Sphere-Meshes for Real-Time Hand Modeling and Tracking

Anastasia Tkach
EPFL

Mark Pauly
EPFL

Andrea Tagliasacchi
University of Victoria

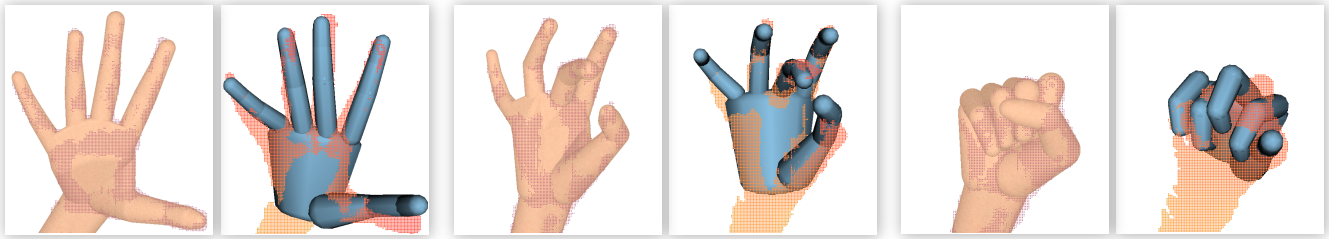

Figure 1: *Three side-by-side comparisons of tracking performance from the* HANDY/TEASER *sequence. Our model allows us to obtain much higher tracking quality. Tracking at a finer scale is instrumental to prevent tracking failure. The whole sequence can be seen in Video [03:53].*

Abstract

Modern systems for real-time hand tracking rely on a combination of discriminative and generative approaches to robustly recover hand poses. Generative approaches require the specification of a geometric model. In this paper, we propose a the use of sphere-meshes as a novel geometric representation for real-time generative hand tracking. How tightly this model fits a specific user heavily affects tracking precision. We derive an optimization to non-rigidly deform a template model to fit the user data in a number of poses. This optimization jointly captures the user's static and dynamic hand geometry, thus facilitating high-precision registration. At the same time, the limited number of primitives in the tracking template allows us to retain excellent computational performance. We confirm this by embedding our models in an open source real-time registration algorithm to obtain a tracker steadily running at 60Hz. We demonstrate the effectiveness of our solution by qualitatively and quantitatively evaluating tracking precision on a variety of complex motions. We show that the improved tracking accuracy at high frame-rate enables stable tracking of extended and complex motion sequences without the need for per-frame re-initialization. To enable further research in the area of high-precision hand tracking, we publicly release source code and evaluation datasets.

Keywords: non-rigid registration, hand tracking, sphere-meshes

Concepts: •**Computing methodologies** → **Tracking; Motion capture;** *Shape representations; Volumetric models;*

1 Introduction

With the imminent advent of consumer-level virtual and augmented reality technology, the ability to interact with the digital world in the most natural way, using our hands, becomes of paramount importance. Over the past two decades a number of techniques have been explored to address this problem, from expensive and unwieldy marker-based mocap [Welch and Foxlin 2002] to instrumented gloves [Dipietro et al. 2008] as well as imaging systems [Erol et al. 2007]. Multi-camera imaging systems can recover the hand pose and hand-objects interactions with high accuracy [Ballan et al. 2012], but the only system capable to approach interactive applications is the 10Hz system of [Sridhar et al. 2013]. Conversely, in this paper we focus on hand motion tracking with a single RGBD sensor (e.g. Intel RealSense or Microsoft Kinect), commonly predicted to be readily available in a typical AR/VR consumer experience.

Tracking: discriminative vs. generative. Modern systems for real-time tracking from RGBD data [Sridhar et al. 2015; Sharp et al. 2015] rely on a combination of discriminative approaches like [Keskin et al. 2012], and generative approaches such as [Oikonomidis et al. 2011]. The *per-frame* re-initialization of discriminative methods prevents error propagation by offering a continuous recovery from tracking failure. As these discriminative models are learnt from data, they typically only estimate a coarse pose. Therefore, generative models are used to refine the estimate by aligning a geometric template of the user hand to the measured point cloud as well as to regularize its motion through time. It is not surprising that the quality of the template directly affects the quality of pose refinement; see Figure 1.

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

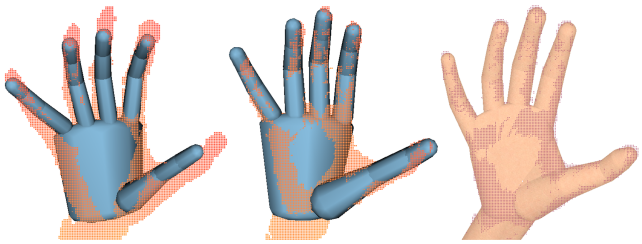

Figure 2: *(left) Tracking when the model from [Tagliasacchi et al. 2015] is used without proper coarse scale calibration. (middle) A roughly manually calibrated model can help increasing the fitting fidelity, but tuning becomes increasingly difficult with more degrees of freedom. (right) The proposed automatically calibrated model.*

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. © 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
SA'16 Technical Papers, December 05 - 08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2980226

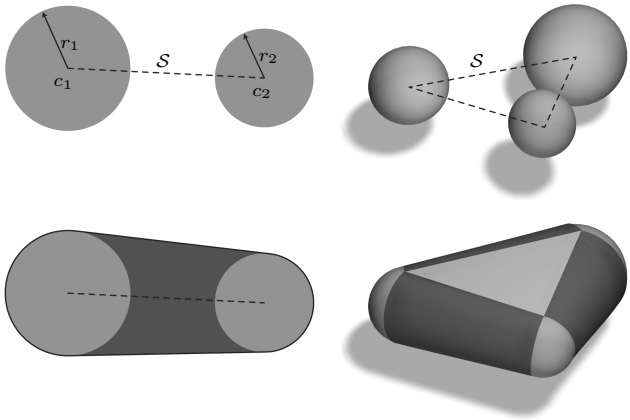

Figure 3: *The sphere-mesh skeleton $\mathcal{S}$ identifies sphere positions and radii. The surface of the object is obtained as the convex-hull of the spheres on the vertices of the skeleton. Sphere-meshes can be rendered through GPU ray-tracing, or by meshing the zero-crossing of their implicit function; see Eq. 1.*

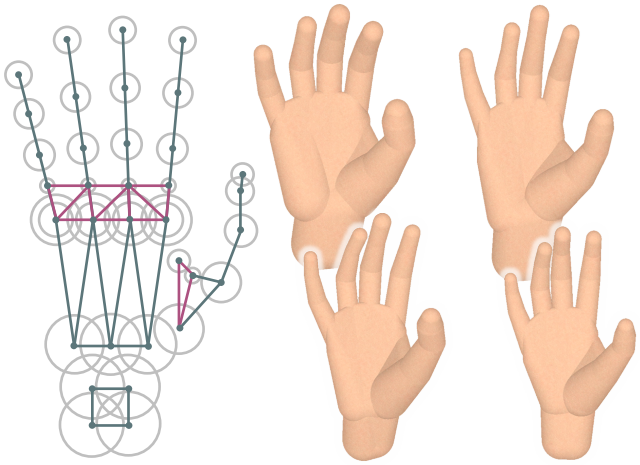

Figure 4: *(left) The skeleton $\mathcal{S}$ parametrizes the sphere-mesh through vertex positions and radii. In our template, articulated components are shown in dark green while flexible components in purple. (right) Calibration instantiates our template by adjusting the skeletal vertex positions and radii.*

specifically, we propose a geometric model that more accurately captures the user's hand geometry, while retaining the ability to answer registration queries in closed form with very high efficiency. In Figure 2 and Video [03:53] we illustrate the importance of employing a tracking template that strikes this delicate balance.

