Efficient and Precise Interactive Hand Tracking Through Joint, Continuous Optimization of Pose and Correspondences

Jonathan Taylor Lucas Bordeaux* Thomas Cashman* Bob Corish* Cem Keskin* Toby Sharp*
Eduardo Soto*† David Sweeney* Julien Valentin*‡ Benjamin Luff§ Arran Topalian§ Erroll Wood¶
Sameh Khamis Pushmeet Kohli Shahram Izadi Richard Banks Andrew Fitzgibbon Jamie Shotton

Microsoft Research‖ McMaster University† University of Oxford‡ University of Abertay§ University of Cambridge¶

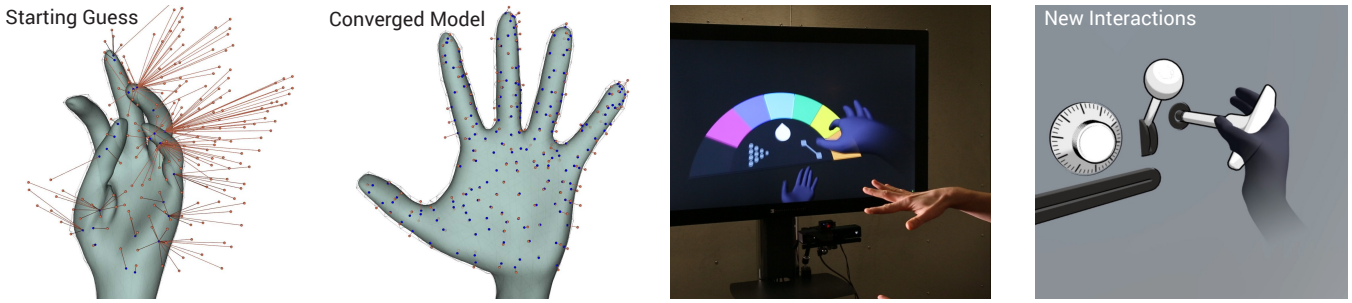


Figure 1: *Our hand tracker uses a smooth model of the hand, allowing a second-order optimizer to jointly search over the high-dimensional space that contains both the hand pose and the model positions that correspond to each depth image pixel. Despite a higher per-iteration cost, this optimization has a wide basin of convergence and yields a real-time system that is both more accurate and more efficient than alternatives, enabling new interactions in virtual reality.*

Abstract

Fully articulated hand tracking promises to enable fundamentally new interactions with virtual and augmented worlds, but the limited accuracy and efficiency of current systems has prevented widespread adoption. Today's dominant paradigm uses machine learning for initialization and recovery followed by iterative model-fitting optimization to achieve a detailed pose fit. We follow this paradigm, but make several changes to the model-fitting, namely using: (1) a more discriminative objective function; (2) a smooth-surface model that provides gradients for non-linear optimization; and (3) joint optimization over both the model pose and the correspondences between observed data points and the model surface. While each of these changes may actually *increase* the cost per fitting iteration, we find a compensating decrease in the number of iterations. Further, the wide basin of convergence means that fewer starting points are needed for successful model fitting. Our system runs in real-time on CPU only, which frees up the commonly over-burdened GPU for experience designers. The hand tracker is efficient enough to run on low-power devices such as tablets. We can track up to several meters from the camera to provide a large working volume for interaction, even using the noisy data from current-generation depth cameras. Quantitative assessments on standard datasets show that the new approach exceeds the state of the art in accuracy. Qualitative results take the form of live recordings of a range of interactive experiences enabled by this new approach.

*Roughly equal contributions ordered alphabetically by last name.

‖All authors were at Microsoft for the majority of the work.

Permission to make digital or hard copies of part or all 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 third-party components of this work must be honored. For all other uses, contact the owner/author(s). © 2016 Copyright held by the owner/author(s).
Publication rights licensed to ACM.

SIGGRAPH '16 Technical Paper, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925965
ACM Reference Format
Taylor, J., Bordeaux, L., Cashman, T., Corish, B., Keskin, C., Sharp, T., Soto, E., Sweeney, D., Valentin, J., Luff, B., Topalian, A., Wood, E., Khamis, S., Kohli, P., Izadi, S., Banks, R., Fitzgibbon, A., Shotton, J. 2016. Efficient and Precise Interactive Hand Tracking Through Joint, Continuous Optimization of Pose and Correspondences. ACM Trans. Graph. 35, 4, Article 143 (July 2016), 12 pages. DOI = 10.1145/2897824.2925965 http://doi.acm.org/10.1145/2897824.2925965.

Keywords: articulated tracking, virtual reality, subdivision surfaces

Concepts: •**Human-centered computing** → **Interaction techniques;** *Virtual reality;*

1 Introduction

Recent rapid advances in display and head-tracking technology is at last bringing virtual reality (VR) and augmented reality (AR) to the consumer mainstream. But perhaps just as important as the ability to *display* virtual objects to the user, is for the user to be able to *interact* with those virtual objects and environments as naturally as possible. In the real world, we use our hands to reach out and touch objects, which react to our interactions according to the laws of physics. Can *uninstrumented* hand tracking allow natural interaction with virtual objects in a physically plausible manner? A further question in VR is the problem of embodiment: how can the system project an accurate (though possibly unrealistic) avatar representation of your body and hands to increase your sense of 'presence'?

Camera-based tracking of the user has long promised technology that could address these questions. The advent of consumer depth cameras with the release of the Kinect system [Shotton et al. 2011] offered a first glimpse outside the laboratory of full-body pose tracking technology. Kinect's tracking proved successful for high-energy scenarios such as video games, but provided only very limited and noisy information about the hands – our primary manipulators. More recently, there has been considerable interest, both academic and commercial, in the task of fully-articulated hand tracking, where the system aims to infer, in real time, a pose vector that accurately describes the detailed motion of a user's hands (see §2).

However, fully articulated hand tracking has not yet become the user interface of choice for AR and VR. We believe that this can be attributed to several factors. **Accuracy and robustness.** Hand tracking from camera input, even depth camera input, is an incredibly hard problem. Occlusions are rife, and the system must rely heavily on priors which cannot work in all situations. Hands rotate

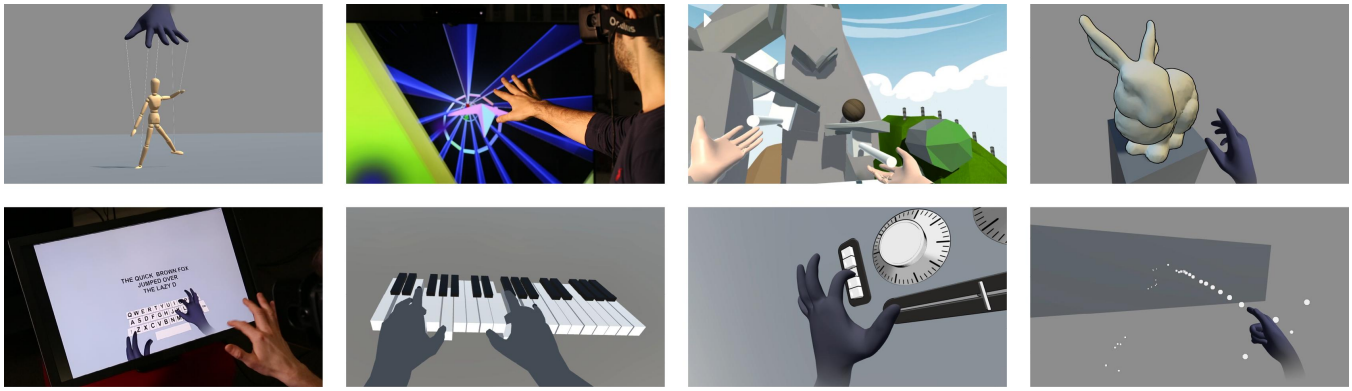


Figure 2: *Our hand tracking system is precise, flexible, and efficient enough to permit designers unskilled in hand tracking algorithms to build a variety of interactive scenarios. A variety of user experiences explore direct physical interaction, pointing, and analog control.*

freely about three axes, radically changing their appearance as they do so. Fingers are difficult to disambiguate locally, thus kinematic constraints must be used to reason about them jointly (e.g. using a hand model). As a result, current systems still have many inaccuracies and failure modes. One particularly difficult obstacle when applying a hand tracker to build end-user experiences is robustness: imagine the frustration of trying to interact with a virtual object if the tracker broke down every few seconds. **Flexibility of setup.** Some systems use sensors that only work for close-range scenarios. If the camera is room-mounted, this can be a limiting factor for VR experiences where the user may be attempting to manipulate virtual objects and environments across a large interaction volume. Head-mounted cameras offer some hope, though are limited in their field of view. **Efficiency.** Many state of the art systems e.g. [Sharp et al. 2015; Tagliasacchi et al. 2015] are computationally expensive. In some cases, the hand tracking algorithm saturates an entire top-end consumer GPU. This makes it difficult to run hand trackers on lower-power devices, or to provide compelling user experiences which may themselves require considerable GPU bandwidth.

Our primary **contribution** is to show that using detailed smooth-surface models in hand tracking allows standard non-linear optimizers such as the Levenberg method to operate in real time with considerably less computational power than existing methods, and with comparable or increased accuracy. This is surprising, because evaluation of these models would superficially appear to be more expensive than simpler models based on geometric primitives. However, the improvements in iteration count and the basin of convergence, coupled with an efficient handling of the Levenberg Jacobian, overcome this expense by allowing for small numbers of iterations and starting points, as well as handling noisy data captured at long range. Our secondary contributions are a new reinitialization strategy, and some novelties in the precise formulation of the energy function.

