A Data-Driven Approach for 3D Human Body Pose Reconstruction from a Kinect Sensor

Li Yao1,*, Xiaodong Peng1, Yining Guo1, Hongjie Ni1, Yan Wan1 and Cairong Yan1

1School of Computer Science and Technology, Donghua University, No.2999 Renmin North Road, Songjiang District, Shanghai, China

*yaoli@dhu.edu.cn

Abstract: In the study of virtual fitting techniques, human body modeling has always occupied a very important position. Whether a model that is roughly consistent with users can be established has a direct impact on the fitting experience of users. Based on this, we propose an automatic human body pose registration algorithm which can efficiently construct a posed model using priori data and a single Kinect sensor, and provide a good foundation for the later shape registration. Because of the complexity of the human body, there are so many methods that just reconstruct a human body model but no rigged animation skeleton inside. To solve this, we use SMPL which is a recently published statistical body model to fit 3D joints acquired by a Kinect sensor. And finally we project the posed model to the corresponding person in the color image to improve the fitting experience. Our experiments show that the speed and the estimation error of the algorithm are within the tolerance of virtual fitting.

1 Introduction

3D human body modeling is a key technology in the virtual fitting system, which plays an important role in the simulation of the 3D garment in the later period. The main difficulties in virtual fitting are the following two aspects: First, we need to ensure that the physical parameters of the virtual body are corresponding to the users in reality. Second, we need to guarantee the matching, authenticity and real-time of the simulated clothing. Based on the requirement of the former, we mainly measure human modeling methods.

Human modeling can be classified according to two criteria, one is based on model prior and the other is Free-form like KinectFusion[1]. The former can adapt to more complex poses of the human body, and only needs a small amount of data to get a rough model. Although the model is lacking in accuracy, it has a higher efficiency. Therefore, this method is more suitable for a virtual fitting system that requires less accuracy but more efficiency. However, the latter usually requires the user to stand still for a long time. Besides, it is no rigged animation skeleton inside. So we focus on measuring model-based human modeling methods.

Among the early work in model-based human body modeling, the SCAPE[2] model has been widely used in 3D human shape and pose modeling. There is a lot of work on estimating human body from multi-camera images, a single image and range data. Considering the convenient configuration of the equipment, we focus on the solutions which only use a single equipment. In some inferring from a single image, SCAPE is fit to the image silhouettes, so they have higher requirements for silhouettes to evaluate pose accuracy. Besides, such solutions are very easy to be ambiguous when inferring 3D from
2D because of the loss of depth information. For range data, there are some solutions that have good performance. However, these solutions based on SCAPE are not compatible with existing rendering engines. In [3] which makes a comparison of six typical parametric models of 3D body which are SCAPE, BlendSCAPE[4], Dyna[5], S-SCAPE[6], SMPL[7] and RealtimeSCAPE[8] respectively, we can see that SMPL has better performance both in speed and accuracy than SCAPE. And more importantly, SMPL is compatible with existing rendering engines. The above mentioned are the reasons why we decide to use a single Kinect and SMPL to infer the human pose.

The specific implementation steps of our algorithm are shown in figure 1: First, we need to acquire skeletal points from a Kinect sensor. It requires the users with an arbitrary pose to stand in front of a Kinect sensor. Then, we need to transform the vertices of the SMPL model and the skeletal points to the same coordinate system using the ICP algorithm. And next we iteratively fit the SMPL model to the Kinect skeletal points by building an objective function. Finally, we use the intrinsic and extrinsic parameters acquired by camera calibration to project the posed model to the corresponding color image so as to achieve the effect of human-computer interaction.

Figure 1. The pipeline of human pose estimation.

2. Related Work
We have conducted research on model-based human modeling methods in recent years. In the survey of Cheng Z Q, we can see that there are 18 applications using SCAPE model to reconstruct 3D body. However, there are only 2 applications using SMPL. As we all know, SMPL model is proposed in 2015, so there are few applications using SMPL model. We mainly introduce the related work in these two areas as follows.

