

# Real-time and robust hand tracking with a single depth camera

Ziyang Ma · Enhua Wu

Published online: 5 December 2013

© Springer-Verlag Berlin Heidelberg 2013

**Abstract** In this paper, we introduce a novel, real-time and robust hand tracking system, capable of tracking the articulated hand motion in full degrees of freedom (DOF) using a single depth camera. Unlike most previous systems, our system is able to initialize and recover from tracking loss automatically. This is achieved through an efficient two-stage *k*-nearest neighbor database searching method proposed in the paper. It is effective for searching from a pre-rendered database of small hand depth images, designed to provide good initial guesses for model based tracking. We also propose a robust objective function, and improve the Particle Swarm Optimization algorithm with a resampling based strategy in model based tracking. It provides continuous solutions in full DOF hand motion space more efficiently than previous methods. Our system runs at 40 fps on a GeForce GTX 580 GPU and experimental results show that the system outperforms the state-of-the-art model based hand tracking systems in terms of both speed and accuracy. The work result is of significance to various applications in the field of human–computer-interaction and virtual reality.

**Keywords** Hand tracking · Virtual reality · Motion capture · User interface · 3D interaction

---

Z. Ma (✉) · E. Wu  
State Key Laboratory of Computer Science, Institute of Software,  
Chinese Academic of Sciences, Beijing, China  
e-mail: maziayang08@gmail.com

Z. Ma  
University of Chinese Academy of Sciences, Beijing, China

E. Wu  
University of Macau, Macau, China

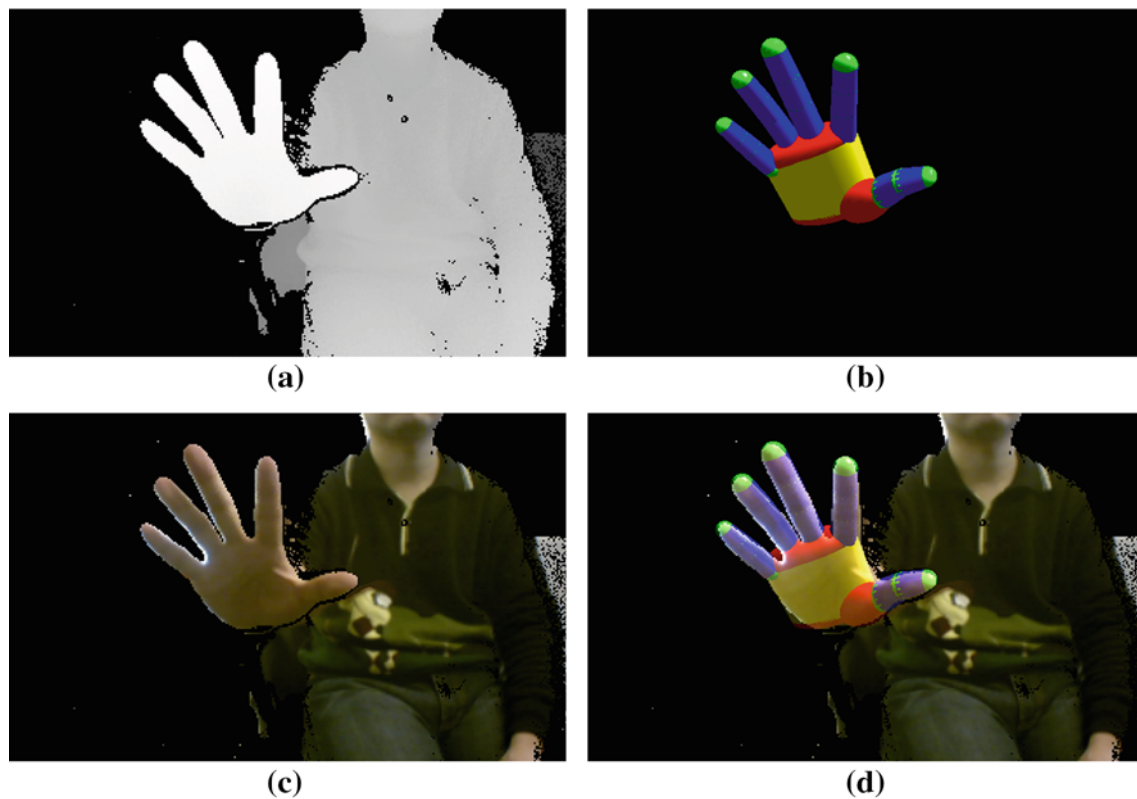
## 1 Introduction

The capturing and tracking of 3D human hand pose is a theoretically interesting problem. Many applications in both human–computer interaction and virtual reality could be built up provided this basic problem is solved effectively and efficiently. Although specific hardware such as visual markers [25] and data gloves [8] have been successfully used, they are too expensive to setup and the device itself might inhibit free hand movements.

As an alternative solution, computer vision-based approach is non-intrusive. However, it is very challenging due to the high dimensionality, fast motion, lack of visible textures and severe self-occlusions. In spite of these difficulties, a significant amount of research has been devoted to the problem of hand pose recovery using visual input. Erol et al. [9] provide a thorough review of this literature and divide vision-based hand tracking approaches into two main categories: appearance-based and model-based ones.

Appearance-based methods [4, 19, 21, 24], also called discriminative methods, formulate the hand pose estimation as a supervised classification problem. They learn a mapping from the input image space into the discrete hand pose configuration space. Although the computation is fast, the discriminative nature makes them only suited for recognizing a small set of known hand poses. Consequently, they are not suited for estimation of continuous parameters for a freely moving hand, which is required in interactive applications.

Model-based methods [12, 14–17, 20, 22], also called generative methods, on the other hand, search the best parameters in the hand configuration space such that the generated hand model matches visual observations. This is achieved by solving an optimization problem whose objective function is the discrepancy between observed cues and cues generated from



**Fig. 1** Illustration of our proposed model-based tracking system. **a** One frame of the input depth sequence captured by a time-of-flight (TOF) camera. **b** The output of our tracking system, which is an articu-

lated 3D hand model with full 26-DOF. **c** The RGB image corresponding to **a**, for better viewing of the input (RGB data is not used in our system). **d** The output in **b** superimposed on **c**, for better viewing of our result

the model hypothesis. The objective function must be evaluated at arbitrary point in the high dimensional parameter space. Due to the high dimensionality of the solution space and high nonlinearity of the objective function, to get optimal solutions, these methods often work on a sequence of input images, and the solution of current frame is usually used as the initial guess for the next frame. These methods are computationally more expensive than the appearance-based ones, and often suffer from the problem of tracking loss [23]. In spite of the above limitations, they do not require training and their outputs are continuous hand motion parameters, which are very useful in HCI and VR applications.

In order to support immersive user experience, four requirements should be met for a useful model-based hand tracking system:

- (1) The tracking should be initialized automatically.
- (2) No extra device except a camera is needed.
- (3) The full DOF tracking is done in real-time.
- (4) It can quickly recover from tracking loss.

