Multimedia Tools and Applications (2019) 78:6745–6762
https://doi.org/10.1007/s11042-018-6452-0

Tracking a hand in interaction with an object based on single depth images

Dongnian Li[1] · Chengjun Chen[1]

Received: 30 November 2017 / Revised: 15 May 2018 / Accepted: 20 July 2018 /
Published online: 30 July 2018

Abstract
Tracking a hand in interaction with an object based on vision is a challenging research topic. The occlusions that occur during the hand-object interaction make it difficult to develop an effective tracking system. To overcome the impacts of occlusions, we build 3D models for both the hand and the manipulated object and propose a model-based tracking method to track the hand and the object simultaneously from single depth images during the hand-object interaction. The most likely hand-object state is searched by an improved particle filtering (PF) tracking algorithm in the high-dimensional hand-object space, which uses Gaussian particle swarm optimization (Gaussian PSO) algorithm to improve the process of particle sampling, moving the particles to the regions with higher likelihood. According to the proposed tracking algorithm, two kinds of hand-object tracking prototype systems are developed by using the graphics rendering engine OSG and off-screen rendering techniques. Experimental results demonstrate that the proposed method can track hand-object motion robustly with few particles.

Keywords Hand tracking · Object tracking · Depth image · Particle filtering · Gaussian PSO · OSG

1 Introduction

3D hand tracking based on computer vision is a challenging research topic and can be applied in many fields, such as robot learning from demonstration, motion capture, human-machine interaction, gesture recognition. Since a hand has multiple degrees of freedom (DOFs), hand tracking is essentially a problem of motion tracking in a high-dimensional state space. The development of an effective hand tracking system is hindered by many complicated factors [7], such as the high-dimensionality of the state space, the self-occlusion during hand movements, and the automatic recovery from a tracking failure. Up to now, there are mainly two kinds of

✉ Chengjun Chen
chenchengjun79@163.com

1 School of Mechanical & Automotive Engineering, Qingdao University of Technology, Qingdao 266520, People's Republic of China

hand tracking methods by computer vision: appearance-based hand tracking and model-based hand tracking.

Appearance-based hand tracking methods [4, 5, 9, 10, 16, 17, 20] establish a mapping through machine learning, which maps the image feature space onto the hand state space. These methods estimate the hand states directly according to the image features. They don't need initialization and usually have high tracking speeds. However, the accuracy of these methods depends on the number of training samples. Model-based hand tracking methods [1–3, 14, 15, 19, 23, 25] generate hand pose hypotheses based on a pre-built 3D hand model, and evaluate the similarity between the features extracted from the model and the features extracted from visual observation. By using some kind of optimization method, the hand pose state with the highest similarity is found and chosen as the solution. These methods can use a lot of prior information (e.g. hand shapes and joint constraints), but the tracking process has to be initialized and searching for the best solution in high-dimensional space will be difficult. Above two kinds of methods are mainly applied to tracking a hand in isolation. However, they can't track hand motion effectively while the hand are in operation with an object, because of the occlusions between the human hand and the operated object.

Nevertheless, an overwhelming majority of hand actions are interactive in the real world. Among them, hand-object interaction is the most common kind of interaction. Therefore, tracking a hand in interaction with an object is a very important research topic. During the interaction, the manipulated object will occlude the hand frequently and the self-occlusion of the hand will be intensified. On the other hand, the useful contextual information provided by the object will facilitate the recognition and estimation of hand movements. Some researchers [8, 11, 13, 18, 21, 22, 24] have conducted research on this topic. Kjellström et al. [11] proposed a method to recognize hand actions and the manipulated objects simultaneously by modeling the action-object correlations using conditional random fields, but this solution can't present the detailed information of hand poses. To make the tracking system robust to occlusions, Hamer et al. [8] used an individual local tracker for each segment of the articulated hand and connected two adjacent hand segments by using a pair-wise Markov random field. The optimal hand state is found with belief propagation (BP). However, this method doesn't model the manipulated object. Romero et al. [21, 22] proposed a non-parametric real-time method for tracking a hand in interaction with objects. This method extracts hand features using histogram of oriented gradients (HoG) and searches the most likely hand pose through nearest neighbor searching (NNS) in a large template database. However, as is appearance-based, it can't track hand motion precisely in high-dimensional space. For tracking hand motion more precisely, some researchers have developed model-based tracking systems by using multi-view cameras [18, 24] or depth cameras [13]. Oikonomidis et al. [18] proposed a model-based tracking method for tracking a hand and the manipulated object simultaneously by using multi-view cameras. The models for the hand and the object are both pre-built, and the tracking problem is considered as a sequential optimization problem which searches for the most likely hand-object state. By using a depth camera, Kyriazis et al. [13] proposed a method that only searches hand states. The object states are inferred according to the hand states and the dynamics model of hand-object interaction. However, the dynamics of hand-object interaction in the real world involves many complicated factors and is difficult to be precisely modeled.

In this paper, a model-based method is proposed to track the hand and the manipulated object simultaneously during the hand-object interaction by using single depth images as observation. The most likely hand-object state is searched by using an improved particle filtering algorithm. Particle filtering (PF) is a robust motion tracking framework in cases of clutter. By propagating multiple hypotheses along with time, it has the capability to describe multi-peak distributions [6]. Many researchers have used PF framework for human body [3, 25] or hand [1, 2, 14] tracking. However, standard PF needs adequate samples to describe the real posterior probability distribution of the system state, and it is difficult to sample the true optimum in the high-dimensional space. So it is easy to cause tracking failure. To address this problem, many researchers have combined some kind of optimization method (e.g. gradient descent [1], genetic algorithm [2] and differential evolution [14]) with PF for hand tracking. Based on the PF framework, these methods use the samples predicted by dynamic models as the initial states of the optimization and describe the importance distribution according to the optimized samples. This paper combines Gaussian particle swarm optimization (Gaussian PSO) [12] with PF for hand-object tracking during the hand-object interaction. Gaussian PSO algorithm is used to improve the PF sampling process, driving the particles to the promising areas in state space with high likelihood values.

2 Overall framework of the tracking system

In this paper, a model-based method is used for tracking hand-object interaction (Fig. 1). 3D models are built for both the hand and the manipulated object, which will be tracked in the state space. During the tracking process, hand and object pose hypotheses are generated by using the 3D hand model and the 3D object model. Then, the matching errors between the model features and the observed features are calculated. The tracking problem is tackled by searching for the best hand-object state with the smallest matching error in the high-dimensional hand-object space. Finally, the best state found by the tracking problem is chosen as the solution for the current frame of system input.

