Accurate, Robust, and Flexible Real-time Hand Tracking

Toby Sharp[†] Cem Keskin[†] Duncan Robertson[†] Jonathan Taylor[†] Jamie Shotton[†]
David Kim Christoph Rhemann Ido Leichter Alon Vinnikov Yichen Wei
Daniel Freedman Pushmeet Kohli Eyal Krupka Andrew Fitzgibbon[⋆] Shahram Izadi[⋆]

Microsoft Research

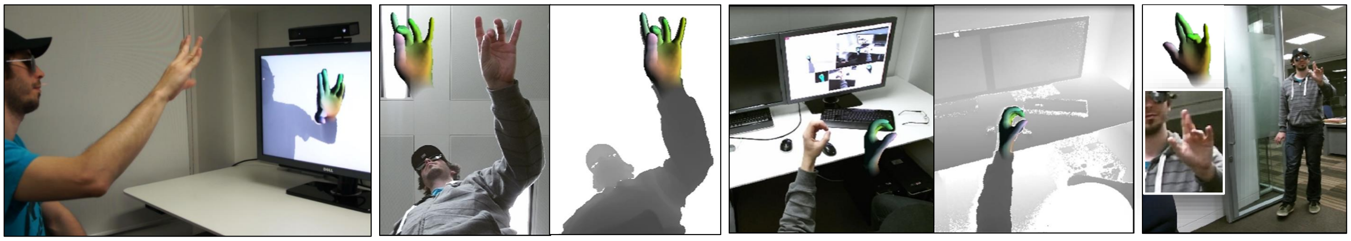

Figure 1: We present a new system for tracking the detailed motion of a user's hand using only a commodity depth camera. Our system can accurately reconstruct the complex articulated pose of the hand, whilst being robust to tracking failure, and supporting flexible setups such as tracking at large distances and over-the-shoulder camera placement.

ABSTRACT

We present a new real-time hand tracking system based on a single depth camera. The system can *accurately* reconstruct complex hand poses across a variety of subjects. It also allows for *robust* tracking, rapidly recovering from any temporary failures. Most uniquely, our tracker is highly *flexible*, dramatically improving upon previous approaches which have focused on front-facing close-range scenarios. This flexibility opens up new possibilities for human-computer interaction with examples including tracking at distances from tens of centimeters through to several meters (for controlling the TV at a distance), supporting tracking using a moving depth camera (for mobile scenarios), and arbitrary camera placements (for VR headsets). These features are achieved through a new pipeline that combines a multi-layered discriminative reinitialization strategy for per-frame pose estimation, followed by a generative model-fitting stage. We provide extensive technical details and a detailed qualitative and quantitative analysis.

INTRODUCTION

The human hand is remarkably dextrous, capable of high-bandwidth communication such as typing and sign language. Computer interfaces based on the human hand have so far been limited in their ability to accurately and reliably track the detailed articulated motion of a user's hand in real time. We believe that, if these limitations can be lifted, hand tracking will become a foundational interaction technology for a wide range of applications including immersive virtual reality, assistive technologies, robotics, home automation, and gaming.

However, hand tracking is challenging: hands can form a variety of complex poses due to their many degrees of freedom (DoFs), and come in different shapes and sizes. Despite some notable successes (e.g. [7, 32]), solutions that augment the user's hand with gloves or markers can be cumbersome and inaccurate. Much recent effort, including this work, has thus focused on camera-based systems. However, cameras, even modern consumer depth cameras, pose further difficulties: the fingers can be hard to disambiguate visually and are often occluded by other parts of the hand. Even state of the art academic and commercial systems are thus sometimes inaccurate and susceptible to loss of track, e.g. due to fast motion. Many approaches address these concerns by severely constraining the tracking setup, for example by supporting only close-range and front facing scenarios, or by using multiple cameras to help with occlusions.

In this paper, we present a system that aims to relax these constraints. We aim for high *accuracy*, i.e. the correctness and fidelity of the final reconstruction across a wide range of human hand poses and motions. Our system is remarkably *robust*, i.e. can rapidly recover from momentary failures of tracking. But perhaps most uniquely, our system is *flexible*: it uses only a single commodity depth camera; the user's hand does not need to be instrumented; the hand and fingers can point in arbitrary directions relative to the sensor; it works well at distances of several meters; and the camera placement is largely unconstrained and need not be static (see Fig. 1).

Our system combines a per-frame *reinitializer* that ensures robust recovery from loss of track, with a *model-fitter* that uses temporal information to achieve a smooth and accurate result. We propose a new reinitializer that uses machine learning to efficiently predict a hierarchical distribution over hand poses. For the model-fitter, we describe a 'golden' objective function and stochastic optimization algorithm that minimizes the reconstruction error between a detailed 3D hand model and the observed depth image. The pipeline runs in real time on consumer hardware. We provide extensive technical details to aid replication, as well as a detailed qualitative and quantitative analysis and comparison on several test datasets. We show how our tracker's flexibility can enable new interactive possibilities beyond previous work, including: tracking across the range of tens of centimeters through to a several meters (for high fidelity control of displays, such as TVs, at a distance); tracking using a moving depth camera (for mobile scenarios);

[†] denotes joint first authorship. [⋆] denotes joint last authorship.

CHI '15, Apr 18 – 23 2015, Seoul, Republic of Korea.

ACM 978-1-4503-3145-6/15/04...$15.00. http://dx.doi.org/10.1145/2702123.2702179.

and arbitrary camera placements, including first person (enabling tracking for head-worn VR systems).

RELATED WORK

Given our aims described above, we avoid encumbering the user's hand with data gloves [7], colored gloves [32], wearable cameras [13], or markers [38], all of which can be barriers for natural interaction. We focus below on vision-based articulated hand tracking. *Discriminative* approaches work directly on the image data (e.g. extracting image features and using classification or regression techniques) to establish a mapping to a predefined set of hand pose configurations. These often do not require temporal information and can thus be used as robust reinitializers [21]. *Generative* (or *model-based*) methods use an explicit hand model to recover pose. *Hybrid* methods (such as [3, 23] and ours) combine discriminative and generative to improve the robustness of frame-to-frame model fitting with per-frame reinitialization.

RGB input Early work relied on monocular RGB cameras, making the problem extremely challenging (see [8] for a survey). Discriminative methods [2, 34] used small databases of restricted hand poses, limiting accuracy. Generative methods used models with restricted DoFs, working with simplified hand representations based on 2D, 2½D or inverse kinematics (IK) (e.g. [24, 35]), again resulting in limited accuracy. Bray et al. [4] used a more detailed 3D hand model. de La Gorce et al. [6] automatically apply a scaling to the bones in the hand model during tracking. Most of these systems performed offline tracking using recorded sequences. An early example of online (10Hz) tracking with a simplified deformable hand model is [10]. This work, as with other RGB-based methods, struggled with complex poses, changing backgrounds, and occlusions, thus limiting general applicability.

Multi-camera input There has also been recent work on high-quality, *offline* (non-interactive) performance capture of hands using multi-camera rigs. Ballan et al. [3] demonstrate high-quality results closely fitting a detailed scanned mesh model to complex two-handed and hand-object interactions. The pipeline takes about 30 seconds per frame. Zhao et al. [38] 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. [33] show complex hand-object interactions by minimizing a silhouette, color, and edge-based objective using a physics engine, though take minutes per frame. Sridhar et al. [23] use a rig comprising five RGB cameras and a time-of-flight (ToF) sensor to track a user's hand using a person-calibrated model at $\sim$10Hz. We provide a direct comparison later in this paper. The above systems can produce highly accurate results, but are impractical for interactive consumer scenarios.