State-of-the-art model-based hand tracking systems proposed by Oikonomidis et al. [14–17] meet requirement 2 well, and track full DOF at 15 fps on a GeForce GTX 580

GPU using a depth + RGB camera. However, they had not considered the tracking loss issue, where abrupt changes in hand gestures or rapid movements at one frame can cause severe tracking failure from that frame on. In addition, the initialization of their system is done through simply asking user to put his/her hand at specified gesture and position. Another state-of-the-art hand tracking system [20] uses three RGB cameras and meets requirement 1 and 4 well, but only limited to four gestures and 6-DOF, and it requires careful setup of three cameras, which is inconvenient and expensive.

This paper introduces a model-based hand tracking system which satisfies all four requirements simultaneously. The input is a depth video sequence captured by a time-of-flight (TOF) camera. Unlike the system proposed by Oikonomidis et al. [14], our system is robust against various lighting conditions and skin colors because no RGB information is required. The output of our system is the tracked 3D hand model with full 26-DOF for each frame. See Fig. 1 for an illustration. We propose a fast two-stage  $k$ -nearest neighbor searching method on a pre-rendered set of small hand depth image database to help automatic initialization and tracking loss recovery. We also propose a robust objective function, as well as an effective variation of the Particle Swarm Optimization (PSO)

algorithm for tracking based on a resampling strategy. Experimental results show that initialization and tracking loss are elegantly solved by database search, while the robust objective function and the improved PSO make the tracking more efficient than existing methods.

### 2 Hand model

The model of the 3D articulated hand for our tracking system is based on the quadric meshes used in [14, 17, 22]. Although more complex hand mesh models are available [2, 10], for visual tracking, the simplest quadric ones are more efficient and flexible approximations without sacrificing precision. We introduce it briefly in this section for completeness.

The model covers five fingers and a palm, and totally 37 quadratic primitives as shown in Fig. 2b. The palm is modeled using an ellipsoid cylinder with two ellipsoids for its caps. The fingers are modeled using three cones and four spheres, except for the thumb which consists of two cones, three spheres and one ellipsoid. These are generated from two basic geometric instances: a sphere and a cylinder, through homogeneous transformations using the following matrix:

$$\begin{pmatrix} r \cdot s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & r \cdot s_z & 0 \\ 0 & 1 - r & 0 & r \end{pmatrix} \tag{1}$$

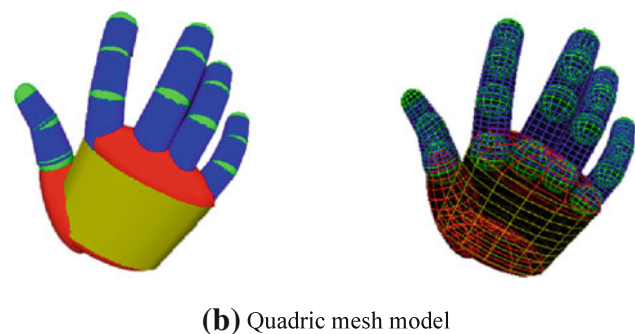
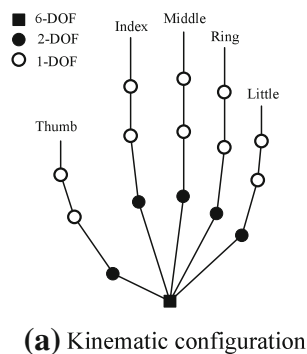


Fig. 2 Quadric mesh-based articulated hand model [14–16]

Here  $s_x, s_y$  and  $s_z$  are scaling factors along three axes,  $r$  is the ratio of the radius between top and bottom faces of the cone, it is set to 1 if we just do scaling like transforming spheres to ellipsoids, and  $<1$  if we are transforming cylinders to cones.

We use three parameters for the global 3D position of the hand at a fixed point on the palm, and four parameters for the global 3D orientation represented using quaternions. The kinematic configuration of the hand is modeled using 20 articulation parameters. Each finger has four parameters, two of them for the root of the finger, and two for the remaining joints. One parameter represents the abduction/adduction angle for the root of each finger, which spreads fingers apart, while all other three parameters represent the flexion/extension angles of each joint: totally 27 parameters to represent the full 26-DOF in hand configuration space, as illustrated in Fig. 2a.

The problem of model-based hand tracking is to search the best hand configurations (i.e., parameters) for each frame of the input image sequence so that the rendered image using the above 3D hand model best matches the observed input.