3 Hand-object model

In this paper, 3D models of the hand and the manipulated object are both pre-built. The hand pose and the object pose are described by one hand-object state vector $\mathbf{x}_{h-o} = (\mathbf{x}_h, \mathbf{x}_o)$. The hand state vector $\mathbf{x}_h$ includes 6 DOF for the global motion and 20 DOF for local finger motion. The object state vector $\mathbf{x}_o$ only includes 6 DOF for the translation and rotation of the object. The hand

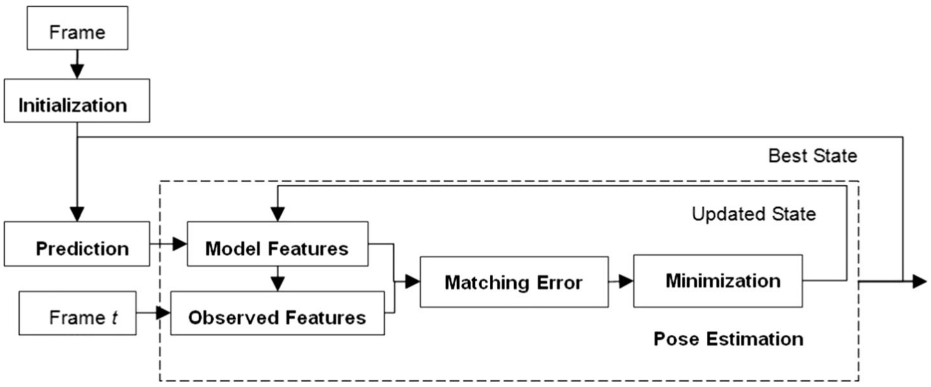


Fig. 1 Overall framework of the model-based tracking system

skeleton model is shown in Fig. 2. All CMC joints are fixed and the palm is modeled as a rigid body with 6 DOF (3 for translation and 3 for rotation). Each finger is modeled with 4 DOF. Each MCP joint except the thumb's has 2 DOF (one for flexion-extension and one for abduction-adduction). The MCP joint of the thumb only has one DOF for flexion-extension. All PIP and DIP joints and the IP joint of the thumb only have a flexion-extension capability. The TM joint of the thumb has 2 DOF. The motion for each finger joint DOF is limited within a certain range according to hand anatomy. These anatomical constraints can ensure the effectiveness of the hand pose solution and reduce the search range of hand states. In this paper, the joint angle values of all fingers except the thumb are limited by the following constraints:

$$\begin{gathered} 0^\circ \le \theta_{MCP\_F} \le 90^\circ \\ 0^\circ \le \theta_{PIP\_F} \le 110^\circ \\ 0^\circ \le \theta_{DIP\_F} \le 90^\circ \\ -15^\circ \le \theta_{MCP\_AA} \le 15^\circ \end{gathered}$$

and the joint angle values of the thumb are limited by the following constraints:

$$\begin{gathered} 0^\circ \le \theta_{TM\_F} \le 30^\circ \\ -15^\circ \le \theta_{TM\_AA} \le 3^\circ \\ 0^\circ \le \theta_{MCP\_F} \le 90^\circ \\ 0^\circ \le \theta_{IP\_F} \le 90^\circ \end{gathered}$$

where θ_{MCP_F}, θ_{PIP_F} and θ_{DIP_F} denote the flexion-extension DOF values of MCP, PIP and DIP joints respectively for all fingers, θ_{MCP_AA} denotes the abduction-adduction DOF value of the MCP joint for each finger except the thumb, θ_{TM_F} and θ_{TM_AA} denote the flexion-extension DOF value and the abduction-adduction DOF value respectively for the TM joint of the thumb, and θ_{IP_F} denotes the flexion-extension DOF value for the IP joint of the thumb.

To balance model accuracy against computational complexity, the hand shape model is constructed based on basic geometric primitives. The palm is built with an elliptic cylinder and

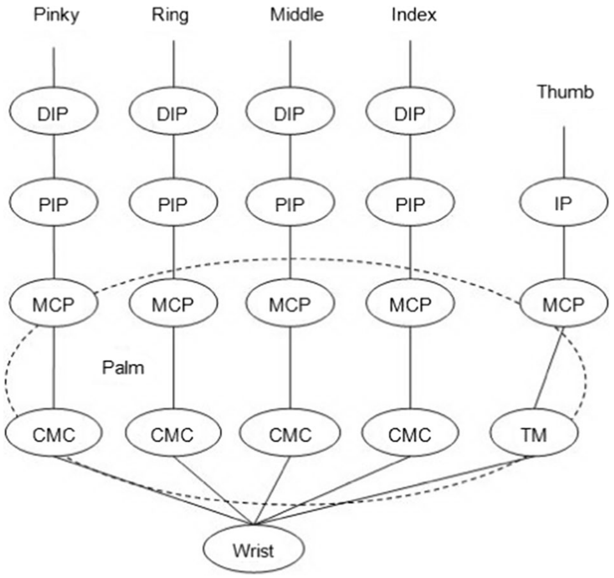


Fig. 2 Hand skeleton model

two spheroids. The finger segments are built with cylinders and truncated cones. The finger joints are built with spheres, while the finger tips and the thumb root are built with hemispheres. The modeling process includes two steps. Firstly, the 3D model of the hand is constructed with geometric primitives in the parametric modeling software PTC Pro/Engineer. Secondly, the hand model is imported into the visualization modeling software Multigen-Paradigm Creator through the intermediate OBJ file format. In Creator, the meshes of the hand model are organized into a tree-like hierarchical structure. Then, the DOF nodes with local coordinate systems are created and added into the hand model. The eventual 3D hand model contains a total of 7104 vertexes, 2368 triangle meshes and 13 DOF nodes. Meanwhile, the manipulated objects are modeled in Creator directly. In this paper, two types of objects (spherical objects and cylindrical objects) in interaction with the hand are considered. However, the proposed method is also suit for tracking a hand in interaction with other types of objects. The 3D model for hand-sphere interaction and the 3D model for hand-cylinder interaction are shown in Fig. 3.

4 Observation model

In this paper, the hand state and the object state are integrated into a hand-object state vector $\mathbf{x}_{h-o} = (\mathbf{x}_h, \mathbf{x}_o)$. The matching error for the current observation $\mathbf{z}$ is determined by both the hand state $\mathbf{x}_h$ and the object state $\mathbf{x}_o$. Meanwhile, the matching error function is established by combining depth features with silhouette features. Using depth images obtained from a Kinect depth camera as input, the foregrounds corresponding to the hand and the manipulated object are extracted by depth segmentation to generate the observed depth map $z_d(\mathbf{z})$. And, for each hand-object pose hypothesis $\mathbf{x}_{h-o}$, the rendered depth map $r_d(\mathbf{x}_{h-o})$ is generated by the rendering process of the 3D hand-object model. Then, the observed silhouette map $z_s(\mathbf{z})$ is generated from the observed depth map ($z_d(\mathbf{z})$) and the rendered silhouette map $r_s(\mathbf{x}_{h-o})$ is generated from the rendered depth map $r_d(\mathbf{x}_{h-o})$. Both $z_s(\mathbf{z})$ and $r_s(\mathbf{x}_{h-o})$ are binary images with value 1 for the hand-object foreground and value 0 for the background.

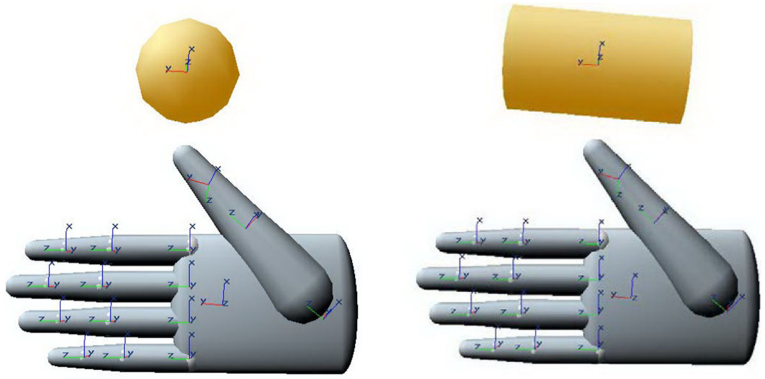

(a) Hand-sphere model (b) Hand-cylinder model

Fig. 3 Hand-object model. The local coordinates of all DOF nodes are marked accordingly

The incompatibility between the current observation $\mathbf{z}$ and the hand-object pose vector $\mathbf{x}_{h\text{-}o}$ is measured by the matching error function. Smaller matching errors reflect higher compatibility. The matching error function is defined as follows:

$$E(\mathbf{z},\mathbf{x}_{h-o}) = \lambda_d E_d(\mathbf{z},\mathbf{x}_{h-o}) + \lambda_s E_s(\mathbf{z},\mathbf{x}_{h-o}) + \lambda_m E_m(\mathbf{x}_{h-o}) \tag{1}$$

$E(\mathbf{z}, \mathbf{x}_{h\text{-}o})$ is composed of a depth term E_d, a silhouette term E_s and a penalty term E_p. λ_d, λ_s and λ_p are the weights of these three terms, respectively.