Implicit vs. explicit templates. In modern digital production the de-facto standard is to represent objects by a surface mesh of their boundary (e.g. triangle or quad meshes). Fast rendering and easy direct manipulation make *explicit* surface representation attractive for many applications. However, unlike *implicit* models [Bloomenthal et al. 1997], explicit representations cannot efficiently answer queries such as the distance from a point to the object's boundary, or whether a point lies inside/outside the model [Botsch et al. 2010, Ch.1]. In tracking applications these queries play a fundamental role, as the optimization attempts to find configurations where the average distance from model to data is minimized. Similarly, a tracker should prevent the model from assuming implausible configurations, for example by preventing self-intersections as measured by inside/outside predicates. For all these reasons, and as demonstrated by compelling results in rigid [Newcombe et al. 2011] and non-rigid [Newcombe et al. 2015] reconstruction, implicit models are highly suitable for registration applications. To address the challenges of *real-time* registration, we propose to employ a *hybrid* model that combines the advantages of explicit and implicit representations.

Hybrid sphere-mesh templates. The model we propose in this paper is a variant of a convolution surface [Bloomenthal and Shoemake 1991]. Its fundamental building blocks are illustrated in Figure 3. The surface is defined as the zero iso-level of the scalar function

$$\phi(\mathbf{x}) = \min_{\mathbf{c}\in\mathcal{S}} \mathcal{B}(\mathbf{x}|\mathbf{c}, r(\mathbf{c})) \tag{1}$$

where $\mathcal{S}$ is a skeletal control mesh (a segment or a triangle in the simple examples of Figure 3), and $\mathcal{B}$ is the implicit function of a sphere given its center $\mathbf{c}$ and radius r:

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

The sphere centers $\mathbf{c}$ span the skeleton $\mathcal{S}$, while the radii are a function of the position $\mathbf{c}$ within an element, linearly interpolated from values $r_* = r(\mathbf{c}_*)$ specified on the skeletal mesh vertices $\mathbf{c}_*$.

This is indeed a *hybrid* model, as Eq. 1 defines an implicit surface $\mathcal{M} = \{\mathbf{x} \in \mathbb{R}^n | \phi(\mathbf{x}) = 0\}$, while the underlying skeleton $\mathcal{S}$ is an explicit representation (i.e. a simplicial complex). We generalize this construct to devise a model suitable to represent a human hand; see Figure 4. Distances to $\mathcal{M}$ can conveniently be computed by querying distances to the piecewise linear elements of $\mathcal{S}$; see Figure 7.

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

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

Overview. The remainder of the paper is structured as follows: We survey related work in Section 2. In Section 3 we describe our generative real-time hand tracking technique, which details how our novel formulation enables efficient correspondence computation. Section 4 explains how we build our template model from 3D scans acquired either through multi-view stereo or from depth maps. In Section 5 we analyze the performance of our model for realtime tracking and provide comparisons to the state-of-the-art. We conclude in Section 6 with limitations and future works.

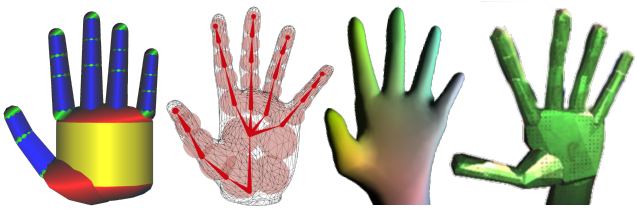

Figure 5: *Several tracking templates employed by recent generative (or hybrid) real-time hand-tracking methods. Images courtesy of (a) [Oikonomidis et al. 2011], (b) [Sridhar et al. 2013], (c) [Taylor et al. 2016], and (d) [Melax et al. 2013].*

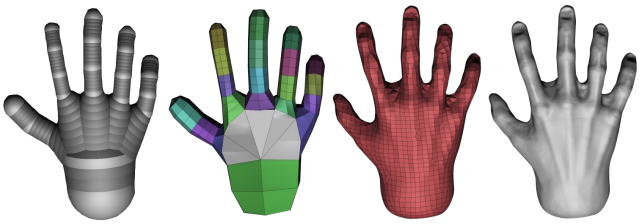

Figure 6: *(a) An artist creates a hand model by first sketching its topological structure as a union of spheres (ZSphere). (b) The model is then converted into a volumetric representation and meshed (Unified Skinning) to be further refined (c,d).*

2 Related Work

The simplest way of tracking a hand in motion is by instrumentation. We can place retro-reflective markers on the hand, wear a data glove with embedded abduction and flexion sensors [Dipietro et al. 2008] or a colored glove [Wang and Popovic 2009]. While effective, active instrumentation can be cumbersome as it requires lengthy preparation and/or calibration. Systems solely relying on computer vision (i.e. color cameras) are highly desirable, but pose estimation relying on color information is extremely challenging [Erol et al. 2007]: the complexity of hand motion, the large number of self-occlusions and rapidly changing backgrounds all contribute to their limited success. Multiple-camera acquisition can mitigate these challenges. In [Ballan et al. 2012], the authors demonstrate high-quality tracking of two-hand and hand-object interactions. However, due to the substantial increase in data bandwidth, these algorithms do not scale to real-time performance. A notable exception is the 10Hz system of [Sridhar et al. 2013], but this acquisition setup also considers data from a depth sensor. While providing accurate results, multi-camera rigs are impractical for consumer-level applications. Therefore, we limit our attention to techniques relying on data from a *single* depth camera; with a slight abuse of wording we refer to these systems as *monocular* acquisition setups.

Hybrid = discriminative + generative. Pose estimation techniques can be grouped into *discriminative* and *generative* techniques, also known respectively as *appearance-based* and *model-based* approaches. Generative approaches fit a template through a temporal sequence of images [Oikonomidis et al. 2011; Melax et al. 2013; Schroder et al. 2014; Tagliasacchi et al. 2015]. Given an accurate template of the user being tracked, these methods can resolve highly accurate motion. As the optimization is initialized from the previous frame, tracking loss can occur, although simple geometric reinitialization heuristics can be employed to overcome this issue [Melax et al. 2013; Qian et al. 2014]. Conversely, discriminative methods estimate the pose by extracting features from each image independently by learning from a large dataset of annotated exemplars [Keskin et al. 2012; Tang et al. 2013; Tejani et al. 2014; Sun et al. 2015]. While discriminative methods avoid drift, they lack the accuracy of generative methods, and joint estimates often violate kinematic constraints, like consistent finger lengths and joint limits. State-of-the-art tracking performance is achieved by *hybrid* algorithms that combine the two approaches. These algorithms estimate (potentially) multiple per-frame coarse poses leveraging discriminative frameworks, and then refine the alignment with a generative fitting [Tompson et al. 2014; Qian et al. 2014; Sharp et al. 2015]. Another class of hybrid algorithms introduce correspondences through a labeling obtained through a per-pixel forest classifier [Sridhar et al. 2015; Fleishman et al. 2015]. Our literature review focuses on generative approaches, while we refer the reader to the very recent work by Taylor and colleagues [2016] for a review of recent discriminative methods. Commercial systems for hand tracking like the *NimbleVR*, the *LeapMotion Orion* and the *Intel Perceptual SDK* also exist, but it is difficult to consider them as the underlying technology is undisclosed.