2.1 SCAPE-based human body modeling
Weiss et al. [9] uses four different views of the depth information and RGB images acquired by Kinect sensor to calculate the shape and pose parameters of the body, and then use these to drive SCAPE model to get the 3D model of target human body. They firstly assume an initial pose, and then correct the pose in the following frame after the fitting of contour. Their work provides a solution for 3D body modeling indoors, but their solution needs 1 hour to complete the complex optimization and cannot handle the condition of people wearing clothes and large-scale motion.

Xu et al. [10] use a depth map, an RGB image and a silhouette image in current frame to infer pose parameters in next frame by using [11]. Their solution is similar to Weiss et al. but they can acquire a more accurate model with large-scale motion. Besides, their method which mitigates the clothes effect
is robust, accurate and effective.

Cheng et al. [12] propose a solution which can reconstruct human body based on sparse key points. They use boosting tree regression framework to train range data and learn a regression function. Then they can get the corresponding key points to fit SCAPE and get the final model in a second.

Zeng et al. [13] assume that the human body is quasi-rigid during slight movements, and propose a model-to-part method to register deformed images to align new objects, and then use the Poisson reconstruction method to reconstruct the surface of the human body. Their solution is limited in human pose.

2.2 SMPL-based human body modeling
Bogo et al. [14] utilize a convolutional neural network (CNN) to predict 2D joints of a color image, and then they fit these joints to the projected 3D body joints of SMPL model by minimizing a simple objective function. Their solution only needs an image to estimate model. However, their pose estimation sometimes is not as accurate.

Marcard et al. [15] are based on inertial measurement units (IMUs) which are one kind of body-worn sensors to deform SMPL model. Their solution has real-time performance but these setups are expensive and require tedious pre-calibration. Therefore, this solution is not suitable for home 3D body scans.

Alldieck et al. [16] can get accurate 3D body models and texture of some people with arbitrary pose and shape from a monocular video. Their solution which extends Bogo et al. by incorporating a silhouette term in pose estimation is more accurate than Bogo et al.

3 Method

3.1 SMPL
SMPL is a body model proposed in recent years which represents pose and shape dependent deformations in an efficient linear formulation and it defines male and female models respectively. SMPL is a parameterized model of naked humans that takes 10 shape and 72 pose parameters and returns a mesh \( M \) with \( N = 6890 \) vertices and \( K = 23 \) joints. Shape parameters \( \beta \) and pose parameters \( \theta \) are applied to a base template \( T_\mu \), which is mean template in the training scans. The SMPL model formulations are as follows:

\[
M(\beta, \theta) = W(T(\beta, \theta), J(\beta), \theta, W) \tag{1}
\]

\[
T(\beta, \theta) = T_\mu + B_s(\beta) + B_p(\theta) \tag{2}
\]

Where shape-dependent deformations \( B_s(\beta) \) and pose-dependent deformations \( B_p(\theta) \) represent the offsets from the template model, some coefficients in these deformations can be learned from registered training meshes. After we add \( B_s(\beta) \) and \( B_p(\theta) \) to the template model, we can get the shape of the body.

Pose parameters \( \theta \) denote the axis-angle of the rotation between two relative joints. \( J(\beta) \) is a function about \( \beta \) that can predict the 3D skeleton joint locations. When we get joint locations, we can use a global rigid transformation \( R_\theta(J(\beta)) \) induced by \( \theta \) to get posed 3D joints. A more detailed analysis of how to get global rigid transformation is presented in section 3.2. \( W \) is a LBS function which takes vertices in the rest pose, \( T_\mu \) joint location, \( J \), a pose, \( \theta \), and the blend weights, \( W \), and returns the posed vertices. We can also use DQBS skinning method to replace LBS. After we use the skinning function \( W \), we can get the pose of the body.

3.2 Change \( \theta \) to a global rigid transformation
As we all know, we can transform the axis angle for every two joints to a local rotation matrix by
using the Rodrigues formula. When we get the local rotation between two relative joints and their location, we can use the transitivity of coordinates to calculate the global rotation transformation and translation transformation.

![Figure 2](image)

**Figure 2.** The transformation of coordinates. (a) The same point in two coordinates. (b) The same point in three coordinates.

