Fast and Robust Hand Tracking Using Detection-Guided Optimization

Srinath Sridhar[1], Franziska Mueller[1,2], Antti Oulasvirta[3], Christian Theobalt[1]
[1]Max Planck Institute for Informatics, [2]Saarland University, [3]Aalto University
{ssridhar,frmueller,theobalt}@mpi-inf.mpg.de, antti.oulasvirta@aalto.fi

Abstract

Markerless tracking of hands and fingers is a promising enabler for human-computer interaction. However, adoption has been limited because of tracking inaccuracies, incomplete coverage of motions, low framerate, complex camera setups, and high computational requirements. In this paper, we present a fast method for accurately tracking rapid and complex articulations of the hand using a single depth camera. Our algorithm uses a novel ***detection-guided optimization*** *strategy that increases the robustness and speed of pose estimation. In the detection step, a randomized decision forest classifies pixels into parts of the hand. In the optimization step, a novel objective function combines the detected part labels and a Gaussian mixture representation of the depth to estimate a pose that best fits the depth. Our approach needs comparably less computational resources which makes it extremely fast (50 fps without GPU support). The approach also supports varying static, or moving, camera-to-scene arrangements. We show the benefits of our method by evaluating on public datasets and comparing against previous work.*

1. Introduction

There is increasing interest in using markerless hand tracking in human-computer interaction, for instance when interacting with 3D applications, augmented reality, smart watches, and for gestural input [11, 13, 30]. However, flexible, realtime markerless tracking of hands presents several unique challenges. First, natural hand movement involves simultaneous control of several (≥ 25) degrees-of-freedom (DOFs), fast motions with rapid changes in direction, and self-occlusions. Tracking fast and complex *finger articulations* combined with global motion of the hand at *high framerates* is critical but remains a challenging problem. Second, many methods use dense camera setups [16, 22] or GPU acceleration [15], *i.e.* have high *setup costs* which limits deployment. Finally, applications of hand tracking demand tracking across many camera-to-scene configurations including desktop, egocentric and wearable settings.

This paper presents a novel method for hand tracking with a single depth camera that aims to address these challenges. Our method is extremely fast (nearly equalling the capture rate of the camera), reliable, and supports varying close-range camera-to-hand arrangements including desktop, and moving egocentric (camera mounted to the head).

The main novelty in our work is a new **detection-guided optimization** strategy that combines the benefits of two common strands in hand tracking research—model-based generative tracking and discriminative hand pose detection—into a unified framework that yields high efficiency and robust performance and minimizes their mutual failures (see Figure 1). The first contribution in this strategy is a novel, efficient representation of both the input depth and the hand model shape as a mixture of Gaussian functions. While previous work used primitive shapes like cylinders [15, 16] or spheres [19] to represent the hand model, we use Gaussian mixtures for both the depth data and the model. This compact, mathematically smooth representation allows us to formulate pose estimation as a 2.5D generative optimization problem in depth. We define a new **depth-only** energy, that optimizes for the similarity of the input depth with the hand model. It uses additional prior and data terms to avoid finger collisions and preserve the smoothness of reconstructed motions. Importantly, since the energy is smooth, we can obtain analytic gradients and perform rapid optimization. While pose tracking on this energy alone could run in excess of 120 fps using gradient-based local optimization, this often results in a wrong local pose optimum.

The second contribution in our strategy is thus to incorporate evidence from trained randomized decision forests that label depth pixels into predefined parts of the hand. Unlike previous purely detection-based approaches [6, 20], we use the part labels as additional constraints in an augmented version of the aforementioned depth-only energy, henceforth termed **detection-guided** energy. The part labels include discriminative detection evidence into generative pose estimation. This enables the tracker to better recover from erroneous local pose optima and prevents temporal jitter common to detection-only approaches. The precondition for recovery is reliability of the part labels. However, even with large training sets it is hard to obtain perfect part classification (per-pixel accuracy is usually around 60%). Thus, pose estimation based on this additional discriminative evidence is also not sufficient.

Our third contribution therefore, is a new **late fusion approach** that combines particle-based multi-hypothesis optimization with an efficient local gradient-based optimizer. Previous work has used particle-based optimizers, but they tend to be computationally expensive [15, 16]. Our approach is fast because we combine the speed of local gradient-based optimization with the robustness of particle-based approaches. At each time step of depth video, a set of initial pose hypotheses (particles) is generated, from which a subsequent local optimization is started. Some of these local optimizers use the depth-only pose energy, some others use the detection-guided energy. In a final late fusion step the best pose is chosen based on the pose fitting energy.

Our approach results in a temporally stable and efficient tracker that estimates full articulated joint angles of even rapid and complex hand motions at previously unseen frame rates in excess of 50 fps, even with a CPU implementation. Our tracker is resilient to erroneous local convergence by resorting to the detection-guided solution when labels can be trusted, and it is not misled by erroneous detections as it can then switch to the depth-only tracking result.

We show these improvements with (1) qualitative experiments, (2) extensive evaluation on public datasets, and (3) comparisons with other state-of-the-art methods.

2. Related Work

In this brief review we focus on previous approaches to markerless hand tracking from depth images. First, we briefly discuss marker-based and multi-camera techniques. Gloves fitted with retro-reflective markers or color patches were used to estimate the kinematic skeleton using inverse kinematics [25, 31, 35]. Research on *markerless* tracking was made popular in the early 2000s (*e.g.* [1, 33]). Some recent solutions assume a multi-camera setup with offline processing [3, 16, 32], while others track at interactive rates [22, 30] of up to 30 fps [23]. However, calibrated multi-camera setups make these methods difficult to adopt for practical applications. The recent introduction of consumer depth sensors has resulted in a number of methods that require only a single depth camera. Some commercial solutions exist, such as the Leap Motion. Although Leap Motion is fast, the approach uses strong priors and fails with complex self-occlusions and non-standard motions (we show an example in Section 7).

