Lifkooee *et al.* Real-time avatar pose transfer and motion generation using locally encoded Laplacian offsets: JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(1): 1–22 January 2018. DOI 10.1007/s11390-015-0000-0

Real-time Avatar Pose Transfer and Motion Generation using Locally Encoded Laplacian Offsets

Masoud Zadghorban Lifkooee[1], Celong Liu[1], Yongqing Liang[1], Yimin Zhu[2], Xin Li*[1]

[1] *Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge 70803, USA.*
[2] *Department of Construction Management, Louisiana State University, Baton Rouge 70803, USA.*

E-mail: mzadgh1@lsu.edu; cliu32@lsu.edu; ylian16@lsu.edu; yiminzhu@lsu.edu; xinli@lsu.edu

Received July 15, 2018 ; revised October 14, 2018.

Abstract We propose a human avatar representation scheme based on intrinsic coordinates, which are invariant to isometry and insensitive to human pose changes, and an efficient pose transfer algorithm that can utilize this representation to reconstruct a human body geometry following a given pose. Such a pose transfer algorithm can be used to control the movement of an avatar model in virtual reality environments following a user's motion in real-time. Our proposed algorithm consists of three main steps. First, we recognize the user's pose and select a template model from the database who has a similar pose; then, the intrinsic Laplacian offsets encoded in local coordinates are used to reconstruct the human body geometry following the template pose; finally, the morphing between the two poses is generated using a linear interpolation. We perform experiments to evaluate the accuracy and efficiency of our algorithm. We believe our proposed system is a promising human modeling tool that can be used in general virtual reality applications.

Keywords Human Body Pose Transfer, Local Intrinsic Coordinates, Avatar Control in Virtual Reality

1 Introduction

An important component of Virtual Reality (VR) environments is the modeling of human body. While the motion of a human character in virtual scenes could be generated automatically, a flexible way in its modeling is still through user's direct input/control. In many VR applications, it is desirable to build an avatar that can automatically mimic a user's motion and pose [1, 2, 3]. For example, in multiplayer VR games, this would allow different users' avatars to see others' behaviors and interact with them. Therefore, this paper aims to build such a human body avatar, whose movement is controlled by a user with motion tracked in real-time.

Human body animations can be defined by three main components: shape, pose, and motion. The motion component of human body animation is a sequence of human geometries in different poses. Hence, we study how to transfer the pose from a user to a digital avatar model in the virtual scene in real-time. The digital avatar may either have a same geometry of the user (e.g., acquired from body scanning) or have a different, pre-designed geometry (e.g., built by modeling software or obtained from templates in database).

Performing such a pose transfer efficiently and realistically, is however, challenging. Humans have a remarkable variety of poses [4]. But having the avatar reproducing the user's motion authentically is important because this is the main way for the users in the VR environment to communicate.

A direct way to achieve this is through real-time

Regular Paper

motion capturing, such as the system developed in [5]. It uses a system including multiple RGB and infrared cameras to capture and transmit the dynamic 3D geometry of the moving human body and the surrounding scene. However, due to the expensive stitching and reconstruction cost involved in performing such a Holoportation, in real-time applications, a trade-off between geometric accuracy and computational efficiency is inevitable.

Another strategy to generate the avatar's motion is through animations. Two widely adopted animation algorithms are *direct mesh deformations* [6, 7, 8] and *skinning-based animations* [9, 10]. The direct deformation strategy converts the tracked motion to positional constraints, following which the deformation should also preserve local geometric detail as much as possible. Such a mesh deformation is usually formulated as a nonlinear optimization. While it is capable of reproducing complex deformation with desirable details, its solving is usually expensive and hard to finish in real-time [11].

Skinning-based methods have been widely adopted as a more efficient character manipulation tool, as it intuitively reduces the deformation to a skeleton subspace in which the computation can be very quick. However, skinning-based methods also have their shortcomings such as the need of tweaking of vertex weights, incapable of describing complex deformation [12], and relying on accurate skeleton tracking.

In this work, we explore the possibility of a data-driven deformation approach that can be both efficient and capable of reproducing deformation details. We generate avatar's motion by integrating pose recognition, template-guided pose transfer and reconstruction, and inter-pose interpolation, to obtain real-time motion generation on a given human avatar model.

Our **main idea** is to design this human avatar and its interactive control using an intrinsic geometric encoding that capture the body geometry in a pose-insensitive manner. The user's pose is tracked and analyzed to guide the placement of a set of feature points on the avatar. Then, the geometry of the avatar under the new pose can be reconstructed using the intrinsic encoding. Specifically, our pipeline consists of four main steps. The first step is done offline, where the intrinsic Laplacian coordinates of the avatar are computed and stored. Then, in the online control phase, we (1) track then estimate the user's pose, and use it to select a reference template pose from the database; (2) transfer the pose of the template onto the avatar; and (3) generate the morphing sequence of the avatar between these key poses. This pipeline is illustrated in Fig. 1.

The **main contributions** of this paper are two-folded. First, we propose to perform a real-time avatar control through a pose reconstruction (pose recognition, then pose transfer) algorithm. With the help of a database containing ever-growing human body geometries/poses, the algorithm is efficient and effective. Second, to perform the real-time pose transfer, we adopt the intrinsic Laplacian encoding which is pose-insensitive, develop an efficient key-pose-frame recognition and geometric reconstruction algorithm. Our experiments have demonstrated that the proposed pipeline has promising applications in VR tasks.

2 Related Work

Designing an efficient human avatar with real-time user-control support is closely related to two technical components. One is the recognition of user's pose, and the other is the deformation of the avatar model according to this pose.

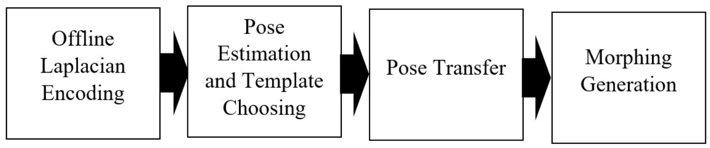


Fig.1. Our main computation pipeline.

2.1 Pose Estimation

The aim of the pose estimation stage is to calculate 2D or 3D positions of joints that characterize a human pose. In order to control a 3D human avatar, we need to have coordinates of 3D joints. These 3D positions can be obtained either directly through tracking sensors attached on the user, or by calculations from images captured by camera(s) on the scene.

Image-based pose estimation is a fundamental but still ongoing research topic in computer vision field. A challenge in image-based 3D joints estimation is collecting proper dataset [13]. To achieve a high performance on pose estimation and classification, having sufficient amount of 3D poses with annotated 2D images (that 2D joints location are determined) is often necessary. This is, unfortunately, expensive and still difficult even with the state-of-the-art motion capturing systems and trained actors [14]. Martinez et al. [15] suggested to collect and utilize only 2D joint information, and designed a deep network architecture to estimate 3D pose from 2D pose data. However, since processing 2D data to support this estimation is highly non-trivial, the generalization ability of this algorithm is yet to be improved. Yasin et al. [13] suggested a method that uses two independent datasets of 3D pose and 2D images. With this, it does not require a large amount of annotated 2D images. The independent 3D poses are projected to 2D plane to train a pictorial structure model (PSM) for 2D pose estimation. Final 3D poses are estimated by minimizing the projection errors from these 2D poses. This method still requires sufficient 3D pose data in training, which is expensive and sometimes prohibited. To solve this issue, Moreno-Noguer et al. [16] developed a 2D-to-3D EDM Regression model with a deep Neural Network that does not rely on 3D pose dataset.

Another challenge of pose estimation especially for real-time applications such as human avatar control is the computation efficiency. The aforementioned methods [16, 13, 15] are not real-time and insufficient for interactive avatar control that we need. In [17], a real-time algorithm is proposed to calculate 2D and 3D joint positions simultaneously. From single RGB images, a kinematic skeleton is fitted and then the 3D joints are calculated through a convolutional neural network.

Although image-based 3D pose estimation has achieved great advancement in the past few years, obtaining reliable and real-time estimation of joints or markers from the user is still not trivial. In this work, we directly adopt sensor-based pose estimation using a tracking vest [18]. With this direct tracking we can have accurate real-time landmark coordinates on the user, without the need to label any image dataset.

