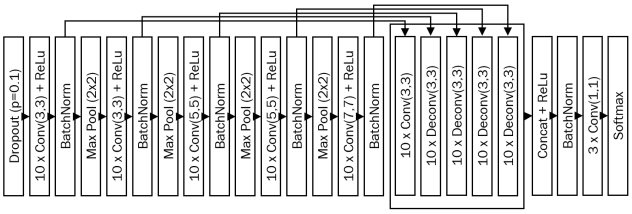

Fig. 4. *The FCN used to segment left and right hands from an input depth image.*

set all values of the probability map P^{right} to zero that are below a high value $\eta^{\text{high}} \in [0,1]$, convolve the output with a large bandwidth Gaussian filter, and then use the location of the maximum value. We then remove outliers from the original segmentation P^{right} by setting to zero the value of any pixels whose probability is less than $\eta^{\text{low}} \in [0, \eta^{\text{high}}]$ or whose 3D location is not contained in a sphere of radius $r^{sphere} \in \mathbb{R}$ around the hand detection. The latter ensures that pixels far from the most prominent hand (e.g. pixels on other peoples hands in the background) do not contaminate the segmentation while the former allows the machine learning method to discard nearby pixels that are recognized as not belonging to the hand (e.g. pixels on the user's chest). The pixels that pass this test are then backprojected into 3D space using the camera parameters to form a point cloud $\{x_n\}_{n=1}^N \subseteq \mathbb{R}^3$ as to define (1).

3.1.1 CNN based Segmenter. To produce these probability we maps, we typically use a CNN that takes a lower resolution version of the depth image and applies a series of transformations to produce the map. To ensure translational equivariance, we refrain from using fully connected layers that discard locality information, and only rely on convolutions, essentially forming a Fully Convolutional Network (FCN)[Long et al. 2014]. FCNs combine coarse, high layer information with fine, low layer information using a series of convolution, pooling and deconvolution operations. We further enhance this network with batch normalization layers placed after each convolution and deconvolution in order to teach the network to preprocess the depth images automatically. We employ dropout ($p = 0.1$) at the input layer to reduce overfitting.

The actual size of the local neighborhood used to infer class labels is determined by the receptive fields of the neurons in the final layer. Unfortunately, a local region around an isolated hand in a depth image may not contain enough information to differentiate between

hand pixels from the left and right hands. We therefore require a large receptive fields large enough to "see" the arms and the body as to disambiguate these pixels. To accommodate this, we convolve pooled feature maps with increasingly large kernels. The entire CNN is depicted in Fig. 4 and Fig. 3 shows a few examples on real test images.

3.1.2 RDF based Segmenter. As a baseline, we also train an RDF classifier to produce the segmentation probability maps. RDFs typically employ depth and translation invariant features that are particularly suitable for processing depth images, which threshold the depth difference of two pixels at depth-normalized offsets around the central pixel [Keskin et al. 2012; Sharp et al. 2015; Shotton et al. 2011; Tang et al. 2015; Tompson et al. 2014]. For each pixel p at coordinate (u, v) in the depth image $\mathcal{I}$, a split node in the tree evaluates the function:

$$\mathcal{I}\left(u+\frac{\Delta u_1}{\Gamma}, v+\frac{\Delta v_1}{\Gamma}\right) - \mathcal{I}\left(u+\frac{\Delta u_2}{\Gamma}, v+\frac{\Delta v_2}{\Gamma}\right) > \tau \tag{5}$$

where Γ is $\mathcal{I}(u, v)$, Δu_i and Δv_i are the two offsets and τ is the threshold for that split node. This feature is well suited for most tasks, but they are inefficient in handling cases where the classification task is invariant to rotations, e.g. a single extended hand. To enhance the feature pool for such subtasks, we introduce a new rotationally invariant family of features. To this end, let $R(u, v, r, \mathcal{I})$ be the sum of the K depth pixels found on a circle in the image of depth-scaled radius r around (u, v). These features then measure the difference

$$\frac{R(u, v, r_1, \mathcal{I})}{K} - \frac{R(u, v, r_2, \mathcal{I})}{K} > \tau \tag{6}$$

of two average depths corresponding to two such co–centric rings. As $R(u, v, r, \mathcal{I})$ gets costlier for larger rings, in practice we approximate this value using a fixed number of points k as

$$R(u, v, r, \mathcal{I}) = \sum_{i=1}^{k} \mathcal{I}\left(u + \frac{r\cos(i2\pi/k)}{\Gamma}, v + \frac{r\sin(i2\pi/k)}{\Gamma}\right). \tag{7}$$

We also define a unary version of this feature as

$$\frac{R(u, v, r_1, \mathcal{I})}{k} - \Gamma > \tau \tag{8}$$

using only a single ring. At training time, we sample from a pool of binary and unary rotationally dependent and invariant features using a fixed ratio, found using grid search. For each considered feature, we uniformly sample multiple τ values from a fixed range, and pick the one that maximizes the information gain.

3.2 Deformable Signed Distance Function

In this section, we formulate a function that, given a pose θ gives the signed distance $D(x, \theta)$ to the posed surface implicitly defined by our hand model. The basic idea is to use a linear blend skinned tetrahedral mesh to deform a precomputed signed distance field into an arbitrary pose (see Fig. 2).[5] To this end, we assume that we are given access to a dense grid of signed distances (see Sec. 5.3) in the base pose θ_0 and use tricubic interpolation [Lekien and Marsden 2005] to define the signed distance $D(x; \theta_0) = \tilde{D}(x) \in \mathbb{R}$ to the surface for any point $x \in \mathbb{R}^3$. Note that tricubic interpolation also gives us access to smooth first and second order derivatives with respect to x.

