Realtime and Robust Hand Tracking from Depth

Chen Qian[1,2] Xiao Sun[1] Yichen Wei[1] Xiaoou Tang[2] Jian Sun[1]
[1]Microsoft Research [2]Chinese University of Hong Kong
{v-xiasun,yichenw,jiansun}@microsoft.com {qc012,xtang}@ie.cuhk.edu.hk

Abstract

We present a realtime hand tracking system using a depth sensor. It tracks a fully articulated hand under large viewpoints in realtime (25 FPS on a desktop without using a GPU) and with high accuracy (error below 10 mm). To our knowledge, it is the first system that achieves such robustness, accuracy, and speed simultaneously, as verified on challenging real data.

Our system is made of several novel techniques. We model a hand simply using a number of spheres and define a fast cost function. Those are critical for realtime performance. We propose a hybrid method that combines gradient based and stochastic optimization methods to achieve fast convergence and good accuracy. We present new finger detection and hand initialization methods that greatly enhance the robustness of tracking.

1. Introduction

Hand tracking is important in many human computer interaction applications and has been intensely studied for decades [10, 28, 4, 2, 7, 14]. Nevertheless, it remains challenging due to its extraordinary complexity. The hand is highly articulated with complex finger interactions. It usually moves fast under large viewpoint variations.

In spite of significant progress in recent years, the state-of-the-art approaches are limited in certain aspects. The hand motion capture work in [13] obtains high accuracy using a complex mesh model, but is limited by a slow local optimization. The works in [8, 9] use a simple polygonal model and achieve real time performance, but require an expensive GPU for model rendering and cost function evaluation. The optimization is purely local and cannot recover from tracking failure. The global techniques [3, 4] search a large parameter space to avoid poor local optima, but the result is usually coarse and the search is slow. The approaches in [21, 25] perform realtime global search and local optimization, but rely on inconvenient setup (a color glove in [21] and multiple cameras in [25]). Other realtime and robust systems are limited in recognizing discrete hand gestures only [31, 5, 6, 29] without optimization, supporting a small number of DOFs [22], or under a fixed viewpoint [12]. Those limitations above are due to difficult tradeoffs between the system complexity and targeted goals. To achieve high accuracy and speed, previous works use complex model, sophisticated cost function, expensive optimization, or specific setup.

We present a new state-of-the-art hand tracking system. It can track free and complex hand motion in realtime on a desktop with high accuracy. Our work is largely inspired by recent advances in human body tracking [24, 1, 15, 30, 27]. However, directly applying existing body tracking techniques to the hand usually works poorly, due to the unique challenges in hand tracking mentioned above. Therefore, careful adaptation and improvement are needed.

We follow a "Local Optimization + Initialization by Part Detection" framework [1, 15, 30] and present several contributions. We adopt a simple hand model that is approximated using a set of spheres, and a fast cost function that measures the distance between the model and a sparse point cloud. These simplifications are critical for realtime performance. Details are given in Section 2.

In spite of its simplicity, the cost function is still effective in that it reaches the global minimum at the correct hand pose in almost all cases. However, it is not smooth enough and has an abundance of local optima in the high dimensional space. Previous gradient based optimization for body tracking [27] and stochastic optimization for hand tracking [8, 9] cannot minimize the cost well, being either too sensitive to local optima or too slow in convergence. Observing the complementary nature of the two methods, we develop a hybrid optimization method that combines the merits of both. As described in Section 3, it converges faster and resists local optima better.

Part detection and part-based initialization have been proven critical to the robustness in body tracking [18, 1, 30, 24]. Inspired by such works, we propose novel and effective methods for finger detection, segmentation, and hand initialization in Section 4. Comprehensive experiments on challenging real data validate the efficacy of our system, as shown in Section 5.

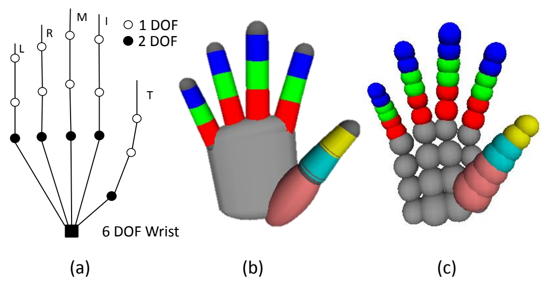

Figure 1. (a) 26 DOFs hand motion model; (b) the hand model in [8]; (c) our hand model that approximates (b) using 48 spheres.

2. Model and Cost Function

Hand Model. To model hand kinematics, we adopt the commonly used 26 degrees of freedom (DOF) hand motion model [21, 9]: 6 DOFs for the global hand pose and 4 DOFs for each finger, illustrated in Figure 1(a). We preserve the kinematic constraints of the hand by enforcing the joint angles in their valid ranges (as defined in [23]) during optimization. The 26 motion parameters are denoted as Θ.