For hand-object pose hypothesis $\mathbf{x}_{h\text{-}o}$, the depth difference between the observed depth map $z_d(\mathbf{z})$ and the rendered depth map $r_d(\mathbf{x}_{h\text{-}o})$ is measured by the depth term

$$E_d(\mathbf{z},\mathbf{x}_{h-o}) = \frac{\sum \min(|z_d(\mathbf{z}) - r_d(\mathbf{x}_{h-o})|, T_d)}{\sum (z_s(\mathbf{z}) \vee r_s(\mathbf{x}_{h-o}))} \tag{2}$$

where depth differences (unit: mm) are calculated in a pixel-wise manner and accumulated over the entire depth map. The result of the accumulation is divided by the total hand-object area for normalizing. Some large depth differences will cause great changes of the matching error, thus affecting the performance of the searching algorithm. Therefore, a constant T_d is introduced to clamp the depth differences in the range $[0, T_d]$.

The silhouette term E_s measures the silhouette incompatibility by calculating the non-overlapping area between the observed silhouette map $z_s(\mathbf{z})$ and rendered silhouette map $r_s(\mathbf{x}_{h\text{-}o})$. E_s is defined as:

$$E_s(\mathbf{z},\mathbf{x}_{h-o}) = \frac{\sum z_s(\mathbf{z})(1 - r_s(\mathbf{x}_{h-o}))}{\sum z_s(\mathbf{z})} + \frac{\sum r_s(\mathbf{x}_{h-o})(1 - z_s(\mathbf{z}))}{\sum r_s(\mathbf{x}_{h-o})} \tag{3}$$

where the first term calculates the area belonging to $z_s(\mathbf{z})$ but excluding from $r_s(\mathbf{x}_{h\text{-}o})$. The second term calculates the area belonging to $r_s(\mathbf{x}_{h\text{-}o})$ but excluding from $z_s(\mathbf{z})$. These two terms are normalized by their denominators, respectively. The introduction of the silhouette term E_s can smooth the objective function and reduce the number of local minima around the global minimum, thus making the optimization process more likely to converge to the real global minimum.

To penalize sudden changes of hand-object poses between two adjacent frames, a smoothing term is introduced into the matching error function $E(\mathbf{z}, \mathbf{x}_{h\text{-}o})$ to smooth the recovered hand-object motion. The smoothing term is defined as:

$$E_m(\mathbf{x}_{h-o,t}) = \left\| \mathbf{x}_{h-o,t} - \hat{\mathbf{x}}_{h-o,t-1} \right\| \tag{4}$$

where $\mathbf{x}_{h\text{-}o,t}$ is a hand-object pose hypothesis of the current frame and $\hat{\mathbf{x}}_{h\text{-}o,t-1}$ is the recovered hand-object pose of the previous frame.

The likelihood function is monotone decreasing with the matching error function $E(\mathbf{z}, \mathbf{x}_{h\text{-}o})$. It is defined as:

$$p(\mathbf{z}|\mathbf{x}_{h-o}) \propto \exp(-\lambda_e \cdot E(\mathbf{z},\mathbf{x}_{h-o})) \tag{5}$$

where λ_e is a constant factor for normalizing and its value is decided by the noise of the observation.

5 The tracking algorithm

In this tracking problem, the hand and the manipulated object are both tracked during the hand -object interaction. For this problem, the state transition model and observation model of this problem are both nonlinear, and the probability distribution of the system state is multi-peak due to incomplete observation and occlusions that occur during the hand-object interaction. In this paper, PF is used for hand-object pose tracking, which is a robust motion tracking framework in cases of clutter. By propagating multiple hypotheses along with time, PF has the capability to describe multi-peak distributions and can be used to solve state estimation problems of nonlinear and non-Gaussian systems [6]. According to a set of weighted particles $\left\{\mathbf{x}_{t-1}^{i}\right\}_{i=1}^{N}$ sampled from the posterior distribution $p(\mathbf{x}_{t-1}|\,\mathbf{z}_{1:t-1})$ of the system state $\mathbf{x}$ at time t-1, the PF algorithm constructs a new set of weighted particles $\left\{\mathbf{x}_{t}^{i}\right\}_{i=1}^{N}$ to approximate the posterior distribution $p(\mathbf{x}_t|\,\mathbf{z}_{1:t})$ of $\mathbf{x}$ at time t by the state transition model $p(\mathbf{x}_t|\,\mathbf{x}_{t-1})$ and the observation model $p(\mathbf{z}_t|\,\mathbf{x}_t)$. Where, the superscript i denotes the number of the particle, $\mathbf{x}_t$ is the system state at time t and in this paper it represents the hand-object pose $\mathbf{x}_{h\text{-}o,\,t}$, w_t is the weight of $\mathbf{x}_t$, and $\mathbf{z}_{1:t}$ is a sequence of observations from time 1 up to time t.

A problem of the standard PF algorithm is that it uses the transition prior $p(\mathbf{x}_t|\,\mathbf{x}_{t-1})$ as the proposal distribution for importance sampling without considering the latest observation $\mathbf{z}_t$, so the importance sampling is suboptimal. In this paper, the tracking problem is a high-dimensional problem. During the tracking process, the standard PF needs large numbers of samples to approximate the real posterior distribution of the system state, making it too slow. Inadequate samples will lead to a tracking failure.

To solve the problem of the standard PF in high-dimensional state space, in this paper, the Gaussian PSO algorithm [12] is integrated into PF to improve the PF sampling process. Gaussian PSO algorithm is an improved PSO algorithm which generates the velocity vector through Gaussian distributions. It is superior to the classic PSO algorithm in convergence ability. The update equations for particle velocities and positions are as follows:

$$\mathbf{v}^{i,k+1} = |randn|\left(\mathbf{p}^{i,k}-\mathbf{x}^{i,k}\right) + |randn|\left(\mathbf{g}^{k}-\mathbf{x}^{i,k}\right) \tag{6}$$

$$\mathbf{x}^{i,k+1} = \mathbf{x}^{i,k} + \mathbf{v}^{i,k+1} \tag{7}$$

where $|randn|$ is the absolute value of the random number generated according to a standard Gaussian distribution $N(0, 1)$. $\mathbf{x}^{i,\,k}$ and $\mathbf{v}^{i,\,k}$ are the position and velocity of the i-th particle at generation k, respectively. In this paper, $\mathbf{x}^{i,\,k}$ represents the hand-object pose hypothesis $\mathbf{x}_{h\text{-}o}^{i,k}$. $\mathbf{p}^{i,\,k}$ is the individual best position which stores the best position ever found by the i-th particle till generation k. $\mathbf{g}^{k}$ is the global best position which stores the best position ever found by the whole particle swarm.