[5]Note that our LBS deformation function includes a single fixed scale hyperparameter that we can manually modify to account for each users hand size.

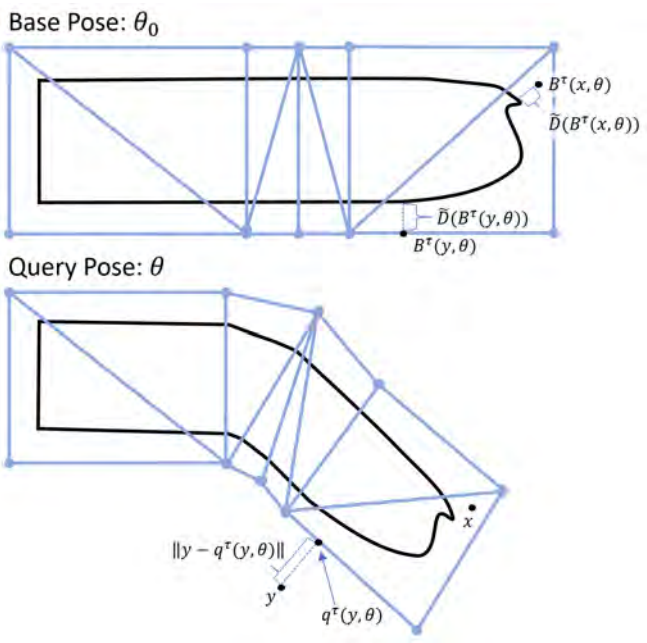

Fig. 5. *An illustration of how a tetrahedral mesh could deform a signed distance field $\tilde{D}(x)$ implicitly encoding a finger. In this 2D example, the cross section of the end of a finger (black curve) in the base pose θ_0 (top) represents the zero crossings $\{p : \tilde{D}(p) = 0\}$ of the signed distance field. The finger is contained inside a triangular mesh (the 2D equivalent of a tetrahedral mesh). When the mesh is deformed (bottom), each triangle defines an affine warp between the base pose and the query pose θ. Using the inverse warps we can implicitly define a signed distance field $D(x, \theta)$ as follows. For a query point x that falls inside the deformed mesh, the triangle τ's affine inverse warp sends the query point to $B^\tau(x, \theta)$ where the distance to the implicitly encoded surface can be queried as $\tilde{D}(B^\tau(x, \theta))$. For a point y that falls* outside *the deformed tetrahedral mesh (bottom), the closest point $q^\tau(y, \theta)$ is used to warp back to the base pose to query its distance to the implicitly encoded surface. To this value, the distance from the query point y to the closest point is added. These are the two cases defined in* (3.2). *Note that the zero crossings $\{x : D(x, \theta) = 0\}$ implicitly defines a deformed finger surface (black curve in the bottom). Also note that the mesh is only used to deform joints, allowing an arbitrary amount of "static detail", such as the finger nail, to be encoded in the original sdf.*

In order to define the signed distance field $D(x, \theta)$ for an arbitrary pose θ, we rely on a tetrahedral volumetric mesh whose vertices are deformed via linear blend skinning (see Sec. 5.3). For any tetrahedron τ, let $V^\tau(\theta) \in \mathbb{R}^{3\times4}$ be a matrix with the positions of the tetrahedron τ's four vertices in pose θ stored in its columns. Let $\hat{\beta}^\tau(x, \theta) \in \mathbb{R}^4$ be the barycentric coordinate of the closest point in the tetrahedron τ under pose θ. That is

$$\hat{\beta}^\tau(x, \theta) = \underset{\beta \in \mathcal{B}}{\arg\min} \|x - V^\tau(\theta)\beta\| \tag{9}$$

where $\mathcal{B} = \{\beta \in [0,1]^4 : \beta^\top \beta = 1\}$. The closest point can then be reconstructed as

$$q^\tau(x,\theta) = V^\tau(\theta)\hat{\beta}^\tau(x,\theta)\ . \tag{10}$$

Similarly, a corresponding point in the base pose can be reconstructed as

$$B^\tau(x,\theta) = V^\tau(\theta_0)\hat{\beta}^\tau(x,\theta)\ . \tag{11}$$

We can use this point in the base pose to perform queries on our precomputed signed distance function $\tilde{D}(\cdot)$. It is possible, however, for the deformation of the tetrahedral mesh to cause the query point x to actually fall in multiple overlapping tetrahedra so let

$$\mathcal{T}(x,\theta) = \{\tau : q^\tau(x,\theta) = x\} \tag{12}$$

be the set of such tetrahedra that contain x. We can then choose a single tetrahedron

$$\tau^*(x,\theta) = \begin{cases} \arg\min_{\tau\in\mathcal{T}(x,\theta)} |\tilde{D}(B^\tau(x,\theta))|, & \mathcal{T}(x,\theta) \neq \emptyset \\ \arg\min_\tau \|x - q^\tau(x,\theta)\|, & \mathcal{T}(x,\theta) = \emptyset\ . \end{cases}$$

that will be used to warp this point back into the base pose. The first case selects the tetrahedron that will return the minimum absolute distance to the unposed surface when it is warped back to the base pose and evaluated in the precomputed signed distance function. The second case deals with the case where the query point does not land in a single tetrahedron by simply selecting the tetrahedron that the point is closest to. We can then define the distance to the surface to be

$$D(x,\theta) = \|x - q^{\tau^*(x,\theta)}(x,\theta)\| + \tilde{D}(B^{\tau^*(x,\theta)}(x,\theta)) \tag{13}$$

where the first term measures the distance to the closest point in the selected tetrahedron and the second term warps that closest point back to the base pose to evaluate the signed distance to evaluate its distance to the surface (see Fig. 5).