Generative tracking models. The capsule model originally proposed by [Rehg and Kanade 1994] has been adopted by a number of researchers [Oikonomidis et al. 2011; Schroder et al. 2014; Fleishman et al. 2015; Tagliasacchi et al. 2015]; see Figure 5(a). Such a coarse representation is suitable to the task given the low signal-to-noise ratio in modern depth sensors, while its simplicity enables the efficient closed-form computation of alignment queries. Cylinders can also be approximated by a small set of disconnected spheres [Qian et al. 2014], but this rough approximation is only sufficient for coarse-scale tracking. An alternative to cylinders and spheres is the use of isotropic [Sridhar et al. 2013; Sridhar et al. 2015], as well as anisotropic Gaussians [Sridhar et al. 2014]; see Figure 5(b). The use of surface meshes, while widespread in other domains (e.g. face tracking [Bouaziz et al. 2013] or offline registration [Loper and Black 2014]), has been limited to the visualization of tracking performance through skinned model animations [Tompson et al. 2014; Schroder et al. 2014]. Sharp et al. [2015] employed mesh models for tracking in a render-and-compare framework, while the very recent work of [Taylor et al. 2016] presents the first attempt towards a continuous registration framework for tracking hands with triangular meshes; see Figure 5(c). Other variants of tracking models include the union of convex bodies from [Melax et al. 2013], a convolutional neural network capable of directly synthesizing hand depth images [Oberweger et al. 2015], and some initial attempts at tracking with implicit templates [Plankers and Fua 2003]. Our sphere-mesh model offers accuracy comparable to triangle meshes used in recent hand trackers, while retaining a compact representation for efficient correspondence queries and effective user adaptation.

Template calibration. Albrecht et al. [2003] pioneered the creation of a realistic hand model (i.e. bones and muscles) by aligning a template mesh to data acquired by a laser-scanned plaster cast. Rhee et al. [2006] use a simpler setup consisting of a single color image to identify approximate joint locations by localizing skin creases, and adapt a mesh template to conform to its silhouette. While these methods focus on a static template, in [de La Gorce et al. 2011] a model is roughly adapted to the user through simple bone scaling to produce the first *animatable* template. Calibration of a cylinder model through particle swarm has been investigated in [Makris and Argyros 2015]. Mesh calibration techniques were proposed in [Taylor et al. 2014] and extended in [Khamis et al. 2015], which introduces compact and linear shape-spaces of human hand geometry. The method in [Taylor et al. 2014] shares some similarities with our work, where the model is adjusted to *jointly* fit a set of depth frames, but with a fundamental difference in the way in which geometry is represented. Our sphere-mesh model is *naturally compact*, leading to straightforward calibration and tracking algorithms.

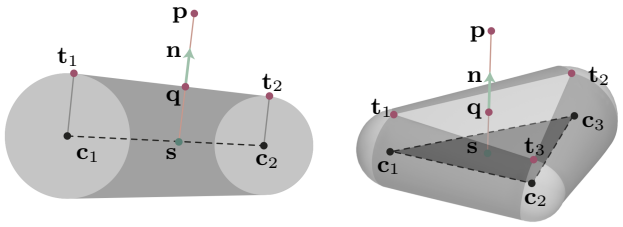

Figure 7: *The computation of closest point correspondences on pill (left) and wedge (right) elements can be performed by tracing a ray along the normal of the line (resp. plane) tangent to the circles (resp. spheres).*

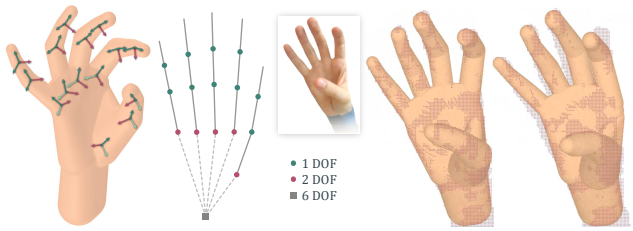

Figure 8: *(a) A visualization of the posed kinematic frames $\bar{T}_*$. (b) The kinematic chain and number of degrees of freedom for posing our tracking model. Tracking quality with (c) optimal and (d) non-optimal kinematic transformation frames.*

Implicit modeling. Implicit sculpting tools have recently become a viable alternative to mesh or spline-based approaches for modeling complex geometries. This paradigm lies at the basis of the success of the *PixoLogic ZBrush* product line. For articulated geometry, it is often convenient to first create a coarse geometric structure analogous to the one described in Eq. 1, a process that *PixoLogic* has re-branded as *ZSphere* modeling; see Figure 6. Editing the radii and centers of the sphere-mesh offers a *natural* way of editing the model, making it easy for both humans and algorithms to calibrate. Note that any geometric model can be approximated, to any desired precision, as a union of spheres [Tagliasacchi et al. 2016]. However, by considering spheres that are linearly interpolated across edges, we can heavily reduce the required number of primitives. Following this principle, [Thiery et al. 2013] recently investigated a method to automatically generate Sphere Meshes provided a (static) input model. Extending this work, [Thiery et al. 2016] proposed a method to fit a model to a sequence of dynamic meshes. While seemingly related, our calibration optimization is solving a fundamentally different problem, as in our technique a template is fixed and provided in input.

3 Tracking

Given a calibrated hand model $\mathcal{M}$, our real-time tracking algorithm optimizes the 28 degrees of freedom $\boldsymbol{\theta}$ (i.e. joint angles) so that our hand model matches the sensor input data; the generation of a calibrated model $\mathcal{M}$ for a user is detailed in Section 4. Directly extending the open source *htrack* framework of [Tagliasacchi et al. 2015], we write our tracking optimization in Gauss-Newton/Levenberg-Marquardt form:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}} \sum_{\mathcal{T} \in \mathcal{T}_{\text{track}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathcal{D}_t, \boldsymbol{\theta}, \boldsymbol{\theta}_{t-1}) \tag{3}$$

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

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

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

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

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

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

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

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

$$E_{d2m} = |\mathcal{D}|^{-1} \sum_{\mathbf{p} \in \mathcal{D}} \| \mathbf{p} - \Pi_{\mathcal{M}(\boldsymbol{\Theta})}(\mathbf{p}) \|_2^1 \tag{4}$$

Adapting this energy, as well as its derivatives, to sphere-meshes requires the specification of the projection operator $\Pi_{\mathcal{M}}$ that is described in Section 3.1.

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

$$E_{m2d} = |\mathcal{M}(\boldsymbol{\Theta})|^{-1} \int_{\mathbf{x} \in \mathcal{M}(\boldsymbol{\Theta})} \| \mathbf{x} - \Pi_{\mathcal{D}}(\mathbf{x}) \|_2^1 \tag{5}$$

In the equation above, the integral is discretized as a sum over the set of pixels obtained through rasterization; see Section 3.2. The rasterization renders the model to the image plane using the intrinsic and extrinsic parameters of the sensor's depth camera.

3.1 Correspondences

Our correspondence search leverages the structure of Eq. 1, by decomposing the surface into several elementary elements $\mathcal{E}^e$, where e indexes the 30 elements of our template; see Video [00:58]. As illustrated in Figure 7, elements are classified into *pill* and *wedge* implicit primitives, with an associated implicit functions ϕ_e. Given a point $\mathbf{p}$ in space, the implicit function of the whole surface can be written by evaluating the expression:

$$\phi_{\mathcal{M}}(\mathbf{p}) = \arg\min_{e=1 \dots E} \phi_e(\mathbf{p}) \tag{6}$$

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

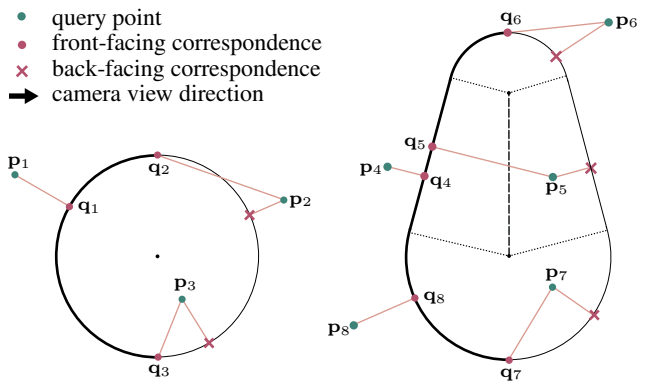

Figure 9: *In monocular acquisition only the front-facing part of the model should be registered to the data. Here the camera is observing (left to right) two elements and the occluded parts of the model are marked. Correspondences whose normals point away from the camera are discarded, and replaced by the closest amongst silhouette correspondences or front-facing portions of wedges.*