A simple geometric model is critical for fast computation of distance and intersection. For example, in [27], the human body is modeled as mixture of spheres and cylinders. In this work, we use the simplest sphere set representation. Specifically, we approximate the polygonal mesh model in [8] using 48 spheres, as illustrated in Figure 1(b) and (c). The number of spheres for each part is manually specified: 6 for each finger (8 for the thumb) and 16 for the palm. The spheres' sizes and centers are also empirically set to approximate the polygonal model at the canonical pose as in Figure 1. In this work, we do not use any personal model adaption (except a global scale as in Section 5) and find the sphere model works well across a few different subjects. Nevertheless, using optimized personal hand model [26] and sphere approximation technique [20] should further improve the accuracy.

The sphere model is denoted as $\mathcal{M}(\Theta) = \{s_i\}_{i=1}^{48}$. Each sphere $s = \{c(\Theta), r\}$ has center $c(\Theta)$ and radius r. The notation implies that the radii are fixed but the centers depend on the parameter Θ through forward kinematics. We drop the notation Θ in the remainder of this paper for conciseness.

Data. We use Intel's Creative Interactive Gesture Camera. The depth resolution is 320×240. For hand segmentation, we use a black band around the wrist to create depth voids, and find the nearest connected component to be the hand region, assuming the hand is closest to the camera. The hand region is further refined by median filtering and morphological opening, denoted as depth map $\mathcal{D}$. It is then converted to a 3D point cloud, denoted as $\mathcal{P}$.

Cost Function. This measures the discrepancy between the hand model and input depth, as well as hand model validity. It is defined as

$$\lambda \cdot \sum_{p \in sub(\mathcal{P})} D(p, s_{x(p)})^2 + \sum_i B(c_i, \mathcal{D})^2 + \sum_{i,j} L(s_i, s_j)^2. \tag{1}$$

The first term $D(\cdot)$ aligns the point cloud $\mathcal{P}$ to the sphere model $\mathcal{M}$. To reduce the computational complexity, the point cloud is randomly down sampled to 256 points, denoted as $sub(\mathcal{P})$. This operation achieves good trade-off between efficiency and accuracy. For each point p, $x(p)$ indexes its closest sphere and $D(\cdot)$ is the distance from that point to the sphere surface,

$$D(p, s) = abs(||p - c||_2 - r). \tag{2}$$

The second term $B(\cdot)$ forces the model to lie inside the point cloud. Each sphere center is projected onto the depth map as $j(c)$. If the depth at $j(c)$ is closer, the sphere center is in front of the depth and receives a penalty being the depth difference. If there is no depth at $j(c)$, the sphere center is outside the silhouette of the depth and receives a penalty being the distance to silhouette, which is efficiently computed by a distance transform of the silhouette[1]. Formally, this term is defined as

$$B(c, \mathcal{D}) = \begin{cases} \max(0, \mathcal{D}(j(c)) - c_z) & \text{if } \mathcal{D}(j(c)) \text{ is valid} \\ dist(j(c), \text{silhouette of } \mathcal{D}) & \text{otherwise} \end{cases} \tag{3}$$

The third term $L(\cdot)$ penalizes model self-collision. The collision cost is

$$L(s_i, s_j) = \max(r_i + r_j - ||c_i - c_j||_2, 0). \tag{4}$$

We observe that most collisions during optimization are between neighboring fingers and therefore only test neighboring fingers for efficiency.

The cost function is simple and effective. The first term matches the visible spheres to the point cloud. It is weighted by $\lambda = |\mathcal{M}|/|sub(\mathcal{P})|$ so its magnitude is the same as the second term. The second term forces the occluded spheres to lie behind depth and complements the first one. It has a similar spirit as [27, 30] but is simpler since it is only evaluated on a few spheres.

The first term has computational complexity $O(|\mathcal{M}||sub(\mathcal{P})|)$ as the nearest sphere for each point needs to be computed. The last two terms have complexity $|\mathcal{M}|$. As both $|\mathcal{M}|$ and $|sub(\mathcal{P})|$ are small, evaluation of the cost function (1) is very fast. This is critical for realtime performance, as any reasonable optimization method would evaluate a cost function many times.

For simplicity we do not consider temporal coherency in the cost function and left this as future work.

[1]The distance is measured in pixels and converted to millimeters using the average input depth.

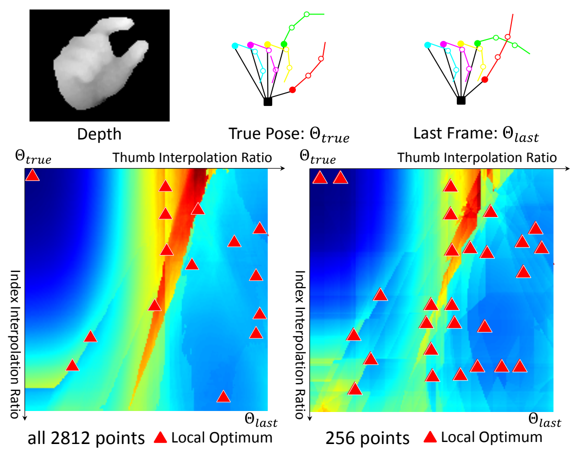

Figure 2. (better viewed in color) Illustration of a real tracking example. Top: due to the fast motion of the thumb and index fingers, the pose from the last frame is a poor initialization for the current frame. Bottom: we generate intermediate poses by interpolating thumb and index finger parameters of the last pose and true pose while keeping other parameters intact. The cost function (1) is densely evaluated on the two interpolation coefficients ($[0, 1]$) and visualized, using dense and sparse 3D point cloud, with local optima overlaid.