2.2 Pose Deformation

The goal of pose deformation is to generate the new body pose and shape for the avatar to match the user's real pose. In this paper we categorize the methods proposed for pose deformation in the literature into three groups: *image based*, *skinning based* and *intrinsic 3D coordinates based* methods.

Image-based methods use 2D images as inputs to generate human body poses. In [19], firstly, twelve 2D images are captured from a person's body in different angles of view. Subsequently, calibration and orientation processes are done on the 2D images. After finding the interest points, the matching points are estimated. Then, the body orientation is calculated based on the matched points for the pair images. Next, The final results are calculated by estimating both interior and exterior orientation. In [20], Seo et al. suggested a method using a statistical modeling of 2D image shapes. First, the contour template of the human body image is determined. The PCA algorithm is applied to parameterize the body shape model based on 3D shapes. In the next stage, the projection of 3D shape is matched with the 2D contour of body shape. Finally, a 3D shape is generated by minimizing the matching error.

In [21], Cheng et al. used Kinect images as input to segment body shapes from the 2D images. In the next phase, some key points are detected based on a regression approach. The human body pose then is parametrized using a sparse key point representation. Although the accuracy reported is high (with the error of 8.2 mm), the computational. cost for each frame takes more than 0.5 second that causes the method not to be suitable for real-time applications The method proposed in [5] is real-time in reproducing digital avatar. In this paper, the pose and texture information are obtained using infrared and RGB cameras respectively. In this method many conditions that causes error in real-time human body reconstruction such as occlusion and topology change are considered and solved. In addition to body, image-based approaches can be used for facial expression representation [22] that is another component of human avatar animation. Although the image-based methods can reconstruct body pose and geometry, the reconstructed body pose may have some salient artifacts and missing parts since the method relies on the visible regions of the provided image.

Skinning is another approach to animate human bodies under different poses. It associates vertices on the human body skin with certain skeletal nodes (bones), then deforms vertices according to transformations of their correlated bones. To adopt skinning approaches, skeletons need to be extracted and the associations need to be computed. However, both real-time extraction/tracking of the skeleton and estimation of bone transformations are non-trivial. While the skeleton extraction from a 2D or 3D shape (i.e., skeletonization) has been widely studied in graphics and vision fields during the past two decades [23, 24, 25],extracting skeletons from incomplete/occluded scans [26], or extracting consistent skeletons on multiple objects [27] (so that deformation can be transferred from one body to another [28]) still cannot be solved in real-time. Finally, while some real-time algorithms such as [17] have been proposed to track the dynamically changing skeleton during human's motion, reliable determination of full bone transformations, i.e., both rotations and translations on all bones, is still challenging.

Another approach to model and transfer human poses is using intrinsic shape representations, or pose-insensitive descriptors to encode both pose and local geometry of human body [29]. By separating the intrinsic local geometry and human pose, designing such a pose-invariant representation becomes possible [29].

The first fundamental form of a surface, defined by the intrinsic metric of the surface, is usually insensitive to postures. In [4], Pishchulin et al. built a statistical shape model for human based on such local coordinates, which are pose insensitive. This will allows the Principle Components Analysis (PCA) to be performed on human bodies with various poses. Another effective coordinates are mesh Laplacian, which provides a mean to represent surfaces using intrinsic bases. In [30], the normalized Laplacian operator is used to calculate the Laplacian offsets. These locally encoded offsets are isometry-invariant, and are used to encode the shape and pose information simultaneously. However, the normalized Laplacian operator is not symmetric nor full rank. Hence, the reconstructions in [30] reduces to an iterative optimization, which is slow and not suitable for online pose transfer. In this work, we modify the model of [30] to make it more efficient for real-time pose transfer.

3 Methodology

Our proposed avatar control pipeline tracks the motion of a user in the field, and selects templates sequentially from the database to guide the avatar's deformation. The algorithm is summarized in Fig. 1. During the offline stage, the geometry of the avatar is encoded using locally encoded Laplacian offsets, which are intrinsic and pose-insensitive (Section 3.1). Then, during the online stage, from the input of the user's pose, described by a set of tracked 3D landmarks, we construct a pose descriptor using the distribution of these landmark points (Section 3.2). Then, in a human body database we find a template model with the most similar pose (Section 3.3). The avatar will be deformed following the template model (Section 3.4). Finally, we animate the motion of the avatar by interpolating shapes between every two consecutive key poses (Section 3.5).

Overall, the goal of our research is to animate a human avatar which can be defined as the digital representative of the user in a 3D space. A default human avatar is selected based on the closest 3D geometry to the user in the dataset that is called source mesh (S) in this paper.

In the next stage, the source mesh iteratively is deformed based on the closest 3D pose to the user in the dataset which is called the template mesh (T). This can be defined by a deformation function (F). So, we have: $S_n = F(S,T)$ where S_n is the deformed source mesh or new mesh.

3.1 Offline Processing: Intrinsic Encoding using Local Laplacian Offsets

To support effective pose recognition and avatar deformation, we need to encode both the pose information and local geometry of a human body shape. The pose information (insensitive to geometry difference) is needed to recognize the user's pose and to match the poses of different persons. The geometry information (insensitive to pose difference) is needed to describe the avatar's own geometric characteristics during deformation, so that the avatar won't deform to another person. We use the Laplacian offsets, encoded in local coordinate system of each vertex. These intrinsic variables will not change under isometry transformation.

We define necessary terminologies as follows. We use $S = \{V_S, E\}$ to denote a *source mesh*, or the avatar mesh. It is the mesh we want to deform according to the user's pose. The avatar mesh S could come from a pre-designed avatar model, or from a body scan of the user. $T = \{V_T, E\}$ denotes a *template mesh.* It is from the human body database where a mesh with a similar geometry and pose is selected. T will guide the deformation of S. Note that we cross-parameterized all the human bodies, so S and T have the same vertex number $\|V_S\| = \|V_T\| = \|V\|$ and the same connectivity E. We use $M = \{m_1, m_2, \ldots, m_k\}$ to denote the (index) set of marker points tracked from the user. They correspond to certain vertices on the mesh. By matching vertices in S with their counterparts in T, we can control the deformation of S following the tracked user's motion.

The *Laplacian offset* vector $\mathbf{\Delta}$ is an $(n \times 3)$-

dimensional matrix ($n = \|V\|$) that can be considered as a discrete 3D vector field defined on every vertex,

$$\mathbf{\Delta} = \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \cdot \\ \cdot \\ \Delta_n \end{pmatrix} = L \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdot \\ \cdot \\ \mathbf{x}_n \end{pmatrix}, \quad (1)$$

where $\mathbf{x}_i$ denotes the 3D coordinates of vertex v_i. The Laplacian Operator L can be discretely represented as an $n \times n$ matrix whose component $l_{i,j}$ is

$$l_{i,j} = \begin{cases} deg(v_i), & \text{if } i = j \\ -1, & \text{if } j \neq i \ \& \ v_j \in N_1(v_i) \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

where $N_1(v_i)$ denotes the one-ring neighborhood of vertex v_i, and deg denotes the valence of vertex v_i.

This Laplacian offset $\mathbf{\Delta}$ encodes the geometry of the human body shape. But it is not invariant under pose change. On the other hand, if we encode this offset under local coordinate frame of each vertex, it becomes intrinsic and is invariant under isometry [30]. Therefore, we project the Laplacian offset onto each vertex's local coordinate system:

$$\Delta_i = \omega_i^1 f_1(v_i) + \omega_i^2 f_2(v_i) + \omega_i^3 f_3(v_i) = F(v_i) \cdot W_i, \quad (3)$$

where $f_1(v_i), f_2(v_i), f_3(v_i)$ are the three orthonormal vectors that define a local coordinates system $F(v_i)$ on vertex v_i in S. The new isometry-invariant coordinates of vertex v_i are $W_i = \{\omega_i^1, \omega_i^2, \omega_i^3\}$.