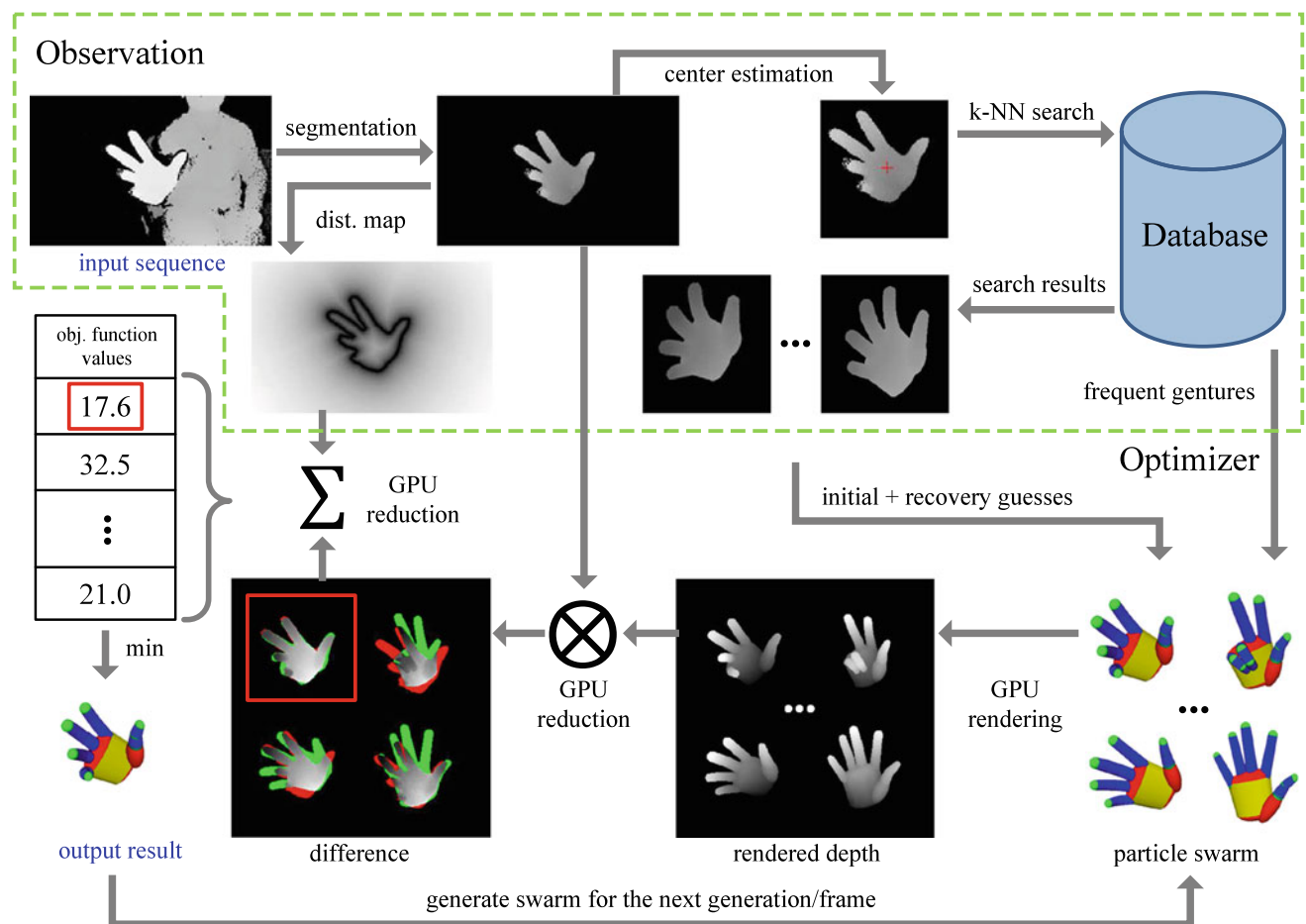
### 3 Hand tracking system

Figure 3 shows an overview of our hand tracking system. The input to our system is a depth image sequence acquired using a TOF camera. Note that unlike [14], we do not rely on the RGB data; this makes our system robust against various skin color and lighting conditions and also increases the performance via lower memory/IO throughput. The output of our system is the tracked 26-DOF result for each frame. Our system mainly consists of three parts: preprocessing,  $k$ -nearest neighbor database searching and particle swarm-based optimizing.

In the preprocessing stage, hand segmentation is performed on the captured raw depth image to remove unrelated objects in the scene and results in a binary mask indicating where the hand is. The distance map of the segmented hand region is computed based on this mask, and a hand center is estimated to cut out a square bounding box of depth image for searching some rough parameter estimations from a pre-rendered database.

The searching results, together with the tracking result coming from previous frame are fed as initial guesses into our tracking system, which is essentially an optimizer based on our improved version of the Particle Swarm Optimization (PSO) to find the optimal solution of hand motion parameters for current frame. The optimizer runs in parallel on a modern GPU and provides instant solution for each frame of the sequence.

The following sections describe each component in detail.



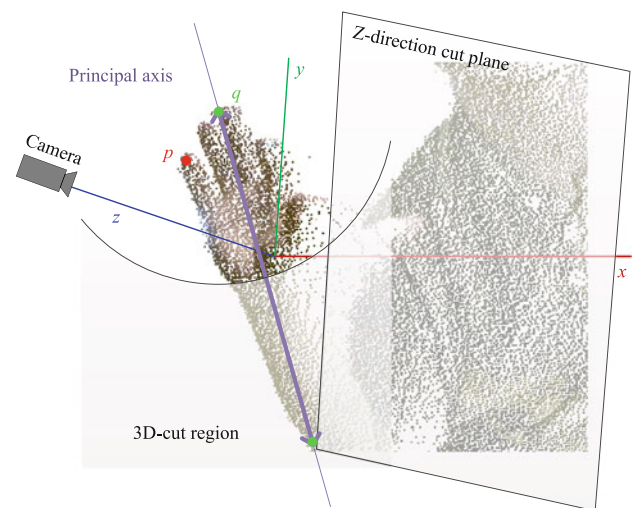
**Fig. 3** Overview of our hand tracking system, green dashed part is on CPU

### 3.1 Depth image preprocessing

The acquired depth images are in general noisy and contain other objects. In this step, we first remove the sensor noise by median filtering; then a simple hand segmentation is used to isolate the hand region.

Hand segmentation (as shown in the upper left corner of Fig. 3) relies on the scenario of our system: hand is the nearest object to the camera, and the camera is facing the hand. This is reasonable in most interactive applications and has been implicitly used in most state-of-the-art hand tracking systems [14–17]. Based on this scenario, our segmentation method works in two steps: a Z-direction cut, followed by a 3D-cut, as shown in Fig. 4.

We first identify the nearest non-trivial connected component to the camera and denote its nearest point by  $p$ . Then all points farther than a pre-defined threshold  $\beta$  from  $p$  are cut out as non-hand regions.  $\beta$  is set to the diameter of the hand (15 cm in our implementation). This step works in Z-direction. It usually cuts out a lot of points corresponding to the body and head regions, but might leave some part of the arm region.



**Fig. 4** Depth image segmentation

All remaining points are grouped in 3D space as a set, and the principal axis is computed from its largest connected component  $\Omega$  using Principal Component Analysis. The two extreme points along this axis are chosen from  $\Omega$  (the green

points in Fig. 4), and the one with minimum distance to the camera is selected and denoted by  $q$ , which is, in our scenario, always located at the boundary of the hand. Finally, all points farther than  $\beta$  away from  $q$  are also cut out. This step works in 3D space and can effectively cut out the arm which might be nearly parallel to the camera plane.

We found that this simple method works well in nearly all cases, and most importantly, unlike other state-of-the-art works on hand segmentation [3, 14–17], our method does not rely on skin color detection, which needs RGB data and might be affected by different lighting conditions and skin color variations.

The  $L^2$  distance transformation of the edge map detected via Sobel detector on the segmented hand mask is then computed for usage in the following steps. The hand center is estimated using this distance map as the point inside the mask with maximum distance to boundary, as shown with red cross in Fig. 3. A squared sub-image centered at this point which contains the whole hand region is cut out and resized into a smaller one ( $40 \times 40$  is used in our system) for searching rough estimates of hand motion parameters (excluding the global position) from a pre-rendered hand depth image database.

### 3.2 Depth image database and $k$ -NN search

For initialization and robustness against tracking loss, we propose a two-stage database searching method to quickly find rough estimates of current hand pose as initial guesses for optimization. The idea of database searching is inspired by the color marker-based data-driven hand pose estimation system proposed by Wang and Popovic [25], and other related work on neighborhood searching with geometric information [13]. However, we are different in both design and usage of the database, and our searching strategy is more suitable for hand shaped depth image retrieval.

We designed two specialized features for hand-shaped depth image, and use them to quickly filter out unmatched items during query processing. The first one locates all bulges in  $Z$ -direction, see Fig. 5a. For each local minimum point (i.e., bulge candidate), we extract the connected component with smoothly varying depth and divide the region between two concentric circles centered at this point into eight bins, then count the number of filled bins (80 % filled in our implementation). The candidate is considered as a bulge only if the number of filled bins is  $< 2$ . The diameters of the two concentric circles are empirically set to one and three times of the sectional diameter of a finger, respectively.

The second feature is a binary coding of the shape of the segmented hand mask. As shown in Fig. 5b, four equally-spaced concentric circles are found around the center of the mask. The regions between two concentric circles are divided into several bins, then a sequence of binary codes indicating

the filling state of each bin are created and concatenated. The length of the binary codes are 120 bits.