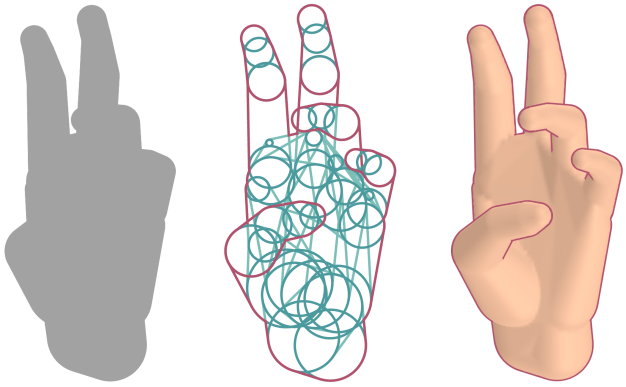

Figure 10: *The image-space silhouette of the model computed by projecting the model in the camera plane (left). The 2D object-space silhouette curves are computed separately for palm and fingers and then composited back together (center). The 3D object-space silhouette (pink) is re-projected in 3D (right).*

of closest-point correspondences takes 250 μs/iteration. We now describe in detail how the projection is evaluated on each element in closed form.

Pill correspondences: $\mathbf{q} = \Pi_{\text{pill}}(\mathbf{p})$**.** A pill is defined by two spheres $\mathcal{B}_1(\mathbf{c}_1, r_1)$ and $\mathcal{B}_2(\mathbf{c}_2, r_2)$. By construction the closest point correspondence lies on the plane passing through the triplet $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{p}\}$, thus allowing us to solve the problem in 2D; see Figure 7-(left). We compute the intersection point $\mathbf{s}$ of the ray $\mathbf{r}(t) = \mathbf{p} + t\mathbf{n}$ with the segment $\overline{\mathbf{c}_1\mathbf{c}_2}$ and parametrize its location in barycentric coordinates as $\mathbf{s} = \alpha\mathbf{c}_1 + (1-\alpha)\mathbf{c}_2$. If $\alpha \in [0, 1]$, our closest point correspondence is given by $\mathbf{q} = \Pi_{\mathcal{L}}(\mathbf{p})$, that is, the intersection of $\overline{\mathbf{c}_1\mathbf{c}_2}$ and $\mathbf{r}(t)$. If $\alpha < 0$ or $\alpha > 1$, then the closest point will be $\mathbf{q} = \Pi_{\mathcal{B}_1}(\mathbf{p})$ or $\mathbf{q} = \Pi_{\mathcal{B}_2}(\mathbf{p})$, respectively.

Wedge correspondences: $\mathbf{q} = \Pi_{\text{wedge}}(\mathbf{p})$**.** A wedge is defined by three spheres $\mathcal{B}_i = \{\mathbf{c}_i, r_i\}$. Figure 3 illustrates how a wedge element can be decomposed in three parts: *spherical*, *conical*, and *planar* elements, associated with vertices, edges, and faces of the sphere-mesh skeleton. For the planar element $\mathcal{P}(\mathbf{t}_1, \mathbf{n})$ with normal $\mathbf{n}$ and tangent $\mathbf{t}_1$ to $\mathcal{B}_1$ we compute the skewed projection $\mathbf{s}$ by finding the intersection of the ray $\mathbf{r}(t) = \mathbf{p} + t\mathbf{n}$ with the triangle $\mathcal{T}$ formed by $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$. According to the position of $\mathbf{s}$ we have two possible solutions: If $\mathbf{s}$ lies inside the triangle $\mathcal{T}$, then our footpoint is $\mathbf{q} = \Pi_{\mathcal{P}}(\mathbf{p})$. Otherwise, we use the barycentric coordinates of $\mathbf{s}$ in $\mathcal{T}$ to identify the closest pill element and compute $\mathbf{q} = \Pi_{\text{pill}}(\mathbf{p})$.

Monocular Correspondences. In monocular acquisition (i.e. single sensor), an oracle registration algorithm aligns the portion of the model that is *visible* from the sensor viewpoint to the available data. Hence, when computing ICP's closest-point correspondences, only the portion of the model currently visible by the camera should be considered [Tagliasacchi et al. 2015]. Given the camera direction $\mathbf{v}$, we can test whether the retrieved footpoint $\mathbf{q}$ is back-facing by testing the sign of $\mathbf{v} \cdot \mathcal{N}_{\mathcal{M}}(\mathbf{q})$, where the second term is the object's normal at $\mathbf{q}$. As illustrated in 2D in Figure 9, whenever this test fails, there are additional candidates for closest point that must be checked: (1) the closest-point on the silhouette of the model (e.g. $\mathbf{p}_{2,3,6,7}$), and (2) the front facing planar portions of elements (e.g. $\mathbf{p}_5$). These additional correspondences for the query point are computed, and the one closest to $\mathbf{p}$ becomes our front-facing footpoint $\mathbf{q}$. The additional computational cost caused by front-facing correspondences with an average of 2500 points/frame is 100 μs/iteration.

Silhouette computation. The *object-space silhouette* $\partial\mathcal{M}$ is a (3D) curve separating front-facing from back-facing portions of a shape [Olson and Zhang 2006, Sec.1]. To simplify the silhouette computation we approximate the perspective camera of the sensor with an orthographic one. We then offset all elements on the 2D camera plane, and perform a cross-section with this plane: spheres are replaced with circles and planes/cylinders with segments; see Figure 10-(left). We then compute an *arrangement*, splitting curves whenever intersection or tangency occurs; see Figure 10-(center). We traverse this graph, starting from a point that is guaranteed to be on the outline (e.g. a point on the bounding box). The traversal selects the next element as the one whose tangent forms the smallest counter-clockwise angle thus identifying the silhouette. Once the 2D silhouette has been computed, it can be re-projected to 3D; see Figure 10-(right). Note the process described above would compute the *image-space silhouette* of our model. Therefore, we apply the process to palm and fingers separately, and merge them in a second phase. The merge process simply checks whether vertices $v \in \partial\mathcal{M}$ are contained within the model, which means it discards those where $\phi_{\mathcal{M}}(v) < 0$. In our experiments the average computation of the silhouette on the CPU takes 150 μs/iteration.

3.2 Rendering

Rendering the sphere-meshes in real time is not only employed for visual verification of tracking performance; e.g. Figure 2. The real-time tracking algorithm reviewed above performs a 2D registration in the image plane that requires the computation of an (image-space) silhouette. There are two alternatives for rendering a sphere-mesh model like the one shown in Figure 4. One possibility is to explicitly extract the surface of individual elements by computing the convex hull of pairs or triplets of spheres; see Figure 3. While this process would be suitable in applications where the model is fixed, it is hardly appropriate in our scenario where we want to calibrate the model to the user. Therefore, similarly to [Thiery et al. 2016], we ray-trace the model on the GPU. We render a unit fullscreen quad and in the fragment shader use the camera intrinsics to compute the camera ray $\mathbf{r}(\mathbf{x})$ associated with each pixel $\mathbf{x}$. Each ray is intersected with each element of our model, and the closest intersection point is retained. Tests are performed with the planar, conical, and spherical primitives that compose each element. Rendering at a resolution of 320×240 pixels provides the best trade-off between accuracy and performance, leading to a total rendering time of $\approx$ 3ms for visualization and $\approx 500\mu s$/iteration for the evaluation of E_{m2d}.