Fig. 2 illustrates the insensitivity of this local coordinate system with respect to pose changes. For a same human body under two different poses (a, b), the coordinates are similar, except on regions that undergo deformations that are far from isometry. This can be seen from (c). On the other hand, these coordinates reflect the geometry difference. Hence, the coordinates on two different human bodies (even with a same pose) are quite different, as shown in (d) and (e).

Note that, unlike [30] which uses a normalized Laplacian operator, we construct the Laplacian offsets using the unnormalized Laplacian operator. This makes the Laplacian matrix symmetric, and it could allow us to more efficiently solve the pose transfer through Cholesky factorization [31] (Section 3.4).

The orthonormal vectors $(f_1(v_i), f_2(v_i), f_3(v_i))$ can be constructed using (1) the normal vector $\mathbf{n}(v_i)$ at each vertex v_i, (2) the normalized projection of $\mathbf{x}_i\mathbf{x}_k$ onto the tangent plane of v_i where v_k is an arbitrary but fixed neighboring vertex of v_i, and (3) the cross product of these two vectors.

3.2 Pose Modeling

We organized and classified available human body mesh data according to their poses. To make the pose estimation consistent with the body tracking, we use a set of selected landmark points on the body. These landmarks are consistent with the sensors being tracked by a wearable body tracker. (Fig. 3(a)). When a user is performing its control motion, the corresponding 3D coordinates of these landmarks (Fig. 3(b)) will be tracked and mapped onto the body mesh space instantly, serving as constraints to guide the avatar deformation.

Using these tracked landmarks, we build a descriptor for pose classification and recognition, using angles between line segments connecting these markers. From all the line segments that connect every pair of markers, we select a subset $L_s = \{l_0, l_1, l_2, \ldots, l_k\}$ of line segments, then build a n-dimensional feature descriptor $F_s = \{\theta_0, \theta_1, \theta_2, \ldots, \theta_n\}$ using angles θ_k between some pairs of adjacent line segments.

$$\theta_k = \arccos \frac{l_i l_j}{\|l_i\| \|l_j\|}, \quad (4)$$

where l_i and l_j are a pair of adjacent line segments. We elaborate the algorithm in selecting line segments and angles in the following.

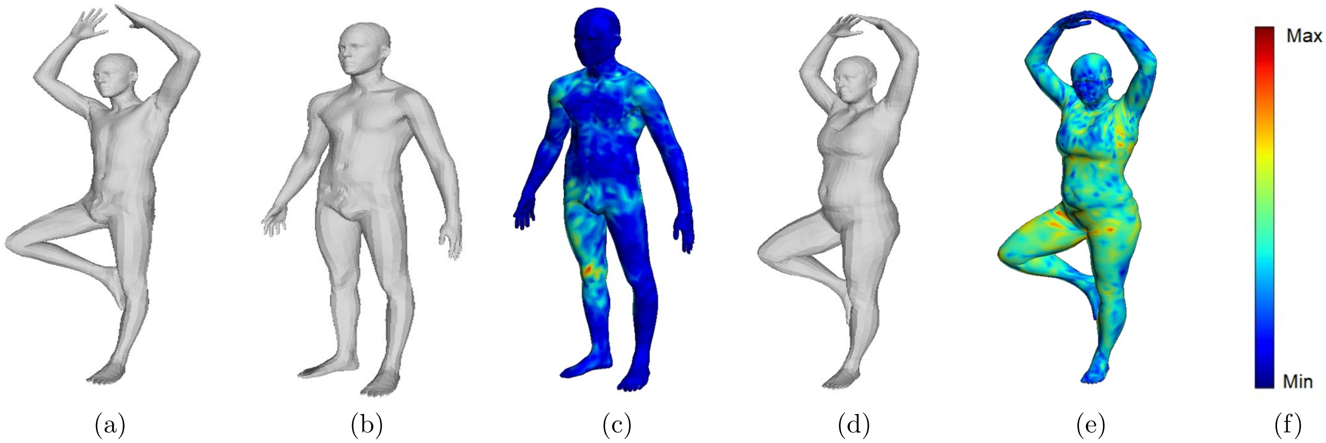
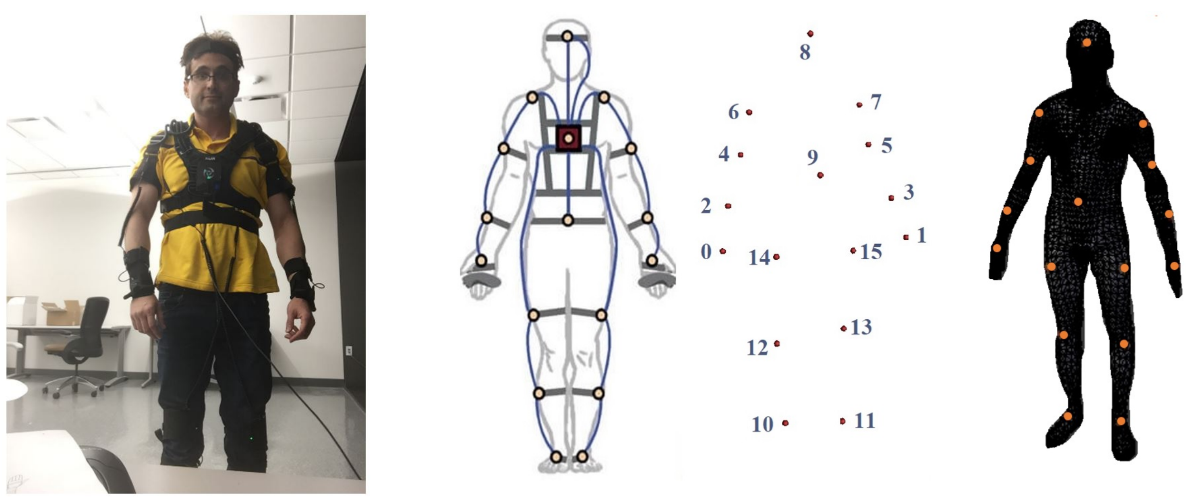


Fig. 2. Pose-insensitivity of Local Laplacian Offsets. (a) and (b) show the two poses belong to one person; (c) shows the colorcoded point-to-point coordinate difference between (a) and (b) In most body regions the deviation is small, near some joints where the deformation is farther away from isometry the deviation is bigger. (d) shows another person that has a similar pose to (a). As shown in (e), the point-to-point coordinate difference is significantly bigger than (c). This indicates that these intrinsic cooridinates are more sensitive to body shape difference, and insensitive to the pose change.



(a) body tracker vest (b) tracked landmarks

Fig. 3. A wearable body tracker vest is used to track feature landmarks on human body (a). 16 corresponding feature points are extracted on the human body mesh template.

3.2.1 *Building the Pose Feature Descriptor.*

We use a decision tree to select the significant line segments and incident angles to build the pose descriptor. First, the 16 landmarks can form $\frac{P_{16}^3}{2}$ angles. From all these angles (variables), we build a decision tree to select the most salient d ones. Considering the symmetric property of the human body and motions and to avoid the imbalance in the training dataset, we "mirrored" all the incident angles: suppose we use $m(i)$ to indicate landmark i's corresponding landmark on the other side, when an incident angle $\theta = \angle(v_i, v_j, v_k)$ is observed in the data, we also add an instance of $\theta = \angle(v_{m(i)}, v_{m(j)}, v_{m(k)})$. These angles are then selected by a decision tree to pick the most salient k variables to form the angle descriptor.

Fig. 4 shows the angle selected when different feature size d is being considered. These selected line segments and angles form the feature descriptors which we used to classify all the pose samples in the dataset.

Fig. 5 illustrates four more example descriptors on two human bodies, with two different poses, respectively. While the feature graphs for two different persons with the same pose are notably similar, these graphs are very different for people in different poses. Therefore, using this graph to describe the pose is effective. More experimental results demonstrating the descriptor's effectiveness are reported in Section 4.

3.3 Pose Recognition and Template Selection

Template Database. The volume of publicly available human pose database has been rapidly growing. We integrated multiple datasets: FAUST [32] (including 500 human body samples in 30 different poses), SCAPE [33] (including a human body in 72 different poses), Human3.6M [34] (including 3.6 million bodies and poses), K3D-Hub [35], CAESAR [36], SHREC'14 [37], and MPI Stitch [38].

