ORIGINAL ARTICLE



Interactive animation generation of virtual characters using single RGB-D camera

Ning Kang^{1,2} · Junxuan Bai¹ · Junjun Pan^{1,2} · Hong Qin³

Published online: 6 May 2019 © Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract

The rapid creation of 3D character animation by commodity devices plays an important role in enriching visual content in virtual reality. This paper concentrates on addressing the challenges of current motion imitation for human body. We develop an interactive framework for stable motion capturing and animation generation based on single Kinect device. In particular, we focus our research efforts on two cases: (1) The participant is facing the camera; or (2) the participant is turning around or is side facing the camera. Using existing methods, camera could obtain a profile view of the body, but it frequently leads to less satisfactory result or even failure due to occlusion. In order to reduce certain artifacts appeared at the side view, we design a mechanism to refine the movement of the human body by integrating an adaptive filter. After specifying the corresponding joints between the participant and the virtual character, the captured motion could be retargeted in a quaternion-based manner. To further improve the animation quality, inverse kinematics are brought into our framework to constrain the target's positions. A large variety of motions and characters have been tested to validate the performance of our framework. Through experiments, it shows that our method could be applied to real-time applications, such as physical therapy and fitness training.

Keywords Character animation \cdot Motion capture \cdot Retargeting \cdot RGB-D camera

1 Introduction

The emergence of commodity depth camera gives rise to the possibilities of rapid creation of character animation for novice users. A prevailing idea is to capture the motion of real human and generate an animated character simultaneously. Although some toolkit developments, such as [12], have facilitated basic process of motion capture for human body, it is impossible to use the captured data to create animation directly due to the low quality. Recent works [15,29]

Ning Kang and Junxuan Bai are joint first authors.

 Junjun Pan pan_junjun@hotmail.com
 Hong Qin qin@cs.stonybrook.edu

State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China

- ² Beihang University Qingdao Institute, Qingdao, China
- ³ Department of Computer Science, Stony Brook University (SUNY Stony Brook), New York, USA

focused on enhancing the capturing result using single RGB-D camera, but they hardly can be operated interactively.

After the motion is captured, the retargeting process will adapt the motion clip to 3D character. Early research papers [3,8,19] mainly focused on the transformation of motions between characters with different sizes or topologies. The transformation was generated by solving an optimization problem which is made up of geometric constraints. Recently, researchers tended to acquire the transformation through an existing database which comprises the matches of corresponding joints [1,28]. However, the retargeting result was unavoidably affected by the content of the database. In this paper, we design an interactive framework for stable motion capturing and animation generation using single Microsoft Kinect device. We capture the motion and retarget the clip to virtual character. Figure 1 summarizes the pipeline of our framework.

According to our observation, two cases often appear for most applications using a single RGB-D camera: (1) The participant is facing the camera; (2) the participant is turning around or is side facing the camera that the camera obtains a side view of the body. Considering the computation requirement, for single RGB-D camera, existing methods fail to



Fig. 1 Framework overview

capture the motion of turning around and motions in side view due to self-occlusion. So we propose a set of schemes to obtain better results when self-occlusion happens. During the motion retargeting stage, for humanoid characters, we control target character in a quaternion-based manner. For non-humanoid characters, we first recognize the user's action by a classifier, and then, we map the motion to the target character. To further improve the animation quality, inverse kinematics (IK) are brought into our framework to constrain the target's positions. Our innovative contributions are listed as follows:

- We propose an adaptive filter to estimate the rotation angle and direction of the user using a single RGB-D camera. It can recognize the action of turning around and capture the motion in real time.
- We design a practical motion retargeting method for humanoid and non-humanoid characters.
- We devise a motion refinement technique based on IK.
 It can enhance the performance of target character's motion.

The remainder of this paper is organized as follows. After briefly discussing the prior works in Sect. 2, we introduce the workflow of our motion caputure approach in Sect. 3. The motion retargeting process is discussed in Sect. 4. Section 5 introduces the algorithms for IK calibration and optimization. Section 6 documents the experimental results and introduces the applications we have developed. Finally, Sect. 7 concludes the paper with necessary discussions.

2 Related works

Our research is closely relevant to motion capture and retargeting techniques. Self-occlusion is a major problem which reduces the quality of motion data. Thus, we also list several methods which handle self-occlusion.