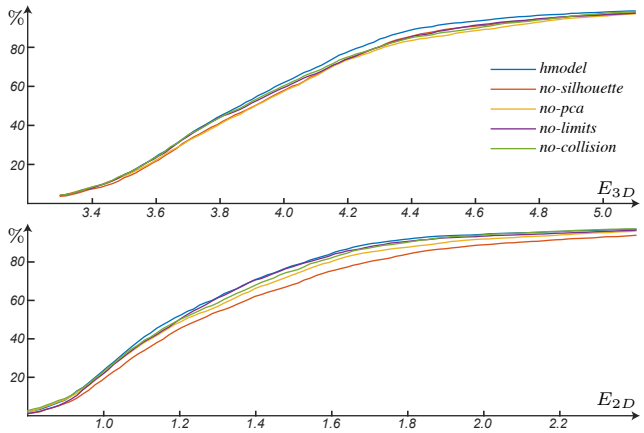

Figure 11: *Each plot visualizes on the y axis the portion of frames with a mean error metric below the value reported on the x axis. We employ the* HANDY/TEASER *sequence for this purpose. Curves closer to the top-left quadrant indicate better performance.*

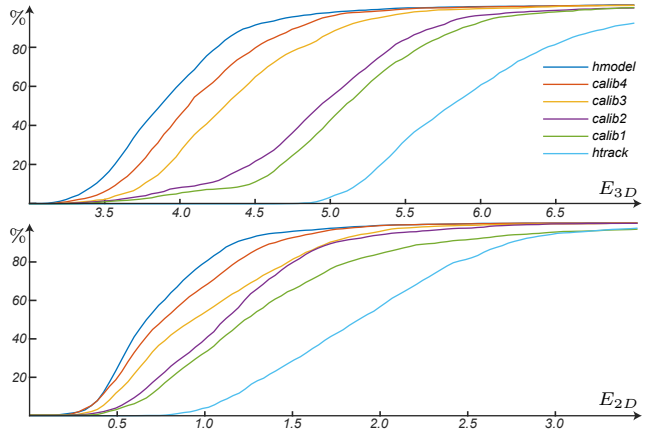

Figure 12: *Calibrating progressively improves the 2D/3D tracking metrics, showing a remarkable improvement in tracking fidelity from [Tagliasacchi et al. 2015] to [Proposed Method].*

4 Calibration

Our calibration procedure adapts our template model to a specific user from a set of N 3D measurements $\{\mathcal{D}_1 \dots \mathcal{D}_N\}$ of the user's hand in different poses. Multiple measurements are necessary, as it is not possible to understand the kinematic behavior by analyzing static geometry, and the redundancy of information improves fitting precision. Further, in monocular acquisition this redundancy is essential, as single-view data is highly incomplete, making the problem ill-posed. In our research we have experimented with datasets $\{\mathcal{D}_n\}$ acquired via multi-view stereo (e.g. *Agisoft Photoscan*), as well as a single RGBD sensor. Our calibration formulation can be employed for both acquisition modalities. Dynamic reconstruction frameworks such as [Newcombe et al. 2015] or [Innmann et al. 2016] could also be used to generate a dynamic template mesh over which sphere-mesh decimation could be executed [Thiery et al. 2016]. However, as no public implementation is currently available, it is currently unclear how well these methods would cope with loop-closure for features as small as human fingers.

Kinematics. The rest-pose geometry of our model is fully specified by two matrices specifying the set of sphere positions $\bar{\mathbf{C}}$ and the set of radii $\bar{\mathbf{r}}$. The geometry is then posed through the application of kinematic chain transformations; see Figure 8a. Given a point $\bar{\mathbf{p}}$ on the model $\mathcal{M}$ at rest pose, its 3D position after posing can be computed by evaluating the expression:

$$\mathbf{p} = \left[\Pi_{k \in K(\bar{\mathbf{p}})} \bar{\mathbf{T}}_k \mathbf{T}_k \bar{\mathbf{T}}_k^{-1}\right] \bar{\mathbf{p}} \tag{7}$$

where $\mathbf{T}_*$ are the *pose* transformations parameterized by $\boldsymbol{\theta}$ and Π left multiplies matrices by recursively traversing the kinematic chain K of point $\bar{\mathbf{p}}$ towards the root [Buss 2004]. Each node k of the kinematic chain is associated with an orthogonal frame $\bar{\mathbf{T}}_k$ according to which local transformations are specified. In most tracking systems, the frames $\bar{\mathbf{T}}_*$ are manually set by a 3D modeling artist and kept fixed across users. However, incorrectly specified kinematic frames can be highly detrimental to tracking quality; see Figure 8(c,d) and Video [02:12]. Therefore, in our formulation, the kinematic structure (i.e. the matrices $\bar{\mathbf{T}}_*$) is directly optimized from acquired data.

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

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

$$\underset{\{\boldsymbol{\Theta}_n\}}{\arg\min} \sum_{n=1}^{N} \sum_{\mathcal{T} \in \mathcal{T}_{\text{calib}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathcal{D}_n, \boldsymbol{\Theta}_n) \tag{8}$$

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

d2m	data to model distance
m2d	model to data distance
rigid	elements are posed rigidly
valid	elements should not degenerate

To make this calibration more approachable numerically, we rewrite Eq. 8 as an alternating optimization problem:

$$\underset{\{\mathbf{C}_n\}, \bar{\mathbf{C}}, \bar{\mathbf{r}}}{\arg\min} \quad \textstyle\sum_{n=1}^{N} \sum_{\mathcal{T} \in \mathcal{T}_{\text{calib}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathcal{D}_n, \mathbf{C}_n, \bar{\mathbf{C}}, \bar{\mathbf{r}}) \tag{9}$$

$$\underset{\{\boldsymbol{\theta}_n\}, \bar{\boldsymbol{\delta}}}{\arg\min} \quad \textstyle\sum_{n=1}^{N} \sum_{\mathcal{T} \in \mathcal{T}_{\text{calib}}} w_{\mathcal{T}} E_{\mathcal{T}}(\mathbf{C}_n, \boldsymbol{\Theta}_n) \tag{10}$$