Note that computing (3.2) is not necessarily the most efficient operation as the simplest algorithm will have to, for each data point, test whether it is contained in every tetrahedron. Nonethless, such an implementation is highly parallel and very fast to execute on a GPU.

Note also that this definition divides the space into a discrete set of cells as $\tau^*(x,\theta)$ jumps from one tetrahedron to another. When x lands in at least one tetrahedra, an affine warp defined by the selected tetrahedron maps the space in the current pose back into the base pose for sdf evaluation. When x lands outside the tetrahedral mesh, the closest tetrahedron is selected and the affine warp is similarly used for sdf evaluation using the closest point on that tetrahedrons boundary. In this case, however, the distance from x to that closest point on the tetrahedron boundary is added. Although the function is continuous everywhere and differentiable within each cell, there are bumps in the function as $\tau^*(x,\theta)$ varies. Nonetheless, the intensity of those bumps are directly related to how different the transformation between neighbouring cells is and so one can add more tetrahedra near the joints to lessen this intensity. Crucially though, in rigid components of the model, neighbouring cells will apply the same transformation and thus it is not beneficial to add more tetrahedra there (e.g. on the back of the hand). Indeed, a key advantage of this formulation is that the representation of static detail occurs in the reference volume and is thus largely independent of the amount of tetrahedra used to warp this detail into different poses. Much like a triangular mesh, more tetrahedra can be added to "smooth" out bumps around joints, but unlike a triangular mesh more tetrahedra *do not* need to be added to add more static detail to the model.

3.3 Energy

In this section, we define an energy that evaluates the goodness of a particular pose θ allowing us to calculate derivatives. In particular, the energy that we seek to minimize is

$$\begin{aligned} E(\theta) =& E_{\text{data}}(\theta) + \lambda_{\text{normal}} E_{\text{normal}}(\theta) \\ &+ \lambda_{\text{prior}} E_{\text{prior}}(\theta) + \lambda_{\text{limit}} E_{\text{limit}}(\theta) \end{aligned} \tag{14}$$

where the date term arrives from simply plugging (13) into (1). Note that this energy does not penalize self or background intersection which allows fingers to self-intersect or leak into the background when left otherwise unconstrained by the data and priors. We hope that future work will resolve these limitations by formulating penalties that properly leverage our articulated SDF formulation.

3.3.1 Normal Term. The normal term $E_{\text{normal}}(\theta)$ leverages the fact that the gradient of the articulated signed distance function with respect to position will point away from the surface, so we estimate the surface normal as

$$\Upsilon(x,\theta) = \frac{\nabla_x D(x,\theta)}{\|\nabla_x D(x,\theta)\|}\ . \tag{15}$$

We thus encourage the surface normal to agree with the estimated normal $x_n^\perp$ at data point n by defining

$$E_{\text{normal}}(\theta) = \sum_{n=1}^{N} \|\Upsilon(x,\theta) - x_n^\perp\|^2\ . \tag{16}$$

Note that although the current formulation leaves discontinuities in $E_{\text{normal}}(\theta)$ as $\tau^*(x,\theta)$ varies, the jumps are generally not substantial. Nonetheless, we leave it as future work to force continuity of this term across cells, either by interpolating a fake normal field or using a higher order interpolant (e.g. quadratic or cubic) in the tetrahedral warp.

3.3.2 Pose Prior. Similar to [Tan et al. 2016; Taylor et al. 2016] we use a multivariate Gaussian with mean $\mu \in \mathbb{R}^{22}$ and covariance matrix $\Sigma \in \mathbb{R}^{22\times 22}$ to provide constraints under occlusion or otherwise under–constrained scenarios. This is embedded in the pose prior term

$$E_{\text{prior}}(\theta) = (\psi(\theta) - \mu)^\top \Sigma^{-1} (\psi(\theta) - \mu) \tag{17}$$

where $\psi(\theta) \in \mathbb{R}^{22}$ extracts the 22 parameters of the pose θ corresponding to joint articulations (i.e. not global orientation).

3.3.3 Joint Limits. Similar to [Tan et al. 2016; Taylor et al. 2016] we encode a set of joint limits $\zeta^{\text{low}} \in \mathbb{R}^{22}$ and $\zeta^{\text{high}} \in \mathbb{R}^{22}$ and penalize violations of these limits through

$$\begin{aligned} E_{\text{limit}}(\theta) =& \sum_{i=1}^{22} \mathcal{I}(\psi_i(\theta) < \zeta_i^{\text{low}})(\psi_i(\theta) - \zeta_i^{\text{low}})^2 \\ &+ \sum_{i=1}^{22} \mathcal{I}(\psi_i(\theta) > \zeta_i^{\text{high}})(\psi_i(\theta) - \zeta_i^{\text{high}})^2 \end{aligned} \tag{18}$$

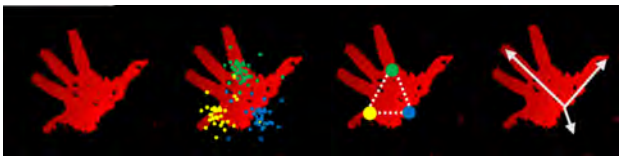

Fig. 6. *6 DOF estimation of the hand pose by the reinitializer. The per–pixel votes for each joint are aggregated and a mode finding method is used to detect the joint locations, which are used to form the global translation and rotation for the reinitialization poses.*

where $\mathcal{I}()$ is the indicator function.

3.4 Energy Optimization

In order to minimize (14) we perform Levenberg optimization from a set of $K+1$ starting poses, one of which comes from the previous frame and the rest of which come from reinitialization. We then report the resulting pose with the lowest energy.