Motion capture techniques based on commodity depth cameras are popular in recent years. Since the emergence of Microsoft Kinect, motion capture algorithms and applications [15,18] based on this type of camera are well studied. Liu et al. [15] created local correspondence between optical captured motion data and Kinect captured motion data, and

built the relationship with Gaussian process. The posture was reconstructed by searching the Kinect input in the local GP models. Several works [4,29,31] focused on the improvement of tracking result with single RGB-D camera. Among these, Ye et al. [31] devised a real-time approach to estimate the pose and shape from depth images by combining Gaussian mixture model with articulated deformation model. The pose was estimated through probability density estimation with a body template.

Other methods of user-character motion capture were implemented in [14,20,27]. Mousas et al. [20] extended the regular structure of the hidden Markov model (HMM) and trained the HMM on a motion dataset containing movements of human dance. The system can predict the corresponding dance motion according to the user's current pose, and the user is able to dance with a virtual character wearing motion capture suit and a head-mounted display (HMD).

Motion retargeting transfers the motion from one character to other virtual characters and generates new natural animation. In skeleton-based animation, motion retargeting is quite helpful. A detailed survey in this field can be found in [10]. The majority of our work is related to motion retargeting techniques. Motion retargeting techniques can be generally divided into two categories: constraints-based method and data-driven method. The former obtains the transformation through a set of geometric constraints or physical constraints [7,24], such as the angles of joints, the positions of end effectors, or the dynamic formulations. Constraintsbased method is still dominant in industry applications for its simplicity and effectiveness for real-time interactions. The latter models the character behavior based on a motion library [5,11,23], which employs decision-making mechanism and transforms the motion to other characters. Data-driven method is useful to generate motions with variations.

In addition to humanoid characters, motion retargeting for non-humanoid had been studied by [25,26,30]. These methods realized arbitrary motion mapping and provided a new way for real-time character control.

Data handling for self-occlusion is used to improve the quality of motion clip before retargeting. Inverse kinematics (IK) methods were commonly employed to deal with occlusion and losses of joints when body moves [9,13,17]. Grochow et al. [9] computed natural looking poses by solving

a constrained optimization problem. Lv et al. [17] formulated data-driven inverse dynamics using a set of reference poses and dynamics data. It succeed in reconstruction of motion using prerecorded dynamics data. In our framework, IK constraints are applied on several joints and solved in parallel.

3 Robust motion capture based on RGB-D camera

In our technique, users perform actions in front of a Kinect. We take the joints extracted by Kinect SDK as inputs, and then, we apply the captured motion to virtual character to generate animation. However, the extracted joints are incorrect in many cases, especially when a user is turning around or performing some actions on his side. Self-occlusion usually occurs in these cases. To improve the captured result, we propose an adaptive filter. Our adaptive filter is illustrated in Fig. 2. The main contribution is to estimate the turning angle and direction. As shown in Fig. 2, the information of three parts, which include shoulders, hips and face, are considered in the filter.

3.1 Estimation of turning angle

As shown in Eq. 1, the turning angle is predicted by three parts E_s , E_h and E_f , which represent the estimation angle based on shoulders, hips and face information. In this equation, $0 \le \lambda_s$, λ_h , $\lambda_f \le 1$ and $\lambda_s + \lambda_h + \lambda_s = 1$. Each term will be explained in the following sections. The turning angle is calculated as follows:

$$\Phi = \lambda_s E_s + \lambda_h E_h + \lambda_f E_f. \tag{1}$$

Estimation from shoulders (E_s) is related to the shoulder spacing L_s . The first step of the estimation is to detect whether the human body is moving or still. We devise a

threshold segmentation method inspired by [22]. We record several groups of shoulder spacing data into motion curves which contains the actions of turning around in front of the camera, and then, we make several statistics on parameters. Finally, we calculate the maximum d_{max} and the minimum d_{min} of shoulder spacing. The calculation for d_{max} is formulated as:

$$d_{\max} = \max\left(w_f * w_t * \left(\sum_{i \in B_f} \frac{L_i}{u_f} - \sum_{j \in B_t} \frac{L_j}{u_t}\right)^2\right). \quad (2)$$

In Eq. 2, w_f is the proportion of points facing the camera in the motion curve, w_t is the proportion of other states points, and u_f and u_t are the numbers of points facing the camera and points in other states. B_f and B_t are the sets of frames, in which the human body is facing the camera or in other states. L_i and L_j are the values of shoulder spacing at the *i*th and the *j*th frame. The minimum d_{\min} is calculated in the same way.

