ANIMATION OF 3D CHARACTERS FROM SINGLE DEPTH CAMERA

Mian Ma[†]*, Feng Xu*[*]*, Yebin Liu*[*]

[†] Department of Computer Science
Beijing University of Posts and Telecommunications

[*] Department of Automation
Tsinghua University

ABSTRACT

As the development of depth acquisition techniques, in recent years, depth cameras achieve to obtain depth maps in real time. In this work, by using a single depth camera, we propose a method to transfer 3D motion of a performer to another character (avatars or other human characters). We capture the motion of a performer by a Kinect camera, which outputs the 3D positions of 15 joints of the performer for each frame. Even though the joints positions are quite noisy and with errors in some frames, our method achieves to calculate joints' rotations with spatial and temporal consistency and transfer the motion to a target skeleton. By using a skinning technique, the captured skeleton motion drives the target 3D model which performs the same motion with the captured performer.

Index Terms— RotationMatrix, Animation, Depth camera, Deformation.

1. INTRODUCTION

In recent years, depth cameras have become a widely used sensor that captures depth images at real time frame rates. Through a single depth camera, the character can be easily distinguished from the background, following which all the joints' positions of the character can be estimated by analyzing depth data [6, 2]. At the same time, 3D modeling is becoming much easier than before. User-friendly systems such as Teddy [4] and Cosmic Blobs have made the creation of 3D characters accessible to novices and children.

In this paper, we get the three dimensional positions of all the joints at real time provided by Kinect SDK, then compute the rotation matrixes of all the joints for the target skeleton with spatial and temporal consistency. Finally, we use the method presented by [3] to animate the target character model.

In skeleton based character animation, the rotation of each joint has a distinguished degree of freedom, so the joints' rotation angles relative to x, y and z axis must be calculated in order to drive a mesh model by the skeleton. However, what we get from the SDK are merely the joints' positions without rotation in each frame. What's more, the position data is quite noisy and with errors, so the length of the bones may vary between every two consecutive frames. Therefore, we ignore the length of the bones and concentrate on estimating the rotation angle from the obtained positions. This seems impossible because it's a problem of solving an undetermined equation. Three rotation angles relative to x, y and z axis are needed to attach the skin to the target model accurately. But we can't calculate all of them just from the corresponding positions get from the SDK. In this work, we propose a simple and effective method which makes impossible to possible but sacrifices some accuracy to estimate rotation matrixes which describe the motion of the captured character. Then the skeleton motion is attached to the mesh surface of the target character, allowing skeleton motion data to animate the character. In general, our proposed method achieves to transfer the captured character's motion to a target model character online using just one depth camera and one mesh model.

Our system consists of three main steps: data acquisition, skeleton transformation and skin attachment. Data acquisition gets the position of each joint at each frame. Skeleton transformation transforms the captured character's skeleton to our target model skeleton. Skin attachment uses the target skeleton to deform our model mesh and generate the retarget model which has the same pose as the captured character. The pipeline is shown in Fig.1. The initial model we used comes from [8].

2. MATERIAL AND METHODS

2.1. Estimate Rotation Matrix

In order to compute the real rotation matrix, we need to get all the joints' rotation angles relative to x, y and z axis. However, merely the joints' positions of every frame is given. Considering that all bones are formed by connecting two joints, we can easily compute the joints' direction vector through the subtraction of two connected joints. Now we want to compute all the joints' rotation matrix $\boldsymbol{A}$ satisfying $y = \boldsymbol{A}x$, which is a problem of solving an undetermined equation. In order to solve it, some additional constraints need to be added. We assume the motion of the character is coherent and the current rotation of each joint is an addition of the rotations before. Based on these assumptions, we compute the rotation matrix$\mathcal{M}(\hat{\boldsymbol{v}}, \theta)$ through equation(1) [1] using a single rotation

$$\mathcal{M}(\hat{v},\theta) = \begin{bmatrix} \cos\theta + (1-\cos\theta)x^2 & (1-\cos\theta)xy - (\sin\theta)z & (1-\cos\theta)xz + (\sin\theta)y \\ (1-\cos\theta)yx + (\sin\theta)z & \cos\theta + (1-\cos\theta)y^2 & (1-\cos\theta)yz - (\sin\theta)x \\ (1-\cos\theta)zx - (\sin\theta)y & (1-\cos\theta)zy + (\sin\theta)x & \cos\theta + (1-\cos\theta)z^2 \end{bmatrix} \quad (1)$$