Depth input The advent of consumer depth cameras such as Kinect has made computer vision more tractable, for example through robustness to lighting changes and invariance to foreground and background appearance. This, together with the use of GPUs for highly-parallel processing, has made *real-time* hand tracking more feasible. However interactive, detailed hand tracking that is simultaneously accurate, robust, and flexible remains an unsolved and challenging problem.

Oikonomidis et al. [17] 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. Qian et al. [20] extend [17] by adding a 'guided' PSO step and a reinitializer that requires fingertips to be clearly visible. Melax et al. [16] use a generative approach driven by a physics solver to generate 3D pose estimates. These systems work only at close ranges, and are based on simple polyhedral models; our approach instead works across a wide range of distances, and exploits a full 3D hand mesh model that is better able to fit to the observed data.

Keskin et al. [12] 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. Tang et al. [27, 26] extend this work demonstrating more complex poses at 25Hz. Whilst more robust to occlusions than [12], neither approach employs an explicit model fitting step meaning that results may not be kinematically valid (e.g. implausible articulations or finger lengths). Xu et al. [36] estimate the global orientation and location of the hand, regress candidate 21-DoF hand poses, and select the correct pose by minimizing reconstruction error. The system runs at 12Hz, and the lack of tracking can lead to jittery pose estimates. Tompson et al. [30] demonstrate impressive hand tracking results using deep neural networks to predict feature locations and IK to infer a skeleton. While real-time, the approach only tackles close-range scenarios. Wang et al. [32, 31] 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.

Commercial systems Beyond this research, there have also been commercial hand tracking systems. The 3Gear Systems [1] (based on prior work of [32, 31]) and the second-generation software for the Leap Motion [15] have shown tracking of a range of complex poses, including two-handed interaction. We compare to both in our results section.

SYSTEM OVERVIEW AND CONTRIBUTIONS

We follow recent approaches to hand and body tracking [3, 17, 28] and adopt an approach based on 'analysis by synthesis' [37]. We use machine learning and temporal propagation to generate a large set of candidate hand pose hypotheses for a new input frame. Each pose hypothesis contains a parameter vector describing the global position and orientation of the hand, as well as the angles between joints in the hand skeleton. Each hypothesis is then rendered as an image using standard graphics techniques, and scored against the raw input image. Finally, the best-scoring hypothesis is output.

One of the main contributions of this paper is to make this approach *practical*, by which we mean accurate, robust, and flexible as described in the introduction. To this end, we make three technical contributions. First, we show how to use a simple 'golden' energy function to accurately distinguish good and bad hypotheses.* Second, we present a discriminative approach (the 'reinitializer') that predicts a *distribution* over hand poses. We can then quickly sample diverse pose hypotheses from this distribution, some of which are likely to be close to the correct answer. Third, we demonstrate that the rendering of a *detailed* articulated hand mesh model and the

*'Energy', 'objective', and 'scoring' functions are notionally equivalent, though 'good' will be used to imply low energy but high objective/score.

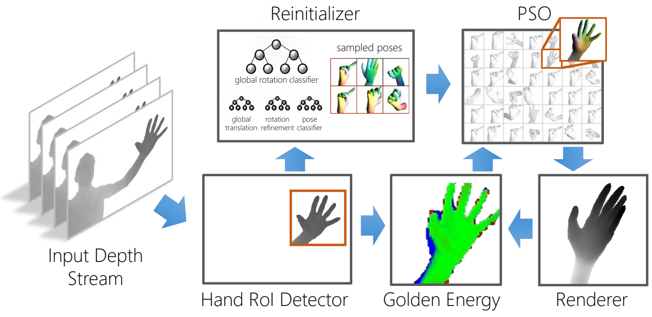

Figure 2: Algorithm pipeline.

golden energy computation can be done in real-time on the GPU. This goes beyond existing approaches that have relied on approximate hand models such as spheres and cylinders.

Pipeline

Each input depth image is processed by a pipeline comprising three steps (see Fig. 2):
1. Hand RoI extraction: Identify a square region of interest (RoI) around the hand and segment hand from background.
2. Reinitialization: Infer a hierarchical distribution over hand poses with a layered discriminative model applied to the RoI.
3. Model fitting: Optimize a 'population' of hand pose hypotheses ('particles') using a stochastic optimizer based on particle swarm optimization (PSO).
We describe each of these components in detail below, but first introduce some preliminaries.

PRELIMINARIES

3D Hand Model

We represent the human hand as a 3D model, represented by a detailed *mesh* of triangles and vertices. The 3D positions of the M mesh vertices are represented as columns in a $3 \times M$ matrix $\mathbf{V}$ that defines the hand shape in a 'base' (rest) pose.

To be useful for pose estimation, we must be able to articulate the wrist, finger, and thumb joints in the model. We use a standard 'kinematic' skeleton for this which specifies a hierarchy of joints and the transformations between them. We use vector θ to denote the *pose* of the hand, including: global scale (3 parameters); global translation (3); global orientation (3); and the relative scale (1) and rotations (3) at each joint (three joints per finger and thumb, plus the wrist). The global rotation is represented as a normalized quaternion which allows us to deal with arbitrary global rotations without risk of 'gimbal lock'. The joint rotations are represented as 'Euler angles' about local coordinate axes which are defined to correspond to 'flexion', 'abduction' and 'twist'.

Given the pose vector θ, the vertices of the *posed mesh* can be computed by a function $\Phi(\theta; \mathbf{V})$, which returns a $3 \times M$ matrix. The columns of this matrix correspond to those in the base pose model $\mathbf{V}$, but are in the pose specified by θ. For function Φ we use a standard technique called linear blend skinning (LBS). The precise details of LBS are unimportant for this paper, though see e.g. [29] for more details.

Input depth image

Our algorithm operates on images captured by a single depth camera. The image is represented as a two-dimensional array of depth values

$$Z = \{z_{ij} \mid 0 \leq i < H, 0 \leq j < W\}$$

where z_{ij} is the depth in meters stored at pixel location (i, j) and W and H are respectively the width and height of the depth map. We assume the depth map has been pre-processed such that invalid pixels (e.g. low reflectance or shadowed from the illuminator) are set to a large background depth value z_{bg}.

SCORING FUNCTION: THE 'GOLDEN ENERGY'

An underlying assumption of all analysis-by-synthesis algorithms is that, given unlimited computation, the best solution can be found by rendering all possible poses of the hand, and selecting the pose whose corresponding rendering best matches the input image (while appropriately accounting for the prior probabilities of poses). For this assumption to hold, a number of details are important.

First, the model should be able to accurately *represent* the observed data. Overly approximate models such as those based on spheres and cylinders will struggle to accurately describe the observed data. While still an approximation, we employ a detailed skinned hand mesh that we believe can describe the observed data much more accurately. Furthermore, given our focus on *flexible* camera setups, the camera may see not just the hand, but also a large portion of the user's body, as well as the background. To avoid having to simultaneously fit an entire model of the hand, body, and background to the depth image, we thus detect and extract (resample) a reasonably tight region of interest (RoI) $Z_{\text{roi}} = \{\bar{z}_{ij} \mid 0 \leq i < S, 0 \leq j < S\}$, comprising of $S \times S$ pixels around the hand, and segment the hand from the background (see next section). Working with a tight RoI also increases the efficiency of the scoring function by ensuring that a large proportion of pixels in the rendered images belong to the hands.