Our experimental evaluation investigates the effect that these technical contributions have on the accuracy and efficiency of our system. We present quantitative comparisons against several existing state of the art approaches, and show in most cases a considerable improvement in accuracy. We also built a number of user experience prototypes that can be seen in the accompanying video. These demonstrate the promise of articulated hand tracking for VR and AR applications, and have also proved incredibly useful as real-world test harnesses as we developed the tracking algorithm.

Many **limitations** remain. This paper does not claim to have 'solved' hand tracking. Indeed, our limitations section might have simply been copied from Rehg and Kanade [1994], more than two decades ago: 'occlusions and complicated backgrounds'. Today we would say 'heavy occlusion' e.g. fists or extreme viewpoints, and of course we would trivially exclude complete occlusions (hands invisible to the sensor). For 'complicated backgrounds' we would include other surfaces, other hands, and scenes where hand segmentation fails. We are not even claiming to have reached some threshold level with hand tracking, but we are confident that the experiences we are showing are qualitatively better with this tracker than with other academic or commercial trackers. At the same time, having built these experiences, we can tell that we need to do more to improve robustness and latency.

2 Related Work

Classical approaches to hand tracking include wearing data gloves [Dipietro et al. 2008], markers [Wang and Popović 2009; Zhao et al. 2012] or wearable cameras [Kim et al. 2012], but this type of user instrumentation can act as a barrier for unencumbered natural interaction. Therefore the field has moved towards vision-based techniques without direct user instrumentation.

Techniques include *discriminative* approaches [Keskin et al. 2012; Xu and Cheng 2013; Sun et al. 2015], which directly estimate hand pose by extracting image features e.g. using classification and/or regression techniques. These often do not rely on temporal information and attempt to directly estimate the hand joint configuration, thus making them useful as robust reinitializers [Shotton et al. 2011]. Conversely, *generative* (or *model-based*) methods use an explicit hand model to recover pose across a temporal sequence of images [Oikonomidis et al. 2011; Tagliasacchi et al. 2015]. Discriminative approaches are typically robust, but lack the accuracy inherent in model fitting. Generative approaches can suffer from a dependency on the previous frame for initialization, making recovery from errors more challenging. *Hybrid* methods address this by combining discriminative and generative approaches to improve the robustness of frame-to-frame model fitting with per-frame reinitialization [Ballan et al. 2012; Sridhar et al. 2013; Sridhar et al. 2015; Qian et al. 2014; Sharp et al. 2015].

Single RGB-based Approaches Early vision-based systems used monocular RGB cameras, making the problem very challenging (see Erol et al. [2007] for a survey). Understandably most approaches were offline. Discriminative approaches used small databases of restricted hand poses [Athitsos and Sclaroff 2003; Wu and Huang 2000] and had limited accuracy. Generative methods used models with restricted degrees of freedom (DoF), working with simplified hand representations based on 2D, 2.5D or inverse kinematics (IK)

(e.g. [Stenger et al. 2001; Wu et al. 2001]), again resulting in limited accuracy. Bray et al. [2004] used a more detailed 3D hand model. Further user-personalization was added by de La Gorce et al. [2011], who automatically applied a scaling to the bones in the hand model during tracking. Heap and Hogg [1996] provided an early example of online (10Hz) tracking with a simplified deformable hand model. This work, as with other RGB-based methods, struggled with complex poses, changing backgrounds, and occlusions, thus limiting general applicability.

In our work, we use a single commodity depth camera; while such cameras typically also provide RGB data, we focus purely on the depth signal, although incorporation of other sensors is interesting future work.

Multi-camera Performance Capture Possibly the first attempt at unencumbered hand tracking was by Rehg and Kanade [1994], using two video cameras to resolve 27 DoF using a Gauss-Newton solver and local edge search based on a capsule model. Their system ran at 10Hz, with the limitation that the hand should be almost entirely unoccluded. There has also been recent work on high-quality *offline* (non-interactive) performance capture of hands using multi-camera rigs. Ballan et al. [2012] demonstrate high-quality results closely fitting a detailed scanned mesh model to complex two-handed and hand-object interactions. Zhao et al. [2012] use a depth camera, motion capture rig, and markers worn on the user's hand to capture complex single-hand poses, again offline. Wang et al. [2013] and Tzionas et al. [2015] show complex hand-object interactions by using a physics engine, but take minutes per frame. The first multi-camera real-time system was that of Sridhar et al. [2013], where five RGB cameras and a time-of-flight (ToF) sensor are used to track a user's hand at $\sim$10Hz.

The above systems can produce highly accurate results, but are impractical for interactive consumer scenarios due to their computational cost and the heavyweight multi-camera setups.

Single Depth Cameras The advent of consumer depth cameras such as Kinect has made computer vision more tractable, for example through robustness to lighting changes and improved invariance to foreground and background appearance. However, most high-accuracy real-time methods are still extremely computationally demanding (typically using the GPU), making their use on mobile devices still limited.

Generative Approaches: Oikonomidis et al. [2011] present a generative method based on particle swarm optimization (PSO) for full DoF hand tracking (at 15Hz) using a depth sensor. The hand is tracked from a known initial pose, and the method cannot recover from loss of track. Other stochastic optimization schemes have also been explored for real-time hand model fitting but again these lack the benefits of discriminative approaches [Makris et al. 2015]. Melax et al. [2013] use a generative approach driven by a physics solver to generate 3D pose estimates. Both Fleishman et al. [2015] and Tagliasacchi et al. [2015] show impressive high speed tracking results using a fast, articulated variant of ICP. These systems however do not recover from tracking loss. Tagliasacchi et al. [2015] require the user to wear a wristband for hand localization, but have also shown state of the art results for hand tracking.

Discriminative Approaches: Keskin et al. [2012] propose a discriminative method using a multi-layered random forest to predict hand parts and thereby to fit a simple skeleton. The system runs at 30Hz on consumer CPU hardware, but can fail under occlusion. A variety of systems extend this work [Tang et al. 2013; Tejani et al. 2014; Tang et al. 2015; Sun et al. 2015] using novel classification or regression architectures to demonstrate more complex poses. Other discriminative methods include systems which estimate hand pose directly in a single shot using different cascaded learning-based architectures [Xu and Cheng 2013; Oberweger et al. 2015; Li et al. 2015; Neverova et al. 2015; Poier et al. 2015]. Wang et al. [2009; 2011] demonstrate a discriminative nearest-neighbor lookup scheme using a large hand pose database and IK for pose refinement. Nearest-neighbor methods are highly dependent on the database of poses, and can struggle to generalize to unseen poses. Purely discriminative systems do not include an explicit model-fitting step, and so results may often not be kinematically valid (e.g. implausible articulations or finger lengths).

Hybrid Approaches: Qian et al. [2014] extend Oikonomidis et al. [2011] by adding a 'guided' PSO step and a reinitializer that requires fingertips to be clearly visible. Sridhar et al. [2014] use a sum-of-Gaussians representation for efficient model fitting with reinitialization. All these systems so far are based on simple hand models; our approach exploits a full 3D hand surface model that, as shown, is better able to fit to the observed data. Tompson et al. [2014] demonstrate impressive hand tracking results using deep neural networks to predict feature locations and IK to infer a mesh skeleton. While real-time, the approach only tackles close-range scenarios. Sharp et al. [2015] use a detailed mesh model and PSO in combination with a robust fern and forest reinitializer to generate impressive results, including far distances or moving camera scenarios, but with heavy demand of a high-end GPU.

Commercial Systems Beyond this research, there have also been commercial hand tracking systems. The work of Wang et al. [2009; 2011] led to the 3Gear/NimbleVR Systems [2013]. The Intel Perceptual Computing SDK [2016] paired with the RealSense camera provides close-range hand tracking but lacks the flexibility and large working volume of our approach. The original release of the Leap Motion [2013] system demonstrated high frame-rate articulated hand tracking but was limited in the range of poses it supported, and only worked in a very limited depth range. This has been re-engineered for head-mounted scenarios, where the hand can also be assumed to be in a limited depth range, as a new release named Orion [2015]. The system robustly tracks complex poses, even when presented with modest two-handed interaction, which translates into compelling user experiences in VR. There often appears, however, to be a non-negligible mismatch between the image and overlaid skeleton, possibly indicating over-reliance on strong priors or the use of an uncalibrated hand model.

3 Method

Our problem statement is as follows. We have a sensor which provides a stream of depth images, $I^t, t \in \{t_0, t_1, ...\}$ where t is the capture time,[1] and we assume the existence of hand segmentation and fingertip detection algorithms (§3.4) which preprocess I^t into:

- a subsampled list of N **3D points** $\{\mathbf{x}_n^t\}_{n=1}^N$, where N is an algorithm parameter (we show in §4.3 that surprisingly small values work well).
- a corresponding list of **3D normals**, estimated from the data, denoted $\{\mathbf{n}_n^t\}_{n=1}^N$.
- a **segmentation mask**, from which we compute a distance transform $D^t : \mathbb{R}^2 \mapsto \mathbb{R}$, where $D^t(\mathbf{x})$ is the 2D distance from 2D point $\mathbf{x}$ to the hand silhouette (zero inside the silhouette).
- a set of F_t **detected fingertips**, denoted $\{\mathbf{f}_f^t\}_{f=1}^{F_t} \subset \mathbb{R}^3$.

[1] Note that much of our system operates independently from frame to frame so the t superscript will often be dropped.