The main approaches to real-time hand tracking can be divided into two classes: (1) generative and (2) discriminative methods.[1] First, [9] proposed a method to track a hand manipulating an object that takes 6.2 s/frame. [15] proposed a model-based method that made use of particle-swarm optimization. This method requires GPU acceleration to achieve 15 fps and uses skin color segmentation which is sensitive to lighting. They showed an extension to interacting hands, although only offline [17, 18]. [14] proposed a tracking method directly in depth by efficient parallel physics simulations. While this method is fast, finger articulations are often incorrectly tracked, as we demonstrate later. Recent real-time surface tracking methods from depth [36] were applied to hands, but are limited to simple motions with no occlusions.

Second, decision forests were used with great success for full body tracking [8, 20] and later adopted to hand tracking with varying success. [10] proposed a method for recognizing finger spelling in depth data using classification forests. [6, 26, 27, 34] also proposed methods based on variants of random forests. Tompson *et al*. [28] track hand motion from depth at $\leq$ 25 fps using feature detections from a convolutional network and further pose refinement through inverse kinematics. However, a common problem with these approaches is jitter due to missing temporal information at each time step. We provide a direct comparison with one recent method [26] to demonstrate this. Moreover, most methods estimate joint positions with temporally varying bone lengths, limiting applicability.

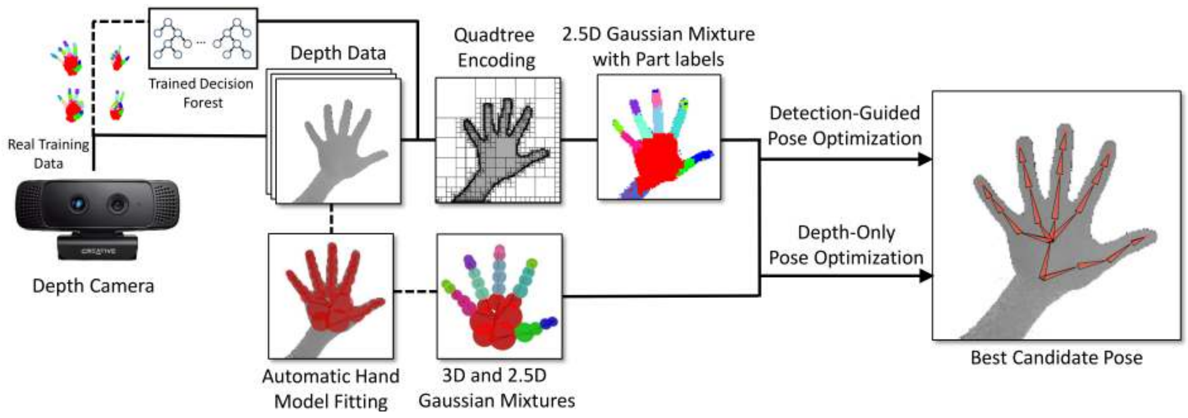
[22] proposed combining discriminative and generative hand pose estimation. Their approach detected only fingertips, which could easily be occluded or misdetected. Offline tracking in RGBD using a combination of discriminative and generative pose estimation was shown in [29]. [19] proposed a method based on optimization in combination with discriminative fingertip detection, achieving 25 fps. However, tracking would be hard with this method when one or more of the fingertips are occluded.

In this paper we present a method that combines decision forests and pose estimation in a unified optimization framework. To our knowledge, ours is the first method to track rapid articulations at 50 fps using a single depth camera and yet achieve state-of-the-art accuracy.

3. Input and Model Representation

In the past, representations such as spheres or cylinders have been used to represent the hand model [15, 19]. Similarly, downsampled images [30, 31] or silhouettes [3] have been used as representations of input data. However, such representations make pose optimization energies discontinuous and difficult to optimize. Our novel representation of depth and 3D model data uses a mixture of *weighted* Gaussian functions to represent both depth data and the hand

[1] There are algorithmic parallels to full-body tracking [2, 7, 12, 20].



Trained Decision Forest
Real Training Data
Depth Camera
Depth Data
Quadtree Encoding
2.5D Gaussian Mixture with Part labels
Detection-Guided Pose Optimization
Depth-Only Pose Optimization
Best Candidate Pose
Automatic Hand Model Fitting
3D and 2.5D Gaussian Mixtures

Figure 1. Overview of our detection-guided tracking method. We develop a novel representation for depth data and hand model as a mixture of 2.5D Gaussians. This representation allows us to combine the benefits of model-based generative tracking and discriminative part detection. Pixels classified using a trained decision forest are directly incorporated as evidence in detection-guided pose optimization. Dashed lines indicate offline computation. Best viewed in color.

shape. We were inspired by [24] who use multiple 2D RGB images and [12] who use depth data. Both methods rely on a uniformly weighted Gaussian mixture, and a 2D or 3D error metric for pose estimation. However, we make important modifications that allows representing 3D depth data using a 2.5D formulation since data from depth sensors contains information only about the *camera-facing* parts of the scene. Thus, we enable pose estimation based on alignment to a single depth image using a 2.5D error metric.

An instance of the input depth or the hand model can be represented as a mixture of Gaussian functions

$$\mathcal{C}(\mathbf{x}) = \sum_{i=1}^{n} w_i \, \mathcal{G}_i(\mathbf{x}; \sigma, \boldsymbol{\mu}), \tag{1}$$

where $\mathcal{G}_i(.)$ denotes a unnormalized Gaussian function with isotropic variance, σ^2, in all dimensions of $\mathbf{x} \in \mathcal{R}^n$, and mean $\boldsymbol{\mu}$. The Gaussian mixture representation has many advantages. First, it enables a mathematically smooth pose estimation energy which is analytically differentiable. Second, only a few Gaussians are needed for representing the input depth and the hand model, an implicit data reduction which makes optimization extremely fast. Finally, it provides a natural way to compute collisions using an analytically differentiable energy. We show later that collisions are important for pose estimation (Section 4). To aid visualization we henceforth represent each Gaussian in the mixture as a sphere ($\mathbf{x} \in R^3$) or circle ($\mathbf{x} \in R^2$) with a radius of $1\,\sigma$. However, Gaussians have infinite support ($\mathcal{C}(\mathbf{x}) > 0$ everywhere) and can produce long range attractive or repulsive *force* during pose optimization.