3.4.1 Reinitialization. Our hand segmenter outputs a segmented depth image $\mathcal{R}$ per hand, which is used by the reinitializer to produce a set of pose estimates $\{\theta_k^R\}_{k=1}^K$ that are used as starting poses to locally optimize from in the energy. We tackle this by using an RDF to estimate the 6DOF hand pose via a per–pixel offset regression [Girshick et al. 2011]. Specifically, we use the RDF to locate three joints on the palm (assumed to be planar) in the world space. We pick the wrist joint q_w, the base of the index metacarpophalangeal (MCP) joint q_i and the base of the pinky MCP q_p, to estimate the 6 DOF pose. We convert these three anchor points into a reinitialization pose by setting the global translation to q_w, and compute the global orientation by finding the orientation of the 3D triangle the anchor points define. We then sample a set of finger poses randomly from the pose prior (see Sec. 3.3.2) to produce the pose estimates $\{\theta_k^R\}_{k=1}^K$.

To locate the anchor points, the RDF is evaluated for each pixel p in $\mathcal{R}$ to produce a single vote for the 3D offset of each joint relative to p. The RDF relies on the feature family given in (5) for this task, as unlike classification, offset regression is rotationally dependent. The trees are trained with a regression objective to minimize the vote variance in the leaves. At test time, each pixel votes for all the joints, which are aggregated separately to form a vote distribution per joint. The modes of these distributions are selected as final estimates of for the anchor points, which are found via mean shift [Comaniciu and Meer 2002] (see Fig. 6).

3.4.2 Levenberg Minimization. To perform local optimization of (14) we first rewrite it, as to perform Levenberg optimization, in a sum of squares form

$$E(\theta) = r(\theta)^\top r(\theta) \tag{19}$$

where $r(\theta) \in \mathbb{R}^D$ is a vector of D residuals. We calculate the Jacobian $J(\theta) \in \mathbb{R}^{D\times 29}$ and perform a Levenberg update as

$$\theta \leftarrow \theta + (J(\theta)^\top J(\theta) + \gamma I_{29\times 29})^{-1} J(\theta)^\top r(\theta) \tag{20}$$

where γ is the standard Levenberg dampener, raised or lowered as steps fail or succeed to lower the energy.

4 EXTENSION TO TWO HANDS

In order to track two hands we jointly optimize over the poses $\Theta = \{\theta^{\text{left}}, \theta^{\text{right}}\}$ and a set of right handed assignments $\Upsilon = \{\eta_n\}_{n=1}^N \subseteq \{0,1\}$ which implicitly define a set of left handed assignments $\Gamma(\Upsilon) = \{1-\eta_n\}_{n=1}^N$. To this end, we reformulate (14) as

$$\begin{aligned} E(\theta;\Upsilon) =& E_{\text{data}}(\theta;\Upsilon) + \lambda_{\text{normal}} E_{\text{normal}}(\theta;\Upsilon) \\ &+ \lambda_{\text{prior}} E_{\text{prior}}(\theta) + \lambda_{\text{limit}} E_{\text{limit}}(\theta) \end{aligned} \tag{21}$$

where the data term is adapted to

$$E_{\text{data}}(\theta;\Upsilon) = \sum_{n=1}^N \eta_n D(x_n;\theta) \tag{22}$$

and likewise for the normal term $E_{\text{normal}}(\theta;\Upsilon)$. We then formulate the full energy to be optimized as

$$\begin{aligned} \tilde{E}(\Theta) =& E(\theta^{\text{left}};\Gamma(\Upsilon)) + E(\theta^{\text{right}};\Upsilon) \\ &+ \lambda_{\text{assign}} \sum_{n=1}^N \left(\eta_n \gamma_n^{\text{right}} + (1-\eta_n)\gamma_n^{\text{left}}\right) \end{aligned} \tag{23}$$

where are γ_n^{left} and γ_n^{right} are penalties output from the segmentation method for assigning data point n to the right and the left hand pose respectively. To optimize this function, we perform alternation between Θ and Υ, updating the former with Levenberg updates and the latter by discretely considering whether assigning the data point to the left or right hand will lower the energy.

5 IMPLEMENTATION DETAILS

5.1 Segmenter Training

To train the CNN and RDF based segmenters explained in Sec. 3.1, we captured large 100K instance dataset for both the ego-centric and front facing camera scenarios from multiple people. The subjects wore colored gloves so that automatic labelling of depth pixels could be performed using a calibrated color camera. A grid search over parameters was done using leave–one–subject–out evaluation in order to optimize for generalization capability. The class imbalance problem was dealt with by simply undersampling the background pixels in each image. We did not preprocess the depth images and in both cases, we aimed for the segmentation to take less than 0.5ms.

The CNN was trained using stochastic gradient descent with an initial learning rate of 0.1, weight decay 0.001 and momentum 0.9. The learning rate and the weight decay were gradually and adaptively decreased during training, which took 300 epochs. Using the entirety of each dataset for parameter estimation proved to be intractable as training took more than 5 days on a Titan X. Therefore, we used every fifth image to reduce training times. We also experimented with the number of pooling operations, kernel sizes and number of filters and picked the network in Fig. 4. The balanced accuracy (i.e. average recall) of the final network is 85.5%.