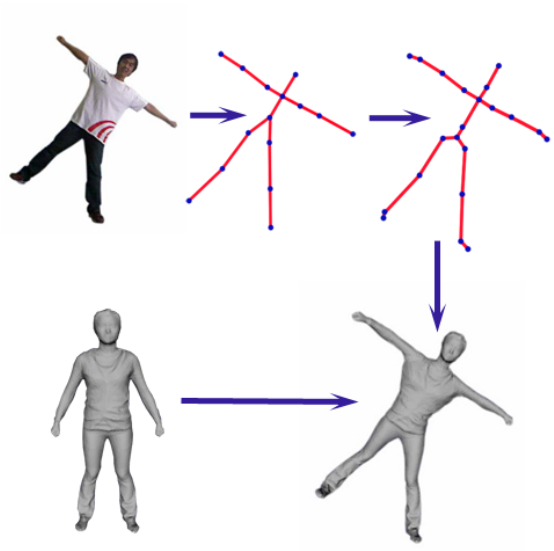


Fig. 1. Pipeline of the system. Left of the first line: input depth data(we showed the RGB image). Middle of the first line: the skeleton we get from Kinect SDK. Right of the first line: the transformed skeleton obtained by our method. Left of the second line: input standard mesh model. Right of the second line: deformed mesh obtained by using the transformed rotation matrix of the skeleton and the standard mesh model.

angle θ and a unit vector $\hat{\boldsymbol{v}} = (x, y, z)$ which perpendicular to $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ showed in Fig.2.

Then the final rotation matrix $\mathcal{R}(j)$ is computed as:

$$\mathcal{R}(j) = \prod_{i=1}^{n} \mathcal{M}(\hat{v},\theta) \quad (2)$$

2.2. Skin Attachment

There are several techniques to deform the mesh and attach the skin. We use the most widespread method – standard linear blend skinning (LBS) which described in [3, 7] to complish this work.

Assume that $\mathbf{v}_j$ is the position of vertex j, $\mathbf{T}^i$ is the transformation of the i^{th} bone, and w_j^i is the weight of the i^{th} bone for vertex j, LBS gives the position of the transformed vertex j as $\sum_i w_j^i \mathbf{T}^i(\mathbf{v}_j)$. The attachment problem is to find bone weights $\mathbf{w}^i$ for the vertices—that is to determing how much each bone transform affects each vertex.

In order to find good weights $\mathbf{w}^i$, there are several properties we must concentrate on. First of all, the weights must

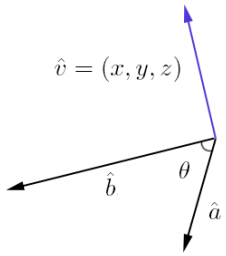


Fig. 2. Example of the rotation. $\hat{\boldsymbol{a}}$ is the vector of one bone of the last frame. $\hat{\boldsymbol{b}}$ is the vector of the same bone of the current frame.

depend on the transformed skeleton. Second, for the results to look good, the weights need to vary smoothly along the surface. Finally, to avoid folding artifacts, the width of a transition between two bones meeting at a joint should be roughly proportional to the distance from the joint to the surface. So we use the analogy to heat equilibrium to find the weights [3]. The principle of the theory is showed in Fig3.

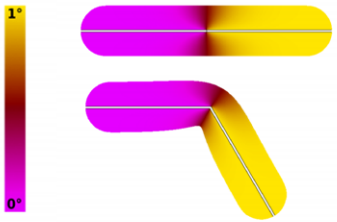


Fig. 3. Top: heat equilibrium for two bones. Bottom: the result of rotating the right bone with the heat-based attachment

2.3. Improvement

The transformed pose of the mesh is basically consistent with the model mesh through our method. But only the rotation between the connected bones is considered, and the anatomy of a human is not taken into consideration. The result looks a little artificial and has some wrong or illogical frames. We mainly confronted three problems: wrong frames, shake between frames and illogical frames. Wrong frames are generated because we use the totally wrong data captured from the Kinect SDK to calculate the positions of all the joints, meanwhile, to deform our mesh. The shake between frames appeared because the topology and the length of the corresponding bones between the captured character and the mesh model

are different, so the retargeted meshes are always sliding on the floor. The illogical frames refer to the frames which are not correspond with our physical structure such as the hand extending into the body. It is also mainly due to the different length between the captured character and the target character.