In this paper, the objective function of Gaussian PSO is the matching error function $E(\mathbf{z}, \mathbf{x}_{h\text{-}o})$ with the latest observation $\mathbf{z}_t$. By searching for the hand-object pose with the smallest matching error in the 32(26 + 6) dimensional space using Gaussian PSO, the particles predicted from the transition prior $p(\mathbf{x}_t|\,\mathbf{x}_{t-1})$ are evolved and moved to the areas with higher likelihood values before the update of particle weights. In this paper, a first-order self-regression model is used as the transition prior

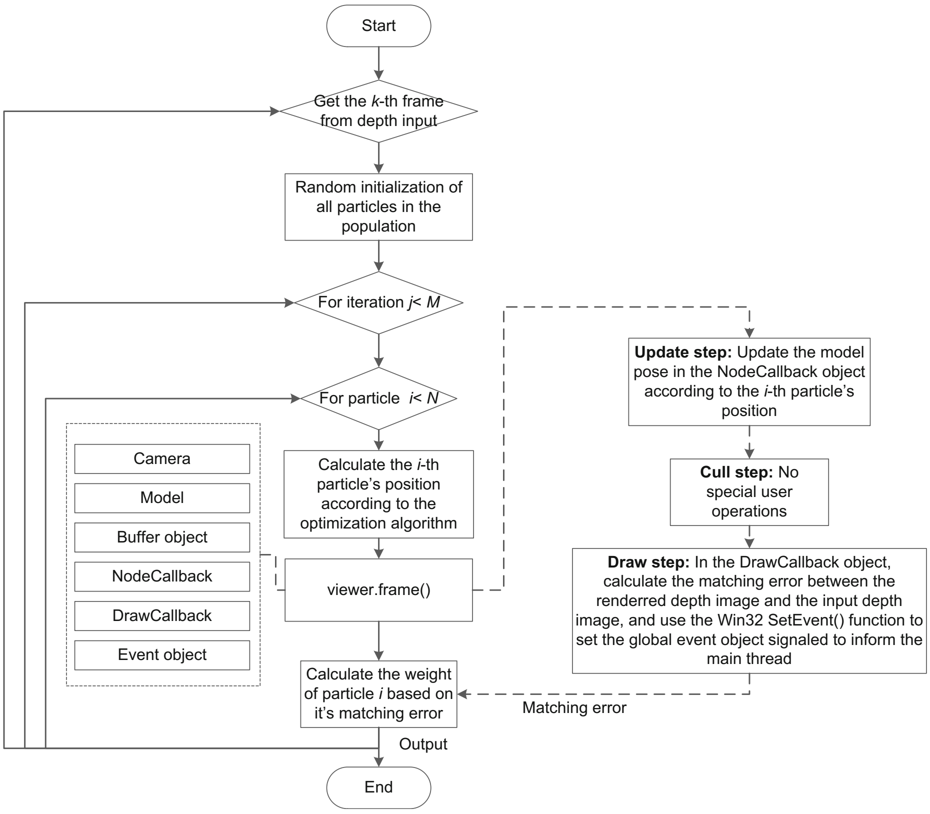

Start
Get the k-th frame from depth input
Random initialization of all particles in the population
For iteration j< M
For particle i< N
Camera
Model
Buffer object
NodeCallback
DrawCallback
Event object
Calculate the i-th particle's position according to the optimization algorithm
viewer.frame()
Update step: Update the model pose in the NodeCallback object according to the i-th particle's position
Cull step: No special user operations
Draw step: In the DrawCallback object, calculate the matching error between the renderred depth image and the input depth image, and use the Win32 SetEvent() function to set the global event object signaled to inform the main thread
Calculate the weight of particle i based on it's matching error
Matching error
Output
End

Fig. 4 Flowchart of the single-virtual-camera system. This system only creates one virtual camera for depth map rendering and calculating of matching errors. Each particle is rendered into its corresponding depth map all through this one virtual camera. Only one particle's matching error is calculated per OSG frame

$p(\mathbf{x}_t|\,\mathbf{x}_{t-1})$ to propagate the particles along with time. The particles at time t are initialized by the following equation

$$\mathbf{x}_{h-o,t}^{i,0} = \mathbf{p}_{h-o,t-1}^{i,K} + \mathbf{r} \tag{8}$$

where $\mathbf{x}_{h\text{-}o,t}^{i,0}$ is the initial value of the i-th hand-object pose hypothesis at time t before the Gaussian PSO optimization, $\mathbf{p}_{h\text{-}o,t-1}^{i,K}$ is the individual best position of the i-th hand-object pose hypothesis converged after K generations of Gaussian PSO optimization at time t-1, and $\mathbf{r}\sim N(0, \boldsymbol{\Sigma})$ is a multi-dimensional Gaussian noise with average 0 and covariance matrix $\boldsymbol{\Sigma}$. The diagonal elements of $\boldsymbol{\Sigma}$ are determined according to the maximum inter-frame angular or translational changes of the hand-object pose. At the first frame, to initialize the model-based tracking process, the hand and the manipulated object are put in their initial positions respectively for calibration.

The proposed hand-object tracking algorithm is summed up as follows:

1) Initialization: Sample N particles $\left\{\mathbf{x}_{h\text{-}o,0}^{i}, 1/N\right\}_{i=1}^{N}$ from the prior distribution $p(\mathbf{x}_{h\text{-}o,\,0})$, and weight each particle as $1/N$.

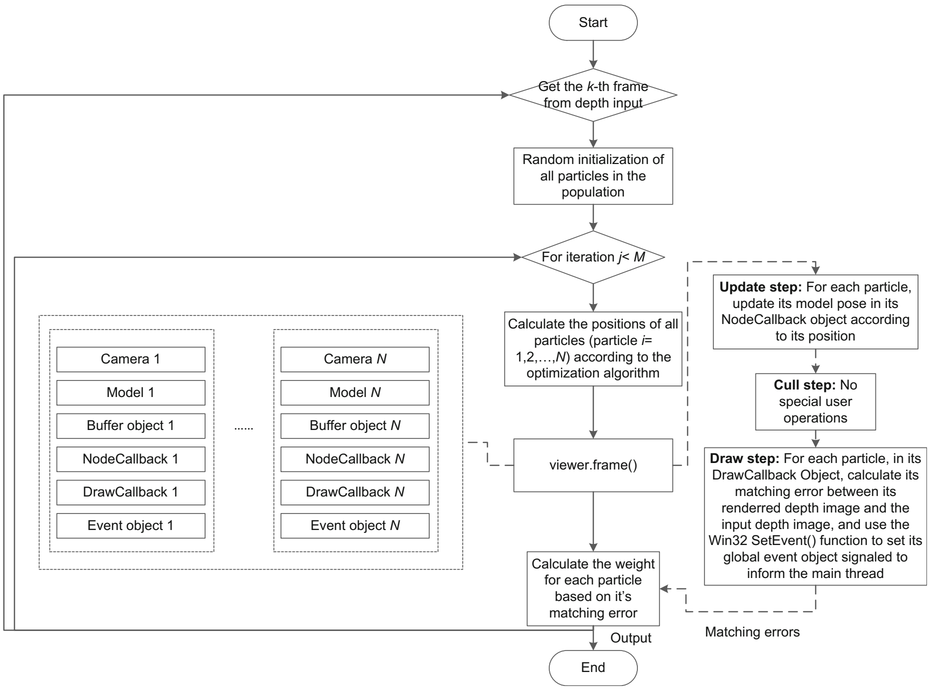
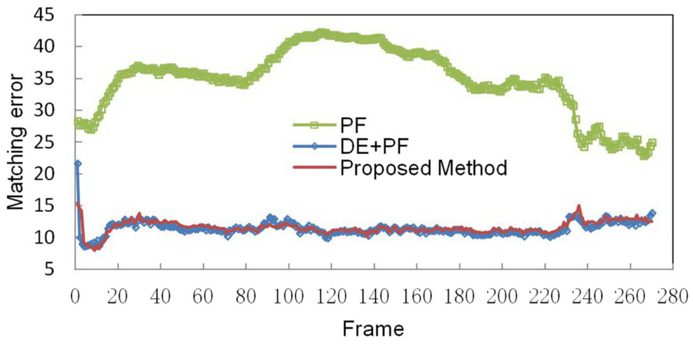