We implemented a GPU based trainer for RDFs and used bagging to ensure variance amongst trees. Tests revealed that four trees of depth 19 gave the best results within our computational budget. Optimal forests preferred a 50%-50% split between rotationally dependent and invariant features, where nodes in the higher levels typically chose the latter type. Data augmentation was done by assigning random rotations, horizontal flips and scale variations to

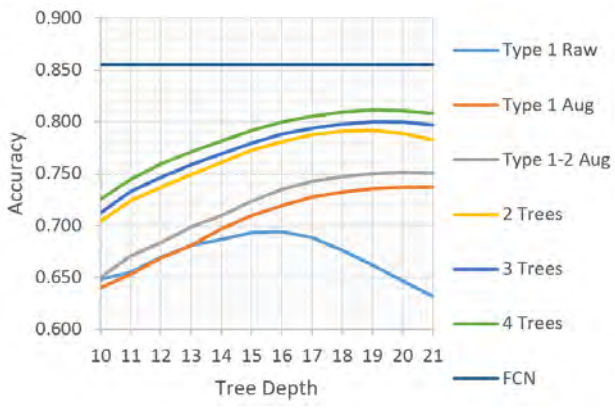

Fig. 7. *Balanced accuracy for leave–one–subject–out tests using i) Type 1 Raw: single tree, rotationally dependent features without data augmentation, ii) Type 1 Aug: same with augmentation, iii) Type 1-2 Aug: both types of features with augmentation, iv) 2 Trees: sames as iii), 2 trees, v) 3 Trees: sames as iii), 3 trees, vi) 4 Trees: sames as iii), 4 trees, and vii)FCN: our CNN based model, which outperforms RDFs.*

each example. Fig. 7 shows the balanced accuracy averaged over left–out subjects, with respect to tree depth, forest size and feature types used. Evidently, data augmentation prevents early overfitting and pushes the optimal depth from 16 to 19. The best forest achieved an accuracy of 81.1%.

5.2 Regression Forest Training

To train the offset regressor, we used the data captured from a multi-view capture system (see Fig. 8 and Sec. 6.1) to generate high quality pose estimates and per-pixel ground truth labels. Optimal parameters were selected with grid search using a leave–one–subject–out technique. Three trees of depth 18 were trained with 12K images each due to GPU memory constraints, and the mean joint error averaged over left-out subjects was 3.8cm.

5.3 Articulated Signed Distance Function

We built a hand model in blender [Blender Online Community 2016] by skinning a coarse triangular mesh to a skeleton using linear blend skinning. To construct a dense signed distance field in the base pose, this mesh can be densified and for each voxel, an exhaustive calculation of distances to all triangles, edges and vertices can be used to find the distance to the mesh. We then dilate the original coarse mesh to create a larger "cage" around the "implicit surface" encoded as zero crossings in the SDF. We then use TetGen [Si 2015] to "triangulate" these vertices into a tetrahedral cage.

6 EXPERIMENTS

In this section, we introduce our new dataset *Cheetah* motivated by the desire to create a multi-handed dataset captured at the high frame rates that our system expects. We also evaluate our tracker against state of the art methods on three publicly available datasets in the one-handed scenario. Furthermore, we qualitatively display the robustness, flexibility and the low latency of our system in our demo video.

Fig. 8. *Our three camera rig captures depth data from three views at 180 fps. We use calibrated RGB cameras to perform segmentation of the two hands, each contained in a different glove from both each other and other objects.*

6.1 The *Cheetah* dataset

Existing 3D hand pose datasets [Sharp et al. 2015; Sridhar et al. 2013; Tang et al. 2016; Tompson et al. 2014; Yuan et al. 2017] vary in capturing device, quantity, modality or annotation. In order to facilitate evaluation of our algorithm in the settings it was designed for (high frame rate, rapid motion, two hands interacting strongly), we collected a new dataset that we call *Cheetah* . This dataset consists of 24 sequences from six subjects, each performing one single-handed, one single-hand-with-an-object, one two-interacting-hands and one two-hands-interacting-with-an-object sequences. Each sequence has 3000 RGBD frames in two different viewpoints: front facing and egocentric. To accomplish this, we built a two camera capture system (see Fig. 8) that captures RGBD data from both front facing and ego-centric views. The subjects wear different colored gloves so that ground truth segmentation masks of the two hands can be extracted (even in the presence of objects). Due to bandwidth and synchronization limitations, we were only able to capture at 180fps – well below the speed of our tracker. To obtain (quasi)-ground truth, we used six different manually customized hand models and fit these models to the full 3D data offline using 20 iterations and 10 starting points. For frames where the fit looked good, we extracted the positions of the five finger tips and the wrist of each hand. We believe that the frame rate, complexity of motions, interactions and the diversity of scenarios (e.g. two hands, objects) of this dataset is far more sophisticated and challenging than what is currently available. Indeed, one can see from the large errors that our own method achieves (see Fig. 11), that there is plenty of room for future improvement.

6.2 Self Analysis

To decide the optimal number of LM iterations, we conducted an experiment on all one-handed sequences from *Cheetah* . Fig. 9 (a) shows that 6-iterations is much better than 3, and the accuracy on difficult cases are further improved by 9-iterations whilst saturation occurs at 12-iterations. Being able to run at a high frame rate is the main advantage of our SDF tracker. To demonstrate that tracking benefits from high frame rate, we subsampled the 180fps one-handed sequences to simulate 60fps and 30fps. Fig. 9 (b) shows that indeed accuracy at 180 fps is the best. Due to the restrictions of capturing device, we were not able to test at higher frame rates.