Pose Recognition. Following the method described in the previous section, pose descriptors for all the meshes in this database are pre-computed on all the template human bodies. When a new pose is given, we can simply compute its descriptor, then compare it with all these precomputed descriptors, and report the most similar template.

To do this comparison efficiently we used Support Vector Machine (SVM) to classify the poses. SVM is well-known for its ability of class separation and low computational cost. Then we used K-Nearest-Neighbors algorithm to choose the best pose the matches the user's body geometry within the classified pose class.

Fig. 6 shows the pipeline of the pose recognition stage. From the tracked landmarks, the pose descriptor is created and compared with representatives from each cluster. A mesh with the most similar pose is selected as the template.

3.4 Geometric Reconstruction

To deform the source mesh S to match the pose of template mesh T, we shall reconstruct S's local geometry, using local Laplacian offset coordinates W computed on S, on the local coordinate frames F defined on T. Specifically, if we recall that

$$L \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdots \\ \mathbf{x}_n \end{pmatrix} = \begin{pmatrix} F^S(v_1)W_1 \\ F^S(v_2)W_2 \\ \cdots \\ F^S(v_n)W_n \end{pmatrix}. \quad (5)$$

Here, we use F^S to indicate the local coordinate frames defined on mesh S and $F^S(v_i)$ is the local frame (a 3×3 matrix) on v_i. W_i is the corresponding local coordinates.

If we denote the deformed source mesh as S^*, then,

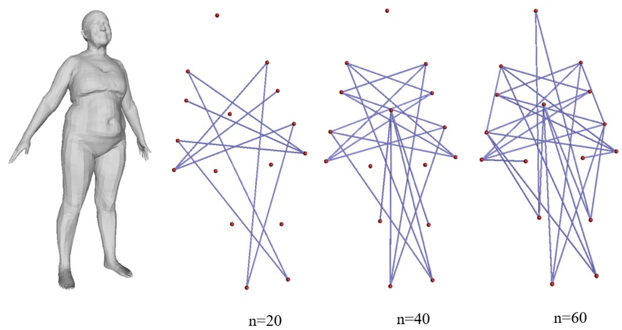


Fig.4. The line segments (angles) selected for constructing the features when different descriptor size is used: $d = 20, 40, 60$.

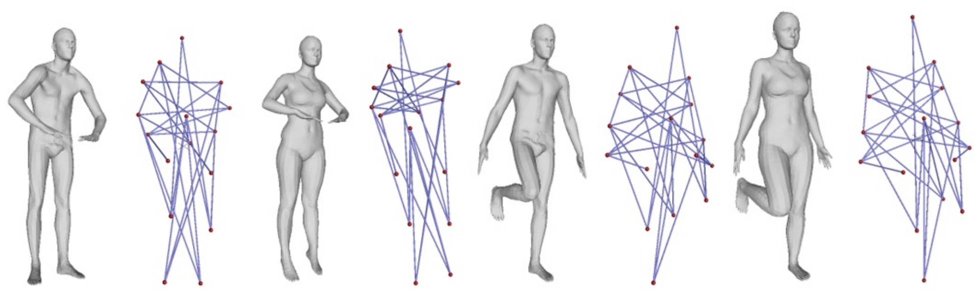


Fig.5. Feature graphs of two human bodies in different poses. The feature size is 50. Note that while the feature descriptor is formed by the incident angles, we plot the these angles' associated line segments for visualization purpose.

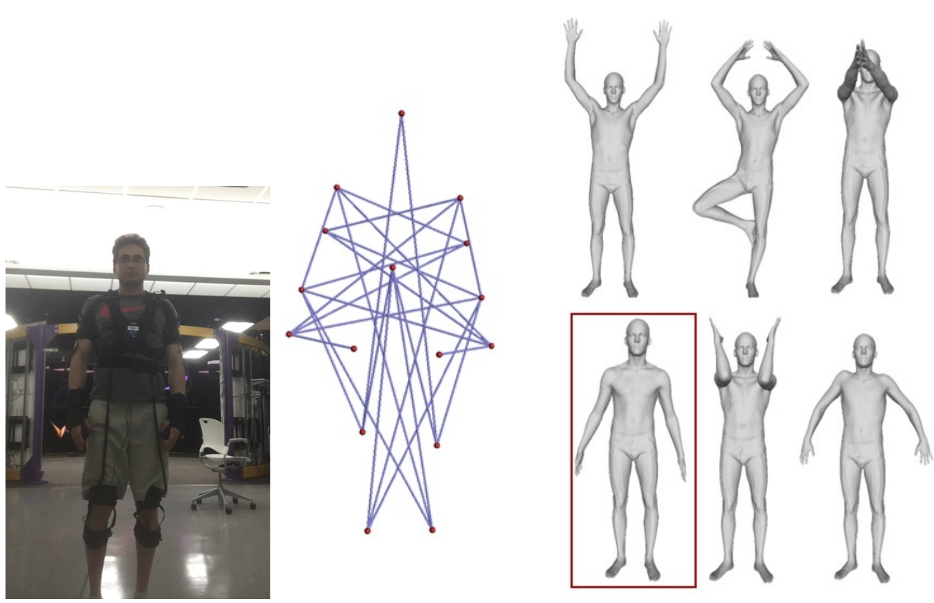


Fig. 6. Pose recognition pipeline. (Left): pose from the user; (Middle): the corresponding pose descriptor (incident line segments visualized as a graph); (Right): matched pose from database.

we also have

$$L \begin{pmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \\ \dots \\ \mathbf{x}_n^* \end{pmatrix} = \begin{pmatrix} F^{S^*}(v_1)W_1 \\ F^{S^*}(v_2)W_2 \\ \dots \\ F^{S^*}(v_n)W_n \end{pmatrix}, \tag{6}$$

where x_i^* are final coordinates of each deformed vertex v_i, and F^{S^*} is the corresponding local frames. Using T to guide this deformation is to make F^{S^*} to follow F^T as much as possible. Hence, we first set F^{S^*} following F^T. And use it to solve $\mathbf{X}^*$, then, update F^{S^*} accordingly. We repeat these iterations until it converges.

Another issue is that the rank of L is $n-1$. Therefore, linear systems of Equations (5, 5) have infinite solutions. This is why [30] uses an iterative solver to find a solution near a given initial guess. In our problem, our tracked landmarks provide with us $c \times 3$ extra constraints on mesh S^*. With these constraints, the system of Equation (6) becomes over-constrained, and we can revise L to a full-ranked symmetric positive definite matrix, and use the more efficient Cholesky decomposition to solve the systems. Furthermore, L will never change, but we will need to resolve the system under different boundary conditions. This strategy will allow us to reuse the decomposition result and get solutions to all these linear systems instantaneously.

Constrained Laplace Linear System. With constraints defined by tracked landmarks, we can simplify the Laplace matrix L by removing the corresponding rows and columns. Specifically, if v_i is a landmark, then its coordinates $\mathbf{x}_i^*$ is known, and we remove the i-th row and i-th column from L and move the corresponding element $l_{ij}x_j^*$ to the right side of the linear system. We use $\mathbf{b}^c$ to denote all these moved components. Suppose there are c landmarks, then after removing all these variables from the system, the coefficient matrix becomes $(n-c) \times (n-c)$. We denote it as L^c. Finally, Equation (6) becomes

$$L^c \begin{pmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \\ \dots \\ \mathbf{x}_{n-c}^* \end{pmatrix} = \begin{pmatrix} F^{S^*}(v_1)W_1 \\ F^{S^*}(v_2)W_2 \\ \dots \\ F^{S^*}(v_n)W_{n-c} \end{pmatrix} + \mathbf{b}^c. \tag{7}$$

When we have more than 2 landmarks, L^c is full-ranked (i.e., positive definite). We can use Cholesky decomposition [31] to decompose L^c into $L^c = TT^*$ where T is a lower triangular matrix with positive diagonal entries and T^* denotes the conjugate transpose of T. Then, we can efficiently reuse T and T^* to solve the linear systems of Eqn. (7) under different boundary constraints when L^c does not change.