We parametrize hand pose following Khamis et al. [2015] using a pose vector $\theta \in \mathbb{R}^{28}$ that includes global translation and rotation, one abduction and three flexion variables for each digit, and one abduction and flexion parameter for the wrist/forearm. We also use the model of both hand shape and pose provided by Khamis et al. [2015]. The model parametrizes neutral hand shape using a set of blend shapes and then, given a fixed neutral hand shape, applies a pose θ to produce an appropriately articulated triangular control mesh $P(\theta) \in \mathbb{R}^{3\times M}$ that defines the smooth surface $\mathcal{S}(\theta) \subseteq \mathbb{R}^3$ of the posed hand (see Fig. 3). To find a fixed neutral hand shape, we simply use the technique of Tan et al. [2016] to personalize the model to each subject. The pose function $P(\theta)$ is defined using linear blend skinning [Jacobson et al. 2014] whereas the smooth surface $\mathcal{S}(\theta)$ is a close approximation [Taylor et al. 2014] to the Loop subdivision surface [Loop 1987] that the control mesh $P(\theta)$ would define. Crucially, Taylor et al. [2014] provide a smooth map $S(u;\theta)$ from a piecewise 2D parameter space Ω to $\mathbb{R}^3$, with corresponding surface normal function $S^{\perp}(u;\theta)$, where $\mathcal{S}(\theta) = \{S(u;\theta)|u \in \Omega\}$. Although points in Ω have the somewhat unusual data type $(\texttt{TriangleID}, \texttt{Real}, \texttt{Real})$, it is known that with careful bookkeeping [Taylor et al. 2014], optimization over Ω can be handled just as easily as over $\mathbb{R}^2$.

The goal of hand tracking is to find pose parameters θ^t such that the 3D surface $\mathcal{S}(\theta^t)$ is a 'good explanation' of the image data in I^t and previous frames. This leads us to a formulation of the problem where the concept of 'good explanation' is encapsulated in an energy function $E^t(\theta)$ whose minimum

$$\hat{\theta}^t = \operatorname*{argmin}_{\theta} E^t(\theta)$$

will be output as the system's estimate of the pose at time t. There is always a trade-off between the fidelity of the energy to the true image formation process and the ease with which it may be minimized, but it appears clear that any effective formulation will involve an energy with multiple local minima, so there must be some form of global optimizer. A common strategy is to use multiple starting points, independent of whether E^t itself requires stochastic optimization. In our system, these starting points are given by forward prediction from previous frames, and from a frame-independent reinitializer based on machine learning (see §3.3.2).

3.1 Overall Architecture

In overview, our hand tracking algorithm comprises these steps at each input frame:

1. Preprocess (§3.4)
2. Generate starting points from reinitializer and temporal prediction
3. From each starting point, optimize E
4. Report the pose which yielded the lowest E

The following sections describe these components, beginning with the energy definition and minimization, as these are the core contribution, and then describing preprocessing and reinitialization.

3.2 Energy Function

The energy function is the core of a modern hand tracker, and a key engineering choice is the complexity of the energy. At one extreme is a full 'render and compare' strategy, sometimes described as a 'golden energy' [Sharp et al. 2015]. In the context of hand tracking, however, such energies have so far been optimized only using stochastic search [Oikonomidis et al. 2011; Sharp et al. 2015] or finite-differenced gradients [Tan et al. 2016], with concomitant high computational expense. An improvement in efficiency is obtained by defining a smooth and differentiable renderer [de La Gorce et al. 2011; Loper et al. 2015] and using gradient-based optimization, but to date we know of no sufficiently efficient implementation to determine whether such approaches can be made real-time. Even with an efficient implementation, it remains to be seen whether the energy surface itself is so complex that an excessive number of starting points is required.

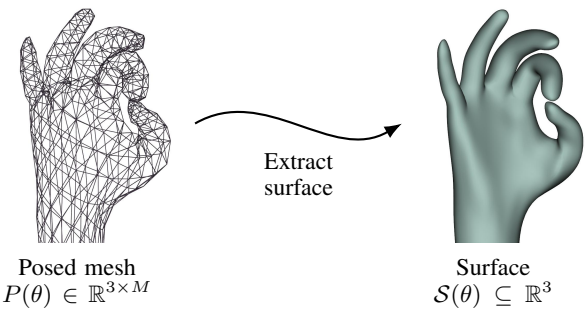


Figure 3: *Our surface model creates a posed mesh using linear blend skinning, which defines a smooth surface (right) using a surface model based on approximate Loop subdivision.*

Our approach is to define a smooth energy function amenable to Gauss-Newton optimization as a weighted sum of several terms, encoding different desiderata of the model fit. These terms are summarized below, with references to recent work that use similar terms.

data	Each data point $\mathbf{x}_n$ should be close to the smooth surface $\mathcal{S}(\theta)$ and have a similar normal at the closest point. [Taylor et al. 2014]
bg	Model points should not project over the background. [Vicente and Agapito 2013]
pose	The pose θ should be a likely human hand pose. [Tagliasacchi et al. 2015; Tan et al. 2016]
limit	The pose θ should obey joint-angle limits. [Khamis et al. 2015]
temp	The temporal sequence of poses should be likely. [Tagliasacchi et al. 2015]
int	The hand model should not self-intersect. [Ballan et al. 2012]
tips	Each detected fingertip $\mathbf{f}_f$ in the data should have a model fingertip nearby. [Sridhar et al. 2013; Tagliasacchi et al. 2015]

Let $\texttt{Terms} = \{\mathsf{bg}, \mathsf{tips}, \mathsf{pose}, \mathsf{limit}, \mathsf{int}, \mathsf{temp}\}$ be the set of non-data term identifiers. Then the overall energy is the weighted sum

$$E(\theta) = E_{\mathsf{data}}(\theta) + \sum_{\tau \in \texttt{Terms}} \lambda_\tau E_\tau(\theta)$$

where the parameters λ_τ are set as described in Table 1. Each term is now defined in detail.

3.2.1 data: Data Term

We penalize the distance from each data point $\mathbf{x}_n$ to the model, and the difference in surface orientation to the associated data normal $\mathbf{n}_n$, using

$$E_{\mathsf{data}}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \min_{u\in\Omega} \frac{\|S(u;\theta) - \mathbf{x}_n\|^2}{\sigma_x^2} + \frac{\|S^{\perp}(u;\theta) - \mathbf{n}_n\|^2}{\sigma_n^2}$$

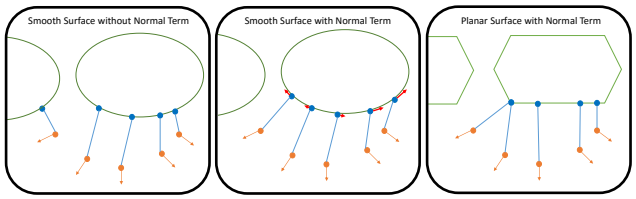


Figure 4: *Illustration of the cross section of the surface (green) of two fingers close to some data (orange) from a single finger. Without the normal term (left) the discrete update causes a correspondence (blue) to select the wrong finger. With the normal term (center) this is not only prevented but forces (shown in red) drive the correspondences towards their correct location, increasing convergence. With a planar model (right) the normal term still allows the discrete update to place the correspondences on the correct finger but these forces are gone, as the surface normal is constant across a face.*

where σ_x^2, σ_n^2 are estimates of the noise variance on points and normals. The normal term allows the energy to use surface orientation to select better locations on the model even when they are far from the data (see Fig. 4).

Relationship to ICP As written, E_{data} involves a search over the entire surface domain Ω for each data point, akin to a closest-point computation in a conventional iterated closest point (ICP) algorithm [Tagliasacchi et al. 2015]. Commonly, when adopting an ICP approach, the model is restricted to some class of geometric primitives such as cones, spheres or triangles, in order that this closest point computation can be conducted efficiently. If instead the algorithm is expressed only in terms of the abstract surface functions S and $S^\perp$, this would need to be implemented as a nonlinear optimization in two variables (a vector in Ω) for each point. In practice, given that the closest-point solution is typically applied over all primitives, and the Ω optimization can be initialized from the nearest vertex, the costs can be quite similar, particularly with the surface-normal terms included.

In our implementation, we avoid these inner iterative steps entirely, using the lifting strategy from Taylor et al. [2014] and Khamis et al. [2015]. Defining

$$\epsilon(u, \theta, \mathbf{x}, \mathbf{n}) := \frac{\|S(u;\theta) - \mathbf{x}\|^2}{\sigma_x^2} + \frac{\|S^\perp(u;\theta) - \mathbf{n}\|^2}{\sigma_n^2} \quad (1)$$

we have

$$E_{\text{data}}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \min_{u\in\Omega} \epsilon(u, \theta, \mathbf{x}_n, \mathbf{n}_n) \quad (2)$$

$$= \frac{1}{N}\sum_{n=1}^{N} \min_{u_n\in\Omega} \epsilon(u_n, \theta, \mathbf{x}_n, \mathbf{n}_n) \quad (3)$$

$$= \min_{\mathcal{U}} E'_{\text{data}}(\theta, \mathcal{U}) \quad (4)$$

where (3) simply renames the variable being minimized over, $\mathcal{U} = \{u_n\}_{n=1}^N$ is a set of now explicitly exposed *correspondences* and

$$E'_{\text{data}}(\theta, \mathcal{U}) = \frac{1}{N}\sum_{n=1}^{N} \epsilon(u_n, \theta, \mathbf{x}_n, \mathbf{n}_n). \quad (5)$$

This allows us to create a 'lifted' energy

$$E'(\theta, \mathcal{U}) = E'_{\text{data}}(\theta, \mathcal{U}) + \sum_{\tau\in\text{Terms}} \lambda_\tau E_\tau(\theta) \quad (6)$$

where for any $\mathcal{U}$

$$E(\theta) \le E'(\theta, \mathcal{U}) \quad (7)$$

and

$$\min_\theta E(\theta) = \min_{\theta,\mathcal{U}} E'(\theta, \mathcal{U}) \,. \quad (8)$$

This lifted energy $E'(\theta, \mathcal{U})$ is easily differentiable with respect to all parameters, making gradient-based minimization possible.