3. ICP-PSO Local Optimization

Tracking is performed by the local optimization of (1) from an initial hand pose, which is either from last frame or finger detection on current frame.

For such point-model alignment tasks, Iterated Closest Point (ICP) method [17, 19] is widely used. It uses alternate and gradient based optimization, converges fast, and is suitable for realtime applications. However, it can be easily trapped in poor local optima and cannot handle non-rigid objects well. Various extensions have been proposed to handle articulated objects [16], and ICP has been successfully adapted for human body tracking [27] recently. Yet, it is still insufficient for high-dimensional articulated hands, especially under free viewpoints. Even worse, for realtime performance we are limited to using a sparse subset instead of the whole point cloud. All above factors lead to an abundance of local optima in the cost function. The fast hand motion also frequently leads to poor initialization.

The challenges are exemplified in Figure 2. There are many local optima that would trap the gradient based optimization from the poor initialization in last frame. The problem deteriorates with a sparse sampling of the point cloud.

Stochastic optimization is necessary to alleviate this problem. We use the Particle Swarm Optimization (PSO) method [11] and briefly review it here. A particle is a high dimensional parameter vector, a swarm is a collection of particles, and PSO is an evolutionary process where particles interact with each other to search the parameter space. During evolution, the global best known particle position of the whole swarm and the local best known position of each particle are remembered. The initial particles are randomly sampled (usually around an initial particle) and their initial velocities are all set to zero. In each generation of evolution, a particle's velocity is updated as a randomly weighted summation of its previous velocity, the velocity towards its local best position, and the velocity towards the global best position. The particle then moves at its velocity from its previous position. After all particles move, the global best and local bests are updated.

PSO can better explore the parameter space and avoid poor local optima by attracting more particles to more promising areas. Recently, it has been successfully used for hand tracking [8, 9]. However, it does not work well in our case. A possible reason is that the cost function in [9] compares all points' depth but our cost function uses sparse points and is less smooth, as shown in Figure 2. We find that even with many particles, PSO still converges slowly and cannot effectively minimize the cost. This problem is called *particle premature* [11] and frequently observed in high dimensional space. Because each particle has a large local volume to search, the random search in PSO is not efficient enough and a particle could be attracted to an incorrect global best too early, even when there exists a good local optimum nearby.

The above analysis indicates that the two approaches are complementary by nature: ICP quickly reaches local optima; PSO explores parameter space more effectively but suffers from premature convergence. We propose a hybrid optimization approach to combine the merits and overcome the drawbacks of both. The key idea is that *each particle takes an additional ICP like gradient descent step before the random particle movement in each PSO generation*. In this way, each particle moves faster and minimizes the cost more effectively, as in ICP. All particles interact with each other to sample the promising area more frequently and a single one has a higher chance to jump from a poor local optimum, as in PSO. Consequently, the combined approach converges faster and resists local optima better than both.

An illustrative example is shown at the top of Figure 3. In generation 0, a few particles are randomly sampled around the poor initialization (bottom right) and the best one (solid green cross) is attracted to the promising area (top left) after ICP. Through PSO update, the entire swarm is gradually attracted to the promising area, and finally reaches the correct solution in generation 10.

We further extend PSO to deal with multiple local optima more effectively. In each generation the particles are divided into multiple clusters using k-means clustering and the average hand joint distance as the particle distance. This dynamic particle re-allocation uses particles more effectively, as a better local optimum usually attracts more particles.

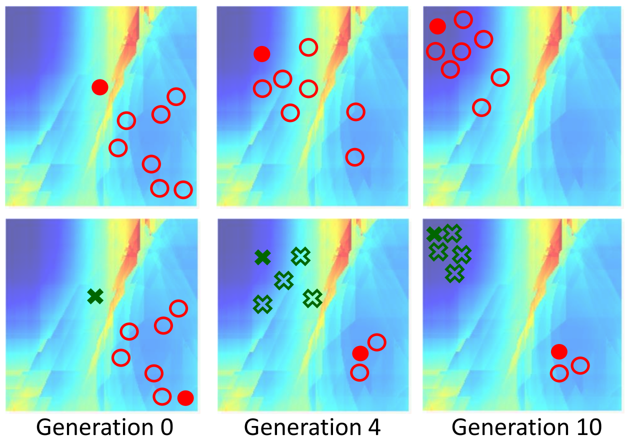

Figure 3. (better viewed in color) Illustration of the ICP-PSO optimization processes using the example in Figure 2 for $k = 1$ (top) and $k = 2$ (bottom). See text for details.

Random particle update is performed within each cluster independently to keep these clusters around their own local optima. This is illustrated at the bottom of Figure 3. When $k = 2$, the particles automatically converge to the two local optima. This further alleviates the particle premature.