3.1. Depth Data Representation

The depth camera outputs depth maps, *i.e.* each pixel has an associated depth value. Depth maps contain only the camera-facing parts of the scene and information about occluded parts is unavailable. We therefore only represent the camera-facing pixels using Gaussian mixtures, which are computed in real-time.

First, we decompose the depth image into regions of homogeneous depth using a quadtree. The quadtree recursively decomposes depth image regions further, until the depth difference between the furthest and nearest point in a region is below a threshold ϵ_c ($\epsilon_c = 20$ mm in all experiments). To each quad in the tree, we fit a Gaussian function with $\boldsymbol{\mu}$ set to the center of the quad, and $\sigma = a/\sqrt{2}$, where a is the side length of the quad. We also set each Gaussian function to have unit weight w_i since we consider all input data to be equally important. This leads us to an analytic representation of the *camera-facing surface* of the input depth, $\mathcal{C}_I(\mathbf{x}) = \sum_{q=1}^{n} \mathcal{G}_q(\mathbf{x})$, where $\mathbf{x} \in \mathcal{R}^2$ and n is the number of leaves in the quadtree. Additionally, each quad has an associated depth value, d_q, which is the mean of all depth pixels within the quad. Figure 1 illustrates the process of converting input depth to a Gaussian mixture.

3.2. Hand Model

We model the volumetric extent of the hand analytically using a mixture of 3D Gaussian functions, $\mathcal{C}_h(\mathbf{x}) = \sum_{h=1}^{m} w_h \, \mathcal{G}_h(\mathbf{x})$ where $\mathbf{x} \in \mathcal{R}^3$ and m is the number of Gaussians. We assume that the best fitting model has Gaussians whose isosurface at $1\,\sigma$ coincides with the surface of the hand. In Section 6 we present a fully automatic procedure to fit such a hand model to a user. Additionally, $\mathcal{C}_h$, is attached to a *parametric*, kinematic skeleton similar to that of [21], *i.e.* each 3D Gaussian is attached to a bone which determines its mean position in 3D. We use $|\Theta| = 26$ skeletal pose parameters in twist representation, including 3 translational DOFs, 3 global rotations, and 20 joint angles.

Model Surface Representation: $\mathcal{C}_I$ is a representation of the *camera-facing surface* while $\mathcal{C}_h$ represents the full volumetric extent of the hand. In order to create an equivalent representation of the hand model that approximates the camera-facing parts, which we later use in pose optimization (Section 4). For each model Gaussian in $\mathcal{C}_h$, we create a new projected Gaussian such that the projected hand model has the form $\mathcal{C}_p = \sum_{p=1}^{m} w_p \, \mathcal{G}_p(\mathbf{x})$ where $\mathbf{x} \in R^2$ and $w_p = w_h \, \forall \, h$. $\mathcal{C}_p$ is a representation of the hand model as seen from the perspective of the depth camera and is defined over the depth image domain. The parameters of each Gaussian $\mathcal{G}_p$ are set to be $(\boldsymbol{\mu}_p, \sigma_p)$, where $\boldsymbol{\mu}_p = \mathbf{K} \, [\, \mathbf{I} \, | \, \mathbf{0} \,] \, \boldsymbol{\mu}_h$. Like [24] we approximate the perspective projection with a scaled orthographic projection, yielding 2D Gaussians with $\sigma_p = \sigma_h \, f / \left[\boldsymbol{\mu}_p\right]_z$. Here f is the focal length of the camera, and $\left[\boldsymbol{\mu}_p\right]_z$ denotes the z-coordinate of the Gaussian mean.

4. Hand Pose Optimization

In this section we describe our new formulation of pose estimation as an optimization problem using the Gaussian mixture representation of 2.5D depth data (See Figure 1). Our algorithm uses two variants of a model-to-image similarity energy, one that is only based on depth data (Section 4.1), and another that is guided by decision forests-based part detection (Section 4.3). Pose estimates obtained with each energy are used by a late fusion approach to find the final pose estimate (Section 5). Input to pose optimization at each time step of depth video is the 2.5D mixture of Gaussians representation of a depth image $\mathcal{C}_I$. The latter is computed after median filtering the depth (to remove flying pixels in time-of-flight data), and for a constrained working volume in depth between 150 mm and 600 mm from the camera. The 3D Gaussian mixture of the hand model is denoted by $\mathcal{C}_h$ and its projected version is denoted by $\mathcal{C}_p$.

4.1. Depth-Only Pose Optimization

Our goal is to optimize for the skeleton pose parameters Θ that best explain the input data and are anatomically plausible. We frame an energy that satisfies our goal while being mathematically smooth and differentiable. These properties make the energy ideal for fast optimization.

4.2. Objective Function

Our new energy has the following general form:

$$\mathcal{E}(\Theta) = E_{sim} - w_c \, E_{col} - w_l \, E_{lim} - w_s \, E_{smo}, \tag{2}$$

where E_{sim} is a measure of 2.5D similarity between $\mathcal{C}_I$ and $\mathcal{C}_p$, E_{col} is a penalty for collisions between Gaussians in $\mathcal{C}_h$, E_{lim} enforces a soft constraint on the skeleton joint limits, E_{smo} enforces smoothness in the tracked motion. In all our experiments, we used fixed weighting factors chosen by searching for the best accuracy over the dataset: $w_c = 1.0$, $w_l = 0.2$, and $w_s = 1.0$. Before describing each of the terms in detail we first introduce a measure of similarity between two Gaussian mixtures which is the basis for many of the terms in the objective.