3.2.2 bg: Background Penetration Penalty

Although the data term encourages the model to explain all the foreground data it does nothing to demand that the model not protrude outside the silhouette and into the background. We therefore check a set $\mathcal{U}^{\text{bg}} = \{u_h^{\text{bg}}\}_{h=1}^H \subseteq \Omega$ of H points on our model and penalize any projection of these points outside the silhouette, using the term

$$E_{\text{bg}}(\theta) = \frac{1}{H}\sum_{h=1}^{H} D(\Pi(S(u_h^{\text{bg}};\theta)))^2 \quad (9)$$

where $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection to the image plane and D is the image-space distance transform. See the supplementary material for the locations we choose for $\mathcal{U}^{\text{bg}}$.

3.2.3 pose: Pose Prior

In order to constrain joints to take on reasonable poses when occluded, we use the pose prior provided by Tan et al. [2016]. This consists of a multivariate Gaussian distribution with mean pose μ and covariance matrix Σ. We penalize the negative log likelihood of the pose θ under this distribution as

$$E_{\text{pose}}(\theta) = \frac{1}{22}(\theta - \mu)^\top \Sigma^{-1} (\theta - \mu) \quad (10)$$

which can also be written in a squared form, as shown for a closely related prior by Tagliasacchi et al. [2015]. The six components of θ that parametrize the global transform are not penalized by this prior.

3.2.4 limit: Joint Limit Constraints

In order to restrict our model from taking on anatomically incorrect poses, such as fingers bent backwards, we rely on a vector of joint angle minima $\psi^{\text{min}} \in \mathbb{R}^{22}$ and maxima $\psi^{\text{max}} \in \mathbb{R}^{22}$ provided by Tan et al. [2016]. Note that instead of using box constraints, we instead softly penalize joint angle settings outside of this feasible area using

$$E_{\text{limit}}(\theta) = \frac{1}{22}\sum_{i=1}^{22} \epsilon(\psi_i^{\text{min}}, \psi_i(\theta), \psi_i^{\text{max}})^2 \quad (11)$$

where

$$\epsilon(a, x, b) = \max(0, a - x) + \max(x - b, 0). \quad (12)$$

3.2.5 temp: Temporal Prior

We encourage temporal consistency of poses between frames as a way to smooth the tracker output but also to integrate observations from the previous frame. A constant position model hypothesizes that the current pose should be near to the pose of the previous frame, which we denote θ^{temp}. We encode this prior into the energy term

$$E_{\text{temp}}(\theta) = \frac{1}{28}\sum_{i=1}^{28} \rho(\theta_i - \theta_i^{\text{temp}}, \tau_i) \quad (13)$$

where the robust Geman-McClure kernel $\rho(e,\tau) = \frac{(e/\tau)^2}{1+(e/\tau)^2}$ models the possibility that the pose from the previous frame could be wrong [Geman and McClure 1987] and $\tau_i \in \{\tau_{\text{trans}}, \tau_{\text{rot}}\}$, selecting τ_{trans} for the translational and τ_{rot} for the rotational components of θ.

3.2.6 int: Self-Intersection Penalty

In order to discourage finger self-intersection, we approximate the volume of the fingers in our deformable hand surface using a set of S spheres (see supplementary material). In particular, we let the sth sphere have radius $r_s \in \mathbb{R}$ and location $c_s(\theta) \in \mathbb{R}^3$ specified as an affine combination of model vertices. That is, we define vectors $T = \{T^s\}_{s=1}^S \subset \mathbb{R}^{M\times 1}$ with $\sum_{m=1}^M T_m^s = 1$ for each s. For the set $\mathcal{P} \subseteq \{1,...,S\}^2$ containing all pairs of spheres where both spheres are not in the same or adjacent joints of the same finger, we penalize self-intersection with

$$E_{\text{int}}(\theta) = \frac{1}{|\mathcal{P}|} \sum_{(s,t)\in\mathcal{P}} \max(0, h_{st}(\theta))^2 \tag{14}$$

where

$$h_{st}(\theta) = (r_s + r_t)^2 - \|c_s(\theta) - c_t(\theta)\|^2 \tag{15}$$

measures the amount of penetration between spheres s and t.

3.2.7 tips: Fingertip Term

The detected fingertips (§3.4) are a set of 3D points $\{\mathbf{f}_f\}_{f=1}^F \subset \mathbb{R}^3$. We likewise identify five model positions $\{u_d^{\text{tip}}\}_{d=1}^5 \subset \Omega$ that correspond to the tip of each finger in our model. We then incorporate the fingertip detector into the model-fitting energy using

$$E_{\text{tips}} = \frac{1}{F}\sum_{f=1}^F \text{softmin}(\mathbf{s}_f; \nu_{\text{tips}})^2 \tag{16}$$

where $\mathbf{s}_f \in \mathbb{R}^5$ is the vector filled with the values $\|S(u_d^{\text{tip}};\theta) - \mathbf{f}_f\|^2$ for $d = 1\ldots5$, and softmin is the differentiable operator

$$\text{softmin}(\mathbf{s};\nu) = \frac{\sum_{s\in\mathbf{s}} e^{-s/\nu^2} s}{\sum_{s'\in\mathbf{s}} e^{-s'/\nu^2}} \tag{17}$$

which encourages each of the detections to be explained by a fingertip in our model.

3.3 Energy Minimization

Having defined an energy $E'(\theta,\mathcal{U})$ parametrized in terms of both the pose θ and the data-to-model correspondences $\mathcal{U} \subset \Omega$, we now show how to efficiently minimize this function. We will concatenate the parameters into $\Theta = \begin{bmatrix}\text{vec}(\mathcal{U})^\top & \theta^\top\end{bmatrix}^\top \in \mathbb{R}^{2N+28}$, and add the overload

$$E'(\Theta) = E'(\begin{bmatrix}\text{vec}(\mathcal{U})^\top & \theta^\top\end{bmatrix}^\top) := E'(\theta,\mathcal{U}).$$

Like Tagliasacchi et al. [2015], we have ensured that all terms described above can be represented as a sum of squares, so the energy is in the Gauss-Newton/Levenberg-Marquardt form:

$$E'(\Theta) = \sum_{k=1}^K r_k^2(\Theta) = \|r(\Theta)\|^2 \tag{18}$$

for the vector of residuals $r(\Theta) \in \mathbb{R}^K$. We use the method 'Square-Rooting the Kernel' described by Zach [2014] in order to handle the robust kernels in the energy.

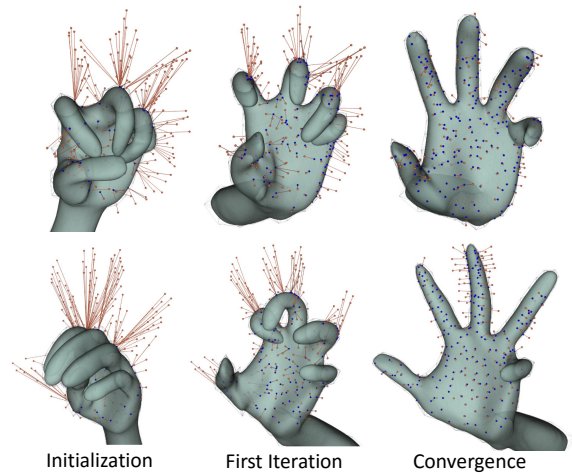


Figure 5: *Examples of our model fitting converging to an energy minimum. Often when initialized far from the true pose the model converges to (presumably) the global minimum (top row) demonstrating the large basin of convergence. In the bottom row the pose is outside that basin as the pinky is too far from its data. Often the first iteration (middle column) dictates the success or failure of the optimization.*

3.3.1 Levenberg Iteration

As we have been careful to make everything differentiable, we can calculate the Jacobian $J(\Theta) \in \mathbb{R}^{K\times(2N+28)}$, which is block sparse (see Fig. 7). We use this to compute a standard Levenberg step using

$$(J(\Theta)^\top J(\Theta) + \gamma I)\Delta\Theta = J(\Theta)^\top r(\Theta) \tag{19}$$

where γ is a damping parameter. A single iteration then updates both the hand pose and correspondences as

$$\Theta \leftarrow \Theta \oplus \Delta\Theta \tag{20}$$

with the $\oplus$ operator defined simply as addition for the pose components of Θ, and the mesh-aware update from Taylor et al. [2014] for the correspondence components, which correctly handles the transitions between the triangular faces of Ω.

Due to the highly sparse and regular structure of the Hessian approximation, we can solve (19) in a very efficient manner by exploiting a Schur complement, as often used in bundle adjustment [Triggs et al. 2000], which observes that in the block-matrix linear system

$$\begin{bmatrix} C & E \\ E^\top & B \end{bmatrix}\begin{bmatrix}\Delta\mathcal{U} \\ \Delta\theta\end{bmatrix} = \begin{bmatrix} c \\ b\end{bmatrix}, \tag{21}$$

we can first solve the symmetric positive definite system for $\Delta\theta$ as

$$(B - E^\top C^{-1} E)\Delta\theta = b - E^\top C^{-1} c\,. \tag{22}$$

Note that $C \in \mathbb{R}^{2N\times 2N}$ is composed of 2×2 blocks along the diagonal and thus C^{-1} can easily be computed by inverting each block independently. Then $\Delta\mathcal{U}$ is obtained cheaply from $\Delta\mathcal{U} = C^{-1}(c - E\Delta\theta)$. Similarly, the structure of these matrices allows for efficient computation of the matrix multiplications above. (22) requires only the decomposition (e.g. Cholesky) of a constant-sized matrix, and thus the algorithm scales linearly with N.