After we tested 50 people, we found that d_{max} is 0.35 and d_{min} is 0.25. When the value of shoulder spacing L_s is larger than d_{max} , we consider that the human body is facing the camera. On the other hand, when L_s is less than d_{min} , the human body is side facing the camera. Otherwise, the human body is between these two states. An example is shown in Fig. 3.

The vibration on the positions of bone joints captured by Kinect will degrade the quality of follow-up animations. Since the angle of body turning is related to the length of bone spacing, if the vibration is not processed, the virtual character will tremble when it rotates. In each frame, we set the compensation value L_{loss} to remove the vibration. L_{loss} is computed by gradient smoothing as follows:

$$L_{\rm loss} = \frac{L_{\rm start} - L_{\rm end}}{d_s} \Delta t, \tag{3}$$



Fig. 2 The procedure of our filter. The symbols in red are the input of our filter, and the symbols in green are the intermediate variables (color figure online)



Fig. 3 The workflow of our filter. In the first row, from left to right, it illustrates the 2D distance between the hip joints, the values of joint at different stages. The second row shows the orientation of the virtual character corresponding to the motion curve

where L_{start} and L_{end} denote the length of shoulder spacing in the start frame and the end frame. Δt is the time interval between the start frame and the *i*th frame. The step length d_s is a constant value. With L_{loss} , we can transform the length of shoulder spacing into angle by the following linear mapping:

$$E_s = \frac{\pi \left(L_s + L_{\text{loss}} \right)}{2 \left(d_{\text{max}} - d_{\text{min}} \right)},\tag{4}$$

where L_s is the value of shoulder spacing and d_{max} and d_{min} are the thresholds mentioned above.

Estimation from hip (E_h) is obtained using the hip spacing L_h . The calculation of E_h is the same as that of shoulders, but the values of d_{max} and d_{min} are different from those of shoulders. Technically, the maximum is 0.17 and minimum is 0.05. According to the experimental experience, we found it is unstable to predict the turning angle only by hip joints or shoulder joints information. Through shoulder, hip and face information, we sum the weighted value to obtain the final turning angle. Since the change of shoulder spacing length is relatively stable, the weight of shoulder (λ_s) is larger (0.6). The weight of E_h is 0.3, and the weight of E_f is 0.1.

Estimation from face (E_f) mainly uses the rotation angle of face Φ_f . Kinect obtains the rotation angle of face according to the position of the five joint points (the mouth corners, eye corners and nose). Φ_f is reliable when the human body is facing the camera, so we set the $E_f = \Phi_f$ and coefficient λ_f to be 0.1. When the body turns more than 60°, the face information will be lost and Φ_f will be incorrect and unstable. Here we will reduce the coefficient λ_f to 0 and increase the coefficient of the crotch λ_h to 0.4.

3.2 Estimation of turning direction

We use the rotation angle of human body to divide the space into two regions. When the person is facing to camera, k = 1. When the person is back to camera, k = -1. The Z-axis information of shoulders represents different rotation directions in different regions, and the parameter *k* represents the region where the previous frame model is located. We explain the algorithm (Algorithm 1) of turning direction estimation.

Algorithm 1 Turning direction estimation.		
Input:		
the depth of left shoulder joint \mathbf{j}_{Ish}^{z} ,		
the depth of right shoulder joint \mathbf{j}_{Rsh}^{z} ,		
the direction of last frame k.		
Output:		
the direction of current frame ε .		
1: Compare the depth values of shoulder joints		
2: if $\mathbf{j}_{I,sh}^{z} > \mathbf{j}_{Rsh}^{z}$ then		
3: the current direction $\varepsilon = k$;		
4: else $\{\mathbf{j}_{Lsh}^{z} < \mathbf{j}_{Rsh}^{z}\}$		
5: the current direction $\varepsilon = -k$;		
6: end if		
7: return ε ;		

Parameter ε is used to control the direction of target model rotation. If ε is 1, it means that the model rotates to the left, otherwise the model turns right. We control the change of ε by comparing the z-axis information of left shoulder joint \mathbf{j}_{Lsh}^{z} and right shoulder joint \mathbf{j}_{Rsh}^{z} . Due to the difficulty of capturing at body turning process, we preset a turnaround animation [6] and retarget to the target model based on the angle of body turning eventually, so that the turning motion of target model can become more realistic.

4 Motion retargeting for virtual characters

In our framework, both humanoid and non-humanoid characters are able to be animated. We transfer the motion from real human to virtual characters in a straightforward manner. For humanoid character, the rotation of each joints which are extracted by Kinect is copied to character. For non-humanoid character, we classify the motion at first, and then, we animate the character using stored motion data.