Our database contains a large amount of small *pre-rendered* hand *depth* images generated using the quadric model described in Sect. 2. Both the number of located bulges, the binary codes, and all parameters except the global position are stored along with each sample for retrieving. This database is used to provide only rough estimations of current hand configuration to the optimizer. It need not cover too much hand gestures, but only the most frequently encountered ones. Hence we select ten gestures from the famous American Sign Language (ASL) [1] to generate a total of 50,000 depth samples of resolution  $40 \times 40$ , with uniformly sampled hand orientation variations. Those selected samples from the ASL dataset are shown in Fig. 6.

In the query stage, the truncated SAD distance measure is usually used to lookup  $k$ -nearest neighbors ( $k$ -NN). The distance between the query image  $d_q$  and database image  $d_D$  is defined as

$$\text{dist}(d_q, d_D) = \iint_x \min(|d_q(x) - d_D(x)|, d_{\max}) dx, \quad (2)$$

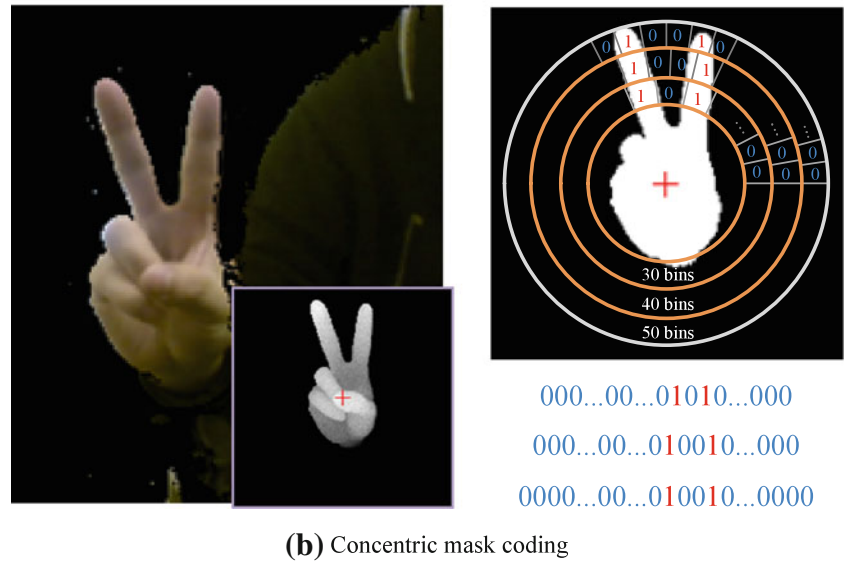
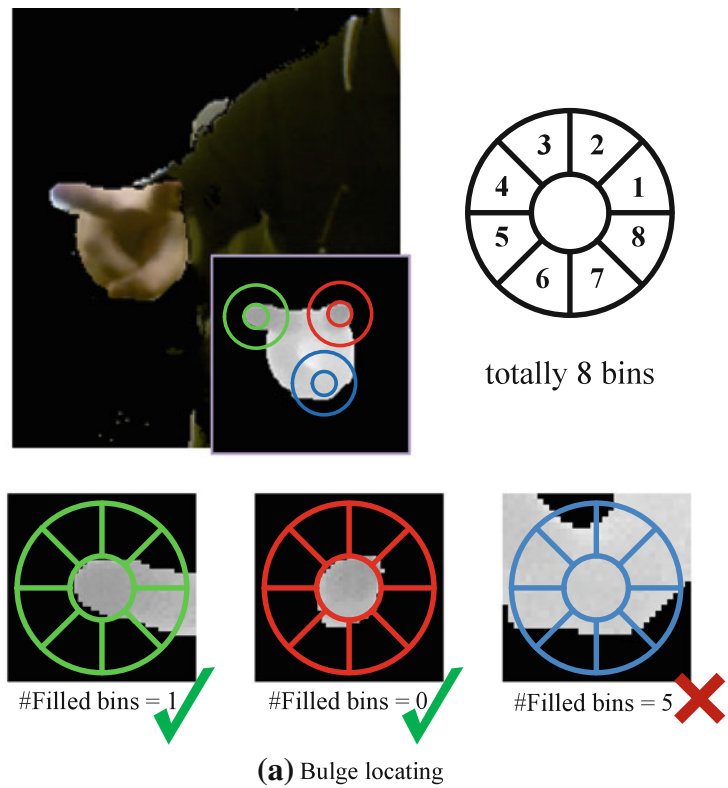
where  $x$  ranges over the entire depth image domain, and  $d_{\max}$  is a threshold, setting to 10 mm to make the distance measure more robust, avoiding noisy pixels dominating the error metric.

However, a brute-force searching through the whole database using Eq. 2 is computationally very expensive. To speed-up the process, we propose a two-stage method. We group all database samples into six sets according to the number of detected bulges, and only the set with the same number of bulges as the query image is considered. In the first stage, all data items in which the binary codes differs significantly (more than 20 % of the total number of bins) from the query are quickly eliminated. Those depth images survived are further considered in the second stage, where the whole distance measure to the query is computed using Eq. 2, and the best  $k$  (set to 10 in our implementation) images, as long as their model parameters are selected as the final searching results. Although other features, e.g., the Fourier descriptor of hand shape on circles [24], could also be used for  $k$ -NN search, we found our simple method very fast in the experiment. We leave more sophisticated approaches, e.g., using decision forest [7], as a future work.

### 3.3 Objective function for hand tracking

Since the goal of our hand tracker is to find the best hand parameters given the input depth frame, an objective function which measures the discrepancy between the rendered hand model depth  $d_p$  using hypothesized parameters  $p$ , and the observed depth image  $d_o$  should be defined. By nature, a

**Fig. 5** Features for pre-selection in database searching



robust objective function often leads to more efficient and easier optimization. In this section, we propose a robust objective function for our tracking system.