Second, several 'short cut' solutions to rendering can invalidate the approach. For example, simply projecting each vertex of the model into the image and measuring distance (either in depth, or in 3D using a distance transform) fails to account for occlusion. We thus define a GPU-based rendering module which takes as input a base mesh $\mathbf{V}$, a vector of pose parameters θ, and a bounding box B within the original depth image Z, and renders a synthetic depth image

$$R_{\text{roi}}(\theta; \mathbf{V}, B) = \{r_{ij} \mid 0 \leq i < S, 0 \leq j < S\} \qquad (1)$$

of the same size as Z_{roi}, where background pixels receive the value $r_{ij} = z_{\text{bg}}$. The bounding box B is used to render R_{roi} perspective-correctly regardless of its position in the original depth image.

Given compatible rendered and acquired images, the 'golden energy' scoring function we use is simple:

$$E^{\text{Au}}(Z_{\text{roi}}, R_{\text{roi}}) = \sum_{ij} \rho(\bar{z}_{ij} - r_{ij}) \,. \qquad (2)$$

Instead of a squared ('L2') error, we employ a truncated 'L1' distance $\rho(e) = \min(|e|, \tau)$ which is much less sensitive to outliers due to camera noise and other factors. Despite its simplicity, this energy is able to capture important subtleties. In particular, where a model hand pixel is rendered over data background, or model background over data from the hand, the contribution to the energy is z_{bg}, which is large enough to always translate to the truncation value τ. Where rendered and data are both z_{bg}, the score is zero.

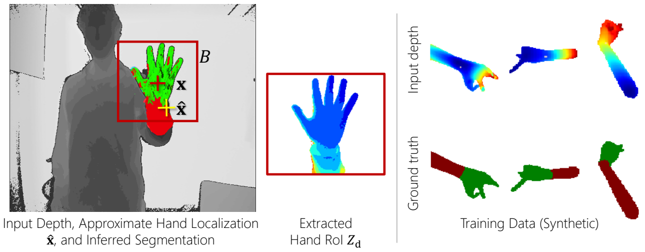
B
x
x̂
Input Depth, Approximate Hand Localization x̂, and Inferred Segmentation
Extracted Hand RoI Z_d
Input depth
Ground truth
Training Data (Synthetic)

Figure 3: Hand Region of Interest (RoI) extraction.

We dub the above function the 'golden energy' to distinguish it from energies that do not directly aim to explain the image data such as [16, 23, 28]. E^{Au} very effectively captures the notion of analysis by synthesis, and experiments suggest that casting tracking as an optimization problem whose sole objective is minimization of E^{Au} yields high-quality results. We found the energy to be reasonably robust to different values of τ; all our experiments used a value $\tau = 100$mm.

REGION OF INTEREST (RoI) EXTRACTION

The RoI extraction process is illustrated in Fig. 3 left. We start from a rough estimate $\hat{\mathbf{x}}$ of the 3D hand location. This can be obtained in several ways, for example using motion extrapolation from previous frames, the output of a hand detector, or the hand position provided by the Kinect skeletal tracker. The estimate $\hat{\mathbf{x}}$ is often fairly approximate, and so a second step is used to more accurately segment and localize the hand using a learned pixel-wise classifier based on [21]. The classifier, trained on 100k synthetic images of the hand and arm (Fig. 3 right), is applied to pixels within a 3D search radius r_1 around location $\hat{\mathbf{x}}$ to classify pixels as hand or arm. To precisely localize the hand, we then search for position $\mathbf{x}$ for which the accumulated hand probability within a 20cm×20cm window is maximal. Finally, we extract the RoI, using nearest-neighbor downsampling to a $S \times S$ pixel image Z_{d} for use in the golden energy. We typically set S between 64 or 128: larger will preserve more detail but will be slower to render in the golden energy computation. Note that we do *not* remove the arm pixels (see below), though we *do* set pixels outside radius r_2 ($< r_1$) of $\mathbf{x}$ to background value z_{bg}. We also record the bounding box B of the RoI in the original depth map.

While in practice we found the classifier [21] to work reliably at improving the *localization* of the RoI around the hand, we decided against attempting to directly *segment* the hand from the forearm, for two reasons. First, the inferred boundary between hand and arm proved rather imprecise, and from frame to frame would typically move up and down the arm by several centimeters. Second, we observed that there is considerable value in observing a small amount of forearm that abuts the edge of the RoI. Unlike existing approaches, we include a forearm in our hand mesh model, allowing our golden energy to *explain* the observed forearm pixels within the RoI and thus help prevent the optimized hand pose from flipping or sliding up and down the arm.

This part of the pipeline is similar to the hand segmentation approach in [30], but does not require training data for all possible backgrounds, can exploit large quantities of synthetic data, is potentially faster since the classifier is only applied to relatively few pixels, and is able to exploit the forearm signal.

ROBUST REINITIALIZATION

Of all parts in our pipeline, we found the reinitializer the most critical in achieving the accuracy, robustness, and flexibility demonstrated in our results. Its primary goal is to output a pool of hypotheses of the full hand pose by observing just the current input depth image, i.e. without temporal information. The hypotheses will in due course be evaluated and refined refined in later stages of the pipeline.

Existing approaches include multi-layer random forests [12], nearest-neighbor look-up [31], fingertip detection [20], or convolutional neural networks [30]. However, our focus on flexible camera setups pushes the requirements for reinitialization far beyond what has yet been demonstrated. In particular, we place no restrictions on the global rotation of the hand. This means that we cannot rely on seeing fingertips, and further, the range of hand appearance that we expect to see is dramatically increased compared to the near-frontal close-range hands evaluated in existing work.

Our reinitialization component makes two significant advances beyond existing work. First, we believe the problem is so hard that we cannot hope to predict a single good pose solution. Instead, our approach predicts a *distribution* over poses, from which our model fitter is free to quickly sample as many poses as desired and use the golden energy to disambiguate the good from the bad candidates. Second, while following existing practice [25, 12] in breaking the regression into two 'layers' (or stages), we focus the first layer exclusively in predicting quantized global hand rotation. By predicting coarse rotation at the first layer, we dramatically reduce the appearance variation that each second-layer predictor will have to deal with.

Synthetic Training Data

Our reinitializers are trained on a corpus of synthetic training data. Given the dexterity of hands and fingers, the potential pose space is large, and to achieve our goal of flexibility, the training data must ensure good coverage of this space while avoiding unlikely poses. We achieve this by sampling from a predefined prior distribution that is specifically designed to give a broad coverage of pose space. We believe it is substantially more general-purpose than existing work which tends to concentrate on close-range frontal poses.

Global hand orientation samples are drawn from a uniform distribution, global translation samples are drawn from a uniform distribution within the view frustum, and wrist pose is randomized within sensible flexion/abduction limits. For the finger and thumb rotations, we employ six manually defined *prototype poses* (or *proto-poses*): Open, Flat, Closed, Pointing, HalfOpen, Pinching. Each proto-pose is able to generate realistic poses for a subset of pose space, with a 'mean' shape designed to look like the proto-pose name, and a set of randomization rules to allow one to draw samples from the proto-pose (see supplementary material for example pseudocode and samples). In future we hope these might instead be learned from a large corpus of hand motion capture data.