Our optimization method is called *ICP-PSO* and is sketched in Algorithm 1. In the initialization, each particle is a random perturbation of the pose in last frame. Whenever a finger detection based hand pose is available, we empirically allocate $1/4$ particles for the second initial. In the random perturbation, each dimension is independently drawn from a 1D gaussian distribution whose center is the initial value and the standard deviation is manually specified as: 5 degrees for angles (the joint angle and global rotation) and 15 mm for the global position. In the ICP part, we use a similar strategy as in [16], *i.e.*, instead of a full Levenberg-Marquardt (LM)-like gradient descent of all 26 parameters, gradient descent is only performed on a randomly selected parameter and the process is repeated a small number of times (empirically set to $m = 10$). This has been shown to be more robust than LM in [16] and we have observed a similar result in our experiments. For conciseness, we do not elaborate PSO part but refer the reader to [9] for more details.

4. Finger Detection for Hand Initialization

Tracking from only the last frame is fragile. Recent advances in human body tracking [24, 1, 15, 30, 27] have proven that the capability of re-initialization on every frame is critical for robust tracking. In this work, we present simple and effective methods for finger detection and hand initialization. They are derived from an intuitive geometrical viewpoint, without using learning such as in [25].

Finger Detection. Recent body part detection methods [18, 1] find extreme points on the 3D point cloud using 3D geodesic distances. We find this approach does not perform well on hand because the underlying 3D graph is quite unstable in case of finger occlusions and depth noises caused by the fast motion. Consequently, finger tips are often not among the top detected extreme points.

Algorithm 1 Our *ICP-PSO* optimization method.

```
1:  Input: initial hand pose(s) from the last frame (and finger detection)
2:  generate random particles around the initial pose(s)
3:  for each generation do
4:      for each particle do
5:          compute point-sphere correspondences
6:          for $m$ times do
7:              gradient descent on a random parameter
8:          end for
9:      end for
10:     $k$-means clustering of all particles
11:     particle swarm update within each cluster
12: end for
13: Output: the best particle
```

The key idea in our method is to exploit the simplicity of finger geometry. Instead of the 3D point cloud, we detect the extreme points on 2D XY plane and 1D Z direction, separately. This is much simpler and more stable. To classify the extreme points as finger tip/non-tip, we grow a finger segment proposal from each extreme point and check whether the segment geometry is similar to a finger. This geometric checking is intuitive, fast and accurate.

XY-Fingers are parallel to the image plane and detected on a mask binarized from the depth map. We initialize the first extreme point as the mask center and compute its geodesic distances to all pixels using distance transform. We then repeatedly add the maximal point in the distance map as a new extreme point and update the distance map in an incremental manner, similar to [1]. From each extreme point, a segment grows on the current distance map until its length or width exceeds a full finger's sizes. Note that the segment length/width are in pixels on the 2D mask and they are converted to millimeters using camera parameters and pixel depths for comparison. The segment is considered as a finger if its length, width and aspect ratio are all close enough to a real finger[2]. The process is repeated six times to find the wrist and (at most) five finger tips.

Z-Fingers are pointing towards the camera and their tips are detected as local minima on the depth map. From each tip, a segment grows by flood fill until its depth exceeds the size of top finger phalange. For classification of tip/non-tip, we observe that a good Z-Finger segment should have the

[2]As finger identity is unknown for now, we simply use the index finger of our hand model for such comparison.

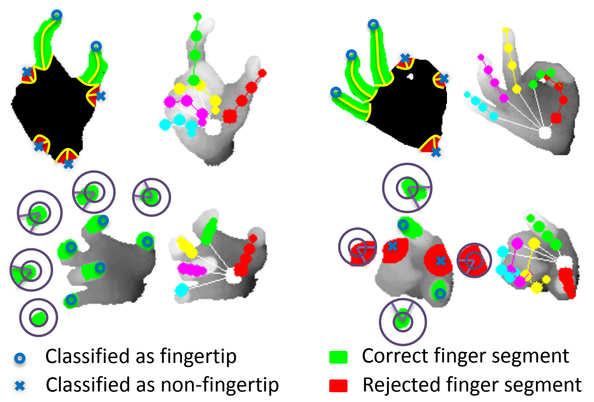

Figure 4. (better viewed in color) Illustration of finger detection and hand pose initialization. For each example, left shows detected extreme points, finger segment proposals, and tip/non-tip classification results. Right shows the estimated hand pose. The top two examples are XY Fingers. Each segment visualizes its length and width lines that are used for geometric checking. The bottom two examples are Z Fingers. Each segment visualizes its two spheres and the sector in the ring for geometric checking.

most pixels in a small sphere that centers on the extreme point and approximates the top phalange. The remaining pixels outside the sphere should reside in a small angular range. Therefore, we check a ring between the small sphere and a larger ($3\times$) sphere, find the sector (spanning 60 degrees) with highest density within the ring, and consider this segment as correct if the sector contains the most pixels in the ring (more than 90%). The sector is efficiently found using a 1D integral histogram that counts the pixel along the angular dimension in the ring.