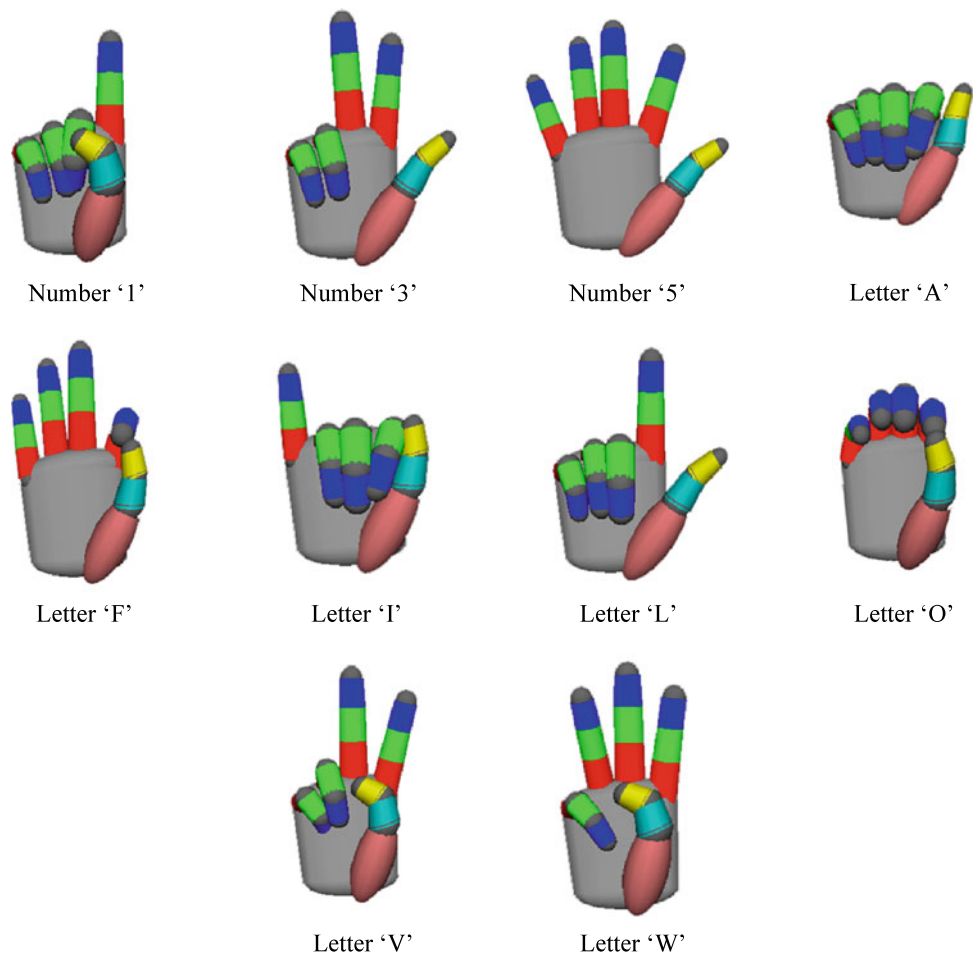
For clarity's sake, three mask images indicating three different regions for evaluation are defined as follows:

$$\begin{aligned}
 \text{msk}_{\text{rnc}}(x) &= (1 - \delta[d_p]) \wedge \delta[d_o] \\
 \text{msk}_{\text{rc}}(x) &= (1 - \delta[d_p]) \wedge (1 - \delta[d_o]) \\
 \text{msk}_{\text{cnr}}(x) &= (1 - \delta[d_o]) \wedge \delta[d_p],
 \end{aligned}
 \tag{3}$$

where  $\delta$  is the Kronecker delta function. Intuitively,  $\text{msk}_{\text{rnc}}$  is one at 'rendered but not captured' pixels,  $\text{msk}_{\text{rc}}$  is one at 'captured and rendered' pixels, and  $\text{msk}_{\text{cnr}}$  is one at 'captured but not rendered' pixels. See the difference images at bottom right of Fig. 3 for a simple illustration, where  $\text{msk}_{\text{rnc}}$  is colored in green, and  $\text{msk}_{\text{cnr}}$  is colored in red.

Given these three mask regions, as long as the distance map  $I_{\text{dist}}$  is computed from the preprocessing stage, we define the measure of discrepancy between  $d_p$  and  $d_o$  as

**Fig. 6** Selected hand configurations from the ASL dataset



$$\begin{aligned}
 D(d_p, d_o) = & \iint_{\text{msk}_{rc}(x)=1} \min(|d_p(x) - d_o(x)|, d_{\max}) dx \\
 & + \lambda \cdot \iint_{\text{msk}_{mc}(x)=1} I_{\text{dist}}(x) dx + \zeta \cdot \iint_{\text{msk}_{cnr}(x)=1} dx \\
 & + \eta \cdot \sum_{(\theta_1, \theta_2) \in Q(p)} -\min(\theta_1 - \theta_2, 0), \quad (4)
 \end{aligned}$$

where  $d_{\max}$  is the same as the threshold used in database search.  $\lambda$  is a relative weight for the distance map term.  $\zeta$  is the penalty for ‘captured but not rendered’ pixels, which is empirically set to 15 mm.  $\eta$  is another weight which is empirically set to 10. The first term is a sum of truncated absolute depth difference in  $\text{msk}_{rc}$  region. The second term is the weighted sum of  $L^2$  distance to the captured hand region boundary, which is important for measuring the shape dissimilarity. The third term is used to penalize the number of ‘captured but not rendered’ pixels. The last term is used to penalize implausible hand configurations, where  $Q(p)$  denotes the three pairs of adjacent fingers, and  $\theta_1, \theta_2$  denote the abduction–adduction angles of two pair of fingers in radians.

Note that our objective function does not rely on the RGB input, which is required in state-of-the-art depth-based hand tracking system [14]. Another important difference from [14] is that we include a more robust distance transform term to measure the dissimilarity in shape contour, which has been successfully used in sketch-based image search system [5].

### 3.4 Resampling-based PSO optimization

As long as the objective function is defined, we minimize it in the bounded 27-dimensional parameter space through an efficient variant of the Particle Swarm Optimization (PSO). The bounds for hand articulation parameters are determined according to anatomical studies in [2]. PSO is an evolutionary algorithm originally proposed by Kennedy et al. [11], and it has been adopted in hand tracking by recent works [14–16]. However, we found that its efficiency can be improved drastically (i.e., using much fewer particles and generations) with the help of our efficient resampling-based strategy introduced in this section.

We briefly describe the original PSO algorithm. The basic PSO works by having a population (i.e., swarm) of particles

(i.e., candidate solutions), in the parameter space. They are moved around iteratively (i.e., in generations) to search for the point with optimal objective function value. Each particle  $i$  has a current position  $x_t^i$  (i.e., current candidate solution) and current velocity  $v_t^i$  in the  $t$ -th generation. It also stores in  $p_t^i$  the best position achieved for this particle up to  $t$ -th generation. The entire swarm also holds the current global best position encountered across all particles in  $g_t$ . In the  $t + 1$ -th generation, each particle updates its velocity and position according to the following formulae:

$$v_{t+1}^i = K(v_t^i + c_1 r_1 (p_t^i - x_t^i) + c_2 r_2 (g_t - x_t^i)) \quad (5)$$

$$x_{t+1}^i = x_t^i + v_{t+1}^i, \quad (6)$$

where  $K$  is a constant constriction factor [6], and  $c_1$  and  $c_2$  are called cognitive component and social component, respectively.  $r_1$  and  $r_2$  are two random numbers sampled uniformly from  $[0, 1]$ . We adopt the parameter settings suggested in [6] here:  $c_1 = 2.8$ ,  $c_2 = 1.3$ ,  $K = 2/|2 - \psi - \sqrt{\psi^2 - 4\psi}|$ , with  $\psi = c_1 + c_2$ .

In [14–16], the particles are initialized as random perturbations of the solution from the last frame by exploring temporal continuity, and the solution for the first frame was simply assumed to be the ‘Number 5’ pose (see Fig. 6) putting at the center of screen. Although this simple strategy has been proven to be worked for careful manual initialization and slow hand motion, it does not consider the tracking loss issue due to such strong dependency on results of the last frames. Abrupt changes in hand gestures or rapid movements at one frame can cause severe tracking failure from that frame on. This problem can be substantially alleviated with our database searching technique. In addition, as observed in our experiments, many particles have wrong positions during evolution; hence a large amount of objective function evaluations are wasted. We next propose a resampling-based strategy for further increasing the efficiency of PSO, as detailed in Algorithm 1.