4.1 Joints matching

A skeleton hierarchy should be generated after Kinect extracts all joints from real human. The first step is to set up the hierarchy. As shown in Fig. 4, the number of joints and the structures is different between characters. We define a dictionary to store the skeleton structure. Each term in the dictionary records the relationship of joints between real human and virtual characters.

The skeleton segment T in virtual character consists of two parts: base joint \mathbf{j}_{base} and leading joint \mathbf{j}_{lead} . \mathbf{j}_{base} contains the positions of parent node of the skeleton segment T and \mathbf{j}_{lead} contains the positions of leaf point. A group of skeleton segments which are connected each other constitute a chain. Technically, the leading joint will rotate around the base joint, and this rotation will be used to implement the motion retargeting in Sect. 4.2.

For the non-humanoid characters, we match a group of human joints with the joints of characters according to the structural similarity. For example, we believe that the leg of real human should control the leg of the deer character, and the spine of real human should affect the neck of the plant model. We set up the connections between them and deform the character accordingly. However, the movement of the whole body cannot be reproduced very well, so we introduce a local classifier to improve the animation in Sect. 4.3.

4.2 Rotation angle of bones

Like the method in [1], we extract the rotation axis and rotation angle of bones from the motion sequence. In each frame of the captured motion sequence, the bones of human will perform a rotation around its base joint. We calculate the direction vectors, **s** and **t**, of all the bones between human body and target model. By calculating the two-frame direction vector of the skeleton segment, the rotation angle θ and rotation axis **c** can be solved. The calculation can be described



Fig. 4 Skeletal structures for different characters

as follows:

 \mathbf{c}_i

$$=\mathbf{s}_i\times\mathbf{t}_i,\tag{5}$$

$$\theta_i = \arccos\left(\frac{\mathbf{s}_i \cdot \mathbf{t}_i}{\|\mathbf{s}_i\|^2 \|\mathbf{t}_i\|^2}\right). \tag{6}$$

Here, $\mathbf{c} = (c_x, c_y, c_z)$ is the axis of rotation, and θ is the angle of rotation. With these axis and angle, the rotation of skeleton segment can be expressed as a quaternion \mathbf{q} :

$$\mathbf{q} = \cos\frac{\theta}{2} + \left(c_x \mathbf{i} + c_y \mathbf{j} + c_z \mathbf{k}\right) \sin\frac{\theta}{2}.$$
 (7)

We simplify the representation of quaternion by $\mathbf{q} = (\mathbf{c}, \theta)$. The bone of target model should rotate θ° around the axis \mathbf{c} , getting the new rotation value for each joint. We bring in $\mathbf{p} = (\mathbf{j}_{\text{lead}}, 0)$, and then, the specific calculation is as follows:

$$Rotate (\mathbf{j}_{lead}, \mathbf{q}) = \mathbf{q} \mathbf{p} \mathbf{q}^{-1}$$
(8)

4.3 Local classifier

Motion cannot be transferred to non-humanoid character directly, so we use another way to create the animation. We recognize the user's action and animate the character using stored motion data. Then, a module in Kinect SDK is employed to recognize action. The module can provide some useful information, including whether it is a whole body movement, whether the movements of the left body and right body are symmetrical, whether it contains hand, etc. The module also contains a classifier which is implemented in SVM.

To train the SVM, we select ten types of actions, and each of them is recorded about one minute. Then, we segment the sequence and label the correct and incorrect sample which amount to 500. After training, the accuracy reaches more than 85%. During the runtime, when the user enters a new action, the user's action will be identified by the trained classifier and the virtual character will perform the corresponding motion. For the unrecognized actions, the framework will wait for the user to re-enter. The classification algorithm (Algorithm 2) is as follows:

5 Constraint of inverse kinematics

Quaternion-based method can transfer the rotations of bones, but it may contain artifacts like foot skating due to missing joints or occlusion. As shown in Fig. 5, when the human body squats down, the feet of target character will be floating using direct retargeting. In order to transfer accurate transformation for joints, we define an offset for each joint

853

Algorithm 2 Motion classification and recognition.