Start
Get the k-th frame from depth input
Random initialization of all particles in the population
For iteration j< M
Calculate the positions of all particles (particle i= 1,2,…,N) according to the optimization algorithm
viewer.frame()
Calculate the weight for each particle based on it's matching error
Output
End
Camera 1
Model 1
Buffer object 1
NodeCallback 1
DrawCallback 1
Event object 1
......
Camera N
Model N
Buffer object N
NodeCallback N
DrawCallback N
Event object N
Update step: For each particle, update its model pose in its NodeCallback object according to its position
Cull step: No special user operations
Draw step: For each particle, in its DrawCallback Object, calculate its matching error between its renderred depth image and the input depth image, and use the Win32 SetEvent() function to set its global event object signaled to inform the main thread
Matching errors

Fig. 5 Flowchart of the multiple-virtual-camera system. This system creates a virtual camera for each particle in the particle set. The matching errors of a whole set of particles are calculated per OSG frame

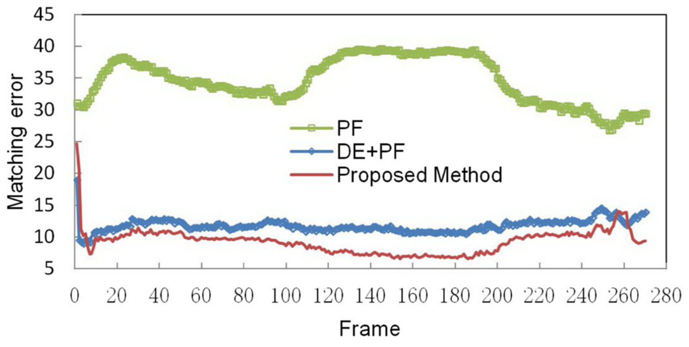
2) State transition: The importance sampling process is realized in this step. At time t, the positions of the particles are initialized according to Eq. (8). By introducing the latest observation $\mathbf{z}_t$ into the objective function to be optimized, the particles are evolved and moved to the areas with higher likelihood values based on Eqs. (6) and (7).
3) Update of weights: Weight the particles as $w_t^i \propto w_{t-1}^i p\left(\mathbf{z}_t | \mathbf{x}_{h\text{-}o,t}^i\right)$ by using the observation likelihood $p(\mathbf{z}_t | \mathbf{x}_{h-o,\,t})$ and then normalize the weights $\left\{w_t^i\right\}_{i=1}^N$. Output the particle with the biggest weight as the solution for the current frame.
4) Resampling: To avoid the degeneracy phenomenon of the particles, resample the particle set $\left\{\mathbf{x}_{h\text{-}o,t}^i, w_t^i\right\}_{i=1}^N$ to generate a new equal-weighted particle set $\left\{\mathbf{x}_{h\text{-}o,t}^i, 1/N\right\}_{i=1}^N$ according to the weights of the particles.
5) If the tracking process is over, exit from the tracking system. Else turn to Step 2.

6 Development of the prototype system

In this paper, a model-based method is used to track the hand poses and the object poses. This method builds 3D models for both the hand and the object. During the tracking, hand and object pose hypotheses are generated by using their 3D models respectively and the matching errors between the model features and the observed features are calculated. This method then searches for the hand-object pose with the smallest matching error in the hand-object state space. In this paper, the hand-object tracking system is developed based on the graphics rendering engine



(a) Matching errors on the real sequence of hand-sphere interaction



(b) Matching errors on the real sequence of hand-cylinder interaction

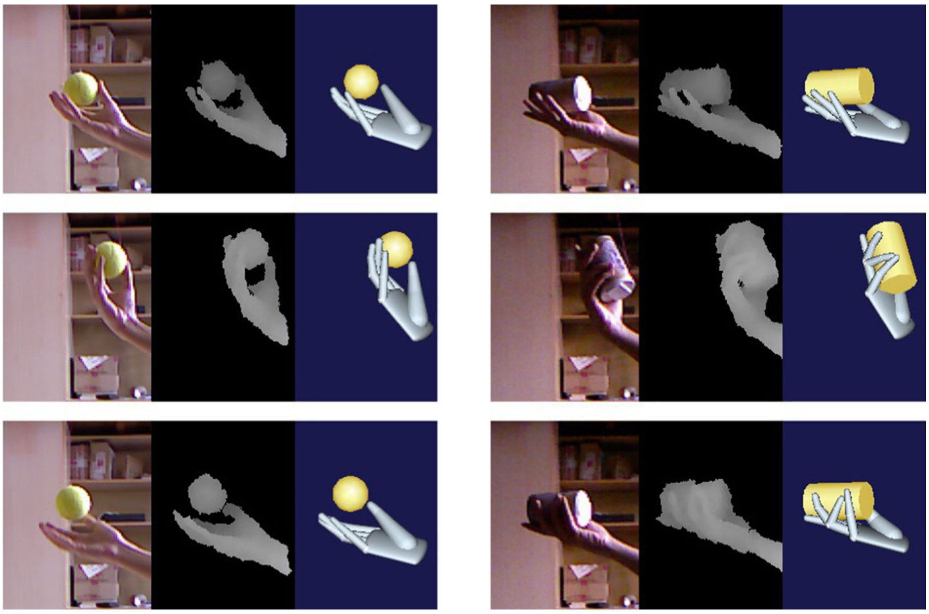
Fig. 6 Matching errors on two real sequences for three tracking methods: PF, DE+PF and the proposed method

OpenSceneGraph (OSG). At first, the hand and object models are loaded into OSG. In the tracking process, hand and object poses are updated by controlling the DOF nodes of the models with the osgSim::DOFTransform class. The hand and object models are rendered into depth maps through off-screen rendering techniques, which are then used for calculating the matching errors of hand-object hypotheses. As an open-source cross-platform rendering engine based on OpenGL, OSG organizes scene models by a tree-like data structure and can provide high-performance rendering based on the adopting of multiple culling methods, render state ordering and multi-thread rendering.

The rendering procedure of each frame in OSG can be divided into three steps, namely, the update traversal step, the cull traversal step and the draw traversal step. During the update traversal, OSG updates the user-defined dynamic data, adjusts the position and pose of the virtual camera, and updates the states of moving objects. During the cull transversal, OSG traverses all the nodes in the virtual scene and cull the invisible nodes. At the same time, the rendering states are optimized and ordered. During the draw traversal, the OpenGL drawing commands are invoked. The geometries and their rendering states are transmitted to the graphics hardware for rendering. In OSG, Camera objects and GraphicsContext objects both can create threads. Under default settings, OSG runs in a multithread mode, which creates a thread for each Camera object and a thread for each GraphicsContext object. Cull operations are performed in Camera threads and draw operations are carried out in GraphicsContext threads. This multithread rendering mode will start the update and cull traversal of a new OSG frame before the draw operations of the last frame are finished in GraphicsContext threads. This method can promote the rendering speed of the system and explore the computing power of hardwares.