Final Algorithm. We summarize our proposed reconstruction algorithm as follows,

1) Initialization: Set $F^{S^*} = F^T$;
2) Solve Equation (7) and get $\mathbf{X}^*$;
3) Update F^{S^*} by re-calculating the local frames;
4) If during the last iteration, both $\mathbf{X}^*$ and F^{S^*} do not change much, STOP; otherwise, go back to Step 2).

An example of reconstruction (pose transfer) result is illustrated in Fig. 7. The source meshes (a, d, g, j) are deformed following the poses of the template meshes. The pose transfer results (c, f, i, l), respectively, have the geometry of each source mesh but its pose mimicks the template's pose.

3.5 Morphing from Source to Target Poses

To generate a sequence of meshes, we only process a few key pose frames.

For every k seconds, its pose is captured and recognized. Suppose the current frame is the i-th capture. With the recognized pose a template $T_{i \times k}$ is selected and used to guide the deformation and obtained a new

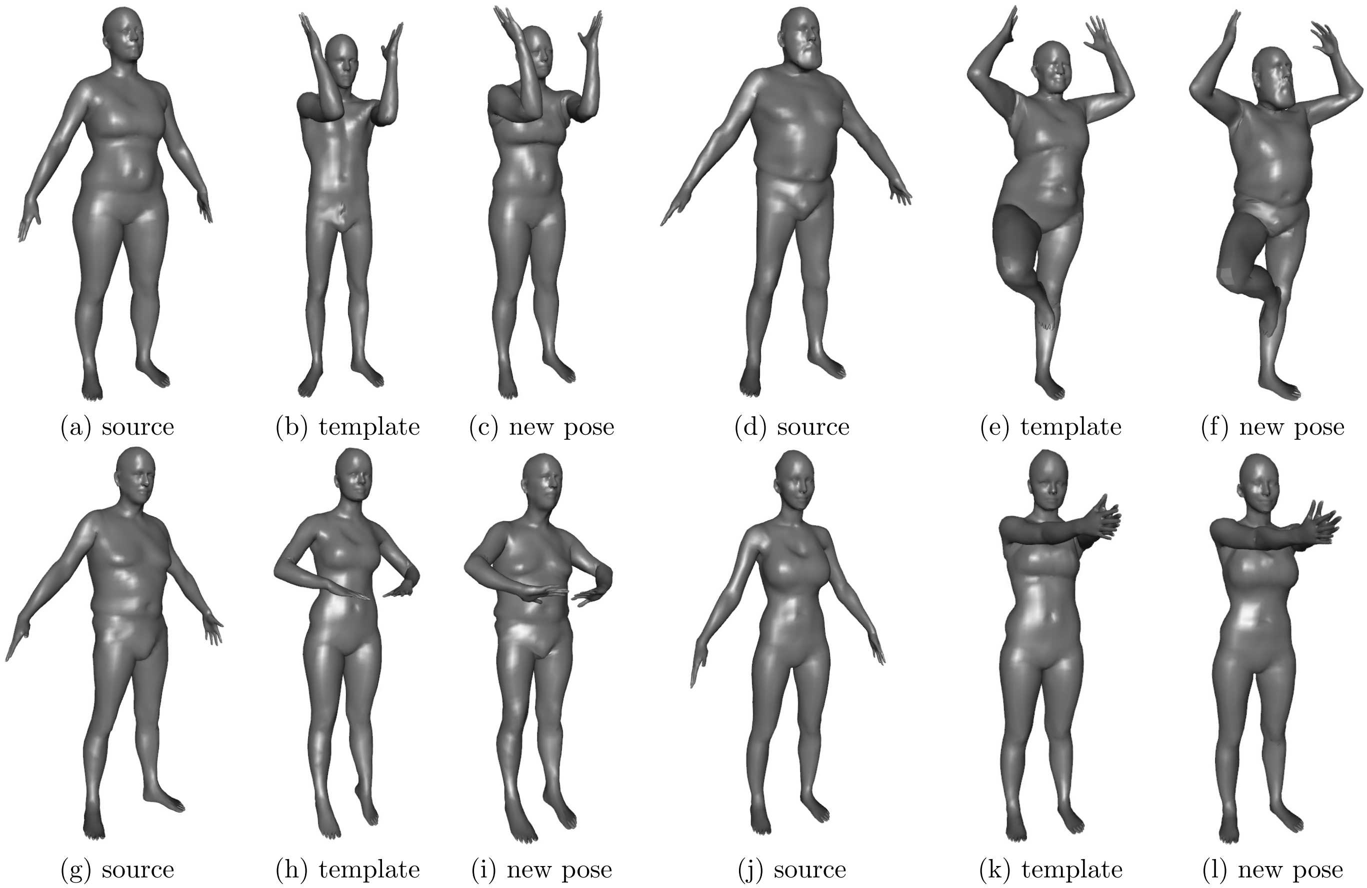


Fig.7. Four Pose Transfer Examples. The source meshes (a, d, g, j), following template meshes (b, e, h, k), are deformed to the new poses (c, f, i, l), respectively.

Fig.8. Morphing based on linear interpolation between two key-frames.

deformed mesh $S_{i\times k}$ from the last key frame $S_{(i-1)\times k}$. Between these two key frames $S_{(i-1)\times k}$ and $S_{i\times k}$, we simply do a linear interpolation to generate the morphing sequence

Key frame interval selection. The interval parameter k balances the quality and computational cost. When k decreases, more intermediate poses are captured, and less interpolation is used. This in general increases the quality of generated motion sequence. However, the computation of pose recognition and reconstruction needs to be finished within this interval. When k increases, we reconstruct fewer poses and rely more on interpolation. The reconstructed motion could be less accurate but the computation is much faster. However, the suitable value for k depends on user's motion. Slower motions can be reconstructed with bigger k, while rapid or drastic motions need smaller k to reproduce. Adaptively selecting k would be ideal; but during the online user-avatar synchronization, performing a real-time prediction then adaptively adjusting k is technically challenging. Therefore, based on multiple experiments and our current implementation on our machine, we select a relatively small interval $k = 1$ for which the computation can always be finished and the reproduced sequence is acceptable for common motions.

Fig. 8 illustrates poses linearly interpolated between two key poses. Fig. 9 illustrates another pose tracking and transfer example in our experiment. The first row shows the sampled pose tracking on the user, and the corresponding computation is finished within such a time interval. The transferred pose on the avatar is rendered in the second row.

4 Experimental Results

In this section, we will describe our experimental setup and demonstrate our results on feature selection, pose classification, and pose transfer.

4.1 Dataset

Human bodies collected in different datasets usually have difference resolutions and connectivities. Fusing all these data and generating a consistently parameterized human body model is necessary for us to use them as templates to guide the pose transfer. However, automatically finding the dense point-to-point correspondences between these human bodies is nontrivial [39, 40]. In this work, we utilize the parametric model, SMPL [41], and perform a fitting on each human body geometry in the database. With this modeling fitting, we obtain the model parameters and use them to reconstruct the consistently parameterized meshes. Every human body in the database is processed this way, and converted into models with the same connectivity. In practice, models within a same database are often registered and consistently parameterized. Then among these models, we only need to perform the above fitting on one representative model; and the cross-shape parameterization can be propagated to all other models in the same database Inspired by [41, 42], to perform a SMPL fitting, we use the 14 landmarks that we are tracking during the motion capturing. Fortunately, human poses can be appropriately encoded by these landmarks because the significant variation in human pose can be defined by these few moving joints.

Fig. 10 shows some examples of FAUST and SCAPE datasets. Fig. 11 illustrates an example of consistently parameterized two human bodies from different datasets. (a, b) are two meshes, from SCAPE and FAUST, respectively, and their zoomed-in wireframe view of the head. (c, d) are their re-parameterized meshes, which now have the same sampling and connectivity.