As training data, we sample 100k poses θ from the above distribution, and for each pose, we generate a synthetic depth image using the same mesh model and renderer as is used for computing the golden energy. The reinitializer acts on a depth RoI rather than the full depth map, and so a tight bounding box is computed around the pixels belonging to hand (not forearm), and then expanded randomly by up to 10 pixels on each side to simulate an imprecise RoI at test time.

Two-Layer Reinitialization Architecture

Given the above data, we train a two-layer [12, 25] reinitializer. Of all elements of the pose vector, global rotation probably causes the largest changes in the hand's appearance. We thus train a first layer to predict global rotation, quantized into 128 discrete bins. For each rotation bin, second layer predictors are trained to infer other elements of the pose vector such as finger rotations. Each second layer predictor is trained only on those images within the respective global rotation bin. This reduces the appearance variation each second layer predictor needs to handle, and thus simplifies the learning problem [12, 25]. The number of bins was chosen to balance the difficulties of the learning problems at the two layers and the memory requirements.

The first layer is trained to predict a distribution $P^{\text{grb}}(q|Z_{\text{roi}})$ over the **g**lobal **r**otation **b**ins. The training data for layer one is thus a set of (Z_{roi}, q) pairs, where $q \in \{0, \ldots, 127\}$ represents the nearest rotation bin to the image's global rotation. Several classifiers were investigated: decision forests [5], decision jungles [22], and discriminative ferns ensembles (DFE) [14]. These are compared in our experiments later. The DFE is trained as a 'holistic' predictor, applied once to the whole RoI. The forest and jungle classifiers are instead trained to predict global rotation distributions at any pixel within the RoI, and at test time multiple randomly-chosen pixel predictions are averaged to produce $P^{\text{grb}}(q|Z_{\text{roi}})$.

At the second layer, we train three predictors for each of the 128 rotation bins q: a global rotation refinement regressor, an offset translation regressor, and a finger proto-pose classifier. Each predictor is trained using the subset of data belonging to the relevant ground truth rotation bin q. We employ per-pixel decision jungles [22] for all layer-two predictors given their small memory requirements and the large number of predictors. The **g**lobal **r**otation **r**efinement regressor is trained to minimize the variance over quaternion global rotation predictions. It predicts a distribution $P_q^{\text{grr}}(\mathbf{q}|Z_{\text{roi}})$ over the global rotation quaternion $\mathbf{q}$. The **o**ffset **t**ranslation **r**egressor [9] is trained to predict the 3D offset from any RoI pixel's 3D camera space location to the wrist joint position (i.e. the global translation). It is trained similarly to minimize the variance of the 3D offsets at the leaf nodes. By aggregating the predictions from multiple pixels at test time, a distribution $P_q^{\text{otr}}(\mathbf{t}|Z_{\text{roi}})$ over the absolute global translation $\mathbf{t}$ is obtained. Finally, the **p**roto-**p**ose **c**lassifier is a trained as a conventional per-pixel classifier over the 6 proto-pose classes. At test time, multiple randomly-chosen foreground pixels' predictions are aggregated to predict a distribution $P_q^{\text{ppc}}(f|Z_{\text{roi}})$.

Making Predictions

At test time, the first layer is evaluated to predict $P^{\text{grb}}(q|Z_{\text{roi}})$. We assume that the top 5 most likely global rotation bins concentrate the majority of the probability mass, and so for speed evaluate the relevant second layer predictors only for these bins, in order to predict the refined global rotation, the global translation, and the proto-pose. At this point we have effectively inferred a hierarchical distribution over hand poses, and can now efficiently sample as many draws from this distribution as possible: first sample a q, then conditioned on q sample a global rotation $\mathbf{q}$ from $P_q^{\text{grr}}(\mathbf{q}|Z_{\text{roi}})$, a global translation $\mathbf{t}$ from $P_q^{\text{otr}}(\mathbf{t}|Z_{\text{roi}})$, and a proto-pose f from $P_q^{\text{ppc}}(f|Z_{\text{roi}})$. Finally, to create a full pose vector θ, we concatenate the global rotation and translation with a random finger pose sampled from proto-pose f.

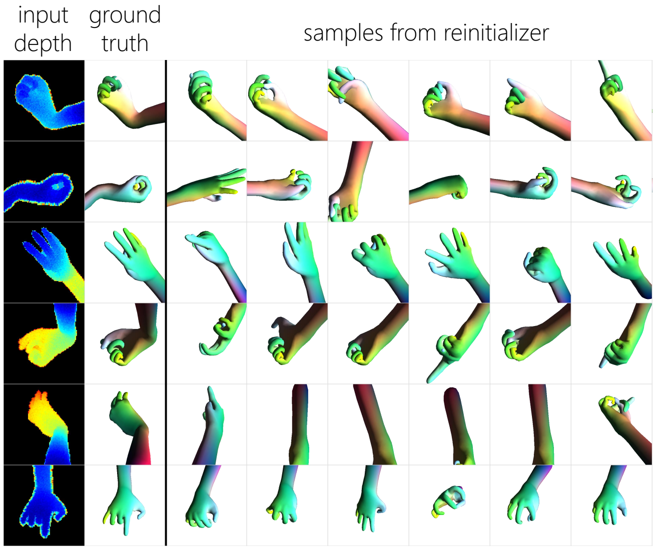

Figure 4: Sample poses inferred by our reinitializer. For each row, we see a false-color image of the input depth map, a rendering of the model in the ground truth pose, and renderings of poses sampled from the reinitializer. Our system is trained to work for arbitrary global rotations to allow for flexible camera setups.

We show some examples of the reinitializer applied to synthetic test images in Fig. 4. Observe that at least some samples from the reinitializer are usually close to the correct hand pose, especially so for global rotation and translation. Comparing rows 3 and 4 one can see the effect of the second layer pose classifier: row 3 has more 'open hand' samples, whereas row 4 has more 'closed hand' samples. Even the incorrect samples often exhibit sensible confusions, for example being flipped along a reasonable axis.

MODEL FITTING

In order to optimize the golden energy and achieve an accurate hand pose estimate, we employ a model fitting algorithm that combines features of particle swarm optimization (PSO) and genetic algorithms (GA) in the spirit of [11, 17, 20]. The model fitter is the component that integrates information from previous frames along with proposals from the reinitializer. It is also the main computational workhorse of our system, and we describe in the supplementary material how our model fitter can be implemented efficiently on the GPU to achieve real-time frame rates.

The algorithm maintains a population of P 'particles' $\{\phi_p\}_{p=1}^P$ (each particle contains a pose vector θ and additional state such as velocity in pose space), and the scoring function is evaluated across the population in parallel on the GPU to yield scores $\{E_p^{\text{Au}}\}_{p=1}^P$. Each such evaluation comprises one 'generation'. The standard PSO algorithm (see e.g. [17]) then specifies an update rule for populating the next generation. This rule incorporates 'momentum' in pose space and attraction towards current local minima of the energy function. We found that the standard update rules did not work well on their own, and describe below our extensions which we found critical for success.

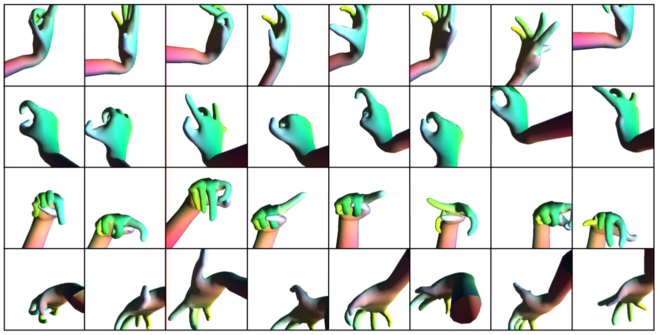

Figure 5: Particle perturbation. The leftmost column shows initial poses, the others show seven draws from the every-third-generation perturbation distribution.