Based on the proposed particle-based tracking algorithm, two hand-object tracking prototype systems are developed by using OSG and off-screen rendering methods. One of them is the single-virtual-camera system. Under the PF framework, single-virtual-camera system only creates one virtual camera for depth map rendering and the calculating of matching errors. Each hand-object particle is rendered into its corresponding depth map all through this one virtual camera. Thus, only one particle's matching error is calculated per OSG frame. Another system is the multiple-virtual-camera system which creates a virtual camera for each particle in the particle set. For this system, the matching errors of a whole set of particles are calculated per OSG frame.

As in Fig. 4, the single-virtual-camera system only creates one virtual camera to render depth maps for hand-object pose hypotheses. This camera has the scene root node as its subnode and is connected with a buffer object which is used to store the rendered depth map. The scene nodes are created by reading the 3D hand-object models into OSG, and the buffer object is connected with the camera through a frame buffer object (FBO). During the rendering process of each OSG frame, the virtual camera renders its scene model into a depth map which is stored in its corresponding buffer object. Additionally, the system creates an osg::NodeCallback object for the scene root node in order to update hand and object poses during the update traversal of each OSG frame. Meanwhile, the system creates an osg::Camera::DrawCallback object for the camera. After the camera renders the updated hand-object model into a



(a) Results of hand-sphere interaction (b) Results of hand-cylinder interaction

Fig. 7 Sample results of the proposed tracking method on real sequences. For (**a**), the results of frames 31, 121, 231 are shown, while for (**b**), the results of frames 31, 171, 231 are shown. For both (**a**) and (**b**), from left to right: real RGB images, real depth images, and the tracking results

depth map, the system will calculate the matching error between the rendered depth map and the observed depth map in the osg::Camera::DrawCallback object. Since OSG runs in a multithread mode under default settings, which creates a thread for each Camera and a thread for each GraphicsContext. This multithread mode will start the update traversal of a new frame before the draw operations of the last frame are finished in the GraphicsContext thread. To avoid data conflict between multiple threads, this system creates an event object for the camera and the synchronization and communication of different threads are realized by using the Win32 SetEvent() function and WaitForSingleObject() function. After the matching error calculating is finished in the GraphicsContext thread, the corresponding event object will be set signaled by the SetEvent() function, which will be then informed to the main thread. The main thread won't perform any other operation until it receives the signal of this event.

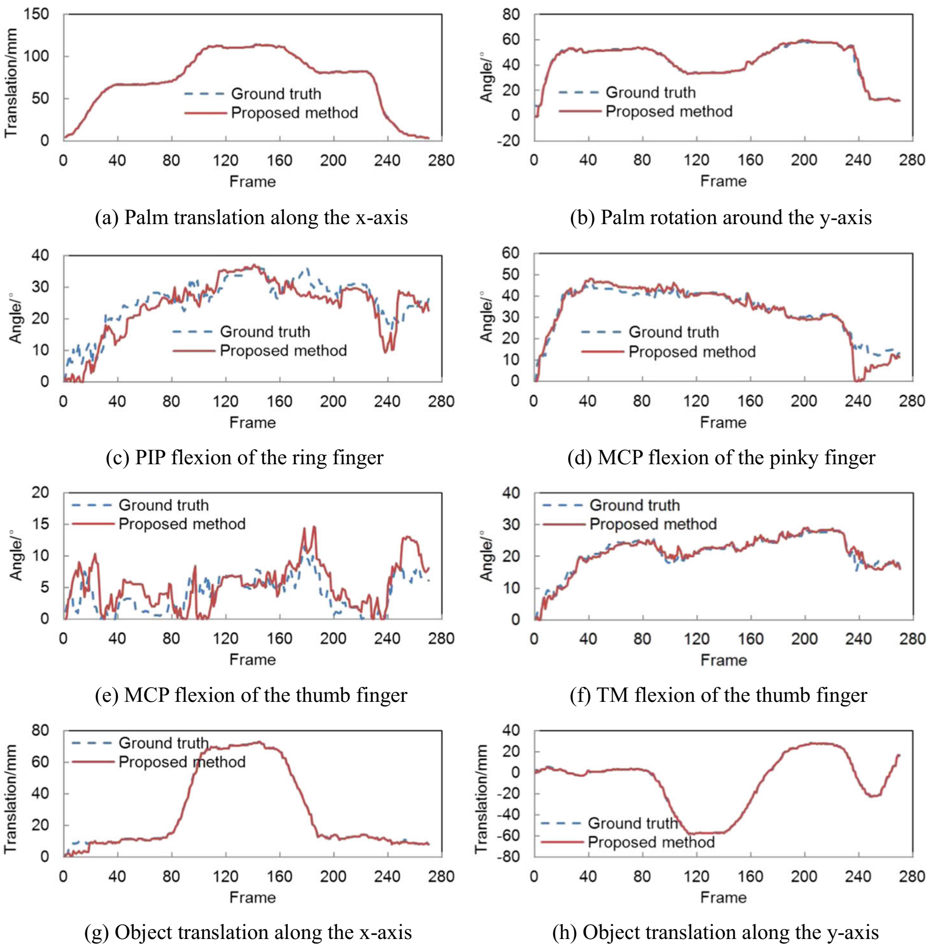
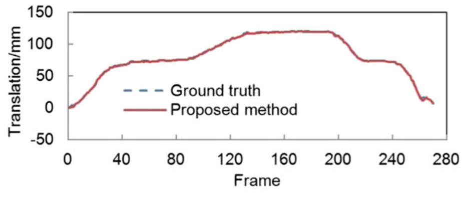
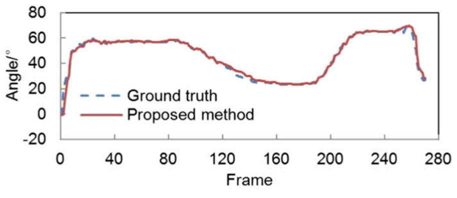


Fig. 8 Comparisons between experimental results and ground truth values on the synthetic sequence of hand-sphere interaction

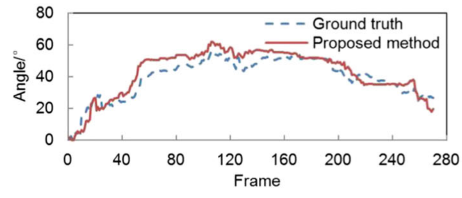
Another system is the multiple-virtual-camera system. This system is developed to explore the potential of the multithread rendering mode of OSG. As shown in Fig. 5, this system creates a virtual camera, a scene root node, a buffer object, an osg::NodeCallback object, an osg::Camera::DrawCallback object and an event object for each particle in the particle set. Given that there are *n* particles in the particle set, this system will start *n* camera threads and *n* GraphicsContext threads per OSG frame to explore the multithread parallel processing capacity of the computer. For thread synchronization, different from the single-virtual-camera system, this system creates an event object for each particle's virtual camera. The synchronization and communication of different threads are realized by using the Win32 SetEvent() function and WaitForMultipleObjects() function. The particle that firstly finishes calculating its matching error in its GraphicsContext thread will firstly set the corresponding event object as signaled through the SetEvent() function and be firstly served by the main thread. The main thread won't turn to any other operation until it has received the signals from all particles. So, the multiple-virtual-camera system is a multi-thread accelerated counterpart of the single-virtual-camera system.