Our methods and results are illustrated in Figure 4. We do not strictly distinguish XY and Z Fingers. Slanted fingers can be usually detected by both. We detect XY-Fingers before Z-Fingers, and discard any segment immediately whenever it grows to touch an existing segment. Our methods use simple operations (incremental distance transform and flood fill) for only a few times (six in XY and typically about five for Z) in small patches, thus very fast. We further down sample the 320×240 depth map to 160×120 for efficiency. The detection takes 2 ms (1 ms for XY and Z) on average. All parameters in segment growth and geometric checking are empirically set to achieve high precision.

Hand Initialization While inverse kinematics is the standard technique to estimate an articulated shape (such as hand) from end effectors (such as finger tip), it is unstable for highly articulated hand. We propose a simpler and more robust approach that also uses finger segments. Given f detected fingers, we assume detected fingers are straight (each 2 DOFs) and undetected fingers are bent (DOFs are ignored). Therefore, the hand pose parameters are simplified to $2f + 6$ DOFs, denoted as Θ'.

Each finger tip t provides 3 constraints. From each finger segment we estimate its direction d by PCA, which provides 2 constraints. We remove the finger segments from the point cloud and use the remaining 3D points to estimate the palm orientation l by PCA, which provides 3 constraints. Therefore we have $5f+3$ constraints, which are sufficient to solve $2f + 6$ unknowns for $f = 1$ to 5.

From forward kinematics, we can derive each finger tip $\hat{t}(\Theta')$, finger direction $\hat{d}(\Theta')$, and palm orientation $\hat{l}(\Theta')$. We find the optimal hand pose that minimizes the differences between those quantities,

$$\begin{aligned}\Theta'_{opt} &= \arg_{\Theta'} \min \textstyle\sum_{i=1}^{f} ||t_i - \hat{t}_i(\Theta')||_2 \\ &+ \textstyle\sum_{i=1}^{f} angle(d_i, \hat{d}_i(\Theta')) + angle(l, \hat{l}(\Theta')).\end{aligned} \quad (5)$$

As the finger identity is unknown, we enumerate several combinations, run optimization for each and choose the solution with the smallest cost in terms of (5). This optimization problem is small and takes less than 1 ms to solve.

Due to the simplifications in Θ' made above, the estimated hand pose is usually rough. However, it is usually well aligned at the fingers and is good enough to initialize the local optimization in Section 3. Our method works well for many useful gestures with extending and visible fingers, *e.g.*, those in sign language. See Figure 4 for example results. It is less effective for complex gestures with bent and occluded fingers, which however, would also challenge any other finger detection method.

5. Experiments

The evaluation of hand tracking in the literature is still primitive. There lacks common datasets, protocols and metrics. This makes cross-approach comparison quite difficult. In this work, we create a real challenging dataset with manually labeled ground truth[3]. Up to our knowledge, its complexity and magnitude is the most comprehensive in the literature. We use strict evaluation processes and metrics that are rarely done before. We hope these could advance the experimentation practices for future work.

Dataset and Metrics. We ask six subjects to make various rapid gestures. A 400-frame video sequence is recorded for each. We manually label the ground truth hand pose for 2400 frames. To account for different hand sizes, a global hand model scale is specified for each subject (see Table 2), but no further personal adaptation is used.

In the evaluation, we measure the average error E of six joints: the five finger tips and the wrist. This measure is strict because these joints are semantically important and present bigger errors than other internal joints. We also measure the success rate S, which is the percentage of good frames which have $E < 10mm$.

[3]Available at http://research.microsoft.com/en-us/people/yichenw/

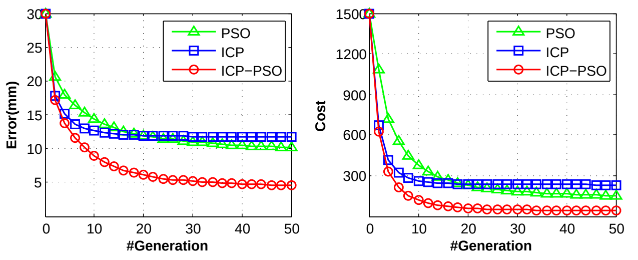

Figure 5. Average error and cost function values decrease as optimization methods run through generations.

Evaluation of Optimization. We compare the proposed ICP-PSO method with the baseline methods ICP and PSO, which are simplified from Algorithm 1. The ICP baseline removes lines 10 to 11 and is equivalent to multiple independent runs. The PSO baseline removes lines 4 to 9 and sets $k = 1$. We use 128, 40, and 32 particles for PSO, ICP, and ICP-PSO, respectively, so that they have approximately the same running time for one generation. All methods run for 50 generations and generally converge.

It is inappropriate to directly compare video tracking results. Because results of later frames highly depend on those of earlier frames, the performance cannot be fully attributed to the methods themselves. Instead, comparison is performed on single frames under the same initialization. For each frame, we randomly perturb the ground truth pose to generate various initial poses. The initial poses are then divided into three difficulty levels. All levels contain 10 initial poses in each frame and their average joint errors are within $[15mm, 25mm]$, $[25mm, 35mm]$, and $[35mm, 45mm]$, respectively.