Input:

the motion of human m_s ,

Output:

- the motion of non-humanoid model m_t .
- 1: Determine whether it is a whole body movement or an upper body movement;
- Determine whether it is a movement of two hands or a movement of single hand;
- 3: Determine whether it contains gesture information;
- 4: Identify the user's motion m_s ;
- 5: Mapping to get the target's motion m_t ;
- 6: return m_t ;



Fig. 5 Pose translation constraint. **a** The input posture of real human; **b** the posture of target character without translation constraint; **c** the posture of target character with translation constraint

based on the coordinates of the root joint. We estimate the offsets for all joints by calculating the offset v_{root} of root joint.

$$\mathbf{v}_{\text{root}} = \sigma \left\| \mathbf{j}_{\text{waist}} - \mathbf{j}_{\text{waist}}^{'} \right\|^{2}, \tag{9}$$

$$\mathbf{v}_i = \begin{cases} \mathbf{v}_{\text{root}}, & \text{if } \mathbf{j}_i \text{ belongs to the upper body} \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$
(10)

The offset of waist joint, which is usually regarded as root joint, is calculated by the subtraction of positions between the current frame \mathbf{j}_{waist} and the last frame \mathbf{j}'_{waist} . Here, the offsets of joints in the upper body are consistent with \mathbf{v}_{waist} , while the y-axis position of joint in the lower body is unaffected by the waist offset. σ is used to map the displacement of real person to the virtual scene according to a certain proportion. The value of σ is greatly relevant to the distance between the human and the camera.

Facing the motion of human jumping, we make the wrist bone point position when the human body is standing still as a demarcation point \mathbf{p}_{wrist} . When the human jumps, the position of the wrist joint will be higher than \mathbf{p}_{wrist} . At this time, we add this offset constraint to the whole body of the model. The position of the wrist joint will be lower than \mathbf{p}_{wrist} when the human squats down. At this point, we only add this constraint to the upper body of the model. The motion retargeting results of the human body jumping is shown in Fig. 6.



Fig. 6 Motion retargeting results when the human jumps

Occlusion may appear and give rise to wrong animation. In order to reduce the artifacts, the posture of character model is calibrated by IK. Given the original motion of human m_s , our aim is to derive a set of constraints to refine the character. In order to be more coherent with the motion, the IK constraints are placed on limbs of the target character, which are interactively specified. Here we consider 5 IK chains, which are reflected in different regions (waist, hands and feet). Combining the constraints of IK, we devise the following algorithm (Algorithm 3) for the motion retargeting process:

5 6 6
Input:
the origin posture $m_s = \{\mathbf{j}_{s_1}, \mathbf{j}_{s_2},, \mathbf{j}_{s_n}\}$ of human,
the posture $m_t = \{\mathbf{j}_{t_1}, \mathbf{j}_{t_2},, \mathbf{j}_{t_n}\}$ of target model,
the skeleton structure dictionary E .
Output:
a new posture m'_t for the target model.
1: Match joints and get bones S from m_s , T from m_t ;
2: if target model is a non-humanoid model then
3: Recognize human motion based on classifier
4: Get the motion data m'_t of target model according to define
mappings
5: else {target model is a humanoid model}
6: for all $(S_i, T_i) \in E$ do
7: Calculate the rotation axis \mathbf{c}_i and rotation angle θ_i ;
8: $\mathbf{q}_i = (\mathbf{c}_i, \theta_i);$
9: $\mathbf{j}_{t_i} = Rotate(\mathbf{j}_{t_i}, \mathbf{q}_i);$
10: $\mathbf{j}_{t_i}' = \mathbf{j}_{t_i} + \mathbf{v}_i;$
11: $m'_{t} = m'_{t} \cup \mathbf{j}'_{t};$
12: end for
13: end if
14: Calibration the posture m'_t by IK
15: return m'_{t} ;

6 Experimental and application

Our framework is implemented in Unity and C#. All the experiments are tested on a desktop with Intel[®] CoreTM i7-6700 CPU (3.40 GHz), 32GB RAM, NVIDIA GeForce GTX 1080 and Microsoft Kinect 2.0.

The experiments are divided into two groups: single user and multiple users. We select several CG models from [2] which include humanoid and non-humanoid characters. We test our framework mainly in four scenarios:

- Single user performs actions in front of the camera.
- Single user turns around or performs actions on his/her side.
- Single user performs physical exercise with equipments.
- Multiple user performs actions in front of the camera.