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

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

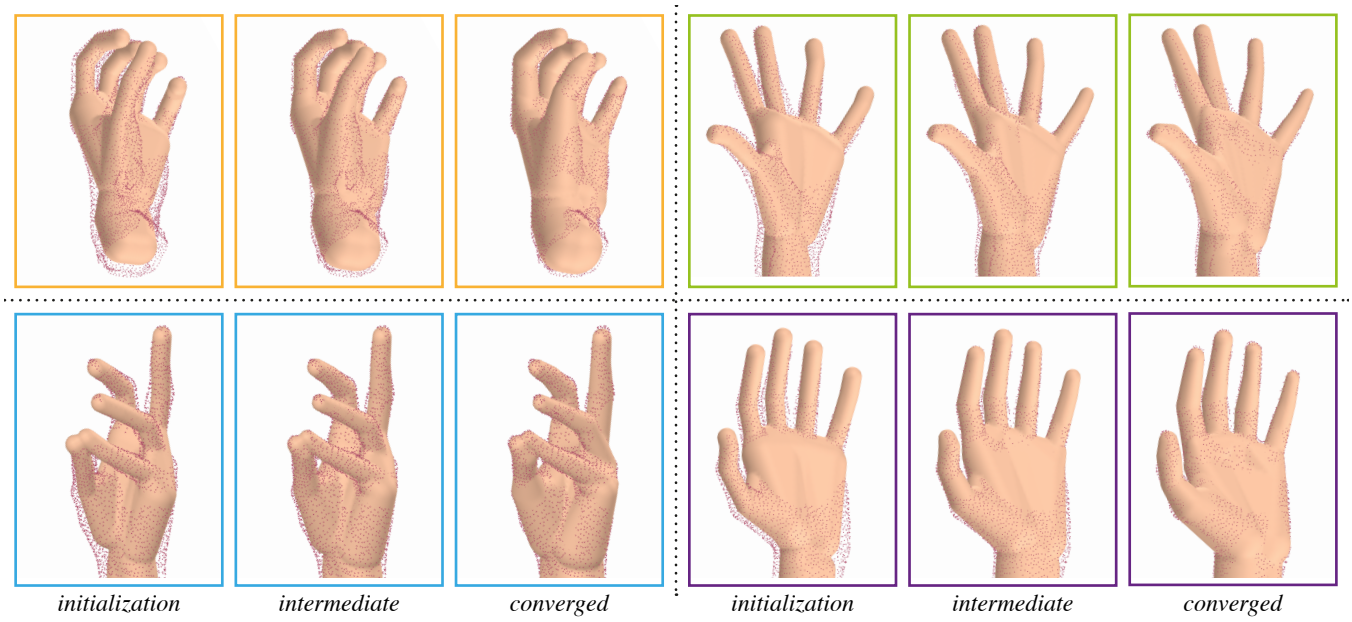

Figure 13: *A visualization of a few iterations of our calibration optimization procedure; see Video [01:30]. Each quadrant displays a data frame $\mathcal{D}_n$, $n = 1 \dots 4$. Within each quadrant we show three iterations of the optimization. The model being calibrated here is the one employed for real-time tracking in Video [02:57].*

4.1 Energies

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

$$E_{d2m} = \sum_{n=1}^{N} |\mathcal{D}_n|^{-1} \sum_{\mathbf{p} \in \mathcal{D}_n} \|\mathbf{p} - \Pi_{\mathcal{M}(\mathbf{\Theta}_n)}(\mathbf{p})\|_2^1 \qquad (11)$$

$$E_{m2d} = \sum_{n=1}^{N} |\mathcal{M}(\mathbf{\Theta}_n)|^{-1} \sum_{\mathbf{x} \in \mathcal{M}(\mathbf{\Theta}_n)} \|\mathbf{x} - \Pi_{\mathcal{D}_n}(\mathbf{x})\|_2^1 \qquad (12)$$

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

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

$$E_{\text{rigid}} = \sum_{ij \in \mathcal{S}} \left(\|\mathbf{c}_{n,i} - \mathbf{c}_{n,j}\| - \|\bar{\mathbf{c}}_i - \bar{\mathbf{c}}_j\| \right)^2 \qquad (13)$$

Note that only a subset of the edges of our control skeleton, as illustrated in Figure 4, are required to satisfy this rigidity condition.

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

$$E_{\text{valid}} = \sum_{\bar{\mathcal{E}} \in \bar{\mathbf{C}}} \sum_{\bar{\mathcal{B}}_i \in \bar{\mathcal{E}}} \chi(\bar{\mathcal{B}}_i) \|\bar{\mathbf{c}}_i - \Pi_{\bar{\mathcal{E}} \setminus \bar{\mathcal{B}}_i}(\bar{\mathbf{c}}_i)\|_2^2 \qquad (14)$$

5 Results

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

Template Calibration. The calibration of our model to a collection of 3D data frames is illustrated in Figure 13; note the same model is rigidly articulated to fit to multiple poses. While for this user we build a model from multi-view stereo data (omni-directional, complete), it is important to notice that the use of multiple frames in different poses is a necessity. Only in this situation can the centers $\{\mathbf{C}_n\}$ be jointly adjusted to create an articulated model that consensually fits the whole dataset. We refer the reader to Video [01:30] for a visualization of our iterative calibration procedure. The calibration from RGBD datasets can be seen at Video [01:45], and the resulting models are illustrated in Figure 4.

Kinematic Calibration. The importance of adjusting kinematic chain transformations is shown in Figure 8, as well as the first images pair in Figure 1. With incorrect transformations, joint limits and the articulation restrictions of the kinematic chain can prevent the model from being posed correctly; see a dramatization in Video [02:12]. In our experiments we discovered it was crucial to identify the *typical* kinematic chain structure using the dataset in Figure 13; user-specific calibration optimization used these transformations as initialization.

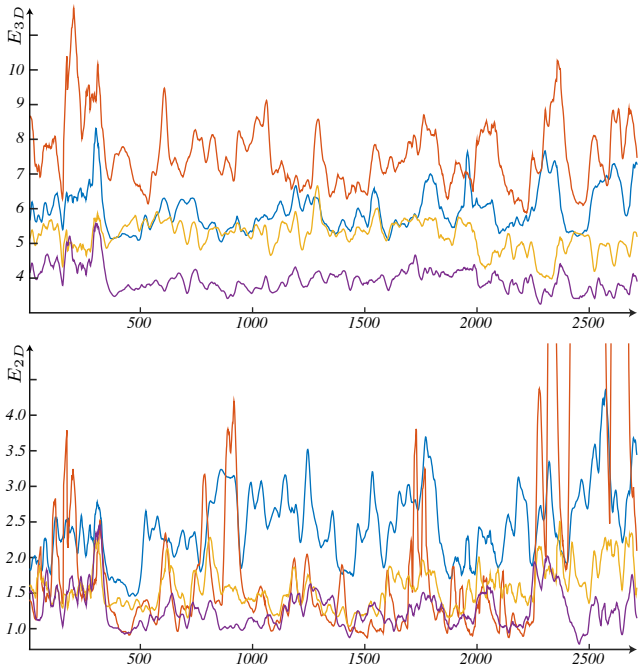

Figure 14: *The [Proposed Method] is quantitatively compared over time to [Tagliasacchi et al. 2015], [Sharp et al. 2015] and [Taylor et al. 2016] on the* HANDY/TEASER *sequence.*

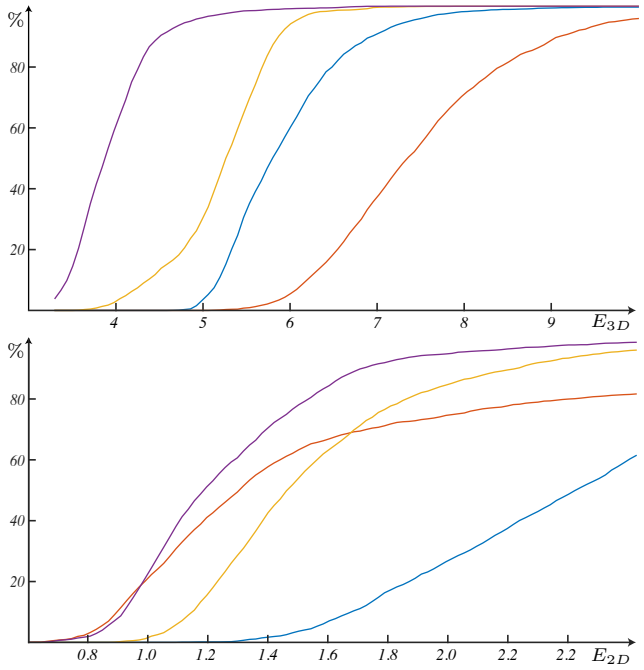

Figure 15: *Aggregated errors are reported for the tracking sequences in Figure 14. These aggregated measures reveal a significant improvement in tracking precision; see legend in Figure 14.*

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

$$E_{\text{3D}} = |\mathcal{D}|^{-1} \sum_{\mathbf{p} \in \mathcal{D}} \|\mathbf{p} - \Pi_{\mathcal{R}}(\mathbf{p})\|_2^1 \tag{15}$$

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

$$E_{\text{2D}} = |\mathcal{R} \setminus \partial\mathcal{D}|^{-1} \sum_{\mathbf{x} \in \mathcal{R} \setminus \partial\mathcal{D}} \|\mathbf{x} - \Pi_{\partial\mathcal{D}}(\mathbf{x})\|_2^1 \tag{16}$$

Each summation term above can be evaluated efficiently by precomputing the 2D Euclidean distance transform of the RoI's (image-space) silhouette $\partial\mathcal{D}$ [Tagliasacchi et al. 2015], where the transform evaluates to zero for a pixel inside the silhouette. Another algorithm agnostic metric is the *golden energy* from Sharp et al. [2015], but this distance does not encode a monocular Hausdorff like ours do.