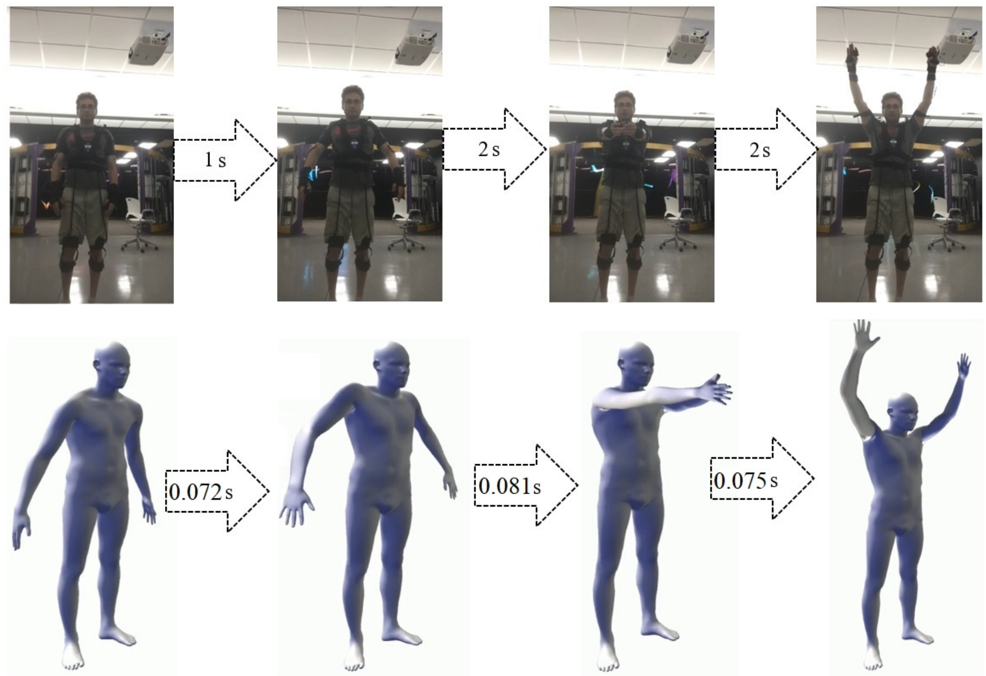


Fig. 9. UP, shows a sequence of real-time captured key-frames and associated run-time between two key-frames; Down, shows the reconstructed body geometries for each captured key-frame and associated run-time between two key-frames. .

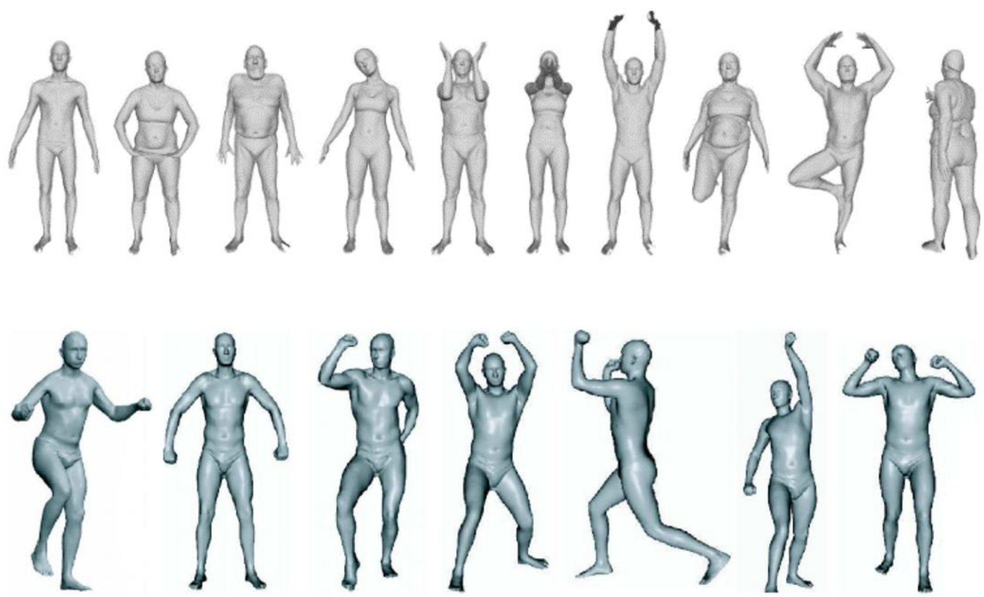
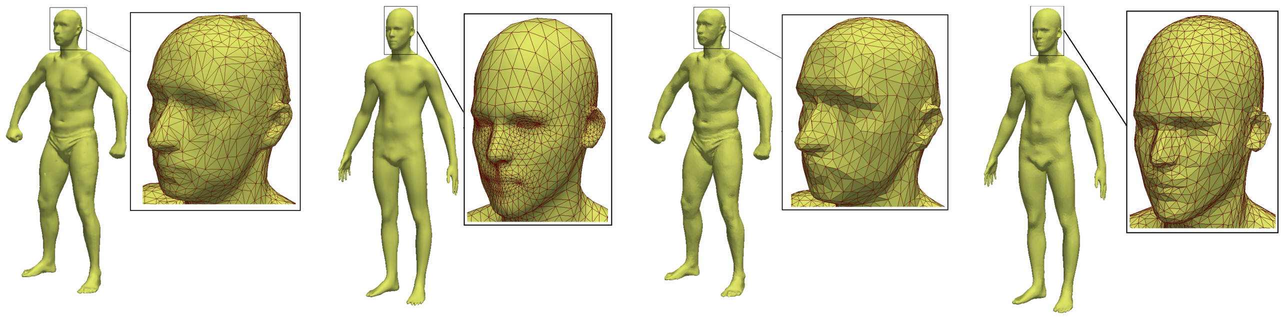


Fig.10. Some examples of body shapes from FAUST [32] (first row) and SCAPE [33] (second row) datasets.



(a) 12500 vertices (b) 6890 vertices (c) 10000 vertices (d) 10000 vertices

Fig. 11. Cross-body Registration and Parameterization using the SMPL model. (a) is a mesh from the SCAPE database and the wireframe view of the head; (b) is a mesh from the FAUST database. (c) and (d) are their re-parameterized meshes after the SMPL fitting, respectively. The re-parameterized meshes have the same resolution and connectivity.

4.2 Results

We demonstrate the experimental results on different phases of the proposed pipeline: feature construction, pose classification, and pose transfer.

4.2.1 *Feature Construction*

We found that the feature selection by the decision tree results in angles that are from joint markers and have high variance. Table 1 shows an example of selected angles and their variance values (the marker indexes of the line segments can be found in Fig. 3). Interestingly, all the selected angles are on joints following the natural skeletal structure of the human model, indicating that these joint angles are significantly more informative and sensitive to the pose change than the rest.

4.2.2 *Pose Estimation*

We used a tracking vest which has low noise error compared to image-based approaches. To estimate the user's pose to obtain the appropriate template mesh, we use the aforementioned feature descriptor derived from the corresponding tracked markers. Fig. 12 visualizes the distribution/clustering of different poses described by our pose descriptors.

For this visualization, we reduce the dimension of the descriptor space to 2 simply using the PCA algorithm. As can be seen in the figure, except for class 1 and class 8 which are remarkably similar poses, all the other classes are appropriately separated. We obtained these results as conceptual experiments to test how our pose classification algorithm is robust. However, in the reality, the number of classes needed for a real-time pose animation is significantly more than 10.

We used SVM to classify the poses on the FAUST dataset. We achieved the average accuracy of 0.98 in this dataset

The classification accuracy is defined by

$$accuracy = (TP + TN)/(TP + TN + FP + FN), \quad (8)$$

where TP, TN, FP, and FN are True Positive, True Negative, False Positive and False Negative, respectively. Pose classification accuracy regarding the dimension of feature descriptors. When the descriptor dimension is 60 and 70, the classification accuracy reaches 1.0.

4.2.3 *Pose Transfer*

Fig. 13 shows a demo of the pose transferring pipeline. As can be seen in the figure, body is tracked using a body tracker. Subsequently, in the pose estimation stage, the template mesh is chosen using the pose feature descriptors. Finally, the pose is transfered based on the template and source mesh.