6.1 Single user motion retargeting

In this section, we evaluate our method on the task of single person motion retargeting, simulating the movement of human body as new motion frames are received. From the first row to the fifth row in Fig. 7, we demonstrate animation results of different characters with the captured human motion data, which fully demonstrates the robustness of our method. Each of these models has 20 joint points, which are connected to 17 groups of bones, but their bone length and shape size are different from each other. We select eight different actions including leg lifting, turning around and some actions with occlusion, such as squating. These actions are well retargeted from real human to virtual characters. The sixth row and seventh row in Fig. 7 show the animation results of non-humanoid characters. For example, the action of lifting leg is retargeted to the deer leg, and the swing of the hand is transferred to the wings of the butterfly. For some complex actions, such as walking and squatting, we identify them through the classifier and then drive the model to make the corresponding movement. Compared with the traditional methods, which use no occlusion action or the processed data



Fig. 7 The retargeting results of different characters from the motion captured by Kinect



Fig. 8 The comparison of side-view result

as input, we can directly deal with the motion data captured by Kinect and improve the performance when occlusions take place.

We also compare the action of turning around. As illustrated in Fig. 8, three cases are enumerated, including "Only Retarget," "Retarget + Filter" and "Retarget + Filter + IK." When the filter and IK are disabled, all results are incorrect. Only using the direction vector of hip bone to calculate the rotation angle of human, it cannot identify the whole process of the human body turn. When the body is over 30° sideways the camera, the model will not track properly. And when the joint point of human body is blocked, the character model will have deformation action and the method is not available. After the filter is enabled, the rotation information of human body can be well captured. The rotation angle of character model is consistent with the angle of the real human rotation, but some poses are incorrect, e.g., squatting. If both the filter and IK are enabled, the transferred poses are the best. The posture solved by IK can be used to calibrate the pose of character model, which makes the retargeting of human sideways movement be more accurate and stable.

To handle some special cases, we need to modify the influence factors λ_s , λ_h and λ_s . Our system uses the shoulder spacing and hip spacing to predict the turning angle. When facing the user only turns the upper body, we need to adjust the influence factor of the hip spacing λ_h to 1, λ_s and λ_f to 0. So when the upper only turns around the upper body but remains the lower body, we can still reproduce the movement of the human body. The results are shown in Fig. 9.

The results of our experiments are subject to the accuracy of the input joint point information. We tested our methods with severe occlusion and compared it with the untreated sit-



Fig. 9 The results of only turning around the upper body



Fig. 10 The comparison of our method with direct retargeted animation

uation. For the occlusion problem, we mainly consider the situation when the human is side to camera. When the human is face to camera or back to camera, we will directly use the data captured by Kinect to retarget and cannot correct the error caused by keypoints missing. We tested our method with severe occlusion and compared the results of ours with direct retargeted animation. Figure 10 shows the experimental results with occlusion at three different angles. When the human is face or back to camera, Kinect can estimate the bone structure of human and the final result only occurred in some areas. However, the joint information predicted by Kinect has a very serious deformation in the case of sideways. For this reason, we introduced filter and IK method to keep the skeleton in a reasonable posture. In addition, we tested our method on a group of postures appeared in existing work [16]. The results are demonstrated in Fig. 11.

6.2 Multi-user motion retargeting

Through parallel processing, we can achieve motion retargeting from multi-person video as shown in Fig. 12 When more than one person appears in front of the camera, we handle multi-person motion data in multi-thread to ensure the realtime performance of the system. In Table 1, we compare the time consumption of our algorithm in different situations.

6.3 Quantitative evaluation

Moreover, we record the motion curves of human and virtual character during the motion. The curves contain values of the joint information. In Fig. 13, we illustrate the motion curves of different joints (left hand and left foot). The red curve is



Fig. 11 The comparison of our method with existing method [16]. From \mathbf{a} -i, the upper is the input motion and the retargeting result from [16], and the lower is the input motion and the retargeting result of our method



Fig. 12 Multi-person motion retargeting

 Table 1
 Computation efficiency of our method

FPS	Computation
60	CPU single thread
50	CPU multi threads
	FPS 60 50

6.4 Applications

a simple e the human motion captured by OptiTrack [21], the green one is captured by Kinect 2.0 and the blue curve is the motion of target model animated by our method. After the comparison, we believe that our method can extract the motion which is close to the real one.

Our framework provides user a real-time interface to control virtual characters and interact with virtual environments. As a simple example, we developed "Personal Trainer" shown in Fig. 14. This application is used to help users feel immersive and interesting when they are doing exercises. Also, this application can provide useful instructions of the fitness postures. In this application, a single Kinect camera is employed to capture the motion. We predefine the reference motion and



Fig. 13 Quantitative evaluation of motion curves recorded by three different approaches. **a** The motion curve of left hand joint; **b** the motion curve of right foot joint

animate the virtual character to instruct users. Users can follow the instructions and create their own animations.