We also conducted a set of experiments with *Cheetah* as a baseline for future comparisons. As expected, the two-handed scenario is more difficult than the one-handed, and hand(s) interacting with objects caused catostrophic failures (see Fig. 11). Moreover, due to

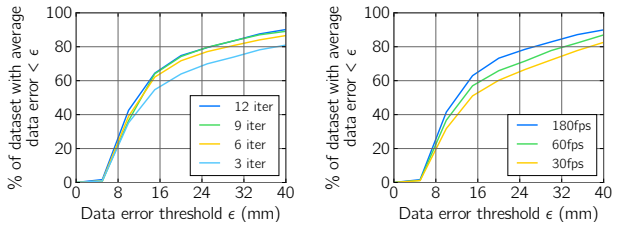

Fig. 9. *(a) Running the SDF tracker with different iterations on one-handed sequences. (b) Running the SDF tracker at different frame rates on one-handed sequences.*

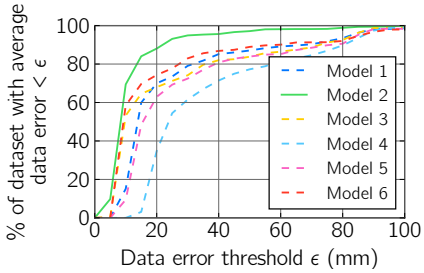

Fig. 10. There are six subjects with personalized models in our Cheetah dataset. To prove the benefits of shape personalization, we fit one sequence of Subject 2 with all six different models. As expected, results show that fitting with Subject 2's model is significantly better.

larger occlusions, the accuracy in the egocentric view is generally lower than from the front-facing view. To remove the dependency on the segmenter, we also ran the SDF tracker with ground truth segmentation obtained from colored gloves. Not surprisingly, in the case of hand(s) interacting with objects, ground truth segmentation helped. However, ground truth segmentation did not drastically improve the two-hand-interacting scenario as much. This is because our segmenter already tries to disambiguate left and right hands, and this information is fed into the energy to be optimized. Qualitative results are shown in Fig. 14.

Note that our system does not currently account for shape variation and thus custom models were used for each user. To understand the impact of using the wrong shape for each user, we tried using all six models on one of the *Cheetah* sequences. As shown in Fig. 10, using the correct model unsurprisingly has a substantial impact on accuracy. Although it is not obvious how to build a parametric shape model for our system as other approaches have done, we are hopeful that our SDF formulation will instead facilitate novel methods for dynamically integrating shape data into our SDF in a non-parametric fashion.

6.3 Comparison on *Dexter*

In order to compare to the state of the art, we use the dataset *Dexter* from [Sridhar et al. 2013]. We would like to emphasize that this dataset is only captured at a frame rate of 25fps and that our tracker is designed for frame rates of more than 180fps. Indeed, at such a slow frame rate large motions between frames can occur and thus a highly accurate reinitialization strategy is crucial, whereas we only use a coarse reinitializer. Nonetheless, we actually exceed the state of the art in the low error regime (see left of curve in Fig. 12) but not surprisingly struggle a bit in the higher error regime as our tracker misses some frames as our coarse reinitializer struggles to reset tracking (see right of curve in Fig. 12).

Subject	1	2	3	4	5	6
DART asym.	32.0	34.4	47.4	21.3	19.1	35.6
DART symm.	14.1	12.0	24.7	14.4	12.6	26.8
SDF Tracker	32.3	12.2	25.2	18.1	33.8	16.5

Table 1. A comparison of average error in mm between our SDF Tracker and [Schmidt et al. 2014] on the six subjects in the *MSRA* dataset.

6.4 Comparison on *MSRA*

In order to compare to [Schmidt et al. 2014] we use the *MSRA* dataset from [Qian et al. 2014]. Not surprisingly we generally outperform the asymmetric version of the algorithm in [Schmidt et al. 2014] that is most similar to our own algorithm and are competitive with the symmetric version. This may indicate the benefit of the more accurate modelling that our formulation can afford using a continous volumetric warp instead of a decomposition into rigid components. This also hints at the value of formulating an appropriate background to obtain the advantages they appear to recieve through their symmetric term.

6.5 Comparison on *Handy*

Lastly, we also compare to the state of the art on the Teaser sequence in *Handy* (see Fig. 15) from [Tkach et al. 2016]. Although we do fairly well here (likely due to the high frame rate), our system seems to respond very poorly to what looks like "flying pixels" consistently derailing our frame to frame tracking.

6.6 Qualitative Comparison to [Tkach et al. 2016]

In our supplementary video, we perform a side-by-side qualitative comparison of our tracker with the tracker from [Tkach et al. 2016]. The first thing to notice is that their tracker requires the use of a wristband to localize and segment the hand. Even if this is not giving an explicit prior on 3-DOF position, this will be implicitly communicated to the tracker through the near perfect segmentation. In contrast, our tracker has to deal with a noisy segmentation at the wrist. The second thing to notice is that the tracker appears to assume that the forearm aims towards the ground, as the hand model immediately jumps up the arm (see Fig. 13).

6.7 User Interaction in Virtual Reality

Many efforts aim to bring vision-based hand tracking to Virtual Reality with the promise to enable dexterous interaction for more seamless and immersive experiences. Technologies, such as the Leap Motion controller already demonstrate compelling applications, in which users can use gestures to interact with objects in simulated physics environments. However, ego-centric hand tracking is challenging due to the non-static nature of the captured background and proneness to self-occlusion. Also, interaction techniques typically do not go beyond interactions with virtual buttons in midair or user