Table 1. Selected Feature (Angles) for Descriptor Construction. They interestingly correspond with joint angles with big variance, following the skeleton structure. Seg-1 and Seg-2 indicate the two line segments forming the angle. The listed indexes for these line segments and markers follow the definition in Fig. 3.

Rank	Seg-1	Seg-2	Marker
1	(0,11)	(11,2)	11
2	(0,10)	(10,2)	10
3	(6,10)	(10,4)	10
4	(6,11)	(11,4)	11
5	(0,12)	(12,2)	12
6	(7,11)	(11,5)	11
7	(4,10)	(10,2)	10
8	(5,11)	(11,7)	11

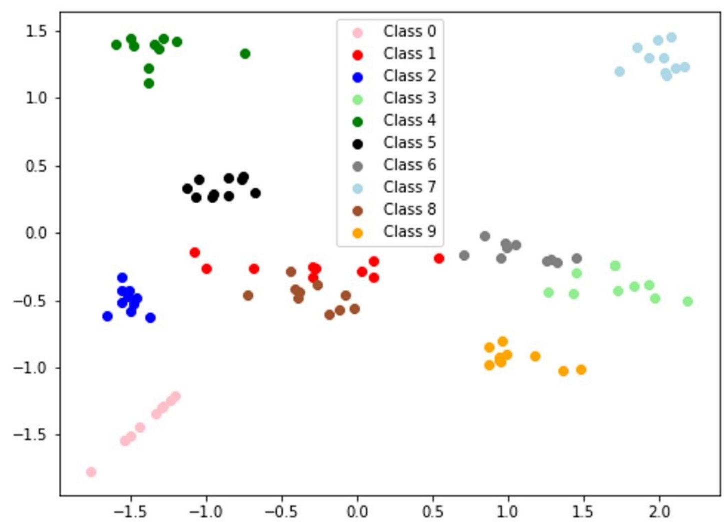


Fig.12. Visualizing the pose estimation, clustered by constructed pose descriptors.

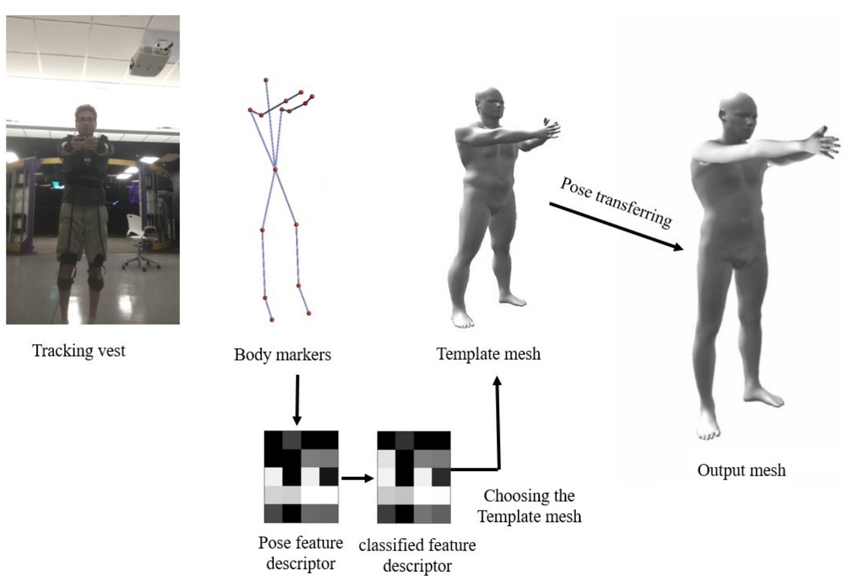


Fig.13. A demo of pose transferring pipeline .

Fig. 14 show animations under different sampling densities, where $k = 2$ (images with green bounding box) and $k = 3$ (images with red bounding boxes), respectively where the poses in the same rows are captured from the same frame. The figure also show the zoomed region obtained by two $k = 2$ and $k = 3$. As can be seen, the hands when k is set to 2 is shrunk and is less natural than when k is set to 3. Generally, as we can see from the figure, with smaller k, the morphing looks smoother and more natural. Naturally, if we keep tracking the user's pose more frequently, we can reproduce the motion more accurately.

Algorithm Efficiency. The runtime statistics (for every computation component) of our pose transfer algorithm is reported in Table. 2. Following the pose transfer algorithm formulated in Section 3.4, we can use a threshold to check the convergence of the pose update. Meanwhile, to ensure the efficiency of the algorithm, we can also limit the iteration number to be smaller than k. In our experiments, we found that the iteration usually converges within 10 steps, and setting $k = 10$ produces good enough result. The linear system solving time in Table. 2 consists of the time in solving three linear systems (for x, y, and z coordinates respectively). The linear interpolation between consecutive key poses is instantaneous. Therefore, the total online computation usually finishes within $12 + 0.8 + 1.9 * 10 < 32$ milliseconds.

Linear Interpolation versus More Advanced Morphing Algorithm. When the interval between captured key poses is big, e.g., $k = 3$ in Fig. 14, morphing generated by the simple linear interpolation can have undesirable artifacts. More advanced morphing strategies [43, 44] could be used to generate the interpolation. However, advanced algorithms for animation morphing through calculating more natural animation paths could be noticeably more expensive, and might delay the online synchronization.

4.3 Discussions and Comparisons

We compare our method with the direct surface deformation method, especially, the direct Laplacian deformation. We also compare with the widely adopted skinning-based character manipulation methods

Laplacian coordinates were used to perform direct surface deformation in [6]. The idea can be summarized as minimizing

$$E(V') = \sum_{i=1}^{n} \|T_i(L(v_i)) - L(v'_i)\| + \sum_{i=m}^{n} \|v'_i - u_i\|, \quad (9)$$

where v_i and v'_i indicate the coordinates of the original and deformed vertices, u_i is v_i's target position (given as user's control), L is the Laplacian operator, T_i is a transformation matrix defined on v_i (which needs to be solved) that consists of rotation, translation, and isotropic scaling. The first term penalizes the deviation of Laplacian coordinates caused by the surface deformation. Solving T_i makes the Laplacian based representation invariant to rigid and iso-scaling transformations. The second term is a soft constraint to attract mesh vertices toward their target positions.

In our method, we perform a pose recognition and then directly use the local frames F_i from a model with similar pose. We minimize

$$E(V') = \sum_{i=1}^{n} \|F_i(L(v_i)) - L(v'_i)\|, \text{s.t. } v'_j = u_j, j = 1, \dots, m,$$

where u_i are a set of tracked landmarks on human body surface. The key difference is that without the need to compute transformations T_i, we can reduce the problem to solving linear systems rather than performing a non-linear optimization. Therefore, our approach is significantly faster and can be used in real-time avatar synchronization

Skinning-based methods: Skinning-based animation methods [28, 10] have been widely adopted in

Table 2. Runtime Table for our Pose Transfer Algorithm.

Components	offline/online	runtime (ms)
Laplacian matrix construction	offline	2.1
Cholesky decomposition	offline	3.9
Pose recognition	online	12.0
Local frames calculation	online	0.8
Linear system solving (per iteration)	online	1.9