The correspondences now contribute only a small computational cost to each iteration, comparing very favourably with ICP, where a closest-point operation must be performed on each iteration for each data point.

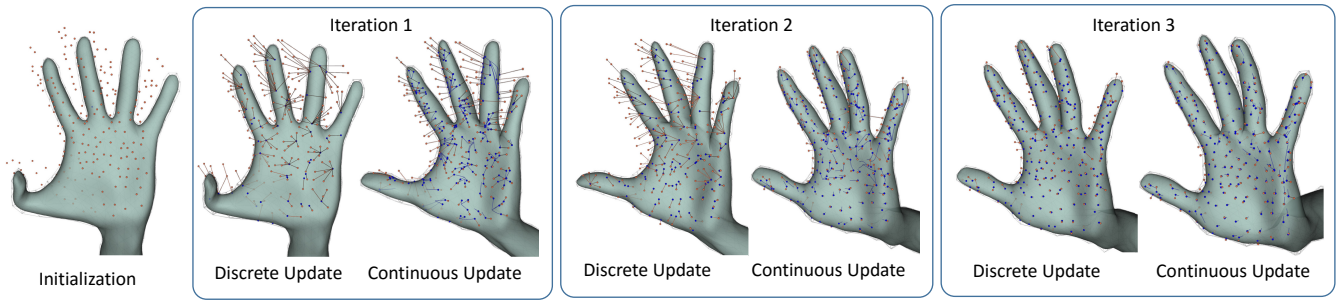


Figure 6: *A typical example of our model-fitting converging to the correct pose. Orange dots are data points $\{\mathbf{x}_n\}_{n=1}^N$. Each correspondence u_n is represented by its blue surface point $S(u_n;\theta)$.* **Initialization:** *The model begins in a pose extrapolated from previous frames or supplied by our discriminative predictor (§3.3.2).* **Iteration 1:** *In the first discrete update, each of the data points is assigned a correspondence from a finite set of model points (§3.3.3). Despite the coarseness of this set, the continuous update (§3.3.1) is able to make large corrections (blue trails from old to new) to these correspondences while simultaneously adjusting the hand pose.* **Iteration 2:** *Many of the grossly misassigned correspondences are corrected by the discrete update allowing the continuous update to make substantial progress towards the true pose.* **Iteration 3:** *The third iteration corrects the remaining correspondences and slides the model further towards the true pose.*

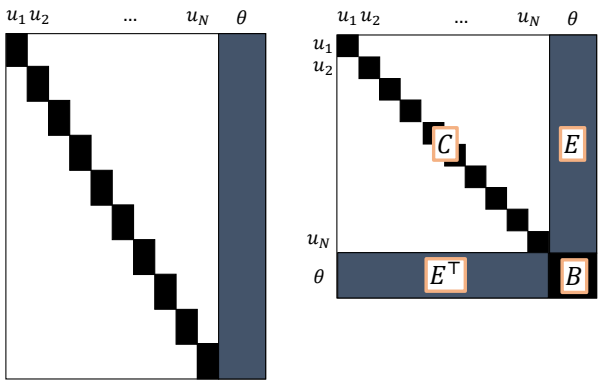


Figure 7: *Left: The Jacobian J has a block sparse structure composed of N blocks of size 6×2 with 28 mostly-dense columns on the right. Right: The Hessian approximation has a block sparse arrowhead structure composed of 2×2 blocks and a mostly dense set of rows and columns forming the arrowhead.*

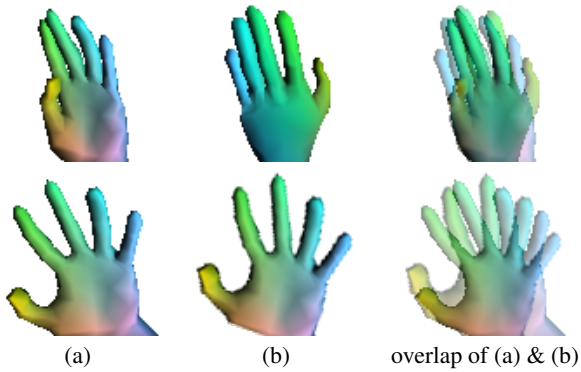


Figure 8: *Visual intuition for the two distances relevant to our reinitializer. Top: Flipping a hand generates an image similar in appearance space, but distant in pose space. The retrieval forest therefore returns a lot of seeds that are flipped versions of the ground truth. A graph building metric that only relies on pose cannot recover from such local minima. Bottom: Retrieval forests can often match hands to their slightly rotated versions. Here, the distance in pose space is small, but the distance in appearance space is large, because the fingers do not overlap.*

3.3.2 Pose Initialization

Since the energy function is highly non-convex, we hope to initiate a local gradient-based optimization from at least one starting point that is sufficiently close to the global minimum for it to be found. There are two sources of starting points for pose parameters. The first uses the predictor

$$\theta^{\text{tracked}} = \theta^{(1)} + \frac{t^{(1)}}{t^{(2)} - t^{(1)}}\left(\theta^{(1)} - \theta^{(2)}\right) \tag{23}$$

where $\theta^{(1)}$ and $\theta^{(2)}$ were the tracked poses for those frames that arrived $t^{(1)}$ and $t^{(2)} > t^{(1)}$ time units earlier than the current frame.[2] When a constant velocity prediction is not available, the constant position prediction $\theta^{\text{tracked}} = \theta^{(1)}$ is substituted instead.

The second is a set of reinitialization poses quickly estimated for each frame. This is an instance of the 'multi-output learning' problem [Guzmán-Rivera et al. 2014], for which we use a modification of a recently-proposed method [Valentin et al. 2016]. This algorithm generates multiple hypotheses for hand poses given a preprocessed and cropped hand image. It quickly generates candidate hand poses using what is coined a 'Retrieval Forest'. Those candidates are then refined using 'Navigation Graphs' to generate a small number of pose proposals. We successfully adapted the algorithm proposed by Valentin et al. [2016] despite the complication that similar appearances do not always imply nearby hand poses and vice versa (see Fig. 8 for some examples). Instead of learning a global translation directly, we instead learned a relative translation from a pre-processed and cropped hand image which could then be mapped into a global translation. For the optimal algorithm settings ($c = 40$, $n = 10$) the search concludes in under 5ms.

3.3.3 Discrete Update of $\mathcal{U}$

While initial estimates for θ are provided by temporal extrapolation and reinitializers, we also need to initialize the correspondences $\mathcal{U}$. A regressor could also be used here [Taylor et al. 2012], but in this

[2] In (23) we implement the addition operator for the global rotation component to maintain angular velocity.

σ_x	σ_n	λ_{bg}	λ_{pose}	λ_{limit}
3mm	0.25	0.1	1	10^2

λ_{temp}	τ_{trans}	τ_{rot}	λ_{int}	λ_{tips}	ν_{tips}
0.15	2cm	4°	10^4	40	2cm

Table 1: *Parameters used for all our experiments and experiences.*

work we perform a simple discrete search. In particular, a set of Q proposal correspondences $\mathcal{U}_i^{\text{prop}} = \{u_{iq}^{\text{prop}}\}_{q=1}^{Q} \subseteq \Omega$ is predefined offline for each iteration i. Before the ith Levenberg step, each entry u_n in $\mathcal{U}$ is updated according to

$$u_n \leftarrow \underset{u \in \mathcal{U}_i^{\text{prop}} \cup \{u_n\}}{\operatorname{argmin}} \epsilon(u_n, \theta, \mathbf{x}_n, \mathbf{n}_n). \tag{24}$$

Note that this update strictly reduces the energy (6) as each correspondence u_n is allowed to retain its old value. This is not only valuable in providing a good initialization but allows correspondences to jump from finger to finger (see Fig. 6), which is a more global, but coarse-grained, update than a gradient-based optimizer can provide. Although we use random sampling to predefine each $\mathcal{U}_i^{\text{prop}}$ (see supplementary material), we leave it as future work to investigate better heuristics, such as using more proposals on the fingers, or regressing the set using features based on the current pose, set of correspondences and image.

3.4 Preprocessing

The preprocessing stages are relatively standard. The hand position is detected using the Kinect body tracker [Shotton et al. 2011]. This position is then used as a seed to grow a segmentation of the hand using a connected components algorithm where neighbouring pixels close in depth are considered to be connected. As pixels within a bounding volume of 19cm centered at the seed pixel are considered for inclusion, pixels from the user's forearm are often included; it is therefore important that our model also has a forearm to explain these pixels.

We then sample the N pixels using a stochastic approximation to farthest-point sampling which simply selects the furthest from 5 candidates to the existing sample set [Mitchell 1991]. The depth of each sampled pixel is used to calculate its 3D position yielding the set of data points $\{\mathbf{x}_n\}_{n=1}^{N} \subseteq \mathbb{R}^3$. For each pixel $\mathbf{x}_n$, we additionally compute a data normal $\mathbf{n}_n \in \mathbb{R}^3$ by first smoothing the depth image and normalizing the average of a set of normalized cross products in a region around the pixel.

Finally, we employ a simple variant of the fingertip detection algorithm of Qian et al. [2014] to find a set of likely fingertips in the image and likewise back-project them to get a set of F positions $\{\mathbf{f}_f\}_{f=1}^{F} \subset \mathbb{R}^3$.

4 Experiments

In this section we evaluate many aspects of our tracker, comparing to the state of the art, investigating the trade-off between accuracy and efficiency, demonstrating the effect of various terms in our energy, and discussing the user experiences shown in our video. All experiments and demonstrations in the accompanying video use the weight settings given in Table 1.