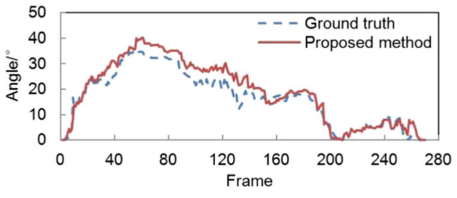
(a) Palm translation along the x-axis



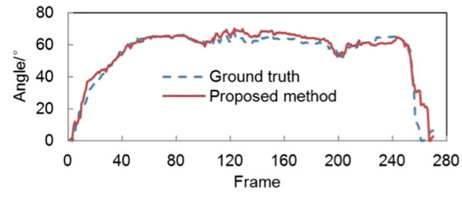
(b) Palm rotation around the y-axis



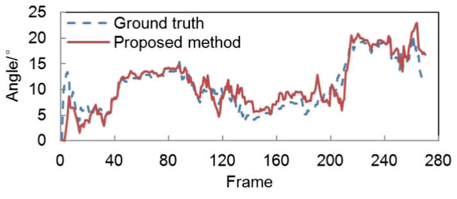
(c) PIP flexion of the middle finger



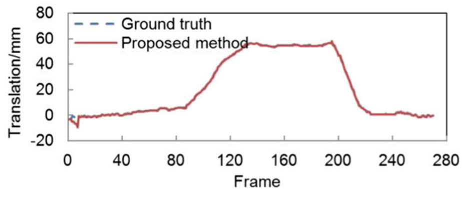
(d) MCP flexion of the ring finger



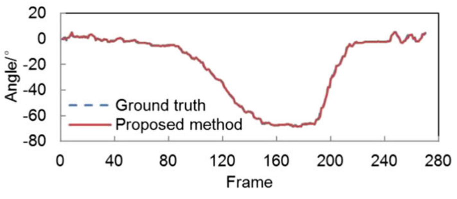
(e) MCP flexion of the pinky finger



(f) TM flexion of the thumb finger



(g) Object translation along the x-axis



(h) Object rotation around the z-axis

Fig. 9 Comparisons between experimental results and ground truth values on the synthetic sequence of hand-cylinder interaction

Table 1 Statistical analysis of the tracking errors on the synthetic sequence of hand-sphere interaction

Pose parameter	Mean error	Standard deviation
Palm translation along the x-axis	0.3453 mm	0.2896 mm
Palm rotation around the y-axis	0.6665°	1.1481°
PIP flexion of the ring finger	3.5181°	2.5534°
MCP flexion of the pinky finger	2.3356°	2.9465°
MCP flexion of the thumb finger	2.2100°	1.8216°
TM flexion of the thumb finger	0.7944°	0.8104°
Object translation along the x-axis	0.6265 mm	1.3522 mm
Object translation along the y-axis	0.3121 mm	0.5192 mm

7 Experiments

The proposed hand-object tracking algorithm was evaluated through experiments based on both real and synthetic sequences. During all experiments in this paper, the proposed tracking algorithm runs with 60 particles and 40 Gaussian PSO generations for each input frame.

7.1 Experiments on real sequences

First, the tracking algorithm is evaluated based on real depth image sequences. The real depth images are acquired by Microsoft Kinect 1.0 depth camera with a resolution of 640×480 pixels and a frame rate of 30 fps. The experiment includes two parts. The first one tracks the hand in interaction with a spherical object and the second one tracks the hand in interaction with a cylindrical object. For evaluation, the proposed method is compared with standard particle filtering (PF) and a tracking algorithm which combines differential evolution with particle filtering (DE+PF) [14]. The matching error calculation is the most time-consuming step in the tracking system. To be fair, standard PF uses 2400 particles, while DE+PF uses 40 particles and 60 generations for DE optimization of each input frame. Hence, for the three methods, the numbers of matching error calculations for tracking one frame are all 2400. Fig. 6 plots the matching errors of the three methods on two real sequences. It can be seen that that standard PF can’t track hand-object motion well with accumulated errors. The proposed method and DE+PF both have obviously better performance than standard PF. For the real sequence of hand-sphere interaction, the proposed method and DE+PF perform equally well. However, for the real sequence of hand-cylinder interaction which is more difficult to track due to occlusions, the proposed method performs better than DE+PF. Sample results of the proposed tracking method on

Table 2 Statistical analysis of the tracking errors on the synthetic sequence of hand-cylinder interaction

Pose parameter	Mean error	Standard deviation
Palm translation along the x-axis	0.5884 mm	0.5236 mm
Palm rotation around the y-axis	1.1793°	2.0406°
PIP flexion of the middle finger	4.7143°	2.9983°
MCP flexion of the ring finger	2.9296°	2.3519°
MCP flexion of the pinky finger	2.9617°	3.9455°
TM flexion of the thumb finger	1.5982°	1.6046°
Object translation along the x-axis	0.3143 mm	0.7198 mm
Object rotation around the z-axis	0.3138°	0.4521°

Table 3 Tracking speed comparison for two tracking systems

Prototype system	Time consumption for tracking one frame
Single-virtual-camera system	5.85 s
Multiple-virtual-camera system	4.41 s

two real sequences are shown in Fig. 7. For each group of results, the first column shows the RGB images captured by the Kinect RGB camera, the second column shows the depth images acquired by the Kinect depth camera, and the third column shows the tracking results on the real depth images. Experimental results demonstrate that the proposed method can track the hand and the manipulated object effectively.

7.2 Experiments on synthetic sequences

Since it is impossible to acquire ground truth from real image sequences directly, synthetic depth image sequences attached with ground truth of hand-object poses are used for quantitative evaluation of the proposed method in this section. The tracking results of the proposed method on the two real sequences are used as the ground truth to define the synthetic sequences. The synthetic sequence of hand-sphere interaction is generated according to the results on the first real sequence, while the synthetic sequence of hand-cylinder interaction is generated according to the results on the second real sequence. The two synthetic sequences are used as the input of the tracking system in this section. Results of the proposed method on some hand-object pose parameters are shown in Figs. 8 and 9 compared with the corresponding ground truth values. The solid curves are the results and the dotted curves show ground truth values. Statistical analyses of the errors of these pose parameters on the whole sequences are listed in Tables 1 and Table 2. It can be seen from Figs. 8, 9, Tables 1 and 2 that the results of the proposed method are consistent with ground truth values.

7.3 Evaluation of tracking speeds

The tracking speeds of the proposed tracking algorithm implemented on both two kinds of prototype systems (the single-virtual-camera system and the multiple-virtual-camera system) are also evaluated experimentally. The experiments are conducted on a PC with a 4 Core i5 2.9 GHz CPU, 4.0 GB memory and an Nvidia GeForce GTX 950 M GPU. For both two systems, using the synthetic sequence of the hand-cylinder interaction as input, the proposed tracking algorithm runs with 60 particles and 40 Gaussian PSO generations for each input frame. During the tracking process, the rendered depth maps for matching error calculating are generated through off-screen rendering using FBO. The time consumptions of the single-virtual-camera system and the multiple-virtual-camera system for tracking one input frame in average are listed in Table 3. Experimental results demonstrate that the tracking speed of the multiple-virtual-camera system is higher than that of the single-virtual-camera system. Although with higher tracking speeds, the multiple-virtual-camera system requires the intrinsic parallel characteristics of the tracking algorithm. Nevertheless, the single-virtual-camera system doesn't have this kind of restriction.