The particles are initially generated from two sources: half of them contain the population generated from perturbations of result in the previous frame, and the rest of them contain random particles in the vicinity of database searching results. The original PSO is invoked at first. After one-third of the maximum number of generations passed, we observed that the best particle often correctly estimates the global hand *position* and *orientation*, but the hand articulation parameters are sometimes not so precise, especially when the database searching results do not match the current input image well. Inspired by this observation, we regenerate one-third of the swarm using Gaussian perturbations of the global best particle and the rest ones using global best pose combined with additional selected gestures from the ASL dataset. Then after each generation, all particles are sorted according to the objective function values attained, and the worst half of

---

### Algorithm 1: Resampling-based PSO

---

**Input:** Captured and preprocessed depth image:  $d_o$   
 k-NN resulting parameters:  $p_1^D, \dots, p_k^D$   
 Tracking result from last frame:  $p^L$   
 Selected ASL gestures:  $p_1^A, \dots, p_{10}^A$   
 PSO evolution parameters:  $c_1, c_2, K$   
 Maximum number of particles:  $i_{max}$   
 Maximum number of generations:  $t_{max}$   
**Output:** Tracked parameter for this frame:  $p^O$

---

```

/*Initialization */
for i = 1 to i_max/2 do
  Randomly generate  $x_0^i$  centered at  $p^L$ ;
  Set  $v_0^i = 0$ ;
end
for i = i_max/2 + 1 to i_max do
  Randomly generate  $x_0^i$  with center randomly chosen from
   $p_1^D, \dots, p_k^D$ ;
  Set  $v_0^i = 0$ ;
end
Initialize objective function values:  $f^i = +\infty$ ;
Set  $p_0^i = x_0^i$ , and  $g_0 = \emptyset$ ;
/*Run the original PSO */
for generation t = 1 to t_max/3 do
  Evaluate objective functions:  $f^i = D(d_{x_{t-1}^i}, d_o)$ ;
  Update  $p_t^i =$  best position for each particle  $i$ ;
  Update  $g_t =$  best position of the entire swarm;
  Evolve one generation according to Eq. 5 and 6;
end
/*Regenerate the whole swarm */
for i = 1 to i_max/3 do
  Regenerate  $x_{t_{max}/3+1}^i$  centered at  $g_{t_{max}/3}$ ;
  Set  $v_0^i = 0$ ;
end
for i = i_max/3 + 1 to i_max do
  Regenerate  $x_{t_{max}/3+1}^i$  with position and orientation centered
  at  $g_{t_{max}/3}$ , but articulation parameters center randomly chosen
  from  $p_1^A, \dots, p_{10}^A$ ;
end
/*Run PSO with resampling */
for generation t_max/3 + 1 to t_max do
  Evaluate objective functions:  $f^i = D(d_{x_{t-1}^i}, d_o)$ ;
  Sort  $f^i$ ;
  Update  $p_t^i =$  best position for each particle  $i$ ;
  Update  $g_t =$  best position of the entire swarm;
  /*Resampling */
  Regenerate the worst half of particles;
  Evolve one generation according to Eq. 5 and 6;
end
Output  $p^O \triangleq g_{t_{max}}$ 

```

---

them are resampled again. Experimental results show that this variation of PSO can quickly eliminate the ‘bad particles’, and needs less particles and generations to find the solution.

### 3.5 GPU accelerated implementation

The computational bottleneck of our pipeline is the rendering of depth images of all particles and the evaluation



of the pixel-wise objective functions in each generation. Both functionalities can be inherently parallelized on modern GPUs.

In our system, we use instanced rendering [18] technique to simultaneously render multiple hand models for all particles in one generation. This requires transferring of only two basic meshes to GPU: a sphere and a cylinder at the beginning. A specialized pixel shader is used to retrieve the depth map and dispatches all rendered images into a big tile for evaluation. The tiled depth image is illustrated in the middle bottom of Fig. 3.

The pixel-wise objective function evaluation using Eq. 4 can be implemented on GPU using the highly efficient CUDA reduction algorithm. Although mip-map reduction can also be used and has been adopted in [14–16], we found the former in general has better performance and easy to implement.

## 4 Experimental results and comparisons

In this section we demonstrate the effectiveness of our improved algorithms both in database search and PSO quantitatively and qualitatively. We also show comparisons with state-of-the-art hand tracking system [14] on real-world sequences. Our system runs on a computer with a quad-core Intel Xeon E5440 CPU, 16GBs RAM, and a 1581 G Flops Nvidia GTX 580 GPU with 1.5 GBs memory; the depth sequence is captured using a time-of-flight (TOF) camera with resolution  $320 \times 200$  at 30 fps. We found running our tracker at half of the resolution (i.e.,  $160 \times 100$ ) does not induce observable lost of precision, but the average frame rate can be boosted from 30 to 40 Hz.

### 4.1 Effectiveness of database search

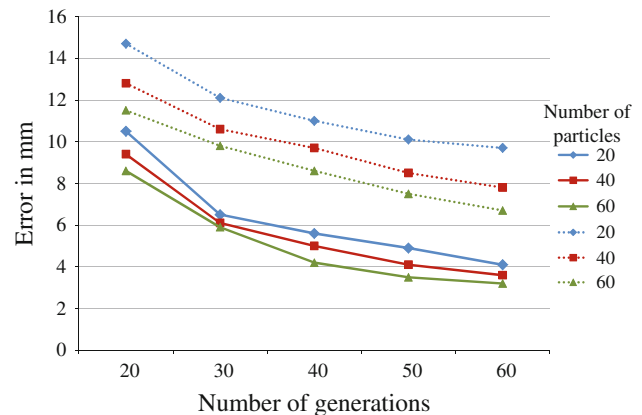
For evaluating the effectiveness of our database search in initialization and tracking loss recovery, we captured and saved three sequences with increasing rapid hand movement. We ran the tracker both with and without enabling database search, and counted the number of failed frames where the total number of  $msk_{mc}$  and  $msk_{nr}$  pixels exceeds the number of  $msk_{rc}$  pixels. Table 1 shows our statistics. As can be seen, our tracking system is much more robust with the help of database search. The running time of our two-stage  $k$ -nearest neighbor search is 5 ms on average, about 12 times faster than a brute-force search.

### 4.2 Effectiveness of resampling-based PSO

Figure 7 shows the number of particles/generations versus the average precision of tracking for those three sequences measured as average absolute difference in the  $msk_{rc}$  region.