To further consolidate our experiments, we also synthesize depth maps for each frame using our labeled ground truth hand pose. We use the polygonal model in Figure 1(b) instead of our sphere model to make the depth more faithful. Table 1 reports the average accuracies on all levels using both real and synthetic depth. It clearly shows that ICP-PSO is better on all difficulty levels, and using multiple clusters ($k > 1$) is better than using one cluster $k = 1$, especially when initial errors are large and local optima problem is severe. These conclusions are consistent in both real and synthetic results, while real results are slightly worse. We fix $k = 4$ in our remaining experiments as it is optimal when using 32 particles.

We then investigate the effect of number of generations and particles, using difficulty level $[15mm, 25mm]$ and real depth. Conclusions remain the same in other cases. Figure 5 shows that our method decreases the cost function and improves the accuracy much more quickly. Figure 6 shows that using more particles can improve accuracy, but ICP and PSO are still worse even with more particles.

Evaluation of Tracking and Initialization During tracking, we initialize the first frame using ground truth. We also implement the state-of-the-art method in [9] (FORTH)[4]. In total, four methods are compared, using (denoted as $*$) and not using finger-based hand initialization. Table 2 reports the average joint errors over all frames for all subjects. Our method already achieves good accuracy without initialization. Using initialization further improves all methods significantly. Figure 7 shows errors over all frames for the first subject. The large errors without using initialization are mostly reduced using initialization. Example results of all methods are shown in Figure 8.

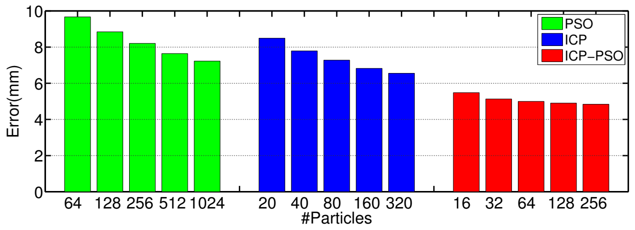

Figure 6. Average errors using different number of particles of the three methods.

Subject	1	2	3	4	5	6
Scale	1.1	1.0	0.9	0.95	1.1	1.0
FORTH	35.4	19.8	27.3	26.3	16.6	46.2
FORTH*	19.8	15.8	19.8	15.4	16.0	21.0
PSO	26.7	14.8	44.7	18.1	15.0	24.3
PSO*	18.6	12.1	21.2	14.4	13.7	22.4
ICP	27.3	20.7	34.4	25.1	17.00	32.8
ICP*	17.9	15.9	19.2	15.6	10.8	25.9
ICP-PSO	9.3	24.1	14.4	13.4	11.0	20.0
ICP-PSO*	**8.0**	**7.4**	**10.8**	**10.9**	**7.3**	**11.7**

Table 2. Model scales and average joint tracking errors (in mm) of 6 subjects. Methods with $*$ use initialization.

We are not aware of any work that reports hand tracking accuracy on challenging real data. Our accuracy is about 10 mm, which compares favorably to the accuracy (around 5 mm) reported on synthetic data [9, 13].

Timing For tracking, we use 128, 40, and 32 particles for PSO, ICP, and ICP-PSO, respectively. All run 20 generations and have similar speed. On an Intel i7 3.4GHz CPU, the run time is: 2 ms preprocessing, 2 ms finger detection, 1 ms hand initialization and 35 ms optimization(four threads). This translates to 25 frames per second (FPS). Note that we can trade a small accuracy loss for higher frame rate by using fewer particles (see Figure 6) and fewer generations (see Figure 5).

We use 64 particles and 40 generations in FORTH implementation, resulting in 14 FPS on an nVidia GeForce 580 GPU. More generations does not improve accuracy.

[4]The public implementation of [9] uses color based skin segmentation. It does not work well in our case.

E_{init}	15-25 mm				25-35 mm				35-45 mm			
Metric	E_{real}	E_{syn}	S_{real}	S_{syn}	E_{real}	E_{syn}	S_{real}	S_{syn}	E_{real}	E_{syn}	S_{real}	S_{syn}
PSO	9.80	7.79	61.4%	83.4%	12.27	10.16	44.2%	67.5%	16.25	14.25	31.3%	50.0%
ICP	10.52	7.71	52.5%	76.9%	14.50	11.75	27.5%	48.8%	19.80	17.71	13.8%	27.3%
ICP-PSO, $k=1$	5.98	3.37	87.7%	95.7%	8.93	5.79	72.2%	85.4%	13.95	10.81	54.2%	68.4%
ICP-PSO, $k=2$	5.65	3.07	89.7%	96.6%	8.39	5.00	74.4%	88.6%	12.75	9.23	58.0%	73.6%
ICP-PSO, $k=4$	**5.53**	**2.91**	**90.8%**	**97.9%**	**7.93**	**4.53**	**76.8%**	**90.2%**	**12.28**	**8.99**	**60.4%**	**74.2%**

Table 1. Averaged performance metrics of three methods on three difficulty levels, using real and synthetic depth maps. Note that k is the k-means parameter in ICP-PSO.