4.1 Metrics

Marker Error We quantify the error of predicted against hand-annotated 3D marker positions, such as the tip of a finger or the location of a finger joint. For this metric we report the number of frames for which either the average or the maximum marker position error was less than a certain threshold (see e.g. Taylor et al. [2012] or Sharp et al. [2015] for more details). Note that the ground truth marker positions have a considerable degree of annotation error, so it is impossible and undesirable to achieve perfect accuracy. Since our algorithm predicts the hand pose, not the markers directly, we must define a mapping from the pose to the required markers. Like Tan et al. [2016], we solve for the affine combination of just four vertices of the posed mesh that best predicts the marker on an equally-spaced 5% subset of the frames in each sequence. Note that the flexibility of this mapping is extremely limited and this procedure simply automates an otherwise manual and biased process.

Classification Error A less prevalent metric based on classifying pixels as belonging to one of several parts of the hand (index finger, palm, etc.) was introduced by Sharp et al. [2015] with the FINGERPAINT dataset. The metric counts the percentage of frames where either the average or the maximum pixel classification error rate falls below a certain threshold. In contrast to marker-based metrics, this metric measures the system's ability to fully and accurately explain every pixel. As such this metric also serves as a proxy for evaluating the validity of the generative model used.

Energy We also report some results on energy values. Since the lifted energy $E'(\theta, \mathcal{U})$ is largely dominated by the current setting of the latent variables $\mathcal{U}$, we use the unlifted energy $E(\theta) = \min_{\mathcal{U}} E'(\theta, \mathcal{U})$. This is calculated by performing a single closest-point update, which we define as a discrete update followed by a continuous optimization that minimizes $\mathcal{U}$ holding θ fixed.

4.2 Comparison to State of the Art

Dexter Dataset In Fig. 9 we compare the accuracy of our tracker with several state of the art methods on the DEXTER dataset [Sridhar et al. 2013]. For all methods we compute per-frame errors for the labeled fingertip markers on the same subset of 2,931 out of 3,155 frames that Sridhar et al. use for error evaluation, and the results are normalized so that each of the 7 sequences in the dataset has equal weight. For our tracker, we use the personalized shape model described by Tan et al. [2016] for the single subject in the Dexter dataset.

Fig. 9 shows that on this dataset, our tracker has much higher accuracy than Sridhar et al. [2015] and Tan et al. [2016]. Our CPU-only tracker achieves almost identical results to those of Tagliasacchi et al. [2015] which requires using a high-end GPU (they report using a Nvidia GTX980). Looking at the qualitative results (see accompanying video), we believe that both our results and those of Tagliasacchi et al. [2015] are operating essentially at the level of human labeling accuracy. We attempted to perform a direct comparison with this implementation to confirm the differing computational requirements of our tracker, but unfortunately it proved difficult to obtain the discontinued Senz3D depth camera that their tracker requires.

NYU Dataset In Fig. 9 we also compare our tracker to a range of recent papers on hand pose estimation by using the NYU dataset [Tompson et al. 2014]. We report results for just the first test sequence of 2,440 frames in order to compare with Tagliasacchi et al. [2015] who did not evaluate their method on the whole test set. The results originally published by Tompson et al. [2014] include

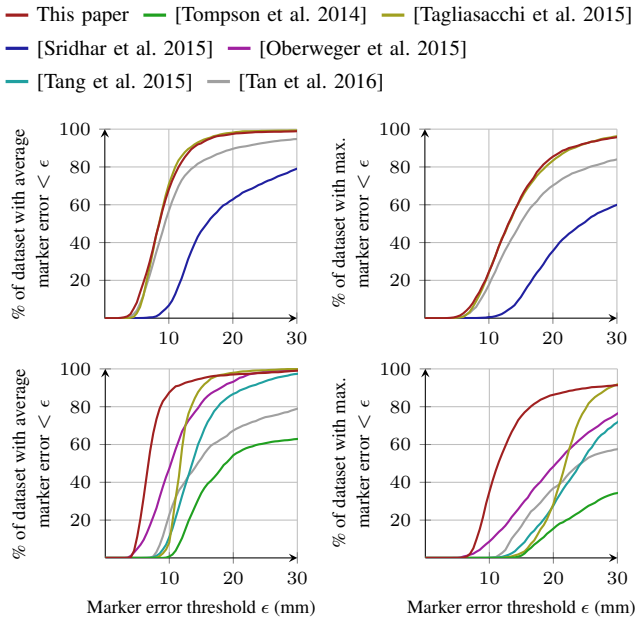


Figure 9: *Comparison to the state of the art on* DEXTER *(top) and* NYU *(bottom).*

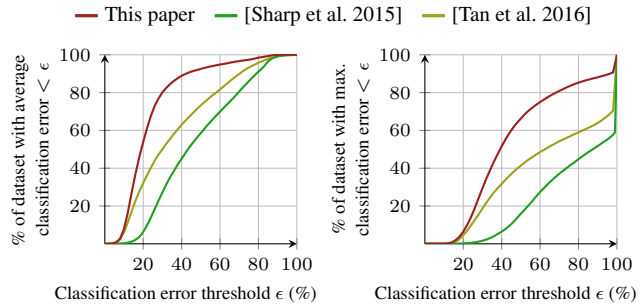


Figure 10: *Comparison to the state of the art on* FINGERPAINT

positions for three locations on the palm that several methods find it difficult to localize accurately; to avoid these, we use a common subset of ten joint positions (two per digit) to compute error in each frame. Our tracker again uses a personalized model for the only subject shown in these frames of the NYU dataset, using the method described by Tan et al. [2016]. On this dataset, our tracker appears to be substantially more accurate than the alternatives.

FingerPaint Dataset In Fig. 10 we compare on the FINGERPAINT dataset [Sharp et al. 2015]. Tan et al. [2016] showed significant improvements over Sharp et al. by using a detailed mesh personalized to each user. However our results, that use the same personalized models, demonstrate further improvements in tracking robustness, despite the compromises made in our generative model to obtain differentiability.

4.3 Computational Efficiency

A key contribution of this paper is the demonstration that both our smooth surface model and joint optimization strategy not only contribute to the state of the art accuracy shown above, but also allow us to largely maintain this accuracy when we enter a low compute scenario. There are three variables that dominantly determine the amount of compute our model fitting procedure requires: (i) the number of data points in our data term; (ii) the number of starting points we initialize the Levenberg algorithm from; and (iii) the number of Levenberg iterations we perform for each starting point.

We found that when a powerful desktop was available we could run 10 starting points for 10 iterations on 192 data pixels while reserving the entire GPU and enough CPU for the demanding VR experiences shown in our videos. This is also the default setting we use for all experiments. As we will see, however, we find that we can scale our tracker down (e.g. to run on a tablet – see video) by adjusting these parameters without a drastic decrease in accuracy.

We demonstrate this graceful degradation in comparison to two different modifications of our algorithm.

- ICP: Each iteration of our algorithm is replaced by a closest point update followed by a continuous optimization of θ holding $\mathcal{U}$ fixed.
- Planar: Our smooth subdivision surface model is replaced by a piecewise planar triangular surface. That is we set $S(u;\theta) \in \mathbb{R}^3$ to be a linear combination of the three vertices u lies on.

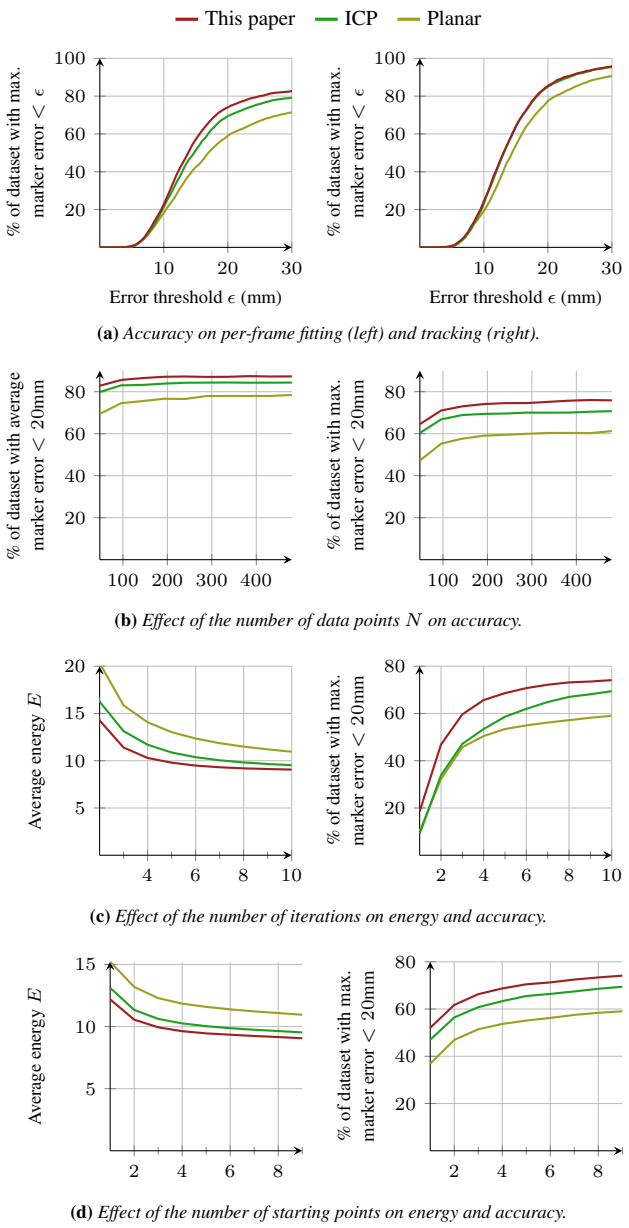
We compare our approach with the ICP and Planar variants using the maximum error metric on DEXTER in Fig. 11a, showing both per-frame and tracking results. The per-frame results show a considerable improvement of our approach over the alternatives. This is important for recovery during rapid motions omnipresent in real world applications such as those presented in §4.6. The improvement is somewhat diluted in the tracked results where our large basin of convergence is less important.