Particle Randomization

Following [17], we found it crucial to 're-randomize' particles regularly. In our approach, use two variants: per-generation and every-third-generation. *Per-generation*, we adjust only fingers: for 50% of particles, a digit (finger/thumb) is chosen uniformly at random, and either (25%) its abduction is adjusted by up to $\pm 2.5°$ or (75%) the flexion of one of its segments is adjusted by $\pm 10°$. *Every-third-generation*, the 50% of particles with highest (worst) golden energy are re-randomized as follows. For a first sub-group (50%), their pose replaced by a new sample from the reinitializer. A second sub-group (30%) are subjected to local random perturbation as illustrated in Fig. 5 and detailed in the supplementary material. For a final sub-group (20%), a 'splicing' or 'crossover' operation is applied: a random particle is chosen from the lowest (best) energy 50% and used to overwrite either the global rotation, translation, and wrist pose, or the remainder (the finger poses) of the pose vector.

Particle Initialization

The set of particles is initialized using the same strategy as every-third-generation, except that all particles are affected, and those particles that are randomly perturbed are perturbed from the result from the previous frame when tracking.

Particle 'Aging'

A per-generation clustering step was recently proposed in [20] to reduce an effect called 'particle collapse' whereby all particles are pulled too close together to a bad solution. We employ a similar but cheaper technique whereby we assign each particle an 'age' a_p. All particles within a given age are treated as an independent swarm. Before particles are re-randomized every third-generation, all particles have their ages incremented, to a maximum value $A_{\max}$. The randomized particles have their ages set to zero.

EVALUATION: QUALITATIVE

We next evaluate our approach and demonstrate state of the art hand tracking, especially regarding robustness and flexibility. The reader is strongly encouraged to also view the addition results in the accompanying video and supplementary document. The results in this section were generated using a standard Kinect V2 time of flight sensor with no modifications. We start by demonstrating the new capabilities of our hand tracker. Fig. 6 and the accompanying video show the results for a wide range of complex poses reconstructed in real-time by our system. We show results across a wide range of adult male and female subjects, both close to and far away (several meters) from the sensor, and during user and camera motion. Our approach is limited by the sensor operating range of 0.5 – 4.5m and the available resolution. In the more distant cases, sometimes fewer than 100 hand pixels remain, but our approach degrades gracefully and is still able to estimate a reasonably accurate pose.

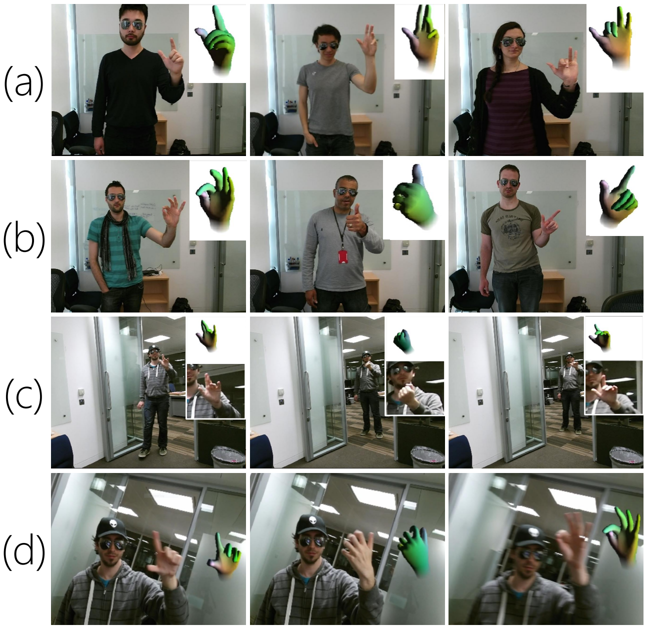

Figure 6: Our system (a,b) can track a wide variety of users and hand shapes, (c) can track at both close and far ranges, and (d) is robust to both scene and user motion.

In Figs. 1 and 7, we further show novel camera placements, such as 'over the shoulder' or 'bottom-up', that do not require the user to be sat directly in front of the sensor at close range. Again our system is able to smoothly track despite being placed in a non-frontal view and even whilst the sensor is moving. We feel these are very compelling configurations allowing integration with VR headsets and other new physical setups such as interacting with displays at larger distances. The accompanying video further demonstrates how our system can deal with fast motions, recover from tracking loss, and cope with changes in lighting (since we only use the depth signal from the camera).

In Fig. 7 and the accompanying video, we qualitatively compare our system to the recent Leap Motion sensor [15] and the 3Gears system [1]. While Leap is not a general depth camera and has different computational constraints, public videos have shown high precision tracking. Our results suggest that indeed the Leap robustly tracks a variety of forward facing poses. However, Leap struggles with more complex poses, seems not to support graceful reinitialization in certain conditions, and can track only at small distances. These are not limitations of our system. Our comparison with 3Gears suggests our system can more faithfully reconstruct a wider range of continuous and complex poses than 3Gears.

EVALUATION: QUANTITATIVE

The few datasets that exist for hand tracking evaluation do not stress-test the flexibility and robustness that we value highly. Therefore, while we first compare with the state of the art system by Sridhar et al. [23] on their published dataset DEXTER1, our main results will be presented on two new and extremely

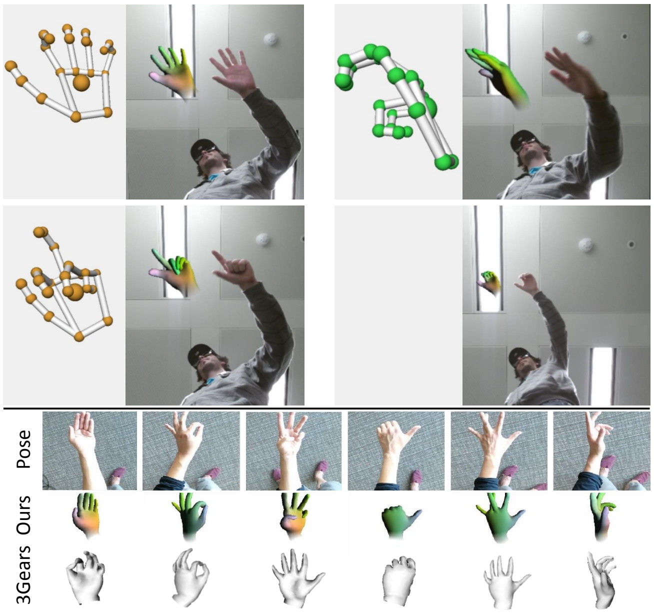

Figure 7: Top half: qualitative comparison of Leap Motion (left panels) with our method (right panels). Bottom half: qualitative comparison of 3Gears (bottom row) with our system (middle row). See text and video.

Sequence	AbdAdd	FingerCount	FingerWave	FlexEx1	Pinch	Random	Tiger	**Mean**
Ours	15.0	12.3	18.2	10.7	12.9	23.3	12.9	**15.0**
[23]	11	14	12.5	17	10	13	12	**12.8**

Table 1: Comparison of our system and [23] on the DEXTER1 dataset. We achieve broadly comparable tracking accuracy but only with a single depth camera (as opposed to 5 RGB + 1 depth), and at higher frame rates.

challenging benchmarks: SYNTHETIC, and FINGERPAINT. We believe a complement of synthetic and real testing to be important: synthetic data allows one to more easily test a full range of hand poses, while real data allows one to ensure the system actually works on real data with real camera noise. More details on these datasets are given below and in the supplementary material.