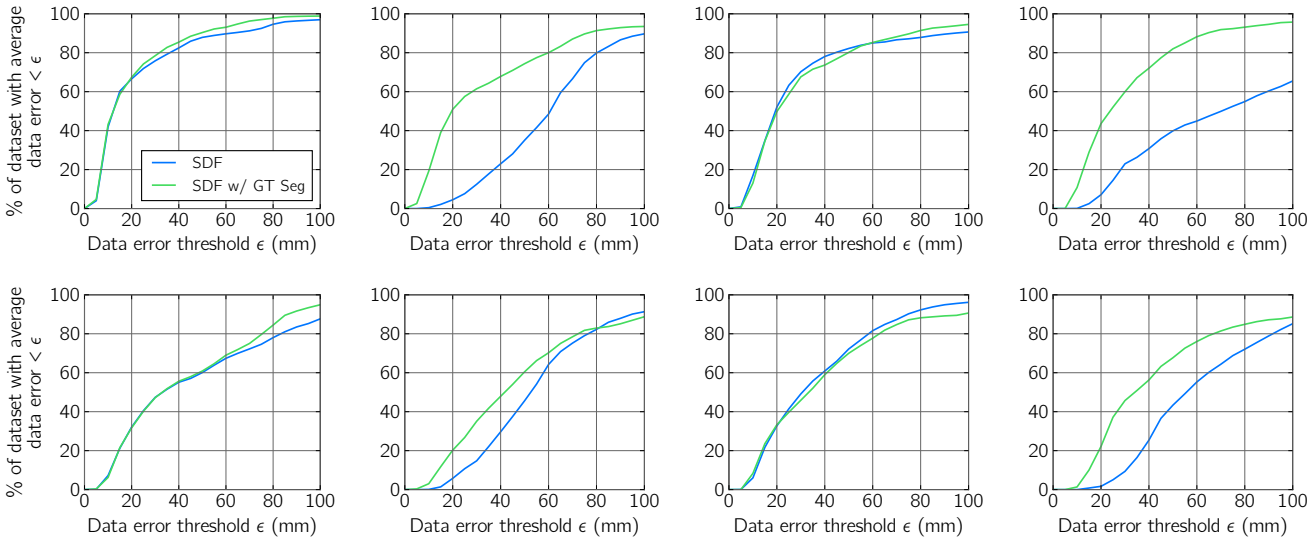

Fig. 11. Quantitative results on our Cheetah dataset. *Row 1: tracked with front facing camera only. Row 2: tracked with ego–centric camera only. Column 1 4: single hand, single hand with an object, two interacting hands, two hands interacting with an object. Results of SDF tracker and SDF tracker conditioned on ground truth segmentation are presented.*

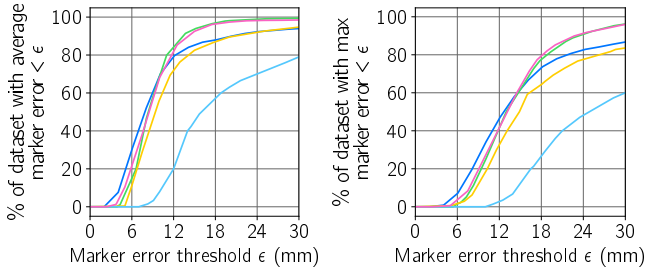

Fig. 12. *Comparison of SDF tracker, [Taylor et al. 2016], [Sridhar et al. 2015], [Joseph Tan et al. 2016] and [Tagliasacchi et al. 2015] on* Dexter.

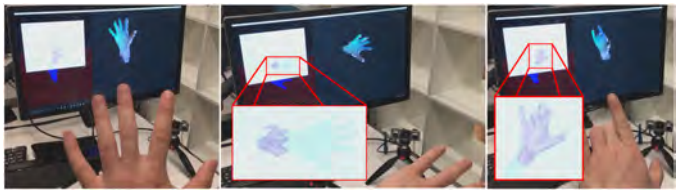

Fig. 13. *Both our tracker and that of [Tkach et al. 2016] track basic poses well (left), but more complex poses, possibly due to frame rate differences between the cameras, seem to fail more often (right) and there seems to be a heavy assumption that the hand is pointed upwards as the model immediately jumps to the back of the forearm when the hand becomes 90 degrees (center). A qualitative side-by-side comparison demonstrating their reliance on a blue wristband and the considerable improvement in robustness and flexibility our tracker offers is demonstrated in the supplementary video.*

interface (UI) elements around the user's hands, dynamic and static hand gestures.

We integrated our hand tracker into the HTC Vive VR headset and explored bi-manual interaction techniques that involve touching UI elements, such as buttons, 1D sliders and 2D track pads, mapped to specific regions of the hands. These techniques come with the advantages of allowing users to interact without looking at the interface itself using their proprioceptive sense and providing passive haptic feedback, something that midair UI lack.

In a preliminary informal study, we identified 16 interior and seven exterior touch points on the hand that are suited for blind touch input that stand out due to their tactile sensitivity and clearly identifiable texture. These points include the inner joint creases (three on each finger, two on the thumb, one on the palm and one on the wrist), and the tips on each digit and the creases on the outer side of the palm. We also identified five areas that are suited for 1D slider input, four along the full length of the inner side of each finger and one along the outer edge of the palm. The palm is suited as a large track pad for relative 2D input, such as panning or scrolling (please see the supplementary video).

7 CONCLUSION

We have presented a system for robustly and accurately tracking the articulations of two hands interacting. At the core of this system is a new implicit representation of an articulated surface that allows for extremely fast gradient based optimization. Crucial to the robustness of this system is an effective method for segmenting out the hands and assigning each pixel a probability of belonging to a left hand, right hand or background. We demonstrate competitive performance against state of the art methods on standard sequences, but as our method is designed to run at 180fps or above, we introduce a new high frame rate dataset *Cheetah* including multiple hands and object interaction.