As shown in figure 2 (a), $A^P$ is the position of a point in coordinate A and $B^P$ is the position of the same point in coordinate B. We can get the $A^P$ by a transformation as follows:

$$
\begin{bmatrix}
A^P \\
1
\end{bmatrix} =
\begin{bmatrix}
A^R & A^P_{borg} \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
B^P \\
1
\end{bmatrix}
$$

(3)

In figure 2(b), we can get the $B^P$ by using the equation (4) to transform $C^P$ to coordinate B and get the $A^P$ by equation (5) to transform $C^P$ to coordinate A.

$$
\begin{bmatrix}
B^P \\
1
\end{bmatrix} =
\begin{bmatrix}
B^R & B^P_{borg} \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
C^P \\
1
\end{bmatrix}
$$

(4)

$$
\begin{bmatrix}
A^P \\
1
\end{bmatrix} =
\begin{bmatrix}
A^R & A^P_{borg} \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
C^P \\
1
\end{bmatrix}
$$

(5)

3.3 Transform the Coordinate of Kinect to SMPL

Because of the difference of coordinates between the joints of SMPL and the joints of Kinect, we need to transform the joints of Kinect and the vertices of SMPL to the same coordinate. One of the most common methods to align two point sets is ICP. ICP is an algorithm which can find the optimal rotation and translation between two sets relying on some corresponding points. If these corresponding points are known, this problem will be easy to solve. The solution we will present is from [17].

There are three steps to find the optimal rigid transformation:

1. Find the centroids of both dataset A and B
2. Bring both dataset to the origin then find the optimal rotation R
3. Find the translation T

Finding the centroids of both dataset is very easy. So we focus on how to find the optimal rotation. There is a common way which is called singular value decomposition or SVD to solve this problem. After moving both dataset to the origin, we need to calculate the covariance matrix $H$, which is defined as follows:

$$
H = \sum_{i=1}^{n} (P_i^A - \text{centroid}_A)(P_i^B - \text{centroid}_B)^T
$$

(6)

And then we use equation (8) to get the optimal rotation.

4
\[
[U, S, V] = \text{SVD}(H)
\]  

(7)

Finally we can get the translation by the equation (9).

\[
T = -R \times \text{centroid}_A + \text{centroid}_B
\]

(8)

(9)

So we can transform the coordinate of Kinect to SMPL by the equation (10).

\[
P_{\text{SMPL}} = R P_{\text{Kinect}} + T
\]

(10)

As is shown in figure 3(a), obviously the distance between the above joints which are Kinect joints and the below joints which are SMPL joints is very far. Because of the rigid of torso, we use the joints of the torso as the corresponding points to find \( R \) and \( T \). The result of the transformation is shown in figure 3(b).

**Figure 3.** The comparison of joints about whether using ICP algorithm. (a) The original joints. (b) The result after using ICP algorithm.

### 3.4 Objective Function

We can get joint locations of SMPL through \( \theta \) which is mentioned in section 3.2. Here is some work we need to do for fitting the pose of SMPL model to the Kinect through \( \theta \). First, we can see that Kinect has 21 joints (actually we delete 4 joints on the hand in Kinect because we don’t need these) in figure 4(a) and SMPL model has 24 joints in figure 4(b) and their joints are not one-to-one. So we need to map the joints of Kinect to the SMPL. For example, the index 12 of Kinect joints maps to the index 1 of SMPL joints. There are three joints which are index 6, 13, 14 in SMPL but not in Kinect, so we need to calculate their values and the simplest way is to calculate the average of the points around them.

**Figure 4.** The comparison of skeletons between Kinect and SMPL. (a) The Kinect skeleton. (b) The SMPL skeleton.
In the second, we need to create an objective function to indicate the difference between the corresponding joints and minimize the objective function as far as possible. That is

\[ E(\beta, \theta) = E_j(\beta, \theta, J_{Kinect}) + w_j \sum_{j=1}^{72} \theta_j^2 \] (11)

Our data term penalizes the weighted 3D distance between Kinect joints \(J_{Kinect}\) and corresponding SMPL joints:

\[ E_j(\beta, \theta, J_{Kinect}) = \sum_{j=1}^{72} w_j (R_\theta(J(\beta)_j) - J_{Kinect,j}) \] (12)

And the latter item of equation (11) is a regular item which can prevent overfitting.