To demonstrate our ability to gracefully trade a small amount of accuracy for lower compute, we investigate varying the parameters that influence computational cost. Fig. 11b shows that our accuracy is remarkably invariant to the number of pixels considered. In Fig. 11c and 11d we show that we can achieve higher accuracy using far fewer iterations and starting points than ICP and Planar. We believe this is facilitated by the smoothness of our energy combined with joint optimization over pose and correspondences, which takes advantage of being able to rotate portions of the surface while sliding the correspondences to compensate (see Fig. 6).

4.4 Evaluation of Energy Terms

Our energy includes a variety of terms and it is important to convey the reasons for the inclusion of each and how we decided on a single set of weights used across all datasets. The weights themselves were set through cursory parameter sweeping combined with modifications that we felt improved the 'feel' of our live hand tracker. Given our desire for the latter, and our use of a single set of weights for all datasets and cameras, we believe that the results presented are not the result of overfitting.

In Fig. 12 we use DEXTER to experiment with turning off one energy term at a time. In order to avoid having the initialization from the previous frame hide details, we only use our reinitializer (see Fig. 11a for the effect of E_{temp} when tracking). Apart from the positional component data_x of the data term, the terms which have the largest effect on model fitting accuracy are the pose prior and the normal component, data_n. The other terms seem to make only a small difference under this metric; nonetheless, we find these terms help the qualitative feel of the tracker for which quantitative metrics may not be a good proxy.



(a) *Accuracy on per-frame fitting (left) and tracking (right).*

(b) *Effect of the number of data points N on accuracy.*

(c) *Effect of the number of iterations on energy and accuracy.*

(d) *Effect of the number of starting points on energy and accuracy.*

Figure 11: *Effect of alternative model fitting strategies and optimizer configurations when fitting to* DEXTER. *Apart from the right-hand plot in* (a)*, every plot shows the results of fitting to each frame individually.*

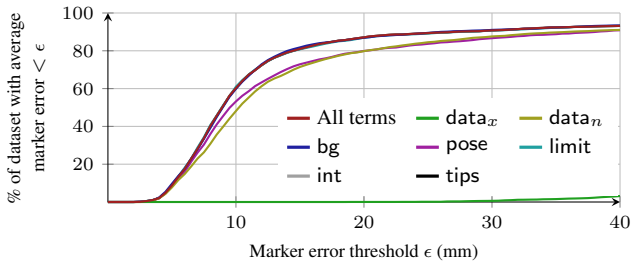


Figure 12: *The effect of setting the weight for each non-temporal term of our energy function E to zero when fitting to each frame of* DEXTER *without tracking, compared to the accuracy with all terms included. See Section 3.2 for a description of each term.*

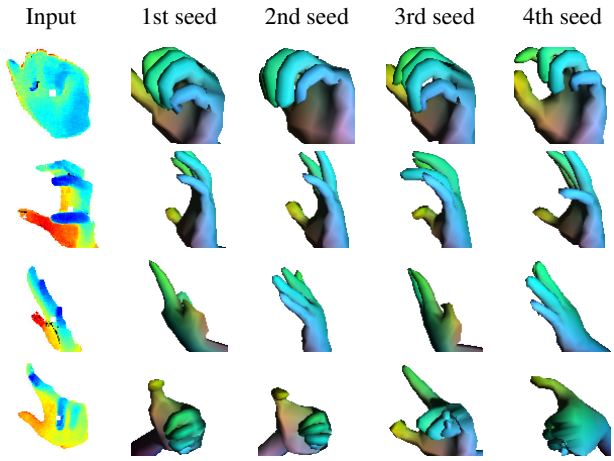


Figure 13: *Our reinitializer is able to estimate reasonable hand poses for many configurations (top two rows). The bottom two rows show failure cases.*

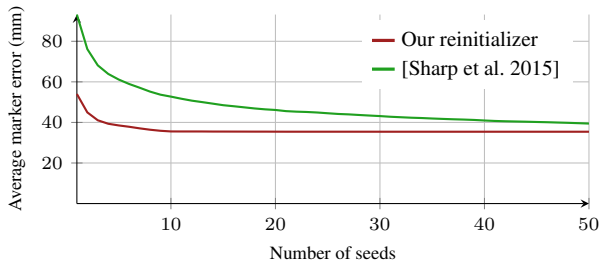


Figure 14: *Comparison of reinitializer strategies on* DEXTER.

4.5 Reinitializer

Our reinitializer is capable of estimating plausible seeds for a wide variety of hand poses as illustrated in Fig. 13. The most significant source of errors come from retrieved poses that are flipped, such as a flat hand facing forward being predicted as facing away.

To demonstrate the precision of our reinitializer, we compare it against the reinitializer from Sharp et al. [2015], which is based on a combination of discriminative fern ensembles [Krupka et al. 2014] and decision jungles [Shotton et al. 2013]. Fig. 14 shows a significant improvement, regardless of the number of seeds. Using only 9 seeds (the real-time setting used for our results) our best seed is on average 17mm closer to the ground truth compared to the best seed of Sharp et al. [2015]. Our reinitializer also runs at a similar speed to that of Sharp et al. [2015].

4.6 Experiences

We demonstrate a number of diverse user experience prototypes in the accompanying video and Figs. 1 and 2. These include examples of discrete gesture recognition (such as a thumb click to engage or disengage), retargeting the hand to new 3D hand avatars, painting, pointing, 3D thumb stick control, and the exciting frontier of physics-based interaction, including: prototypes of piano playing and typing, controlling a marionette via virtual strings attached to the fingers, deforming a 3D model, scratching records, and interacting with a 3D GUI through pure physics. We also demonstrate a play-through of a prototype game experience, codenamed 'Huge', that combines many

of the individual interaction metaphors into a coherent experience. Most of these experiences were built in Unity hooked up to the Oculus DK2 VR headset and using a Kinect V2 camera for input.

Several aspects set these experiences apart from existing work. First, their number and diversity: the level of precision and robustness of our system means that it was extremely easy to put each experience together. As an example, the typing experience took under an hour for a designer with no technical background in hand tracking to create. Our ability to track the hands reliably at several meters from the camera enables a much larger working volume that is typically possible (e.g. compared to a Leap Motion sensor) meaning that there is less risk of the user accidentally breaking the experience by moving their hands out of view. Finally, the Huge demonstrator shows for the first time a coherent end-to-end user experience, and highlights how our system can reliably distinguish between several interaction modalities across multiple minutes of gameplay.

An ongoing concern regarding virtual object manipulation is the importance of haptic feedback. We believe there are several ways to address this. First, with the stereo disparity cues offered by VR headsets, it is easy to position your hand relative to virtual objects, and visual and auditory cues can then indicate interaction events to the user. We were genuinely surprised how well this works and how natural this feels. Second, one can design the experience such that affordances are thin and the user ends up making a pinch. Finally, research in in-air haptics continues apace [Monnai et al. 2014; Ultrahaptics Ltd 2013].

5 Conclusion

We have presented a new system for detailed articulated tracking of the human hand. Our approach combines a smooth, differentiable surface model with a new energy function that is continuously optimized jointly over both pose and correspondences. The optimization exhibits a large basin of convergence, meaning that fewer starting points need to be explored. Further, despite a possibly higher per-iteration cost, convergence is much faster than alternating algorithms such as ICP. These technical contributions result in quantitative accuracy that exceeds that of multiple competing systems across a number of datasets, at a much lower computational cost. The resulting system runs in real-time on lightweight devices without any GPU compute. Finally we have shown compelling user experience prototypes that demonstrate the exciting range of possibilities that articulated hand tracking will enable. We believe our approach represents a significant milestone in the problem of fitting deformable surface models to observed data.

Acknowledgements

This paper makes advances in both technology and design, and the co-evolution of these areas was crucial in building a compelling system. On the engineering side, Taylor, Bordeaux, Cashman, Sharp, Wood, Khamis, Fitzgibbon, and Shotton contributed the mathematical and technical advances enabling the subdivision-surface tracker, while Keskin, Valentin, Kohli, and Izadi contributed the machine-learning front end that advances over their previous work [Valentin et al. 2016]. On the design side, Soto, Sweeney, Corish, and Banks envisioned and built many of the core interactions, while Soto, Topalian, and Luff designed and built the full playable game concept 'Huge'. We would also like to thank Federica Bogo, Alina Kuznetsova, Ian Ormesher, Abigail Sellen, Srinath Sridhar, and Richard Stebbing for helpful discussions, technical assistance, and engineering.

References

3GEAR SYSTEMS INC, 2013. Gesture recognizer. `http://threegear.com`, Jan.

ATHITSOS, V., AND SCLAROFF, S. 2003. Estimating 3D hand pose from a cluttered image. In *Proc. CVPR*, vol. 2, II–432.

BALLAN, L., TANEJA, A., GALL, J., GOOL, L. V., AND POLLEFEYS, M. 2012. Motion capture of hands in action using discriminative salient points. In *Proc. ECCV*, 640–653.

BRAY, M., KOLLER-MEIER, E., AND VAN GOOL, L. 2004. Smart particle filtering for 3D hand tracking. In *Proc. Automatic Face and Gesture Recognition*, 675–680.