**Table 1** Effectiveness of database search on tracking loss recovery

Sequence No.	#Total frames	#Failed frames with database search	#Failed frames w/o database search
1	380	11	93
2	333	17	159
3	282	25	239



**Fig. 7** Effectiveness of resampling-based PSO (solid lines) versus traditional PSO (dotted lines)

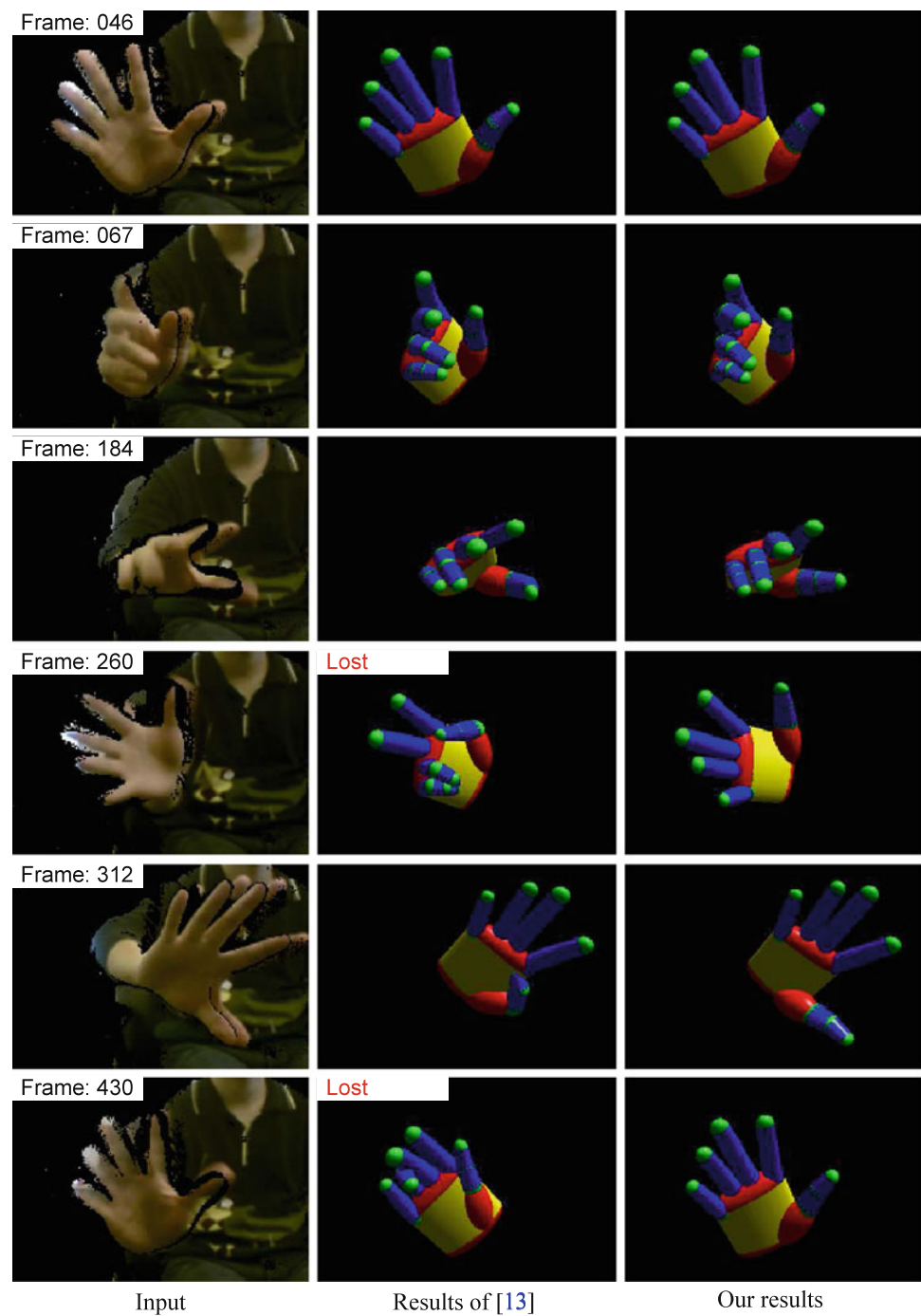
As can be observed, the resampling-based PSO needs only less than half of the particles and generations to achieve the same precision as traditional PSO. Another interesting observation from this figure is that the quality of our resampling-based PSO does not strongly depend on the number of particles as traditional PSO does, but the number of generations is more important. This is reasonable because the resampling strategy makes the usage of particles more efficient.

The computation time of our improved PSO (40 particles and 30 generations) is 15 ms on average, which is more than two times faster than the one without resampling strategy.

### 4.3 Visual comparison with state-of-the-art

Figures 8 and 9 show the comparison of our hand tracking system with state-of-the-art system [14] (we use the SDK provided by the authors from their web site) on selected frames from captured sequence 2 and 3 (we show colored input here for better viewing). All results show that our hand tracking system substantially outperforms state-of-the-art, especially on the more challenge sequence 3 where rapid hand movements significantly challenge the methods without tracking loss recovery mechanisms. See more video results in the supplemental material.