Gaussian Similarity Measure: We define a similarity measure between any two pairs of Gaussian mixtures $\mathcal{C}_a$ and $\mathcal{C}_b$ as,

$$E(\mathcal{C}_a, \mathcal{C}_b) = \sum_{p \in \mathcal{C}_a} \sum_{q \in \mathcal{C}_b} D_{pq}, \tag{3}$$

$$\text{where, } D_{pq} = w_p \, w_q \int_{\Omega} \mathcal{G}_p(\mathbf{x}) \, \mathcal{G}_q(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{4}$$

Ω denotes the domain of integration of $\mathbf{x}$. This Gaussian similarity measure has a high value if the spatial support of the two Gaussian mixtures aligns well. It bears resemblance to the Bhattacharyya Coefficient [4] used to measure the similarity of probability distributions while being computationally less expensive.

Depth Similarity Term (E_{sim}): The 2.5D depth similarity term measures the quality of overlap between the projected model Gaussian mixture $\mathcal{C}_p$ and the image Gaussian mixture $\mathcal{C}_I$. Additionally, this measure also incorporates the depth information available for each Gaussian in the mixture. Figure 2 explains this term intuitively. Two Gaussians that are close (in 2D pixel distance) in the depth image obtain a high value if their depth values are also close. On the other hand, the same Gaussians obtain a low value if their depths are too far apart. Formally, this term is defined as,

$$E_{sim}(\mathcal{C}_p, \mathcal{C}_I) = \frac{1}{E(\mathcal{C}_I, \mathcal{C}_I)} \sum_{p \in \mathcal{C}_p} \sum_{q \in \mathcal{C}_I} \Delta(p, q) \, D_{pq} \tag{5}$$

where D_{pq} is as defined in Equation 4 and the *depth similarity factor* is

$$\Delta(p,q) = \begin{cases} 0, & \text{if } |d_p - d_q| \geq 2\,\sigma_h \\ 1 - \frac{|d_p - d_q|}{2\,\sigma_h}, & \text{if } |d_p - d_q| < 2\,\sigma_h \end{cases}. \tag{6}$$

Here, d_p and d_q are the depth values associated with each Gaussian in $\mathcal{C}_p$ and C_q respectively, and σ_h is the standard deviation of the *unprojected* model Gaussian $\mathcal{G}_h$. The surface depth value of each Gaussian in C_p is computed as $d_p = [\boldsymbol{\mu}_h]_z - \sigma_h$. The factor $E(\mathcal{C}_I, \mathcal{C}_I)$ is the similarity measure from equation 3 of the depth image with itself and serves to normalize the similarity term. The Δ factor has a support $[0, 1]$ thus ensuring the similarity between a projected model Gaussian and an image Gaussian is 0 if they lie too far apart in depth.

Collision Penalty Term (E_{col}): The fingers of a hand are capable of fast motions and often come in close proximity with one another causing aliasing of corresponding depth pixels in the input. Including a penalty for collisions avoids fingers *sticking* with one another and Gaussian interpenetration. The 3D Gaussian mixture representation of

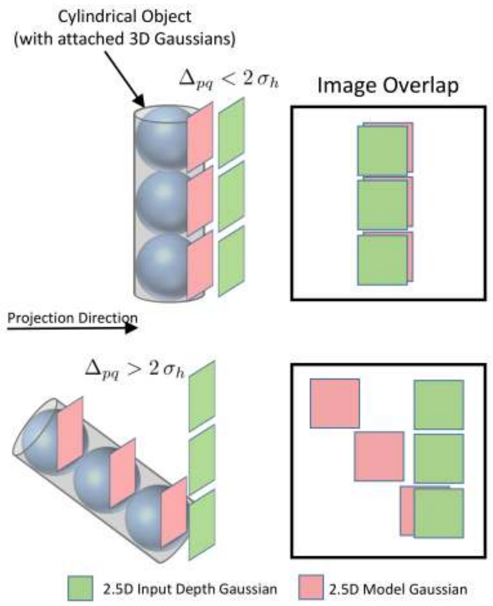

Cylindrical Object
(with attached 3D Gaussians)
$\Delta_{pq} < 2\,\sigma_h$
Image Overlap
Projection Direction
$\Delta_{pq} > 2\,\sigma_h$
2.5D Input Depth Gaussian
2.5D Model Gaussian

Figure 2. **Depth Similarity Term**: Consider the similarity value (E_{sim}) for a cylindrical shape represented by 3 Gaussians ($x \in \mathcal{R}^3$). The top figure shows a case where the value of E_{sim} is high since the image overlap is high and the depth difference Δ_{pq} is low. The bottom figure shows a case where the image overlap is moderate but $\Delta > 2\,\sigma_h$ thus making $E_{sim} = 0$.

the hand model ($\mathcal{C}_h$) offers an efficient way to penalize collisions because they implicitly act as collision proxies. We define the penalty for collisions as,

$$E_{col}(\Theta) = \frac{1}{E(\mathcal{C}_h, \mathcal{C}_h)} \sum_{p \in \mathcal{C}_h} \sum_{\substack{q \in \mathcal{C}_h \\ q > p}} D_{pq}, \tag{7}$$

where $E(\mathcal{C}_h, \mathcal{C}_h)$ is the similarity measure from equation 3 for the hand model and serves to normalize the collision term. The collision term penalizes model Gaussians that collide with others but not if they collide with themselves. As we show in the results, the collision term has a large impact on tracking performance.