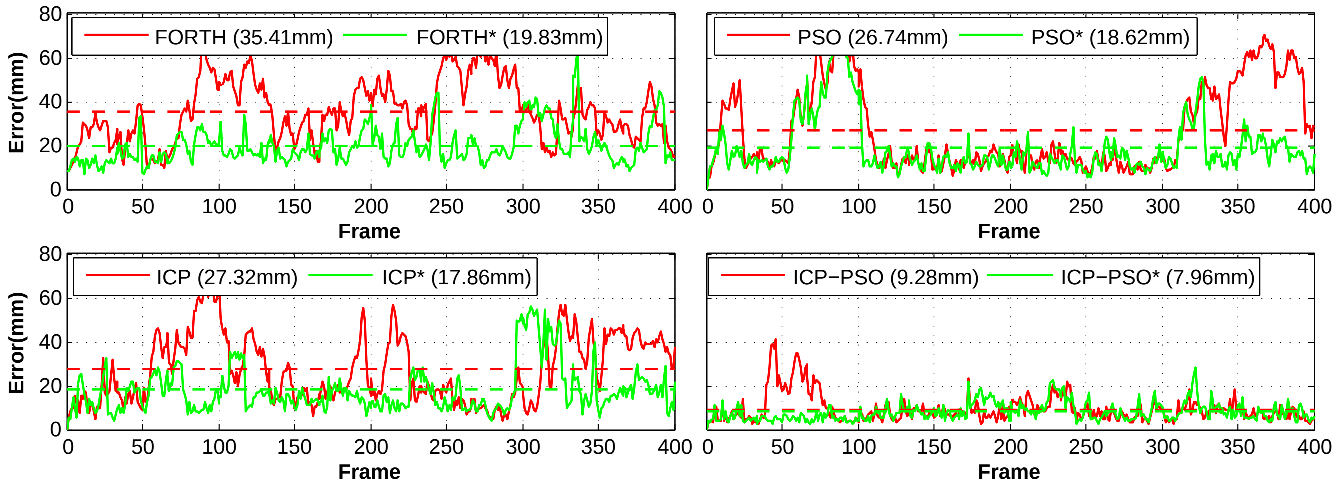

Figure 7. Average joint error in all frames of first subject. Each plot shows the results of one method using ($*$) and not using initialization. The horizontal dotted lines are mean errors of each method over all frames.

6. Conclusion

We present a new state-of-the-art hand tracking system, realized as the synergy of a simplified model, a fast cost function, and effective methods for optimization and initialization. Its realtime and robust performance on a desktop makes it useful for many applications such as user interface, sign language recognition, and virtual reality control.

References

[1] 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 *ICCV*, 2011. 1, 4

[2] A.Erol, G.Bebis, M.Nicolescu, R.D.Boyle, and X.Twombly. Vision-based hand pose estimation: A review. *CVIU*, 2007. 1

[3] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR*, 2003. 1

[4] B.Stenger, A.Thayananthan, P.H.S.Torr, and R.Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *PAMI*, 2006. 1

[5] C.Keskin, F.Kirac, Y.E.Kara, and L.Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*, 2012. 1

[6] D.Tang, T.Y, and T.K.Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *ICCV*, 2013. 1

[7] H.Hamer, K.Schindler, E.K.Meier, and L.V.Gool. Tracking a hand manipulating an object. In *ICCV*, 2009. 1

[8] I.Oikonomidis, N.Kyriazis, and A.A.Argyros. Markerless and efficient 26-dof hand pose recovery. In *ACCV*, 2010. 1, 2, 3

[9] I.Oikonomidis, N.Kyriazis, and A.A.Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, 2011. 1, 2, 3, 4, 6

[10] J.M.Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In *ECCV*, 1994. 1

[11] J. Kennedy and R. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks*, 1995. 3

[12] D. Kim, O. Hilliges, S. Izadi, A. Butler, J. Chen, I. Oikonomidis, and P. Olivier. Digits: Freehand 3d interactions anywhere using a wristworn gloveless sensor. In *UIST*, 2012. 1

[13] L.Ballan, A.Taneja, J.Gall, L.V.Gool, and M.Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV*, 2012. 1, 6

[14] M.L.Gorce, D.J.Fleet, and N.Paragios. Model-based 3d hand pose estimation from monocular video. *PAMI*, 2011. 1

[15] M.Ye, X.Wang, R.Yang, L.Ren, and M.Pollefeys. Accurate 3d pose estimation from a single depth image. In *ICCV*, 2011. 1, 4

[16] S. Pellegrini, K. Schindler, and D. Nardi. A generalisation of the icp algorithm for articulated bodies. In *BMVC*, 2008. 3, 4

[17] P.J.Besl and N.MacKay. A method for registration of 3d shapes. *PAMI*, 1992. 3

[18] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-time identification and localization of body parts from depth images. In *ICRA*, 2010. 1, 4

[19] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling*, 2001. 3

[20] R.Wang, K.Zhou, J.Snyder, X.Liu, H.Bao, Q.Peng, and B.Guo. Variational sphere set approximation for solid objects. *Visual Computer*, 2009. 2