### 3.5 Nonlinear optimization

There are two common methods for solving nonlinear optimization problems, one is Gauss-Newton and the other is Levenberg-Marquardt (LM). In Gauss-Newton method, \(J^T J\) may return a singular or an ill-condition which will cause the algorithm not to converge. So we decide to use the latter to optimize our objective function which is more stable than former. LM is a trust region method which means that in the trust region, we can think of the approximate solution as valid. Generally speaking, this is due to that the Taylor expansion can only have a good approximation near the deployment point. A good method to determine the range of the trust region is based on the difference between our approximate solution and actual solution like equation (13).

\[ \rho = \frac{f(x + \Delta x) - f(x)}{J(x)\Delta x} \] (13)

If \(\rho\) is too small, it means that the actual reduced value is far less than the approximate reduced value and we need to narrow the approximate range; otherwise, we need enlarge the range.

### 3.6 Projection

The model we finally fitted needs to be projected onto the corresponding color image. In this way, we can more intuitively see if the human pose is fitted.

As we all know, we can project 3D vertices onto the 2D screen through the camera matrix \(P\), like the equation (15):

\[
\lambda v = PV \\
\lambda P = K[r | t] \] (14) (15)

where \(V\) is 3D vertices, and \(v\) is 2D vertices, but in homogeneous coordinates, \(V\) consists of 4 elements, and \(v\) consists of 3 elements. \(\lambda\) is the inverse depth of the 3D point, we can use it to normalize the last element of homogeneous coordinates to 1. The camera matrix \(P\) consists of two parts, one is the intrinsic matrix \(K\) and the other is the extrinsic matrix which contains rotation matrix \(r\) and translation matrix \(t\). The intrinsic matrix describes the projection properties of the camera and the extrinsic matrix determines the pose of the camera.

In general, \(K\) can be written as:

\[
K = \begin{bmatrix}
f_x & 0 & c_x \\
0 & f_y & c_y \\
0 & 0 & 1
\end{bmatrix}
\] (16)

Where \(f_x\) is focal length, \(f_x = \alpha f_y\), \(\alpha = \text{height} / \text{width}\), \(\text{height}\) and \(\text{width}\) is the resolution of the color image. Because our model is in OpenGL, we cannot multiply \(K\) by the vertices of our model. We need to transform \(K\) to a perspective projection matrix \(p\) in OpenGL.

And we can get \(\text{fovy}\) and \(\text{aspect}\) by equation (17) and equation (18) separately.
\[ f_{\text{ovy}} = 2 \cdot \arctan(0.5 \cdot \frac{\text{height}}{f_y}) \cdot \frac{180}{\pi} \]  
(17)

\[ \text{aspect} = \frac{\text{width} \cdot f_y}{\text{height} \cdot f_x} \]  
(18)

The model we finally get is in world coordinate, so we need to transform it to color camera coordinate through the equation (19).

\[
\begin{bmatrix}
X_{\text{rgb}} \\
Y_{\text{rgb}} \\
Z_{\text{rgb}} \\
I
\end{bmatrix} = p \begin{bmatrix}
r^{-1} & -r^{-1}t & X_w \\
0 & 1 & Y_w \\
0 & 1 & Z_w \\
1 & & 1
\end{bmatrix}
\]  
(19)

We can get \( K, r \) and \( t \) through the script of camera calibration in matlab.

4. Analysis
We have done 50 sets of experiments, and their average running time is about 2 seconds. And the average errors between the joints with the number of iteration are shown in figure 5. We can get a conclusion that our solution is convergent.

\[ \text{Average joints error} = 10 \]  

Figure 5. The average joints error with the number of iteration

There are 4 sets of effect figures shown as follows, and we can see that the poses of the resulting models and the corresponding human bodies are consistent if their arms are not bent. However, as shown in figure 6(h), the effect of arms is not good.
Figure 6. The result of 4 kinds of poses. (a), (b), (c) and (d) The original human bodies. (e), (f), (g) and (h) The corresponding posed models.