Joint Limit Penalty Term (E_{lim}): We add a penalty for poses that exceed predefined joint angle limits. This forces biomechanically plausible poses to be preferred over other poses. The joint limit penalty is given as,

$$E_{lim}(\Theta) = \sum_{\theta_j \in \Theta} \begin{cases} 0, & \text{if } \theta_j^l \le \theta_j \le \theta_j^h \\ ||\theta_j^l - \theta_j||^2, & \text{if } \theta_j < \theta_j^l \\ ||\theta_j - \theta_j^h||^2, & \text{if } \theta_j > \theta_j^h \end{cases} \tag{8}$$

where θ_j^l and θ_j^h are the lower and higher limits of the parameter θ_j which is defined based on anatomical studies of the hand [21]. The result is a tracked skeleton that looks biomechanically plausible.

Smoothness Penalty Term (E_{smo}): During frame-by-frame pose optimization, noise is introduced which manifests as jitter in tracking. To prevent this we penalize fast motions by adding a penalty as done by [24]. This term is given as,

$$E_{smo}(\Theta) = \sum_{j=0}^{|\Theta|-1} \left(0.5\,\left(\Theta_j^{t-2} + \Theta_j^t\right) - \Theta_j^{t-1}\right)^2 \tag{9}$$

where, Θ^t denotes the pose at time t. This term acts as a regularizer and prevents jitter in the tracked pose.

4.3. Detection-Guided Pose Optimization

To increase chances of recovery when the estimated pose is at a wrong local pose optima, we use a second pose optimization energy that includes evidence from hand part detection. In particular we use pixel labels computed with a trained random forest [5]. Decision forests have been used before for 3D pose and joint position detection [10, 26, 27, 34]. We are interested in part labels and therefore follow an approach similar to [20] and [10]. The evidence from the part labels is incorporated in our tracking.

We use 12 part labels for the hand (see Figure 1) and found this to be an ideal trade-off between classification accuracy and sufficient evidence for detection-guided optimization. We adopt the same depth features as [20]. We use 50,000 labeled training images spanning the hand pose space. As opposed to previous work [10] that use synthetic data, we use **real** hand motions with part labels which were obtained using the depth-only version of our method and tracking motions slowly without causing tracking failure. During training we trained 3 trees, each with a maximum depth of 22. For each training image, we sampled 2000 random, foreground pixels, and evaluated 4000 candidate threshold-feature response pairs.

During quadtree clustering of the depth (Section 3.1) each quad is endowed with a part label, l_q which is the label with the highest number of votes among all pixels in the quad. We can now tightly integrate the part labels in the optimization by defining a pose fitting energy identical to Equation 2 with one exception: the depth similarity factor from Equation 6 is replaced by the following *label similarity factor*.

$$\Delta_l(p,q) = \begin{cases} 0, & \text{if } l_p \ne l_q \text{ or } |d_p - d_q| \ge 2\,R_i \\ 1 - \frac{|d_p - d_q|}{2\,R_i}, & \text{if } l_p = l_q \end{cases},$$

where l_p and l_q are the part labels , d_p and d_q are the depth values , R_i refers to the radius of influence which is set to 200 mm in all our experiments. Intuitively, Δ_l has a value of zero if the labels are different and a value of one if the two Gaussians have identical labels and are perfectly aligned in 2.5D. The labels l_p are obtained from preassigned labels of each Gaussian in the hand model.

5. Late Fusion

The goal of optimization is to find the pose Θ such that $-\mathcal{E}(\Theta)$ is minimized. Our energies—both with and without detection—are well suited for gradient based optimization because we can derive the analytic gradient with respect to the DOFs Θ. For efficiency, we adopt the fast gradient-based optimizer with adaptive step length proposed by [24].

To improve robustness, especially with changes in direction and global rotation, we use multiple *pose particles* for optimizing each frame. Multiple particles improve the chances of a good initialization for optimization. Each particle P_i is initialized using the pose parameters from two previous time steps Θ^{t-1} and Θ^{t-2} with different extrapolation factors α_{ij} for each DOF j. This is given as $P_i = \theta_j^{t-1} + \alpha_{ij}\, \theta_j^{t-2}, \forall j$. We sample α_{ij} from a normal distribution with mean fixed at the initial value of θ_j. All but one of these particles is optimized using the depth-only energy. Finally, the pose particle which converges with the best energy value is chosen as the winning pose. In all our experiments, we found that 2–3 particles were sufficient to obtain more robust results. Increasing the particles had a negative effect and caused jitter in the final pose. Each particle used 10–30 iterations per frame. We justify the choice of these parameters in Section 7.

6. User Specific Hand Modeling

Accounting for the fact that there are large variations in anthropometric dimensions, our pose optimization method works best with a customized hand model for each user. Our method does not necessitate laser scans, manual tuning of the model, nor semi-automatic bone model optimization as used by existing methods [24, 22].

We observed in our experiments that the primary variations in hand dimensions are finger thickness, hand length and width. We developed a simple strategy where a default hand model is scaled using three parameters: hand length, width, and variance of Gaussians. To find the scaling parameters for a user, we perform a greedy search over a fixed range for each scaling parameter. At each point on this parameter grid we evaluate the energy function value from Equation 2. The parameters that obtain the best energy are selected as the model scaling parameters. This method is fast and takes less than a second to find a user-specific hand model. Figure 3 shows some qualitative results from our model fitting strategy for different users.

7. Results and Evaluation

We provide quantitative and qualitative evidence for performance with fast motions and finger articulations. Evaluation of hand tracking algorithms is challenging because ground truth data is difficult to obtain. Marker-based motion capture is often problematic due to self-occlusions. Many

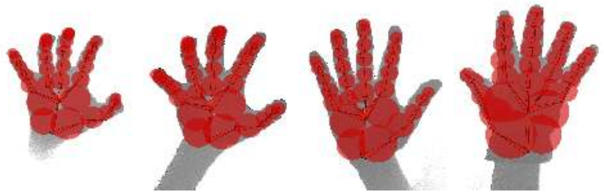


Figure 3. Automatic fitting of user specific hand model for 4 subjects, one of whom is wearing a thick glove to simulate variability in hand dimension. The red spheres denote 3D Gaussians.

methods have therefore resorted to evaluation on synthetic data [15, 16] which, however, is not representative of real hand motions. There are also no established benchmark datasets with accepted error metrics, and only a few implementations have been made public.