Handy dataset. We create the new HANDY tracking dataset for the evaluation of high-precision generative tracking algorithms. Our dataset contains $\approx$ 30k depth and color images recorded with an *Intel RealSense* SR300 sensor. The dataset is designed to cover the entire range of motions that has been surveyed in recent techniques. As detailed in Figure 16, we identified three main axes of complexity in the hand tracking literature, and devised the TEASER dataset to thoroughly sample this space; see Video [02:51].

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

Video [04:53] – TAYL1	rigid and clenching
Video [05:40] – SRID1	fingers extension
Video [05:56] – SRID2	fingers contact
Video [06:15] – SRID3	crossing fingers
Video [06:29] – SRID4	pinching
Video [07:33] – SHAR1	fast and complex
Video [08:07] – SHAR2	fast rigid
Video [08:37] – SHAR3	rotating fist

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

Self evaluation. In Figure 11, we adopt the self-evaluation visualization proposed by [Taylor et al. 2016]. We study the changes in algorithm performance as we disable the tracking energy terms in Eq. 3 on the HANDY/TEASER sequence – in all tests, the *d2m* term is never disabled, as otherwise immediate loss of tracking occurs. Not surprisingly, we identify the *m2d* and *pose* terms to be the ones dominating tracking performance. Similarly to [Taylor et al. 2016], while the contribution of other terms is small, we found that it still yields a visually noticeable improvement.

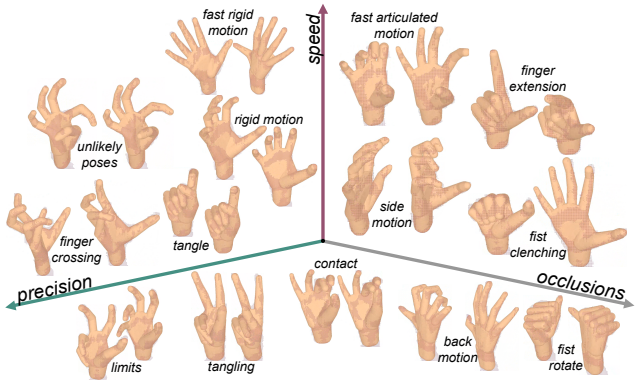

Figure 16: *Our dataset contains a wide range of motions. We identify three main axes of complexity by analyzing recent hand-tracking papers; see Video [02:51]. The distance to the origin indicates the level of tracking difficulty.*

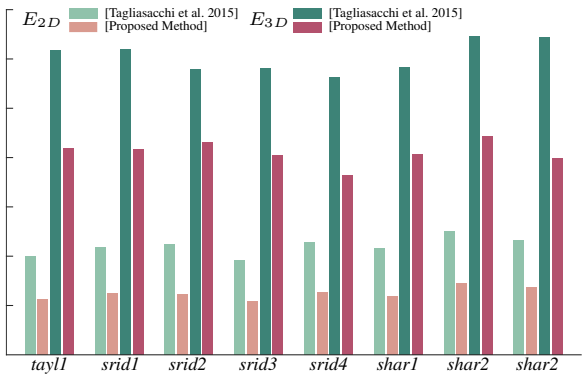

Figure 17: *Average 2D/3D tracking performance metrics of the proposed method compared to [Tagliasacchi et al. 2015]. In the additional material we report error plots through time for the aggregated data above.*

Quantitative comparison. Our algorithm has been tested with the *Intel RealSense* SR300 (QVGA@60Hz). We have tailored the method of [Tagliasacchi et al. 2015] to support this sensor to enable quantitative comparisons. In Figure 14, our two metrics are plotted per-frame as multiple tracking algorithms are executed on the HANDY/TEASER sequence, while Figure 15 reports aggregated errors; see Video [03:53]. It is important to note our metrics are designed to evaluate fitting precision; the method of [Sharp et al. 2015] still achieves good tracking robustness on the test sequences, but the lack of user calibration heavily biases this metric. Aggregated performance comparisons are also reported in Figure 17 for each sequence in the HANDY dataset; see Video [04:50]. These metrics reveal a consistent and significant increase in performance. Figure 12 quantitatively illustrates the tracking benefits of template calibration.

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

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

6 Discussion

Our analysis demonstrates how sphere-meshes tracking templates are particularly well suited for real-time hand-tracking. Our calibration and tracking algorithms are simple to implement, efficient to optimize for, and allow for the geometry to be represented with high fidelity. While the calibration algorithm is currently implemented in Matlab, we are confident real-time performance can be achieved with a simple C++ port of our code. The recently proposed system of Taylor and colleagues [Taylor et al. 2016], has also demonstrated excellent tracking quality. Their formulation employs a triangular mesh model and optimizes it in a semi-continuous fashion. However, as their model is articulated through linear blend skinning, joint collapse artifacts can occur. Conversely, our model is volumetric and naturally overcomes this shortcoming; see Video [01:18]. Although it is difficult to predict whether surface or volumetric models will eventually prevail, we believe the simplicity of our representation will lead to extremely performant articulated tracking algorithms.

Generative tracker. In this paper we demonstrated a generative algorithm that yields unprecedented levels of robustness to tracking failure. We would like to stress that our real-time tracking algorithm is (almost) *purely* generative: a discriminative technique [Qian et al. 2014] is only employed in the first frame for tracking initialization. We believe our robustness is due to the quality of the calibrated model, and to the ability to optimize at a constant 60Hz rate. Discriminative algorithms could still be necessary to compensate for situations where the hand re-appears from complete occlusions, but their role in real-time tracking will diminish as RGBD sensors will start offering imaging at frequencies above $60Hz$. To highlight our high frame-rate dependancy, in Video [11:02] we analyze the performance on the tracker with varying frame-rates (60Hz, 30Hz, 15Hz and 7.5Hz) while the additional material reports the corresponding tracking metrics. In Video [10:30] we further investigate tracking failures that include long phases of total occlusion; note how in these scenarios the mean-pose (Probabilistic PCA) regularizer described in [Tagliasacchi et al. 2015, Eq.6] helps tracking recovery.

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

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

7 Conclusion

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

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

Acknowledgements. We would like to thank the reviewers for their feedback, Andrii Maksai, Sofien Bouaziz and Misha Kazhdan for insightful discussions, Tom Cashman, Jonathan Taylor and Andrew Fitzgibbon for their help generating the quantitative comparisons to [Taylor et al. 2016; Sharp et al. 2015], Chris Wojtan for his help rendering Figure 3, Merlin Nimier-David, Pei-I Chen and Thomas Joveniaux for their help in creating calibration datasets, Matthew Murray, Laura Grondhal, Nicolas Guillemot for their help proofreading the paper. This research is supported by the NSERC Discovery grant #2016-05786 and the Swiss NSF grant #200021-153567.

References

ALBRECHT, I., HABER, J., AND SEIDEL, H.-P. 2003. Construction and animation of anatomically based human hand models. In *Proc. of the Symposium on Computer Animation (SCA)*.

BALLAN, L., TANEJA, A., GALL, J., VAN GOOL, L., AND POLLEFEYS, M. 2012. Motion capture of hands in action using discriminative salient points. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

BLOOMENTHAL, J., AND SHOEMAKE, K. 1991. Convolution surfaces. In *Computer Graphics (Proc. SIGGRAPH)*.