Comparison with [23]

We directly compare with [23] on their DEXTER1 dataset. Table 1 compares mean joint position errors obtained by our tracker and that of [23] (numbers are estimated from a graph in their paper). We achieve broadly comparable accuracy, despite our system running 3 times faster (at 30 fps vs. 10 fps) and only using the depth sensor input ([23] use a combination of the depth sensor and 5 RGB cameras as input).

Experiments on synthetic data

We designed a new dataset, SYNTHETIC, to evaluate our reinitialization strategy and model fitting algorithms in the absence of temporal information. It was designed to stress-test robustness and we believe it is considerably more challenging than existing datasets. The dataset comprises 1000 individual rendered depth image frames (not sequences) with corresponding ground truth hand poses. To generate each frame, the full hand pose is heavily randomized: global rotation is uniformly randomized; global translation is randomized within the view frustum; finger/thumb/wrist flexions/abductions are either fully randomized within sensible limits or sampled from a proto-pose. The dataset thus contains hands in pretty much arbitrary poses, including both close to and far from the camera. The left column in Fig. 4 shows some examples. For each frame we additionally specify a 'starting pose'. In 50% of frames this is perturbed from the ground truth according to the every-third-generation method described above. In the remaining 50% of frames, the starting pose is fully randomized to simulate attempting to recover from complete loss of track.

Model	Size	Depth	Accuracy	Memory
Ferns	20	12	70.5%	20MB
Ferns	50	13	73.3%	100MB
Forest	1	22	66.9%	150MB
Jungle-512	3	20	30.9%	100KB
Jungle-2048	3	45	47.5%	500KB

Table 2: Comparison of the first layer classifiers in terms of accuracy and memory for various settings.

We believe that the average joint metric used by [23] hides interesting information, and so for SYNTHETIC we advocate the 'proportion of frames correct' metrics proposed in [28]. This gives a graph that plots the proportion of joints that have Euclidean error less than ϵ, where the value of ϵ is plotted on the x-axis (see e.g. Fig. 9). Under this metric, correctness under approximately $\epsilon = 0.03$m corresponds to a visually good result. The supplementary material includes results using a 'worst case' variant of this metric.

First layer reinitializer. The first layer of our two-layer reinitializer classifies the depth RoI into one of 128 global rotation bins. We compare ferns, forests, and jungles. We use 100k images to train each model, and applied a grid search using 10k validation images to select the best parameters for each method: the model size (i.e. number of trees, jungles, and ferns), model depth, and model width (for jungles only). Within the range of parameter values tested, the highest accuracies are achieved with the following parameter settings: (i) 50 ferns of size 13, (ii) 1 randomized tree of depth 22, and (iii) 3 jungles of width 2048 and depth 45. The bin classification accuracies achieved by each model and their corresponding memory requirements are given in Table 2. Ferns achieve the highest absolute accuracy, though jungles provide the best accuracy-memory ratio. Training forests beyond depth 22 would add significantly to their accuracy, but the memory requirements quickly become prohibitive (trees grow as 2^{depth}).

Second layer reinitializer. The models in the second layer are selected based on the predictions of the first layer model. The large differences in accuracy between ferns, forests, and jungles seen in the first layer disappear in the second layer, possibly because the amount of variation is highly reduced inside each global rotation bin. Specifically, trees of depth 18 performed only marginally better than jungles of depth 30 and width 256, even though they are 20 times larger in size. Because 128 predictors need to be kept in memory, we opted for the jungles in the second layer. We also experimented with three training sets of sizes 10k, 100k and 200k. It was possible to get a substantial improvement in accuracy by using 100k images rather than 10k images, but there was no noticeable difference between 100k and 200k images.

One layer vs. two layers. To empirically justify our two-layer model, we compared our final results with a more traditional one-layer model. To do this, we trained a forest to depth

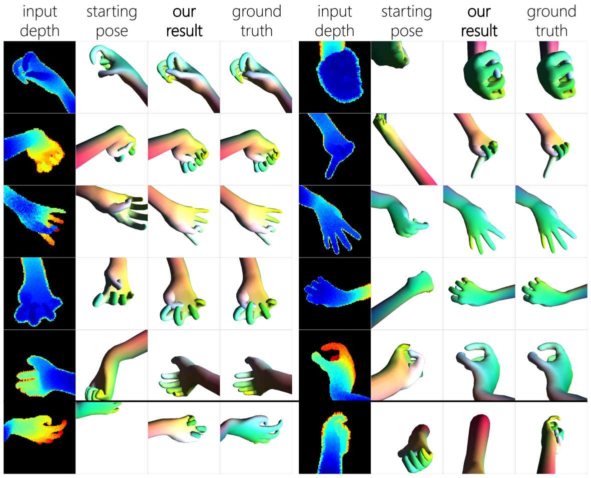

Figure 8: Example results on the per-frame Synthetic test set. Despite simulated input noise and many frames having wildly incorrect starting poses (simulating loss of track), our reinitialization + model fitting algorithm is often able to infer the full-DOF pose (global rotation, translation, and finger poses). The final row shows two failure cases, which would likely be easily corrected if part of a tracked sequence.

25 for each of the three layer-two reinitialization tasks (global rotation, translation, and pose cluster estimation) using the full dataset, i.e. not conditioned on a layer-one global rotation bin. This one-layer model achieved the same accuracy as our two-layer model, but used 6x more memory. Moreover, if we let the the two-layer model use more memory, it considerably exceeded the accuracy of the one-layer model. We we unable to improve the accuracy of the one-layer model by training deeper as we ran out of memory, even on our high-end PC.

Full optimization. We use Synthetic to validate specific algorithmic choices made and (on held-out validation data) to optimize the parameters of our algorithm. We first show some example image results in Fig. 8. These results were computed on each input image independently. The model fitting is initialized for each image at the starting pose shown to simulate tracking (when perturbed from the ground truth) or loss of track (when completely randomized). We can observe robust and accurate convergence of the model fitting for both global and finger poses, even from extremely incorrect starting positions. This is due largely to our reinitialization algorithm which provides a reliable set of candidate hand poses that are then refined by the model fitting and golden energy pipeline.

We next present quantitative results on Synthetic. Fig. 9(a) compares the model fitting result for variants of reinitializer with different components turned off. When a reinitialization component is turned off, it is replaced by random sampling. Note how reinitialization makes an enormous difference to accurate convergence of the optimization, and that each component contributes to the full reinitializer's power. Fig. 9(b) shows the effect of different numbers of particles in the model fitter on the final accuracy. Clearly, more is better, but the improvement from reinitialization far outweighs the number of particles. We show the effect of different components of our optimization algorithm in Fig. 9(c). Starting with the full optimization, we cumulatively remove the components in the order in the figure. We see that splicing, PSO particle aging, and re-randomization all play a substantial role in getting high accuracy. Fig. 9(d) plots the convergence of the optimization for various error threshold levels. In most cases the optimization converges around 30-40 generations.