We use the dataset from [22] which consists of seven challenging sequences (abduction–adduction, finger counting, finger waving, flexion–extension, pinching, random motions, grasping) that are further split into slow and fast parts. The fingertips are annotated manually in the depth data thus making it possible to compare with the multi-view approaches of [22] and [23]. Additionally, we also compare with the discriminative method of [26] on 3 sequences. We also motivate the need for our fusion strategy, parameter selection in optimization, and analyze the effects of different components of our objective function. We also provide details about our framerate and qualitative evidence of improvements over [14] and the Leap Motion. Please see the supplementary material for more results.

Error Metrics: Our evaluations concern the average fingertip localization error which correlates well with overall pose accuracy. For each sequence, we compute Euclidean error of the 5 fingertip positions averaged over all frames. Additionally, we use a second error metric [19] which is the percentage of frames that have an error of less than x mm where $x \in \{15, 20, 25, 30\}$. This is a stricter measure that highlights reliability.

7.1. Quantitative Evaluation

Accuracy: Figure 4 shows our average error compared with that of [22], [23], and [26]. Our method produces the lowest average error of **19.6** mm while using only a single depth camera. The multi-view approaches of [23] and [22] have errors of **24.1** mm and **31.8** mm respectively. The detection-based discriminative method of [26] has an error of **42.4** mm (3 sequences only) highlighting the need for using temporal information. We observe that our method does particularly well for motions that involve articulation of fingers such as `flexex1`. Our worst performance was on the `random` sequence involving fast global hand rotation.

Error Frequency: Table 1 confirms the trend that our method performs well for finger articulations. In 6 out of 7

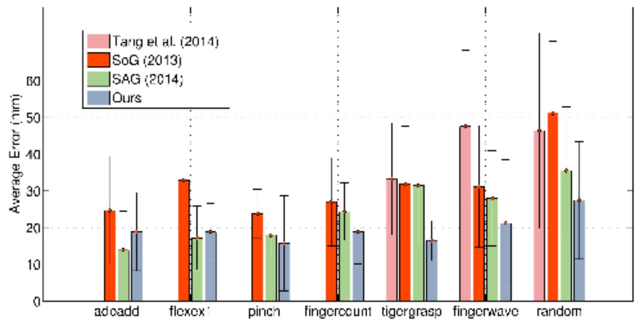


Figure 4. Average error over the 7 sequences in Dexter 1 and comparison with the **multi-view** methods of [22] and [23], and the **detection-based** method of [26]. Our method achieves the lowest error on 5 of the 7 sequences and the best average error (**19.6** mm).

Error $<$ (mm)	**adbadd**	**fingercount**	**fingerwave**
15	56.6	50.0	56.2
20	70.6	66.5	71.2
25	76.2	77.7	78.3
30	84.9	85.8	85.0

Error $<$ (mm)	**flexex1**	**pinch**	**random**	**tigergrasp**
15	53.7	56.7	19.1	62.9
20	68.1	83.9	40.7	80.6
25	76.7	93.1	59.0	87.3
30	85.5	97.4	70.6	91.8

Table 1. Percentage of total frames in a sequence that have an error of less x mm.

sequences, our method results in tracking errors of less than 30 mm in 85% of the frames. A closer examination shows that these sequences contain complex finger articulations.

Robustness: We measure robustness as the ability of a tracker to recover from tracking failures. To demonstrate how our late fusion strategy and the different terms in the energy help achieve this, we show the frame-wise error over the `flexex1` sequence (Figure 6). Using the depth-only energy with all terms except E_{sim} disabled (2 particles) results in catastrophic tracking failure as shown by the accumulating error. Adding the other terms, especially the collision penalty (E_{col}) term, improves accuracy but results are still unsatisfactory. The results from the late fusion approach show large gains in accuracy. The errors also remain more uniform which results in temporally stable tracking with less jitter.

Number of Particles and Iterations: Figure 5 shows the effect of varying the number of particles and iterations during optimization. As the number of particles increased we noticed very little increase in accuracy. In fact, the best accuracy was with 2 particles which we use throughout. We noticed a reduction in error when using more number of

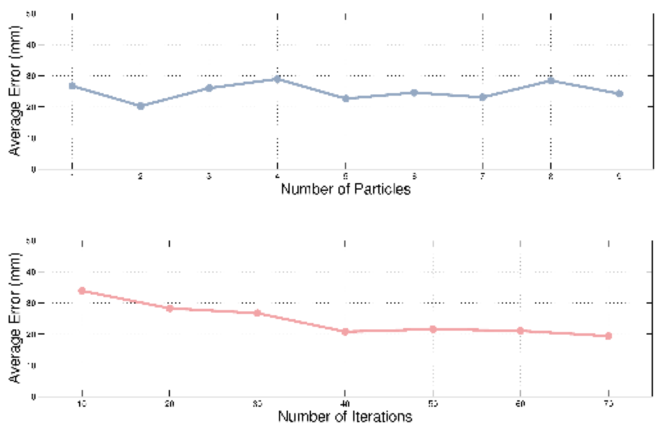


Figure 5. Effect of varying the number of particles and iterations during optimization. We found that increasing the number of particles resulted in diminishing returns.

iterations per particle but at the expense of runtime. We therefore fixed the number of iterations to be 10.