Fig. 14. Qualitative results on the Cheetah dataset. *Each row is the result from one frame in* Cheetah . *Blue points are from the front facing camera and green points from the ego–centric one. Colored gloves are for ground truth annotation and not used during testing. Due to larger occlusion, failures in ego–centric view happen more frequently than the front facing view, e.g. , row 3 and row 7.*

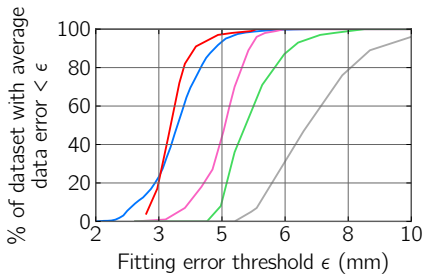

Fig. 15. *Comparison of SDF tracker, [Taylor et al. 2016], [Sharp et al. 2015], [Tkach et al. 2016] and [Tagliasacchi et al. 2015] on* Handy.

REFERENCES

Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision*. Springer, 640–653.

Blender Online Community. 2016. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam. http://www.blender.org

Thomas J Cashman and Andrew W Fitzgibbon. 2013. What shape are dolphins? Building 3D morphable models from 2D images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35, 1 (2013), 232–244.

Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence* 24, 5 (2002), 603–619.

Martin de La Gorce, David J Fleet, and Nikos Paragios. 2011. Model-based 3d hand pose estimation from monocular video. *IEEE transactions on pattern analysis and machine intelligence* 33, 9 (2011), 1793–1805.

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan

Taylor, et al. 2016. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 114.

Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 2015. 3d scanning deformable objects with a single rgbd sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 493–501.

Sean Ryan Fanello, Julien Valentin, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, Carlo Ciliberto, Philip Davidson, and Shahram Izadi. 2017a. Low Compute and Fully Parallel Computer Vision with HashMatch. In *ICCV*.

Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, Philip Davidson, and Shahram Izadi. 2017b. UltraStereo: Efficient Learning-based Matching for Active Stereo Systems. In *CVPR*.

Andrew W Fitzgibbon. 2003. Robust registration of 2D and 3D point sets. *Image and Vision Computing* 21, 13 (2003), 1145–1153.

Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 2016. Robust 3D Hand Pose Estimation in Single Depth Images: From Single-View CNN to Multi-View CNNs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ross Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew Fitzgibbon. 2011. Efficient regression of general-activity human poses from depth images. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 415–422.

Je Hyeong Hong and Andrew Fitzgibbon. 2015. Secrets of Matrix Factorization: Approximations, Numerics, Manifold Optimization and Random Restarts. In *Proceedings of the IEEE International Conference on Computer Vision*. 4130–4138.

Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 559–568.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5610–5619.

Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. 2012. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proceedings of the IEEE International Conference on Computer Vision*. 852–863.

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.

Francois Lekien and J Marsden. 2005. Tricubic interpolation in three dimensions. *Internat. J. Numer. Methods Engrg.* 63, 3 (2005), 455–471.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2014. Fully Convolutional Networks for Semantic Segmentation. *CoRR* abs/1411.4038 (2014). http://arxiv.org/abs/1411.4038

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, 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 *Proceedings of International Conference on Computer Vision (ICCV)*. 10. http://handtracker.mpi-inf.mpg.de/projects/OccludedHands/

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 343–352.

Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. 2015a. Hands Deep in Deep Learning for Hand Pose Estimation. *CoRR* abs/1502.06807 (2015). http://arxiv.org/abs/1502.06807

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

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

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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Edoardo Remelli, Anastasia Tkach, Andrea Tagliasacchi, and Mark Pauly. 2017. Low-Dimensionality Calibration through Local Anisotropic Scaling for Robust Hand Model Personalization. In *Proceedings of the International Conference on Computer Vision*.

Tanner Schmidt, Richard A Newcombe, and Dieter Fox. 2014. DART: Dense Articulated Real-Time Tracking.. In *Robotics: Science and Systems*.

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 the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3633–3642.

Jamie Shotton, Andrew Fitzgibbon, Andrew Blake, Alex Kipman, Mark Finocchio, Bob Moore, and Toby Sharp. 2011. Real-Time Human Pose Recognition in Parts from a Single Depth Image. https://www.microsoft.com/en-us/research/publication/real-time-human-pose-recognition-in-parts-from-a-single-depth-image/

Hang Si. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)* 41, 2 (2015), 11.

Ayan Sinha, Chiho Choi, and Karthik Ramani. 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*. 4150–4158.

Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. 2015. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3213–3221.

Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. 2013. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proceedings of the IEEE International Conference on Computer Vision*. 2456–2463.

Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. 2015. Cascaded hand pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 824–832.

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

Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. 2016. Latent Regression Forest: Structured Estimation of 3D Articulated Hand Posture. In *Transactions on Pattern Analysis and Machine Intelligence*.

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 *Proceedings of the IEEE International Conference on Computer Vision*. 3325–3333.

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, Richard Stebbing, Varun Ramakrishna, Cem Keskin, Jamie Shotton, Shahram Izadi, Aaron Hertzmann, and Andrew Fitzgibbon. 2014. User-specific hand modeling from monocular depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 644–651.

Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. 2016. Sphere-Meshes for Real-Time Hand Modeling and Tracking. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* (2016).

Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. 2017. Online Generative Model Personalization for Hand Tracking. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia, conditionally accepted)* (2017).

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 (ToG)* 33, 5 (2014), 169.

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).

Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. 2017. Crossing Nets: Dual Generative Models with a Shared Latent Space for Hand Pose Estimation. *arXiv preprint arXiv:1702.03431* (2017).

Qi Ye, Shanxin Yuan, and Tae-Kyun Kim. 2016. Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation. In *European Conference on Computer Vision*. Springer International Publishing, 346–361.

Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhand Jain, and Tae-Kyun Kim. 2017. BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis. *arXiv preprint arXiv:1704.02612* (2017).

Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. 2016. Model-based Deep Hand Pose Estimation. In *IJCAI*.