In order to eliminate the outliers(wrong frames) and make the retarget sequence smoothly, we propose a method with following steps. The first stage is to remove the outliers (The experiments reveal that the simplest method which will be introduced immediately is enough). Considering the fact that the number of outliers is few and they do not exist next to each other, we divide the long skeleton sequence into several short sequences and compute the average variance of every shot sequence. Then we set a variance threshold σ. All the shot sequences whose variance are larger than the threshold σ will be treat as an outliers. Base on this rule, all the outliers can be found and then eliminated. Secondly, we concentrate on eliminating the shake. We define $\{\boldsymbol{\theta}_j\}$ as the j^{th} joint's observed rotation angle sequence, and $\{\hat{\boldsymbol{\theta}}_j\}$ as the optimized rotation angle sequence we want to get. Then we find the $\{\hat{\boldsymbol{\theta}}_j\}$ sequence by equation3.

$$\arg\min_{\hat{\boldsymbol{\theta}}_j} \sum_{i=0}^{n} \|\theta_j^i - \hat{\theta}_j^i\|^2 + \triangle\|\hat{\theta}_j^i\|^2 \qquad (3)$$

We optimized the angle sequence separately for every joint. The minimization of $\|\boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j\|^2$ determine that the calculated $\{\hat{\boldsymbol{\theta}}_j\}$ sequence will approximate the observed value $\{\boldsymbol{\theta}_j\}$. The minimization of $\triangle\|\hat{\boldsymbol{\theta}}_j\|^2$ ensure the $\{\hat{\boldsymbol{\theta}}_j\}$ sequence is as smooth as possible. Then we solve the optimization problem3 to make the motion of captured character more smoothly. Because the optimization of the second order gradient can be converted into ℓ_2 [9, 5] optimization problem, we can easily get the optimized $\{\hat{\boldsymbol{\theta}}_j\}$ sequence of every joint through least square method.

3. RESULT AND DISCUSSION

In the process of calculation, only the depth data is used to compute the positions of all the joints. For easier understanding, we just show the RGB images and the deformed mesh. Fig.4 shows the final result in which the retarget motion is the same with the motion of the performer moving in front of the Kinect camera. Fig.5 shows some ordinary pose and the retarget result. Fig.6 demonstrates several results in more complex situations.

Real time result without the improvement at several moments is shown in Fig.7. From Fig.7 we can see the deformed mesh's pose is basically the same as our captured character, but errors still exists (denoted in red circles). In the first row we see the right hand of the mesh has extended into the body because the length of the model's arm is longer than our captured character. In the second row is the case where the

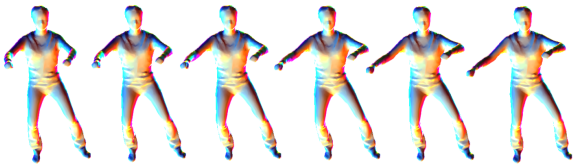
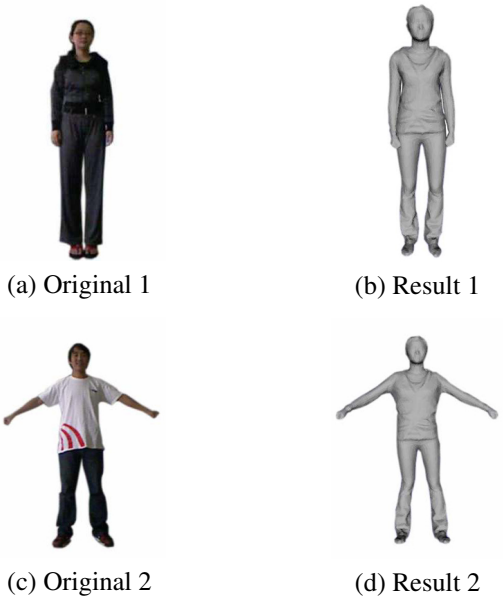


Fig. 4. The motion stereo.

mesh's leg has crossed abnormally because the skeleton we get from the SDK is totally wrong at this frame. Therefore, we did some refinement as described in section 2.3 to eliminate these errors. After the elimination, the result is immune to the artifacts, but it is no longer that similar with the captured character's pose because of error accumulation in the process of smoothing.