Tracking Speed: We tested the tracking speed of different variants of our method on a 3.6 GHz Intel Processor with 16 GB of RAM. Our method was parallelized using OpenMP but no GPU was used. All tests were done with the Intel Senz3D depth camera with a depth resolution of 320×240 and capture rate of 60 fps. The decision forest when loaded in memory used 1 GB because the trees were stored as full trees. This can be avoided by loading only nodes that are valid. The depth-only energy when used with 1 particle, and 10 iterations per particle ran at 120 fps. When 2 particles were used, the speed came down to 60 fps. The late fusion approach, when used with 2 particles (10 iterations per particle), achieved a framerate of **50** fps. Image acquisition, part labeling, preprocessing, and creating the Gaussian mixture representation took 2 ms. The optimization took between 18 and 20 ms.

7.2. Qualitative Results

We present several qualitative results from realtime sequences in Figure 7. The examples show motions with a wide range of finger articulations involving abduction, adduction, flexion, extension, and considerable occlusions. They also include common gestures such as the *v-sign* and pointing. In the boxes, we also show comparison with [14] and the Leap Motion on similar poses. We observe a finger sliding effect in both these methods. Pinching is an important gesture, but the Leap Motion produces sliding fingertips which makes it hard to detect pinching gestures from the tracked skeleton. Our method reproduces pinching faithfully as is evident from the skeleton overlaid on the depth image. Occasional tracking failures occur with large global rotations but the detection-guided energy eventually

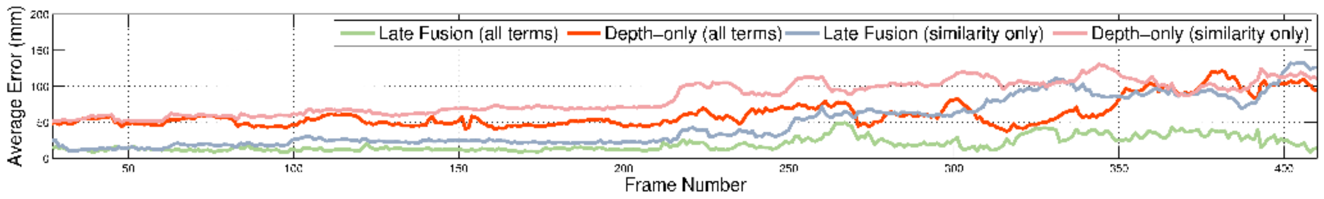


Figure 6. Plot of the error for the depth-only tracking and late fusion approach. Each approach was run with only the similarity term E_{sim} and with all terms. Notice the catastrophic tracking failure with the depth-only energy. The late fusion strategy is robust and prevents error accumulation. The collision penalty term also results in large accuracy gains. Best viewed in color.

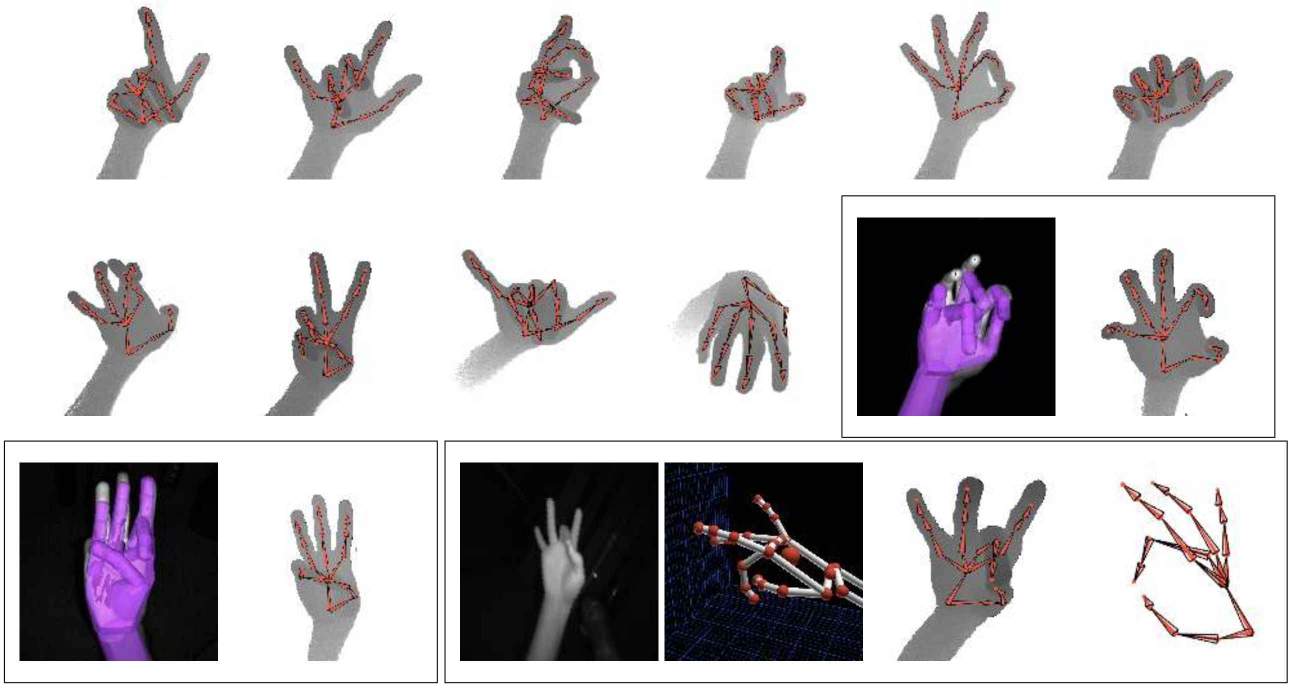


Figure 7. Qualitative results from our tracking approach (top row and four leftmost in the second row). The highlighted boxes show comparison with [14] and the Leap Motion both of which produce a finger sliding effect. Our method tracks the pinch faithfully.

reinitializes to the correct pose. Please see the supplementary materials for more results on different camera arrangements, failure cases, and tracking with different subjects.