BLOOMENTHAL, J., BAJAJ, C., BLINN, J., CANI, M.-P., ROCKWOOD, A., WYVILL, B., AND WYVILL, G. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LÉVY, B. 2010. *Polygon Mesh Processing*. A. K. Peters.

BOUAZIZ, S., WANG, Y., AND PAULY, M. 2013. Online modeling for realtime facial animation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*.

BUSS, S. R. 2004. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*.

DE LA GORCE, M., FLEET, D. J., AND PARAGIOS, N. 2011. Model-based 3D hand pose estimation from monocular video. *Pattern Analysis and Machine Intelligence (PAMI)*.

DIPIETRO, L., SABATINI, A. M., AND DARIO, P. 2008. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics*.

EROL, A., BEBIS, G., NICOLESCU, M., BOYLE, R. D., AND TWOMBLY, X. 2007. Vision Based Hand Pose Estimation: A Review. *Computer Vision Image Understanding*.

FLEISHMAN, S., KLIGER, M., LERNER, A., AND KUTLIROFF, G. 2015. Icpik: Inverse kinematics based articulated-icp. In *Proc. of the IEEE CVPR Workshops (HANDS)*.

INNMANN, M., ZOLLHÖFER, M., NIESSNER, M., THEOBALT, C., AND STAMMINGER, M. 2016. Volumedeform: Real-time volumetric non-rigid reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

KESKIN, C., KIRAÇ, F., KARA, Y. E., AND AKARUN, L. 2012. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

KHAMIS, S., TAYLOR, J., SHOTTON, J., KESKIN, C., IZADI, S., AND FITZGIBBON, A. 2015. Learning an efficient model of hand shape variation from depth images. *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

KRUPKA, E., VINNIKOV, A., KLEIN, B., BAR HILLEL, A., FREEDMAN, D., AND STACHNIAK, S. 2014. Discriminative ferns ensemble for hand pose recognition. In *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

LOPER, M. M., AND BLACK, M. J. 2014. OpenDR: An approximate differentiable renderer. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

MAKRIS, A., AND ARGYROS, A. A. 2015. Model-based 3d hand tracking with on-line shape adaptation. In *Proc. of British Machine Vision Conference (BMVC)*.

MELAX, S., KESELMAN, L., AND ORSTEN, S. 2013. Dynamics based 3d skeletal hand tracking. In *Proc. of Graphics Interface*.

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHLI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

NEWCOMBE, R. A., FOX, D., AND SEITZ, S. M. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

OBERWEGER, M., WOHLHART, P., AND LEPETIT, V. 2015. Training a Feedback Loop for Hand Pose Estimation. In *Proc. of the Intern. Conf. on Computer Vision (ICCV)*.

OIKONOMIDIS, I., KYRIAZIS, N., AND ARGYROS, A. A. 2011. Efficient model-based 3D tracking of hand articulation using kinect. In *Proc. of British Machine Vision Conference (BMVC)*.

OLSON, M., AND ZHANG, H. 2006. Silhouette extraction in hough space. *Computer Graphics Forum (Proc. of EuroGraphics)*.

PLANKERS, R., AND FUA, P. 2003. Articulated soft objects for multi-view shape and motion capture. *Pattern Analysis and Machine Intelligence (PAMI)*.

QIAN, C., SUN, X., WEI, Y., TANG, X., AND SUN, J. 2014. Realtime and robust hand tracking from depth. In *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

REHG, J. M., AND KANADE, T. 1994. Visual tracking of high dof articulated structures: An application to human hand tracking. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

RHEE, T., NEUMANN, U., AND LEWIS, J. P. 2006. Human hand modeling from surface anatomy. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*.

SCHRODER, M., MAYCOCK, J., RITTER, H., AND BOTSCH, M. 2014. Real-time hand tracking using synergistic inverse kinematics. In *Proc. of the Intern. Conf. on Robotics and Automation (ICRA)*.

SHARP, T., KESKIN, C., ROBERTSON, D., TAYLOR, J., SHOTTON, J., LEICHTER, D. K. C. R. I., WEI, A. V. Y., KRUPKA, D. F. P. K. E., FITZGIBBON, A., AND IZADI, S. 2015. Accurate, robust, and flexible real-time hand tracking. In *Proc. of ACM Special Interest Group on Computer-Human Interaction (CHI)*.

SRIDHAR, S., OULASVIRTA, A., AND THEOBALT, C. 2013. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proc. of the Intern. Conf. on Computer Vision (ICCV)*.

SRIDHAR, S., RHODIN, H., SEIDEL, H.-P., OULASVIRTA, A., AND THEOBALT, C. 2014. Real-time hand tracking using a sum of anisotropic gaussians model. In *Proc. International Conference on 3D Vision (3DV)*.

SRIDHAR, S., MUELLER, F., OULASVIRTA, A., AND THEOBALT, C. 2015. Fast and robust hand tracking using detection-guided optimization. In *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

SUN, X., WEI, Y., LIANG, S., TANG, X., AND SUN, J. 2015. Cascaded hand pose regression. In *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

TAGLIASACCHI, A., AND LI, H. 2016. Modern techniques and applications for real-time non-rigid registration. *Proc. SIGGRAPH Asia (Technical Courses)*.

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

TAGLIASACCHI, A., DELAME, T., SPAGNUOLO, M., AMENTA, N., AND TELEA, A. 2016. 3d skeletons: A state-of-the-art report. *Computer Graphics Forum (Proc. of EuroGraphics)*.

TAN, D. J., CASHMAN, T., TAYLOR, J., FITZGIBBON, A., TARLOW, D., KHAMIS, S., IZADI, S., AND SHOTTON, J. 2016. Fits like a glove: Rapid and reliable hand shape personalization. In *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

TANG, D., YU, T.-H., AND KIM, T.-K. 2013. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proc. of the Intern. Conf. on Computer Vision (ICCV)*.

TAYLOR, J., STEBBING, R., RAMAKRISHNA, V., KESKIN, C., SHOTTON, J., IZADI, S., HERTZMANN, A., AND FITZGIBBON, A. 2014. User-specific hand modeling from monocular depth sequences. In *Proc. of Comp. Vision and Pattern Recog. (CVPR)*.

TAYLOR, J., BORDEAUX, L., CASHMAN, T., CORISH, B., KESKIN, C., SOTO, E., SWEENEY, D., VALENTIN, J., LUFF, B., TOPALIAN, A., WOOD, E., KHAMIS, S., KOHLI, P., SHARP, T., IZADI, S., BANKS, R., FITZGIBBON, A., AND SHOTTON, J. 2016. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*.

TEJANI, A., TANG, D., KOUSKOURIDAS, R., AND KIM, T.-K. 2014. Latent-class hough forests for 3d object detection and pose estimation. In *Proc. of the European Conf. on Computer Vision (ECCV)*.

THIERY, J.-M., GUY, E., AND BOUBEKEUR, T. 2013. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*.

THIERY, J.-M., GUY, E., BOUBEKEUR, T., AND EISEMANN, E. 2016. Animated mesh approximation with sphere-meshes. *ACM Transactions on Graphics (TOG)*.

TOMPSON, J., STEIN, M., LECUN, Y., AND PERLIN, K. 2014. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (TOG)*.

VAILLANT, R., BARTHE, L., GUENNEBAUD, G., CANI, M.-P., ROHMER, D., WYVILL, B., GOURMEL, O., AND PAULIN, M. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*.

VAILLANT, R., GUENNEBAUD, G., BARTHE, L., WYVILL, B., AND CANI, M.-P. 2014. Robust iso-surface tracking for interactive character skinning. *ACM Transactions on Graphics (TOG)*.

WANG, R. Y., AND POPOVIC, J. 2009. Real time hand tracking with a colored glove. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*.

WELCH, G., AND FOXLIN, E. 2002. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Comput. Graph. Appl.*.