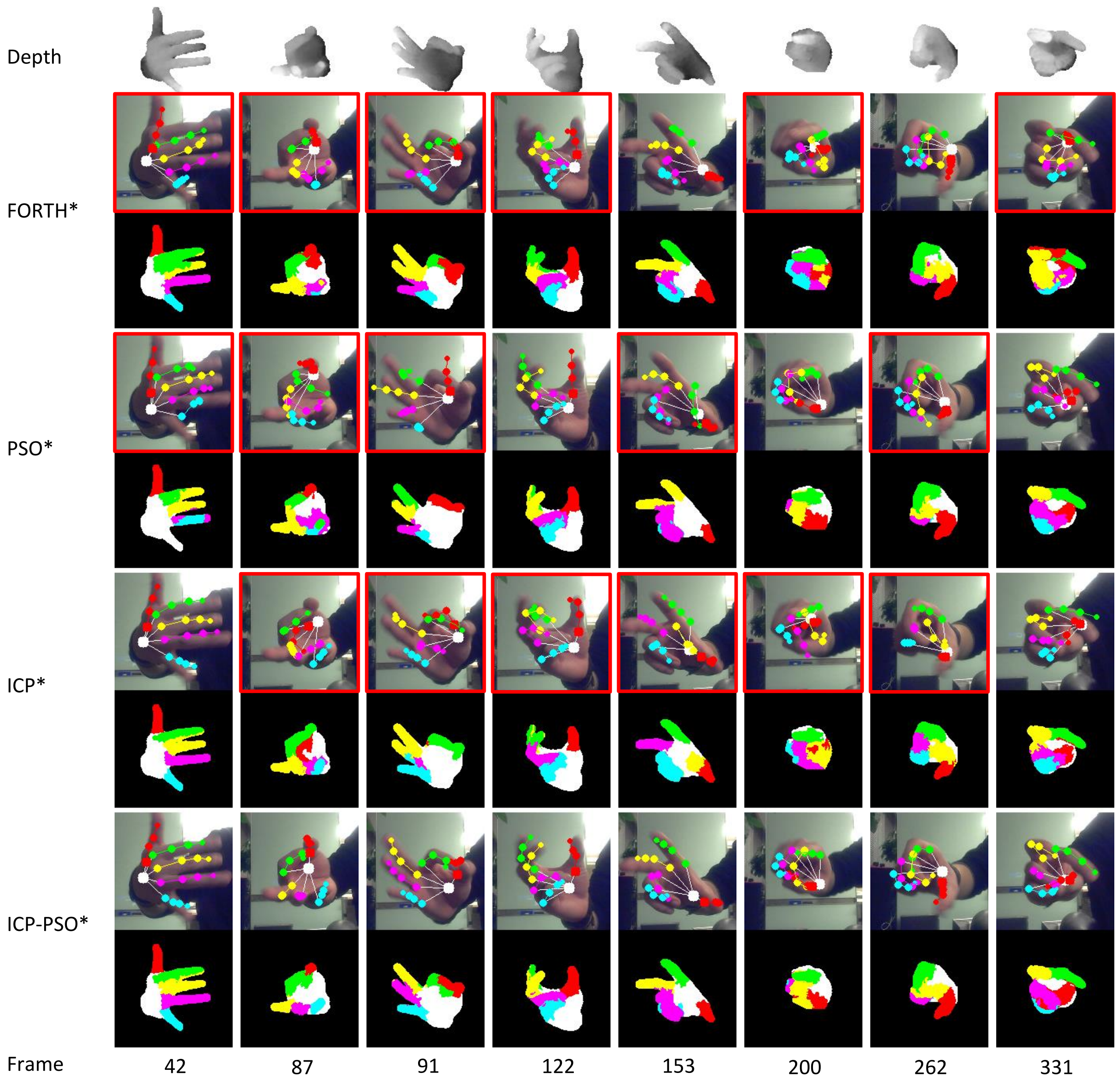

Figure 8. (better viewed in color) Example tracking results of the first subject. Those with red frames contain large errors. Color coded correspondence map of each result is also shown for better visualization.

[21] R.Y.Wang and J.Popovi. Real-time hand-tracking with a color glove. In *SIGGRAPH*, 2009. 1, 2

[22] R.Y.Wang, S.Paris, and J.Popovic. 6d hands: Markerless hand tracking for computer aided design. In *UIST*, 2011. 1

[23] E. S. Serra. Kinematic model of the hand using computer vision, 2011. 2

[24] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *PAMI*, 2013. 1, 4

[25] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *ICCV*, 2013. 1, 4

[26] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Fitzgibbon, and A. Hertzmann. User-specific hand modeling from monocular depth sequences. In *CVPR*, 2014. 2

[27] V.Ganapathi, C.Plagemann, D.Koller, and S.Thrun. Real-time human pose tracking from range data. In *ECCV*, 2012. 1, 2, 3, 4

[28] Y. Wu, J. Y.Lin, and T. S.Huang. Capturing natural hand articulation. In *ICCV*, 2001. 1

[29] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. In *ICCV*, 2013. 1

[30] X.Wei, P.Zhang, and J.Chai. Accurate realtime full-body motion capture using a single depth camera. In *Siggraph Asia*, 2012. 1, 2, 4

[31] Z.Mo and U.Neumann. Real-time hand pose recognition using low-resolution depth images. In *CVPR*, 2006. 1