Experiments on real data

The labor-intensive and error-prone nature of manually tagging hand images makes it expensive to annotate extended sequences. Our new FingerPaint dataset instead provides a semi-automatic means of obtaining ground truth. We simultaneously captured video of painted hands using both a prototype time of flight depth sensor and a standard RGB camera. Having calibrated the cameras, an automatic color segmentation algorithm can be used to give pixel-wise ground truth hand part labels across extended sequences. The labels are manually corrected where color segmentation is incorrect. This provides a 'proxy' ground truth: we do not have joint angles/positions, but achieving high accuracy on a pixel-segmentation metric (see below) is only achievable with accurate pose estimation. Further, these labels cover the whole hand, unlike some datasets e.g. only labeling fingertips.

The dataset consists of five subjects (three are pictured in Fig. 11 and all in the supplementary material). For each subject, three sequences were captured: 'global' (large global movements, relatively static fingers); 'poses' (relatively static global position, moving fingers); and 'combined'. The combined sequences exhibit movement that appears to be considerably more challenging than that present in existing datasets.

To evaluate tracking accuracy on FingerPaint, we measure our ability to reproduce the ground truth segmentation labels. We texture-map our model to define parts that mimic the ground truth label, and can then render a synthetic label image from the inferred pose to compare to the ground truth label image. For part label l, the accuracy is defined as the proportion of ground truth pixels with label l that are inferred as label l. Consistent with the Synthetic metric, we then plot the proportion of frames where we achieve an average classification accuracy (across labels) of a certain threshold.

Results on FingerPaint are shown in Fig. 10. As expected, that the tracker achieves its highest accuracies on the easier 'global' sequences and its lowest accuracies on the harder 'pose' and 'combined' sequences which contain complex finger articulations and inter-finger occlusions that are particularly hard to disambiguate from a single noisy depth sensor. Also of note is the reduced accuracy in tracking the second subject. This subject is a child, and thus her hand occupies fewer depth pixels and has significantly different proportions and features than the others.

SYSTEM PARAMETERS AND TIMINGS

Our timings are measured on a dual-processor workstation with an Nvidia GTX Titan graphics card. Using our default PSO settings of 100 particles and 30 generations with a tile size of 64x64, we spend on average 6ms per frame in reinitialization and 26ms per frame in model fitting. This achieves the 30Hz (our camera rate) tracking shown in the accompanying video. (Some of the synthetic experiments above vary these parameters and may run faster or slower.) The main bottlenecks in model fitting are in rendering for golden energy computation, and (in our current implementation) transferring candidate hand poses between main and GPU memory. The algorithm is heavy on GPU compute, but does not need much CPU power except for training the reinitializers which can

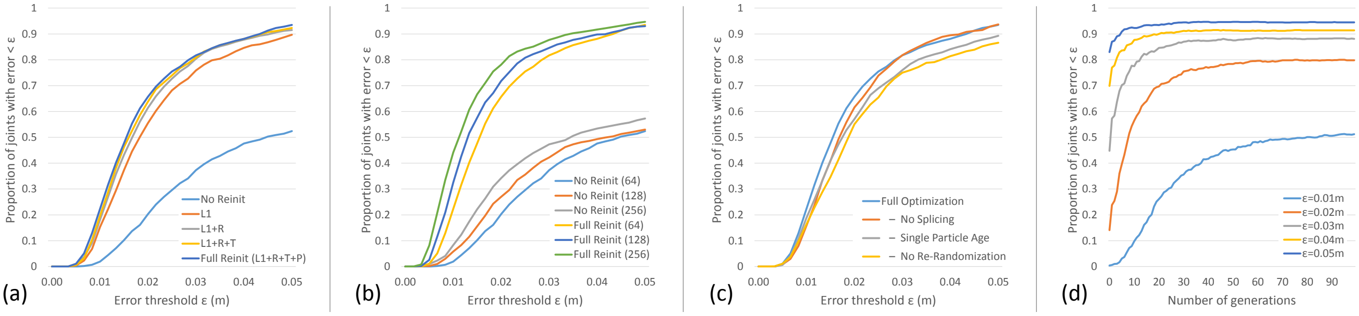

Figure 9: Experiments on our per-frame SYNTHETIC test set using 'average' error metric (see text) (a) Effect of reinitialization on model fitting accuracy. (L1 = layer one quantized rotation classification, R = layer two rotation regression, T = layer two offset translation regression, P = layer two pose classification). (b) Accuracies achieved using different numbers of particles for both full and no reinitialization. (c) Effect of removing components of the model fitting optimization. (d) Convergence plots for varying levels of error threshold.

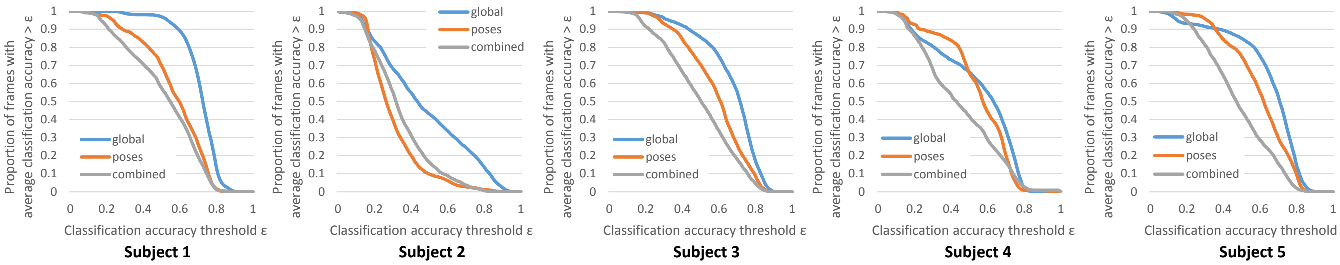

Figure 10: Experiments on the FINGERPAINT dataset for the five subjects.

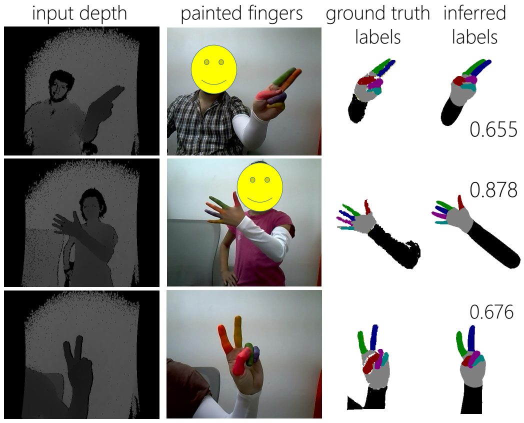

Figure 11: Examples from our FINGERPAINT dataset. For each row, we see the input depth image, the calibrated RGB image in which the painted fingers can be seen, the resulting ground truth labeling of the depth image, and the inferred labels that our tracker produces. (Numbers shown are classification accuracies).

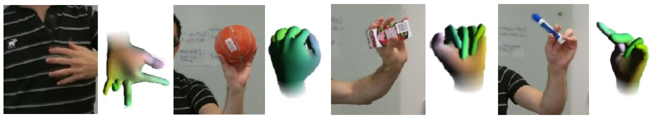

Figure 12: Example failure cases for our method.

take several hours. The trained two-layer fern/jungle reinitializer occupies about 30MBs of memory.