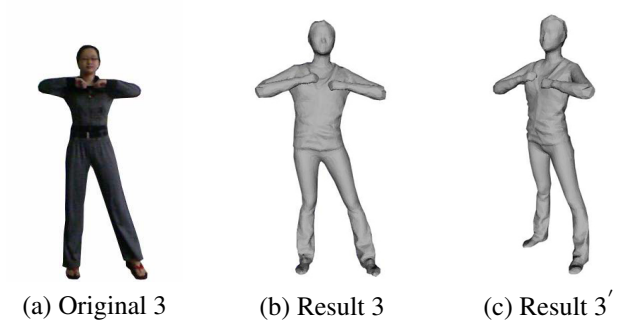
(a) Original 1 (b) Result 1

(c) Original 2 (d) Result 2

Fig. 5. Left column: the original input picture. Right column: the synthetic model mesh with the same angle of view.

4. CONCLUSION AND FUTURE WORK

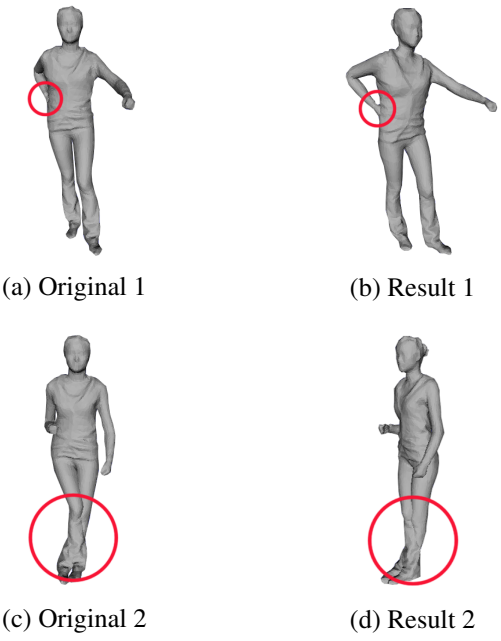
In this paper we propose a method to animate a 3D character model online from noisy and erroneous joint position information obtained by a single depth camera. It can be used in 3D immersive sense games, computer interaction and some other areas. The online result shows 92% of the frames are deformed correctly, but still needs to be improved. Then we further present some methods to ameliorate the retarget result. The shake and illogical frames have been eliminated, but as the accumulating of the error, the deformed model's pose is not that similar to the captured character. In the future, we



(a) Original 3 (b) Result 3 (c) Result 3′

(d) Original 4 (e) Result 4 (f) Result 4′

Fig. 6. Left column: the original input picture. Middle column: the synthetic model mesh with the same angle of view. Right column: the synthetic model mesh with different angle of view.



(a) Original 1 (b) Result 1

(c) Original 2 (d) Result 2

Fig. 7. The wrong frames at different angle of view.

will propose some methods which can not only refine false frames but also be able to perform in real time. In this paper, the skinning method we used behaves well most of the time, but in some special cases, unrealistic presentation may appear. For example when the character lifting his arm too high, we will find the vertexes around armpit deform unrealistic because it does not really correspond with the heat equilibrium model so much. In the future, we will adopt more effective methods to attach the skin to make the model lifelike and look veritably.

5. ACKNOWLEDGEMENT

This work was supported by the Project of NSFC (No. 61035002 & 61073072 &61100120).

6. REFERENCES

[1] wikipedia. http://en.wikipedia.org/wiki/Rotation_matrix, Nov. 2011.

[2] A. Baak, M. Müller, G. Bharaj, H.P. Seidel, and C. Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera.

[3] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics (TOG)*, 26(3):72–es, 2007.

[4] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2007 courses*, pages 21–es. ACM, 2007.

[5] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *ACM Transactions on Graphics (TOG)*, 24(3):536–543, 2005.

[6] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR11), Colorado Springs, USA*, 2011.

[7] R.Y. Wang, K. Pulli, and J. Popović. Real-time enveloping with rotational regression. In *ACM SIGGRAPH 2007 papers*, pages 73–es. ACM, 2007.

[8] F. Xu, Y. Liu, C. Stoll, J. Tompkin, G. Bharaj, Q. Dai, H.P. Seidel, J. Kautz, and C. Theobalt. Video-based characters–creating new human performances from a multi-view video database.

[9] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for l1-minimization with applications to compressed sensing. *SIAM J. Imaging Sci*, 1(1):143–168, 2008.