8 Conclusions

Tracking a hand in interaction with an object based on vision is very challenging due to the occlusions between the hand and the manipulated object. To overcome the impacts of occlusions, we build 3D models for both the hand and the manipulated object and propose a model-based tracking method in this paper to track the hand and the object simultaneously during the hand-object interaction. Depth images acquired by a Kinect depth camera are used as system input and an improved PF algorithm is adopted to search for the most likely hand-object state. To overcome the difficulty of particle sampling in the high-dimensional space, Gaussian PSO is combined with the PF algorithm to improve the process of particle sampling, driving the particles to the regions with higher likelihood values. According to the proposed tracking algorithm, the hand-object tracking prototype systems are developed based on OSG and off-screen rendering techniques. Experimental results have proved that the proposed method can track hand-object motion robustly with few particles.

As our tracking method is a model-based method, the tracking process needs to be initialized and searching for the best hand-object state in high-dimensional space has high computational complexity. In the future, we will combine a learning-based method with our model-based tracking method. For each frame, the initial estimation of the hand-object state will be carried out by a learning-based method, while the follow-up model-based tracking process will then be used to refine the results. By this way, the hand-object tracking can be initialized automatically and the difficulty for searching the best solution in high-dimensional space can be reduced. In addition, the performance of the tracking system is significantly sensitive to the noise of the depth information acquired by Kinect. To make the system more robust, in the future, multiple depth cameras will be used to improve the quality of depth images. And as the calculation of the matching error is the most time-consuming step in the tracking system which is also easy to be parallelized, future research will apply GPU programming to accelerate the system.

Acknowledgements This work was funded by the National Natural Science Foundation of China (NO. 51475251 and 51705273).

References

1. Bray M, Koller-Meier E, Van Gool L (2004) Smart particle filtering for 3D hand tracking. In: Proceedings of the sixth IEEE international conference on automatic face and gesture recognition, pp 675–680
2. Cui J, Sun Z (2004) Visual hand motion capture for guiding a dexterous hand. In: Proceedings of the sixth IEEE international conference on automatic face and gesture recognition, pp 729–734
3. Deutscher J, Blake A, Reid I (2000) Articulated body motion capture by annealed particle filtering [C]. In: IEEE computer society conference on computer vision and pattern recognition (CVPR), vol 2, pp 126–133

4. Do MQ, Hung CH, Lin CH (2017) Robot navigation control using vision based steering wheel. Multimed Tools Appl 76(22):24569–24588
5. Doliotis P, Athitsos V, Kosmopoulos D, Perantonis S (2012) Hand shape and 3D pose estimation using depth data from a single cluttered frame. In: International symposium on visual computing (ISVC), vol 1, pp 148–158
6. Doucet A, Godsill S, Andrieu C (2000) On sequential Monte Carlo sampling methods for Bayesian filtering. Stat Comput 10(3):197–208
7. Erol A, Bebis G, Nicolescu M, Boyle RD, Twombly X (2005) A review on vision-based full DOF hand motion estimation. In: IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 75–82
8. Hamer H, Schindler K, Koller-Meier E, Van Gool L (2009) Tracking a hand manipulating an object. In: IEEE 12th international conference on computer vision (ICCV), pp 1475–1482
9. Keskin C, Kiraç F, Kara YE, Akarun L (2012) Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In: Proceedings of the 12th European conference on computer vision (ECCV), vol 6, pp 852–863
10. Keskin C, Kiraç F, Kara YE, Akarun L (2013) Real time hand pose estimation using depth sensors. In: Fossati A, Gall J, Grabner H, Ren X, Konolige K (eds) Consumer depth cameras for computer vision. Springer, London, pp 119–137
11. Kjellström H, Romero J, Martinez D, Kragic D (2008) Simultaneous visual recognition of manipulation actions and manipulated objects. In: European conference on computer vision (ECCV), vol 2, pp 336–349
12. Krohling RA (2004) Gaussian swarm: a novel particle swarm optimization algorithm. In: Proceedings of the 2004 IEEE conference on cybernetics and intelligent systems, vol 1, pp 372–376
13. Kyriazis N, Argyros AA (2013) Physically plausible 3D scene tracking: the single actor hypothesis. In: IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 9–16
14. Li D, Zhou Y (2015) Combining differential evolution with particle filtering for articulated hand tracking from single depth images. Int J Signal Process Image Process Pattern Recognit 8(4):237–248
15. Liang H, Yuan JS, Thalmann D, Zhang Z (2013) Model-based hand pose estimation via spatial-temporal hand parsing and 3D fingertip localization. Vis Comput 29(6-8):837–848
16. Lu G, Nie L, Sorensen S, Kambhamettu C (2017) Large-scale tracking for images with few textures. IEEE Trans Multimedia 19(9):2117–2128
17. Nie L, Yan S, Wang M, Hong R, Chua TS (2012) Harvesting visual concepts for image search with complex queries. In: Proceedings of the 20th ACM international conference on multimedia, pp 59–68
18. Oikonomidis I, Kyriazis N, Argyros AA (2011) Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In: IEEE international conference on computer vision (ICCV), pp 2088–2095
19. Oikonomidis I, Kyriazis N, Argyros AA (2011) Efficient model-based 3D tracking of hand articulations using Kinect. In: Proceedings of the 22nd British machine vision conference (BMVC), pp 101.1–101.11
20. Prisacariu VA, Reid I (2012) 3D hand tracking for human computer interaction. Image Vis Comput 30(3): 236–250
21. Romero J, Kjellström H, Kragic D (2009) Monocular real-time 3D articulated hand pose estimation. In: International conference on humanoid robots, pp 87–92
22. Romero J, Kjellström H, Kragic D (2010) Hands in action: real-time 3D reconstruction of hands in interaction with objects. In: IEEE international conference on robotics and automation, pp 458–463
23. Taylor J, Bordeaux L, Cashman T, Corish B, Keskin C, Sharp T, Soto E, Sweeney D, Valentin J, Luffx B, Topalianx A, Wood E, Khamis S, Kohli P, Izadi S, Banks R, Fitzgibbon A, Shotton J (2016) Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. ACM Trans Graph 35(4):143
24. Wang Y, Min J, Zhang J, Liu Y, Xu F, Dai Q, Chai J (2013) Video-based hand manipulation capture through composite motion control. ACM Trans Graph 32(4):43
25. Zhang Z, Seah HS, Quah CK, Sun J (2013) GPU-accelerated real-time tracking of full-body motion with multi-layer search. IEEE Trans Multimedia 15(1):106–119

Dongnian Li received his Ph.D. and B. Eng. degrees from School of Mechanical Engineering, Shandong University, China, in 2015 and 2009 respectively. He is currently working as a lecturer at School of Mechanical & Automotive Engineering, Qingdao University of Technology, China. His major research interests include computer vision, graphics and virtual reality.

Chengjun Chen is currently working as an associate professor at School of Mechanical & Automotive Engineering, Qingdao University of Technology, China. He received his Ph.D. and B. Eng. degrees from School of Mechanical Engineering, Shandong University, China, in 2008 and 2003 respectively. His major research interests include virtual reality and augmented reality in industrial applications.