LIMITATIONS AND FUTURE WORK

We believe our system has dramatically pushed the boundaries of articulated hand tracking, particularly in terms of flexibility and robustness. But there is still further work to be done. The accompanying video and Fig. 12 highlight some of our current limitations. We only attempt to track a single hand, and two or more interacting hands and objects can cause confusion. Proofs of concept [19, 18] suggest that extending our approach to deal with these cases should not be too challenging, at least for the model fitter. Our pipeline is heavy on GPU compute, introducing latency and limiting deployment on mobile devices. We hope that by improving the relatively cheap discriminative reinitialization stage, and model fitting using a more efficient continuous optimization, we can reduce the compute requirements to a level suitable for mobile devices while maintaining high robustness and accuracy. We are keen to build more applications on top of our tracking system, and perform user studies to investigate their usefulness, as well as to explore the correlation between our various quantitative metrics and the perceived accuracy or task completion speed. Finally, we plan to explore the effect of the user's hand shape on tracking accuracy. We believe that building an accurate personalized model of each user's hand [29] can only improve the quality of tracking.

CONCLUSIONS

We presented a new real-time articulated hand tracker that combines fast learned reinitialization with model fitting based on stochastic optimization of a 'golden' objective function. Our evaluation demonstrated not only highly *accurate* hand pose estimates, but also dramatic improvements over the state of the art in *robustness*, recovering quickly from failure and tracking reliably over extended sequences, and *flexibility*, working for arbitrary global hand poses, at extreme distances from the camera, and for both static and moving cameras. We compared to three state-of-the-art approaches, showing accuracy comparable to a multi-camera system, and results exceeding both the Leap Motion and 3Gears commercial systems. We also extensively evaluated on challenging new real and synthetic datasets. We feel our system can help open up a variety of new interactions, with for example large displays at a distance, mobile devices equipped with depth sensors, and virtual or augmented reality systems.

REFERENCES

1. 3Gear Systems Inc, 2013. http://threegear.com.
2. Athitsos, V., and Sclaroff, S. Estimating 3D hand pose from a cluttered image. In *Proc. CVPR* (2003), II–432.
3. Ballan, L., Taneja, A., Gall, J., van Gool, L., and Pollefeys, M. Motion capture of hands in action using discriminative salient points. In *Proc. ECCV* (2012), 640–653.
4. Bray, M., Koller-Meier, E., and Van Gool, L. Smart particle filtering for 3D hand tracking. In *Proc. IEEE Conf. Automatic Face and Gesture Recognition* (2004), 675–680.
5. Criminisi, A., and Shotton, J. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.
6. de La Gorce, M., Fleet, D. J., and Paragios, N. Model-based 3D hand pose estimation from monocular video. *IEEE Trans. PAMI 33*, 9 (Feb. 2011), 1793–1805.
7. Dipietro, L., Sabatini, A. M., and Dario, P. A survey of glove-based systems and their applications. *IEEE Trans. Sys., Man, and Cybernetics C 38*, 4 (2008), 461–482.
8. Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. Vision-based hand pose estimation: A review. *Comp. Vis. and Image Understanding 108* (2007), 52–73.
9. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., and Fitzgibbon, A. Efficient regression of general-activity human poses from depth images. In *Proc. ICCV* (2011), 415 – 422.
10. Heap, T., and Hogg, D. Towards 3D hand tracking using a deformable model. In *Proc. IEEE Conf. Automatic Face and Gesture Recognition* (1996), 140–145.
11. Juang, C.-F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans. Sys., Man, and Cybernetics B 34*, 2 (April 2004), 997–1006.
12. Keskin, C., Kiraç, F., Kara, Y. E., and Akarun, L. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proc. ECCV* (2012), 852–863.
13. Kim, D., Hilliges, O., Izadi, S., Butler, A. D., Chen, J., Oikonomidis, I., and Olivier, P. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proc. ACM UIST* (2012), 167–176.
14. Krupka, E., Bar Hillel, A., Klein, B., Vinnikov, A., Freedman, D., and Stachniak, S. Discriminative ferns ensemble for hand pose recognition. In *Proc. CVPR* (2014).
15. Leap Motion Inc, 2014. http://leapmotion.com/product.
16. Melax, S., Keselman, L., and Orsten, S. Dynamics based 3D skeletal hand tracking. In *Proc. Graphics Interface* (2013), 63–70.
17. Oikonomidis, I., Kyriazis, N., and Argyros, A. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proc. BMVC* (2011), 1–11.
18. Oikonomidis, I., Kyriazis, N., and Argyros, A. A. Full DoF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proc. ICCV* (2011), 2088–2095.
19. Oikonomidis, I., Kyriazis, N., and Argyros, A. A. Tracking the articulated motion of two strongly interacting hands. In *Proc. CVPR* (2012), 1862–1869.
20. Qian, C., Sun, X., Wei, Y., Tang, X., and Sun, J. Realtime and robust hand tracking from depth. In *Proc. CVPR* (2014).
21. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. Real-time human pose recognition in parts from a single depth image. In *Proc. CVPR* (2011).
22. Shotton, J., Sharp, T., Kohli, P., Nowozin, S., Winn, J., and Criminisi, A. Decision jungles: Compact and rich models for classification. In *NIPS* (2013).
23. Sridhar, S., Oulasvirta, A., and Theobalt, C. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proc. ICCV* (Dec. 2013).
24. Stenger, B., Mendonça, P. R., and Cipolla, R. Model-based 3D tracking of an articulated hand. In *Proc. CVPR* (2001), II–310.
25. Sun, M., Kohli, P., and Shotton, J. Conditional regression forests for human pose estimation. In *Proc. CVPR* (2012), 3394–3401.
26. Tang, D., Chang, H. J., Tejani, A., and Kim, T.-K. Latent regression forest: Structured estimation of 3D articulated hand posture. In *Proc. CVPR* (2014), (in press).
27. Tang, D., Yu, T.-H., and Kim, T.-K. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proc. ICCV* (2013).
28. Taylor, J., Shotton, J., Sharp, T., and Fitzgibbon, A. The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proc. CVPR* (2012), 103–110.
29. Taylor, J., Stebbing, R., Ramakrishna, V., Keskin, C., Shotton, J., Izadi, S., Hertzmann, A., and Fitzgibbon, A. User-specific hand modeling from monocular depth sequences. In *Proc. CVPR* (2014).
30. Tompson, J., Stein, M., LeCun, Y., and Perlin, K. Real-time continuous pose recovery of human hands using convolutional networks. In *ACM Trans. Graph.* (2014).
31. Wang, R., Paris, S., and Popović, J. 6D hands. In *Proc. ACM UIST* (New York, New York, USA, Oct. 2011), 549–558.
32. Wang, R. Y., and Popović, J. Real-time hand-tracking with a color glove. In *ACM Trans. Graph.*, vol. 28 (2009), 63:1–63:8.
33. Wang, Y., Min, J., Zhang, J., Liu, Y., Xu, F., Dai, Q., and Chai, J. Video-based hand manipulation capture through composite motion control. *ACM Trans. Graph.* (2013).
34. Wu, Y., and Huang, T. S. View-independent recognition of hand postures. In *Proc. CVPR* (2000), II:88–94.
35. Wu, Y., Lin, J. Y., and Huang, T. S. Capturing natural hand articulation. In *Proc. CVPR* (2001), II:426–432.
36. Xu, C., and Cheng, L. Efficient hand pose estimation from a single depth image. In *Proc. ICCV* (2013), 3456–3462.
37. Yuille, A., and Kersten, D. Vision as Bayesian inference: analysis by synthesis? *Trends in cognitive sciences 10*, 7 (2006), 301–308.
38. Zhao, W., Chai, J., and Xu, Y.-Q. Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data. In *Proc. Eurographics Symposium on Computer Animation* (2012), 33–42.