8. Conclusion

In this paper, we presented a method for realtime hand tracking using detection-guided optimization. Our method is robust and tracks the hand at 50 fps without using a GPU. We contribute to the tracking literature by proposing a novel representation of the input data and hand model using a mixture of Gaussians. This representation allows us to formulate pose estimation as an optimization problem and efficiently optimize it using analytic gradient. We also showed how additional evidence from part detection can be incorporated into our tracking framework to increase robustness. We evaluated our method on a publicly available dataset and compared with other state-of-the-art methods. An important direction for future work is the tracking of multiple hands interacting with each other or with objects. We believe that the strong analytic formulation offered by our method can help solve this.

Acknowledgments: This research was funded by the ERC Starting Grant projects CapReal (335545) and COMPUTED (637991), and the Academy of Finland. We would like to thank Christian Richardt.

References

[1] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. In *Proc. of CVPR 2003*, pages II–432–9 vol.2. 2

[2] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Proc. of ICCV 2011*, pages 1092–1099. 2

[3] L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys. Motion capture of hands in action using discrimina-

tive salient points. In *Proc. of ECCV 2012*, volume 7577 of *LNCS*, pages 640–653. 2

[4] A. Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhya: The Indian Journal of Statistics (1933-1960)*, 7(4):401–406, July 1946. 4

[5] A. Criminisi and J. Shotton. *Decision forests for computer vision and medical image analysis*. Springer, 2013. 5

[6] S. R. Fanello, C. Keskin, S. Izadi, P. Kohli, D. Kim, D. Sweeney, A. Criminisi, J. Shotton, S. B. Kang, and T. Paek. Learning to be a depth camera for close-range human capture and interaction. *ACM TOG*, 33(4):86:1–86:11. 1, 2

[7] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. In *Proc. of ECCV 2012*, volume 7577 of *LNCS*, pages 738–751. 2

[8] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *Proc. of ICCV 2011*, pages 415–422. 2

[9] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *Proc. of ICCV 2009*, pages 1475–1482. 2

[10] C. Keskin, F. Kirac, Y. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *Proc. of ICCV Workshops 2011*, pages 1228–1234. 2, 5

[11] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, I. Oikonomidis, and P. Olivier. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proc. of UIST 2012*, pages 167–176. 1

[12] D. Kurmankhojayev, N. Hasler, and C. Theobalt. Monocular pose capture with a depth camera using a sums-of-gaussians body model. In *Pattern Recognition*, number 8142 in LNCS, pages 415–424. Jan. 2013. 2, 3

[13] J. Lee, A. Olwal, H. Ishii, and C. Boulanger. SpaceTop: integrating 2D and spatial 3D interactions in a see-through desktop environment. In *Proc. of CHI 2013*, pages 189–192. 1

[14] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3D skeletal hand tracking. In *Proc. of I3D 2013*, pages 184–184. 2, 6, 7, 8

[15] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using kinect. In *Proc. of BMVC 2011*, pages 101.1–101.11. 1, 2, 6

[16] I. Oikonomidis, N. Kyriazis, and A. Argyros. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proc. of ICCV 2011*, pages 2088–2095. 1, 2, 6

[17] I. Oikonomidis, N. Kyriazis, and A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *Proc. of CVPR 2012*, pages 1862–1869, June 2012. 2

[18] I. Oikonomidis, M. Lourakis, and A. Argyros. Evolutionary quasi-random search for hand articulations tracking. In *Proc. of CVPR 2014*, pages 3422–3429. 2

[19] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *Proc. of CVPR 2014*, pages 1106–1113. 1, 2, 6

[20] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proc. of CVPR 2011*, pages 1297–1304. 1, 2, 5

[21] E. Simo Serra. *Kinematic Model of the Hand using Computer Vision*. PhD thesis, Institut de Robòtica i Informàtica Industrial, 2011. 3, 5

[22] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proc. of ICCV 2013*, pages 2456–2463. 1, 2, 6, 7

[23] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt. Real-time hand tracking using a sum of anisotropic gaussians model. In *Proc. of 3DV 2014*, pages 319–326. 2, 6, 7

[24] C. Stoll, N. Hasler, J. Gall, H. Seidel, and C. Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *Proc. of ICCV 2011*, pages 951–958. 3, 4, 5, 6

[25] D. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, 1994. 2

[26] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proc. of CVPR 2014*, pages 3786–3793. 2, 5, 6, 7

[27] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proc. of ICCV 2013*, pages 3224–3231. 2, 5

[28] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM TOG*, 33(5):169:1–169:10, Sept. 2014. 2

[29] D. Tzionas, A. Srikantha, P. Aponte, and J. Gall. Capturing hand motion with an RGB-D sensor, fusing a generative model with salient points. In *Proc. of GCPR 2014*, pages 1–13. 2

[30] R. Wang, S. Paris, and J. Popović. 6D hands: markerless hand-tracking for computer aided design. In *Proc. of UIST 2011*, pages 549–558. 1, 2

[31] R. Y. Wang and J. Popović. Real-time hand-tracking with a color glove. *ACM TOG*, 28(3):63:1–63:8, July 2009. 2

[32] Y. Wang, J. Min, J. Zhang, Y. Liu, F. Xu, Q. Dai, and J. Chai. Video-based hand manipulation capture through composite motion control. *ACM TOG*, 32(4):43:1–43:14, July 2013. 2

[33] Y. Wu, J. Lin, and T. Huang. Capturing natural hand articulation. In *Proc. of ICCV 2001*, volume 2, pages 426–432. 2

[34] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. In *Proc. of ICCV 2013*, pages 3456–3462. 2, 5

[35] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. *SIGCHI Bull.*, 17(SI):189–192, May 1986. 2

[36] M. Zollhöfer, M. Niessner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger. Real-time non-rigid reconstruction using an RGB-D camera. *ACM TOG*, 33(4):156:1–156:12, July 2014. 2