7 Conclusions and limitations

In this paper, we develop a real-time framework for producing virtual character animation using single Microsoft Kinect device. Our framework is capable of handling complex human motions like turning around as well as self-occlusion. The success of our framework is hinging upon the following key factors: (1) The adaptive filter is capable of estimating the rotation of human body and recovering the motion when turning around in 360° ; (2) the quaternion-based method for motion retargeting is helpful for reducing vibration and



Fig. 14 An example of interactive application. **a** User chooses the type of fitness; **b** user does exercises according to the reference motion

noises; and (3) the motion refinement adjusts the posture of the entire body through IK constraints which enhance the motion transfer result in side view.

There are several limitations of our method. First, our method is dependent on the accuracy of the input joint information. The missing keypoints in the capture stage cannot be restored in our final result. Second, the influence factors are set manually at present. It is practical for specific scenes, but we believe it may increase the estimation results when the settings are modified dynamically. Third, although IK constraints improve the final motion, it seems unnatural for some cases. In the future, we plan to employ data-driven approach to enhance the animation. In the meantime, immersive devices such as HMD are considered to be coupled with our technique in its application.

Acknowledgements We thank reviewers for their valuable comments and suggestions. We also thank Yansheng Fu, Jian Su, Zhongwang Zhang, Ying Xin for the experiments.

Funding This research has been supported by National Key R&D Program of China (Grant No. 2017YFB1002602), National Natural Science Foundation of China (Grant Nos. 61532002, 61672149, 61872020, 61872347), NSF IIS-1715985, NSF IIS-1812606.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

- Abdulmassih, M., Yoo, I., Benes, B.: Motion style retargeting to characters with different morphologies. Comput. Graph. Forum 36(6), 86–99 (2017)
- 2. Adobes mixamo. https://www.mixamo.com/

- Baciu, G., Iu, B.K.C.: Motion retargeting in the presence of topological variations. Comput. Anim. Virtual Worlds 17(1), 41–57 (2006)
- Bogo, F., Black, M.J., Loper, M., Romero, J.: Detailed fullbody reconstructions of moving people from monocular RGB-D sequences. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, pp. 2300–2308 (2015)
- Celikcan, U., Yaz, I.O., Capin, T.: Example-based retargeting of human motion to arbitrary mesh models. Comput. Graph. Forum 34(1), 216–227 (2015)
- Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E., Thelen, D.G.: Opensim: open-source software to create and analyze dynamic simulations of movement. IEEE Trans. Biomed. Eng. 54(11), 1940–1950 (2007)
- Fang, A.C., Pollard, N.S.: Efficient synthesis of physically valid human motion. ACM Trans. Graph. 22(3), 417–426 (2003)
- Gleicher, M.: Retargeting motion to new characters. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, pp. 33–42 (1998)
- Grochow, K., Martin, S.L., Hertzmann, A., Popovic, Z.: Stylebased inverse kinematics. ACM Trans. Graph. 23(3), 522–531 (2004)
- Guo, S., Southern, R., Chang, J., Greer, D., Zhang, J.: Adaptive motion synthesis for virtual characters: a survey. Vis. Comput. 31(5), 497–512 (2015)
- Hecker, C., Raabe, B., Enslow, R.W., DeWeese, J., Maynard, J., van Prooijen, K.: Real-time motion retargeting to highly varied user-created morphologies. ACM Trans. Graph. 27(3), 27:1–27:11 (2008)
- 12. Kinect windows app development. https://developer.microsoft. com/en-us/windows/kinect
- Kwon, T., Hodgins, J.K.: Momentum-mapped inverted pendulum models for controlling dynamic human motions. ACM Trans. Graph. 36(1), 10:1–10:14 (2017)
- Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K., Pollard, N.S.: Interactive control of avatars animated with human motion data. ACM Trans. Graph. 21(3), 491–500 (2002)
- Liu, Z., Zhou, L., Leung, H., Shum, H.P.H.: Kinect posture reconstruction based on a local mixture of Gaussian process models. IEEE Trans. Vis. Comput. Graph. 22(11), 2437–2450 (2016)
- Liu, Z., Zhou, L., Leung, H., Shum, H.P.H.: Kinect posture reconstruction based on a local mixture of Gaussian process models. IEEE Trans. Vis. Comput. Graph. 22(11), 2437–2450 (2016)
- Lv, X., Chai, J., Xia, S.: Data-driven inverse dynamics for human motion. ACM Trans. Graph. 35(6), 163:1–163:12 (2016)
- Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H., Xu, W., Casas, D., Theobalt, C.: Vnect: real-time 3d human pose estimation with a single RGB camera. ACM Trans. Graph. 36(4), 44:1–44:14 (2017)
- Monzani, J., Baerlocher, P., Boulic, R., Thalmann, D.: Using an intermediate skeleton and inverse kinematics for motion retargeting. Comput. Graph. Forum 19(3), 11–19 (2000)
- Mousas, C.: Performance-driven dance motion control of a virtual partner character. In: 2018 IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018, pp. 57–64 (2018)
- 21. Optitrack motion capture. https://www.optitrack.com/motioncapture-virtual-reality/
- 22. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. 9(1), 62–66 (1979)
- Park, M.J., Shin, S.Y.: Example-based motion cloning. J. Vis. Comput. Anim. 15(3–4), 245–257 (2004)
- Popovic, Z., Witkin, A.P.: Physically based motion transformation. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999, pp. 11–20 (1999)