5 Conclusions
We proposed an automated method for estimating 3D body pose from 3D joints acquired by a Kinect sensor. This method only requires the users with an arbitrary pose to stand in front of a Kinect sensor. When the Kinect sensor collected 3D joints, we will automatically fit the 3D joints to the SMPL model and project the posed model back to the corresponding person in the color image. The resulting model can be used in many directions for future work. In particular, we can align the posed model to the point cloud data acquired by a Kinect sensor in order to get a consistent model in shape and then we can measure the model to verify the accuracy of our solution. Additionally, when we get the consistent model in pose and shape, we can move it to Unity3D engine to achieve the effect of dynamic tracking by an existing Kinect script. Besides, in order to achieve fully automated human modeling, our gender-specific model will be decided by predicting the gender of the users. In the end, we need to add garment to the model and achieve the physical simulation of the garment.

References
[1] Newcombe R A, Izadi S, Hilliges O, et al 2012. KinectFusion: Real-time dense surface mapping and tracking[C]. IEEE International Symposium on Mixed and Augmented Reality. IEEE, 127-136.
[2] Anguelov D, Srinivasan P, Koller D, et al 2005, 24(3). SCAPE: shape completion and animation of people[J]. Acm Trans Graph, 408-416.
[3] Cheng Z Q, Chen Y, Martin R R, et al 2017. Parametric modeling of 3D human body shape—A survey[J]. Computers & Graphics.
[4] Hirshberg D A, Loper M, Rachlin E, et al 2012. Coregistration: Simultaneous Alignment and Modeling of Articulated 3D Shape[C]. European Conference on Computer Vision. 242-255.
[5] Pons-Moll G, Romero J, Mahmood N, et al 2015, 34(4). Dyna:a model of dynamic human shape in motion[J]. Acm Transactions on Graphics, 1-14.
[6] Jain A, Seidel H P, Theobalt C 2010. MovieReshape: tracking and reshaping of humans in videos[C]. ACM SIGGRAPH Asia. ACM, 148.
[7] Loper M, Mahmood N, Romero J, et al 2015, 34(6). SMPL: a skinned multi-person linear model[J]. Acm Transactions on Graphics, 248.
[8] Chen Y, Cheng Z Q, Lai C, et al 2016, 22(8). Realtime Reconstruction of an Animating Human Body from a Single Depth Camera.[J]. IEEE Transactions on Visualization & Computer Graphics, 2000-2011.
[9] Weiss A, Hirshberg D, Black M J 2011. Home 3D body scans from noisy image and range data[C]. International Conference on Computer Vision. IEEE Computer Society, 1951-1958.
[10] Xu H, Yu Y, Zhou Y, et al 2013, 13(9). Measuring Accurate Body Parameters of Dressed
Humans with Large-Scale Motion Using a Kinect Sensor[J]. Sensors, 11362-84.

[11] Gauvain J L, Lee C H 1994, 2(2). Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains[J]. IEEE Transaction on Speech & Audio Processing, 291-298.

[12] Cheng K L, Tong R F, Tang M, et al 2016, 22(11). Parametric Human Body Reconstruction Based on Sparse Key Points.[J]. IEEE Transactions on Visualization & Computer Graphics, 2467-2479.

[13] Zeng M, Cao L, Dong H, et al 2015, 151. Estimation of human body shape and cloth field in front of a kinect[J]. Neurocomputing, 626-631.

[14] Bogo F, Kanazawa A, Lassner C, et al 2016. Keep It SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image[J], 561-578.

[15] Marcard T V, Ponsmoll G, Rosenhahn B 2016, 38(8). Human Pose Estimation from Video and IMUs[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1533-1547.

[16] Alldieck T, Magnor M, Xu W, et al 2018. Video Based Reconstruction of 3D People Models[J].

[17] Besl P J, Mckay N D 2002, 14(2). A method for registration of 3-D shapes[J]. IEEE Trans Pattern Analysis & Machine, 239-256.