**Fig. 8** Visual comparison with state-of-the-art [14] on sequence 2



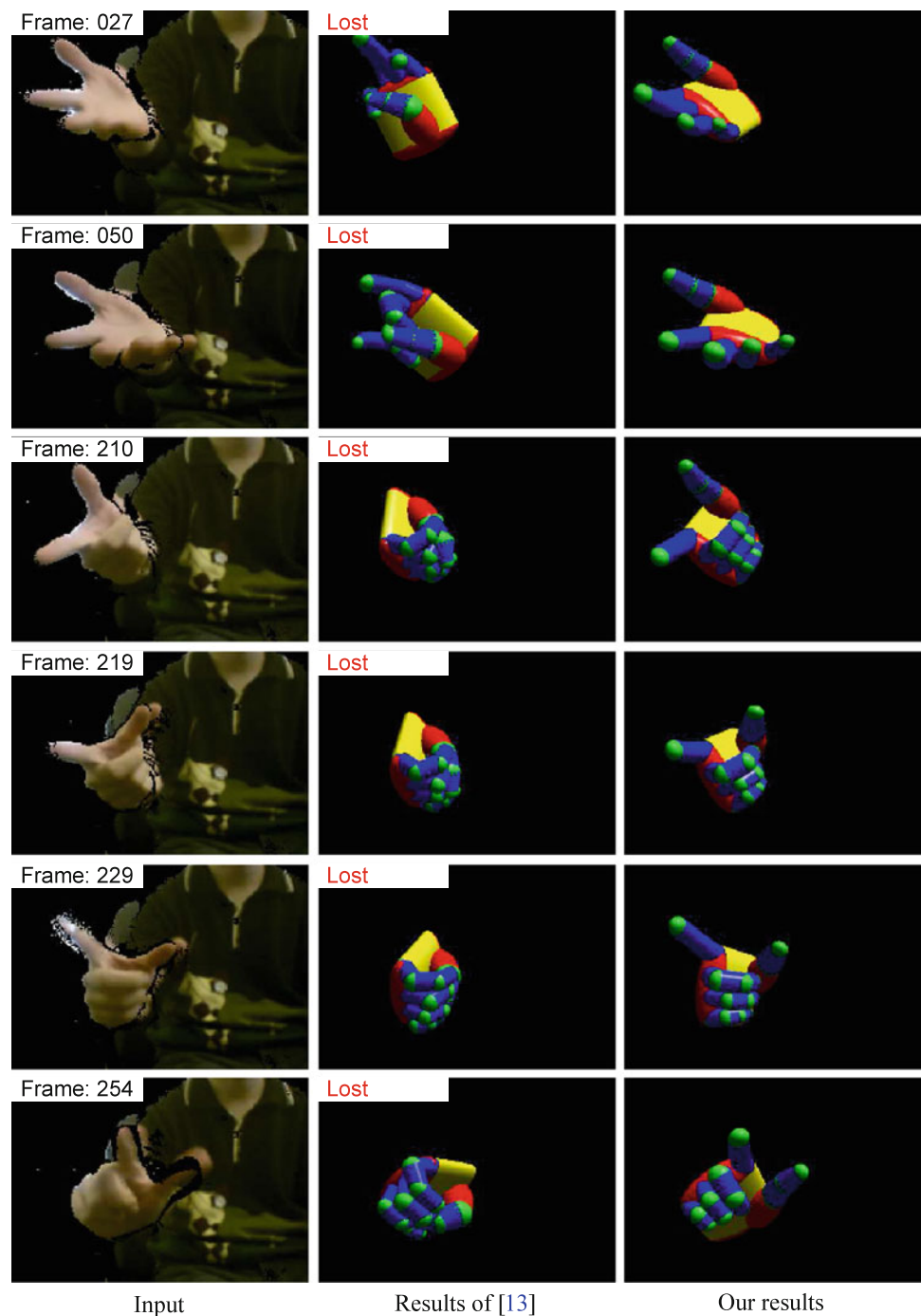
## 5 Conclusions and future work

We have presented a real-time and robust full DOF hand tracking system using a single depth camera. Our system incorporates efficient  $k$ -NN database search as long as resampling-based PSO for effective tracking. We show that

our system outperforms state-of-the-art both quantitatively and qualitatively.

Since an individual in general has particular size and shape in his/her hands, model calibration is important for our system to produce high-quality tracking results. Currently, in the calibration stage, we use a fixed shape and simply treat

**Fig. 9** Visual comparison with state-of-the-art [14] on the more challenging sequence 3



size ratio as an additional variable in the optimizer to match the size of each person. We run the optimization to find the best size ratio for a specified hand pose, but we leave a more sophisticated approach for hand shape calibration as a future work.

We believe that our system will benefit a very large amount of HCI, VR and graphics applications, and we will investigate some of them in the future.

**Acknowledgments** The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This research is supported by National 973 Program of Basic Research on Science and Technology (2009CB320800), NSFC (61272326) and the research grant of University of Macau.

## References

1. [http://en.wikipedia.org/wiki/American\\_Sign\\_Language](http://en.wikipedia.org/wiki/American_Sign_Language)

2. Albrecht, I., Haber, J., Seidel, H.P.: Construction and animation of anatomically based human hand models. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 98–109. Eurographics Association (2003)
3. Argyros, A.A., Lourakis, M.I.: Real-time tracking of multiple skin-colored objects with a possibly moving camera. In: European Conference on Computer Vision (ECCV) (2004)
4. Athitsos, V., Sclaroff, S.: Estimating 3D hand pose from a cluttered image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2003)
5. Cao, Y., Wang, C., Zhang, L., Zhang, L.: Edgel index for large-scale sketch-based image search. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011)
6. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **6**(1), 58–73 (2002)
7. Criminisi, A.: Decision forests: a unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Found. Trends Comput. Graph. Vis.* **7**(2–3), 81–227 (2011)
8. Dipietro, L., Sabatini, A.M., Dario, P.: A survey of glove-based systems and their applications. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **38**(4), 461–482 (2008)
9. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X.: Vision-based hand pose estimation: a review. *Comput. Vis. Image Underst.* **108**(1), 52–73 (2007)
10. Huang, H., Zhao, L., Yin, K., Qi, Y., Yu, Y., Tong, X.: Controllable hand deformation from sparse examples with rich details. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 73–82. ACM (2011)
11. Kennedy, J.F., Kennedy, J., Eberhart, R.C.: *Swarm intelligence*. Morgan Kaufmann Publisher, San Francisco (2001)
12. de La Gorce, M., Paragios, N., Fleet, D.J.: Model-based hand tracking with texture, shading and self-occlusions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
13. Liu, Y.J., Zheng, Y.F., Lv, L., Xuan, Y.M., Fu, X.L.: 3D model retrieval based on color + geometry signatures. *Vis. Comput.* **28**(1), 75–86 (2012)
14. Oikonomidis, I., Kyriazis, N., Argyros, A.: Efficient model-based 3D tracking of hand articulations using kinect. In: British Machine Vision Conference (BMVC) (2011)
15. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Markerless and efficient 26-DOF hand pose recovery. In: Asian Conference on Computer Vision (ACCV) (2010)
16. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In: IEEE International Conference on Computer Vision (ICCV) (2011)
17. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Tracking the articulated motion of two strongly interacting hands. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
18. Pharr, M., Fernando, R.: *GPU gems 2: programming techniques for high-performance graphics and general-purpose computation*. Addison-Wesley Professional, Boston (2005)
19. Rosales, R., Athitsos, V., Sigal, L., Sclaroff, S.: 3D hand pose reconstruction using specialized mappings. In: IEEE International Conference on Computer Vision (ICCV) (2001)
20. Schlattmann, M., Kahlesz, F., Sarlette, R., Klein, R.: Markerless 4 gestures 6 DOF real-time visual tracking of the human hand with automatic initialization. In: *Computer Graphics Forum*, vol. 26, pp. 467–476. Wiley Online, Library (2007).
21. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, In (2011)
22. Stenger, B., Mendonça, P.R., Cipolla, R.: Model-based 3D tracking of an articulated hand. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2001)
23. Stenger, B.D.R.: *Model-based hand tracking using a hierarchical bayesian filter*. Ph.D. thesis (2004)
24. Tomasi, C., Petrov, S., Sastry, A.: 3D tracking= classification+ interpolation. In: *IEEE International Conference on Computer Vision (ICCV)* (2003)
25. Wang, R.Y., Popović, J.: Real-time hand-tracking with a color glove. In: *ACM SIGGRAPH*, vol. 28 (2009)



**Ziyang Ma** received his B.Sc. degree in pure and applied mathematics from Nankai University, Tianjin, China, in 2008. He is currently a Ph.D. candidate at the State Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. His research interests include computer graphics, computer vision, machine learning, numerical computation and optimization.



**Enhua Wu** received his B.Sc. degree in 1970 from Tsinghua University, Beijing, and his Ph.D. degree in 1984 from University of Manchester, UK. Since 1985, he has been working at the Institute of Software, Chinese Academy of Sciences, and from 1997 he has been also teaching in University of Macau. He is a member of IEEE & ACM. His research interests include realistic image synthesis, virtual reality, physically based modeling, and scientific visualization.