- Rhodin, H., Tompkin, J., Kim, K.I., Varanasi, K., Seidel, H., Theobalt, C.: Interactive motion mapping for real-time character control. Comput. Graph. Forum 33(2), 273–282 (2014)
- Rhodin, H., Tompkin, J., Kim, K.I., de Aguiar, E., Pfister, H., Seidel, H., Theobalt, C.: Generalizing wave gestures from sparse examples for real-time character control. ACM Trans. Graph. 34(6), 181:1–181:12 (2015)
- Roth, D., Lugrin, J., Buser, J., Bente, G., Fuhrmann, A., Latoschik, M.E.: A simplified inverse kinematic approach for embodied VR applications. In: 2016 IEEE Virtual Reality, VR 2016, pp. 275–276 (2016)
- Villegas, R., Yang, J., Ceylan, D., Lee, H.: Neural kinematic networks for unsupervised motion retargetting. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, pp. 8639–8648 (2018)
- Wang, K., Zhang, G., Xia, S.: Templateless non-rigid reconstruction and motion tracking with a single RGB-D camera. IEEE Trans. Image Process. 26(12), 5966–5979 (2017)
- Yamane, K., Ariki, Y., Hodgins, J.K.: Animating non-humanoid characters with human motion data. In: Proceedings of the 2010 Eurographics/ACM SIGGRAPH Symposium on Computer Animation, SCA 2010, pp. 169–178 (2010)
- Ye, M., Shen, Y., Du, C., Pan, Z., Yang, R.: Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. IEEE Trans. Pattern Anal. Mach. Intell. 38(8), 1517– 1532 (2016)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Ning Kang is a master student in Beihang University. From 2017, he studied in the State Key Laboratory of Virtual Reality Technology and Systems in China. At present, he is studying at the Beihang Qingdao Research Institute. His research interests include 3D animation and human pose estimation.



Junxuan Bai is a Ph.D. student in Beihang University. From 2012, he studied in the State Key Laboratory of Virtual Reality Technology and Systems in China. He received his MS in Computer Science from Beihang University in 2015. His research interests include computer animation, virtual surgery and 3D visualization.



Junjun Pan received both B.Sc. and M.Sc. degree in School of Computer Science, Northwestern Polytechnical University, China. In 2006, he studied in National Centre for Computer Animation (NCCA), Bournemouth University, UK, as Ph.D. candidate with full scholarship. In 2010, he received Ph.D. degree and worked in NCCA as Postdoctoral Research Fellow. From 2012 to 2013, he worked as a Research Associate in Center for Modeling, Simulation and Imaging in Medicine,

Rensselaer Polytechnic Institute, USA. In Nov 2013, he was appointed as Associate Professor in School of Computer Science, Beihang University, China. His research interests include virtual surgery and computer animation.



Hong Qin is a full professor of Computer Science in the Department of Computer Science at Stony Brook University (SUNY). He received his B.S. and his M.S. in Computer Science from Peking University, China. He received his Ph.D. in Computer Science from the University of Toronto. Currently, he serves as an associate editor for the Visual Computer, Graphical Models, and Journal of Computer Science and Technology. His research interests include geometric and solid modeling,

graphics, physics-based modeling and simulation, computer-aided geometric design, human-computer interaction, visualization and scientific computing.