DE LA GORCE, M., FLEET, D. J., AND PARAGIOS, N. 2011. Model-Based 3D Hand Pose Estimation from Monocular Video. *IEEE Trans. PAMI 33*, 9, 1793–1805.

DIPIETRO, L., SABATINI, A. M., AND DARIO, P. 2008. A survey of glove-based systems and their applications. *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews 38*, 4, 461–482.

EROL, A., BEBIS, G., NICOLESCU, M., BOYLE, R. D., AND TWOMBLY, X. 2007. Vision-based hand pose estimation: A review. *CVIU 108*, 1-2, 52–73.

FLEISHMAN, S., KLIGER, M., LERNER, A., AND KUTLIROFF, G. 2015. ICPIK: Inverse kinematics based articulated-ICP. In *Proc. CVPR Workshops*, 28–35.

GEMAN, S., AND MCCLURE, D. E. 1987. Statistical methods for tomographic image reconstruction. *Bulletin of the International Statistical Institute 52*, 4, 5–21.

GUZMÁN-RIVERA, A., KOHLI, P., GLOCKER, B., SHOTTON, J., SHARP, T., FITZGIBBON, A. W., AND IZADI, S. 2014. Multi-output learning for camera relocalization. In *Proc. CVPR*, 1114–1121.

HEAP, T., AND HOGG, D. 1996. Towards 3D hand tracking using a deformable model. In *Proc. Automatic Face and Gesture Recognition*, 140–145.

INTEL CORPORATION, 2016. RealSense SDK. `http://software.intel.com/realsense`, Jan.

JACOBSON, A., DENG, Z., KAVAN, L., AND LEWIS, J. 2014. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*, #24.

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. ECCV*, 852–863.

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. In *Proc. CVPR*, 2540–2548.

KIM, D., HILLIGES, O., IZADI, S., BUTLER, A. D., CHEN, J., OIKONOMIDIS, I., AND OLIVIER, P. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proc. UIST*, 167–176.

KRUPKA, E., BAR HILLEL, A., KLEIN, B., VINNIKOV, A., FREEDMAN, D., AND STACHNIAK, S. 2014. Discriminative ferns ensemble for hand pose recognition. In *Proc. CVPR*, 3670–3677.

LEAP MOTION INC, 2013. Motion Controller. `http://leapmotion.com/product`, Jan.

LEAP MOTION INC, 2015. Orion. `http://developer.leapmotion.com/orion`, Feb.

LI, P., LING, H., LI, X., AND LIAO, C. 2015. 3D hand pose estimation using randomized decision forest with segmentation index points. In *Proc. ICCV*, 819–827.

LOOP, C. T. 1987. *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis, University of Utah.

LOPER, M., MAHMOOD, N., ROMERO, J., PONS-MOLL, G., AND BLACK, M. J. 2015. SMPL: a skinned multi-person linear model. *ACM Trans. Graphics 34*, 6, #248.

MAKRIS, A., KYRIAZIS, N., AND ARGYROS, A. 2015. Hierarchical particle filtering for 3D hand tracking. In *Proc. CVPR Workshops*, 8–17.

MELAX, S., KESELMAN, L., AND ORSTEN, S. 2013. Dynamics based 3D skeletal hand tracking. In *Proceedings of the 2013 Graphics Interface Conference*, 63–70.

MITCHELL, D. P. 1991. Spectrally optimal sampling for distribution ray tracing. In *Proc. SIGGRAPH*, 157–164.

MONNAI, Y., HASEGAWA, K., FUJIWARA, M., YOSHINO, K., INOUE, S., AND SHINODA, H. 2014. HaptoMime: Mid-air haptic interaction with a floating virtual screen. In *Proc. UIST*, 663–667.

NEVEROVA, N., WOLF, C., NEBOUT, F., AND TAYLOR, G. 2015. Hand pose estimation through weakly-supervised learning of a rich intermediate representation. *arXiv preprint 1511.06728*.

OBERWEGER, M., WOHLHART, P., AND LEPETIT, V. 2015. Training a feedback loop for hand pose estimation. In *Proc. ICCV*, 3316–3324.

OIKONOMIDIS, I., KYRIAZIS, N., AND ARGYROS, A. 2011. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proc. BMVC*, 101.1–101.11.

POIER, G., RODITAKIS, K., SCHULTER, S., MICHEL, D., BISCHOF, H., AND ARGYROS, A. A. 2015. Hybrid one-shot 3D hand pose estimation by exploiting uncertainties. In *Proc. BMVC*, 182.1–182.14.

QIAN, C., SUN, X., WEI, Y., TANG, X., AND SUN, J. 2014. Realtime and robust hand tracking from depth. In *Proc. CVPR*, 1106–1113.

REHG, J. M., AND KANADE, T. 1994. Visual tracking of high DOF articulated structures: an application to human hand tracking. In *Proc. ECCV*, 35–46.

SHARP, T., KESKIN, C., ROBERTSON, D., TAYLOR, J., SHOTTON, J., KIM, D., RHEMANN, C., LEICHTER, I., VINNIKOV, A., WEI, Y., FREEDMAN, D., KOHLI, P., KRUPKA, E., FITZGIBBON, A., AND IZADI, S. 2015. Accurate, robust, and flexible realtime hand tracking. In *Proc. CHI*, 3633–3642.

SHOTTON, J., FITZGIBBON, A., COOK, M., SHARP, T., FINOCCHIO, M., MOORE, R., KIPMAN, A., AND BLAKE, A. 2011. Real-time human pose recognition in parts from a single depth image. In *Proc. CVPR*, 1297–1304.

SHOTTON, J., SHARP, T., KOHLI, P., NOWOZIN, S., WINN, J., AND CRIMINISI, A. 2013. Decision jungles: Compact and rich models for classification. In *NIPS*, 234–242.

SRIDHAR, S., OULASVIRTA, A., AND THEOBALT, C. 2013. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proc. ICCV*, 2456–2463.

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. 3DV*, 319–326.

SRIDHAR, S., MUELLER, F., OULASVIRTA, A., AND THEOBALT, C. 2015. Fast and robust hand tracking using detection-guided optimization. In *Proc. CVPR*, 3213–3221.

STENGER, B., MENDONÇA, P. R., AND CIPOLLA, R. 2001. Model-based 3D tracking of an articulated hand. In *Proc. CVPR*, vol. 2, II–310.

SUN, X., WEI, Y., LIANG, S., TANG, X., AND SUN, J. 2015. Cascaded hand pose regression. In *Proc. CVPR*, 824–832.

TAGLIASACCHI, A., SCHRÖDER, M., TKACH, A., BOUAZIZ, S., BOTSCH, M., AND PAULY, M. 2015. Robust articulated-ICP for real-time hand tracking. *Computer Graphics Forum 34*, 5, 101–114.

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. 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. ICCV*, 3224–3231.

TANG, D., TAYLOR, J., KOHLI, P., KESKIN, C., KIM, T.-K., AND SHOTTON, J. 2015. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *Proc. ICCV*, 3325–3333.

TAYLOR, J., SHOTTON, J., SHARP, T., AND FITZGIBBON, A. 2012. The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proc. CVPR*, 103–110.

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. CVPR*, 644–651.

TEJANI, A., TANG, D., KOUSKOURIDAS, R., AND KIM, T.-K. 2014. Latent-class Hough forests for 3D object detection and pose estimation. In *Proc. ECCV*, 462–477.

TOMPSON, J., STEIN, M., LECUN, Y., AND PERLIN, K. 2014. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Graphics 33*, 5, #169.

TRIGGS, W., MCLAUCHLAN, P., HARTLEY, R., AND FITZGIBBON, A. 2000. Bundle adjustment — A modern synthesis. In *Vision Algorithms: Theory and Practice*, LNCS. 298–372.

TZIONAS, D., BALLAN, L., SRIKANTHA, A., APONTE, P., POLLEFEYS, M., AND GALL, J. 2015. Capturing hands in action using discriminative salient points and physics simulation. *arXiv preprint 1506.02178*.

ULTRAHAPTICS LTD, 2013. Haptics System. `http://ultrahaptics.com`, Jan.

VALENTIN, J., DAI, A., NIESSNER, M., KOHLI, P., TORR, P., IZADI, S., AND KESKIN, C. 2016. Learning to navigate the energy landscape. *arXiv preprint 1603.05772*.

VICENTE, S., AND AGAPITO, L. 2013. Balloon shapes: reconstructing and deforming objects with volume from images. In *Proc. 3DV*, 223–230.

WANG, R. Y., AND POPOVIĆ, J. 2009. Real-time hand-tracking with a color glove. *ACM Trans. Graphics 28*, 3, #63.

WANG, R., PARIS, S., AND POPOVIĆ, J. 2011. 6D hands. In *Proc. UIST*, 549–558.

WANG, Y., MIN, J., ZHANG, J., LIU, Y., XU, F., DAI, Q., AND CHAI, J. 2013. Video-based hand manipulation capture through composite motion control. *ACM Trans. Graphics 32*, 4 (July), 43:1–43:14.

WU, Y., AND HUANG, T. S. 2000. View-independent recognition of hand postures. In *Proc. CVPR*, vol. 2, 88–94.

WU, Y., LIN, J. Y., AND HUANG, T. S. 2001. Capturing natural hand articulation. In *Proc. ICCV*, vol. 2, 426–432.

XU, C., AND CHENG, L. 2013. Efficient hand pose estimation from a single depth image. In *Proc. ICCV*, 3456–3462.

ZACH, C. 2014. Robust bundle adjustment revisited. In *Proc. ECCV*, 772–787.

ZHAO, W., CHAI, J., AND XU, Y.-Q. 2012. Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data. In *Proc. Symposium on Computer Animation*, 33–42.