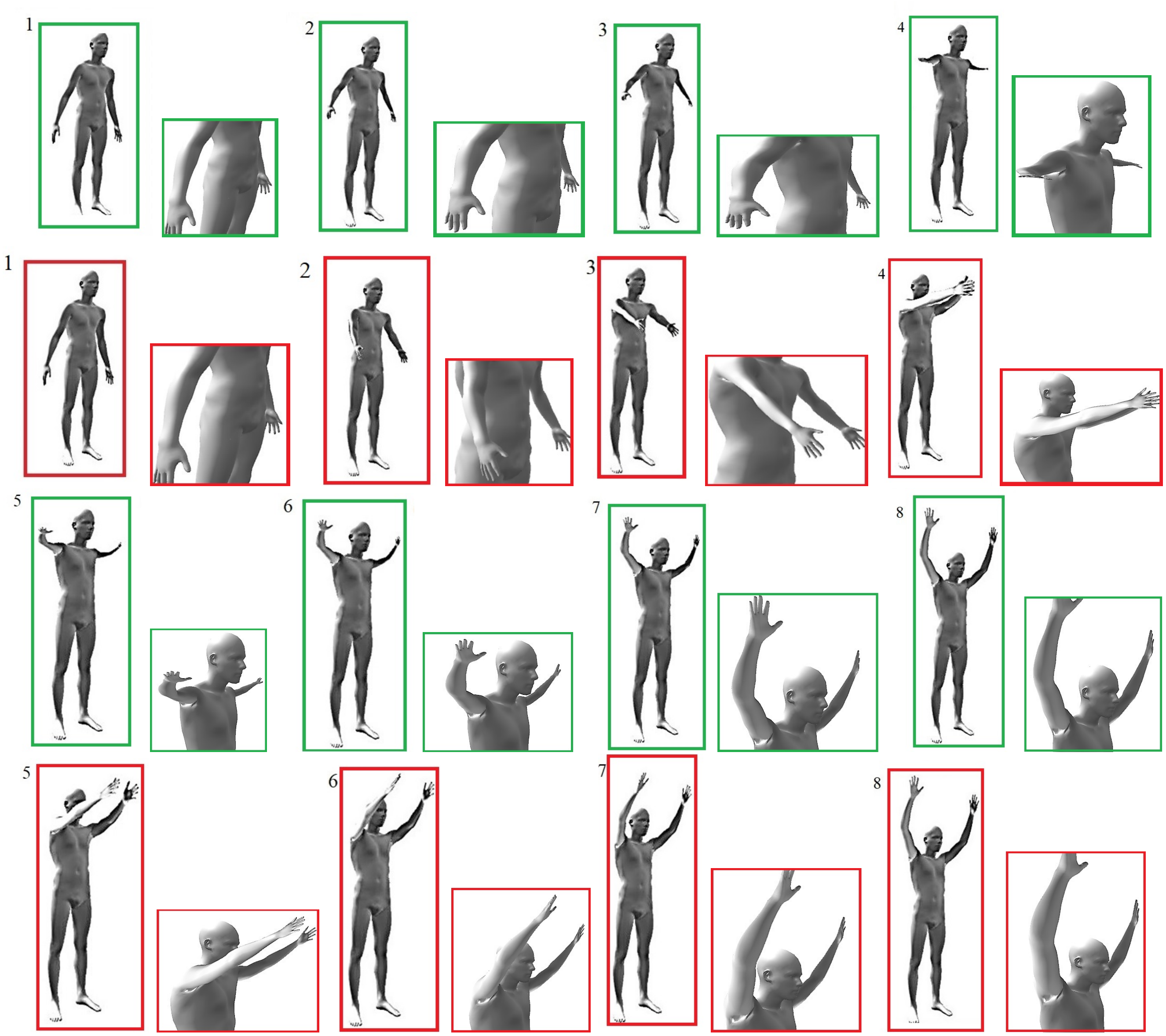


Fig.14. Comparisons of interpolated poses with different key pose intervals: $k = 2$ (green bounding box) and $k = 3$ (red bounding box). A zoomed-in figure is put to the right of each pose for clearer comparison.

generating animations. They usually first do the skinning by extracting skeletal bones and computing bone-vertex association from a sequence of animated meshes, then use the deformation of the skeleton to drive the deformation of surface vertices.

One difficulty for skeleton-driven body deformation is the accurate skeleton tracking from the field. Although commercial APIs from the RGB-D sensors like Kinects have been developed to support skeleton extraction from the field, and recent research on pose estimation from RGB cameras has also made great performance improvement [17], the skeleton tracking is still not always reliable. When the motion is uncommon, dramatic, or there is salient occlusion, tracked skeletons could have missing nodes or incorrect topology. This could affect subsequent animations. Therefore, we use the tracking vest which can more accurately and reliably track a set of landmarks on the body surface, and avoid this problem.

5 Conclusions

We designed a human avatar representation approach for avatar control in virtual reality environments using a wearable body tracking vest. The suggested method consists of two main phases of pose recognition and pose transfer. We developed a pose descriptor which the pose can be effectively estimated. For pose transfer, we adopted an intrinsic coordinates using locally encoded Laplacian offsets. The transfer reduces to solving of sparse linear systems and can be computed rapidly. Using interpolation between the key-poses obtained from previous step, a fast human avatar animation can be achieved.

Limitations and Future Work. Currently, we generate the morphing sequence using the simple linear interpolation. This could lead to artifacts, especially when the two consecutive poses change dramatically. With a denser sampling of the human poses, this could become less an issue. However, processing densely sampled poses requires a big database that contains many more poses. Without sufficient classification of different poses, selected templates for different key poses could be the same, and hence, their interpolation does not help refine the morphing. But with the collection/integration of more human body datasets, this issue will be alleviated gradually.

Skinning-based body deformation is a widely adopted strategy for human motion animation. In general, if the skeleton tracking is accurate, skinning-based methods could better handle the non-isometry deformation than our Laplacian-based deformation. In our future work, we will explore more reliable real-time skeleton tracking algorithm, and also study avatar synchronization through skinning-based deformation for noisy or incomplete skeletons

Acknowledgements

We thank anonymous reviewers for their constructive comments. This work was partly supported by the National Science Foundation IIS-1320959 and the National Natural Science Foundation of China 61728206.

References

[1] Nicolas Robitaille, Philip L. Jackson, Luc J. Hebert, and Catherine Mercier. A virtual reality avatar interaction (vrai) platform to assess residual executive dysfunction in active military personnel. *Disability and Rehabilitation: Assistive Technology*, 12(7):758–764, 2017.

[2] Masoud Zadghorban Lifkooee, Celong Liu, Maoqing Li, and Xin Li. Image-based human character modeling and reconstruction for virtual reality exposure therapy. In *2018 13th International Confer-*

ence on Computer Science & Education (ICCSE), pages 1–5. IEEE, 2018.

[3] Nikilesh Urella, Jennifer Hughes, Erich Conrad, Junsong Zhang, and Xin Li. A vr scene modelling platform for ptsd treatment. In *Computer Science and Education (ICCSE), 2017 12th International Conference on*, pages 257–262. IEEE, 2017.

[4] Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 67:276–286, 2017.

[5] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 741–754. ACM, 2016.

[6] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184. ACM, 2004.

[7] Lin Shi, Yizhou Yu, Nathan Bell, and Wei-Wen Feng. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph.*, 25(3):1108–1117, 2006.

[8] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Subspace gradient domain mesh deformation. In *ACM SIGGRAPH 2006 Papers*, pages 1126–1134, 2006.

[9] Binh Huy Le and Jessica K Hodgins. Real-time skeletal skinning with optimized centers of rotation. *ACM Transactions on Graphics (TOG)*, 35(4):37, 2016.

[10] Edilson De Aguiar, Christian Theobalt, Sebastian Thrun, and Hans-Peter Seidel. Automatic conversion of mesh animations into skeleton-based animations. In *Computer Graphics Forum*, volume 27, pages 389–397. Wiley Online Library, 2008.

[11] Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J Black, and Sung-Hee Lee. Data-driven physics for human soft tissue animation. *ACM Transactions on Graphics (TOG)*, 36(4):54, 2017.

[12] Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph.*, 26(3), July 2007.

[13] Hashim Yasin, Umar Iqbal, Bjorn Kruger, Andreas Weber, and Juergen Gall. A dual-source approach for 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4948–4956, 2016.

[14] Zhao Liu, Jianke Zhu, Jiajun Bu, and Chun Chen. A survey of human pose estimation: the body parts parsing based methods. *Journal of Visual Communication and Image Representation*, 32:10–19, 2015.

[15] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *International Conference on Computer Vision*, volume 1, page 5, 2017.

[16] Francesc Moreno-Noguer. 3d human pose estimation from a single image via distance matrix regression. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1561–1570. IEEE, 2017.

[17] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. volume 36, 2017.

[18] Priovr dev kit. `https://yostlabs.com/priovr/`, Dec. 2018.

[19] Fabio Remondino. 3-d reconstruction of static human body shape from image sequence. *Computer Vision and Image Understanding*, 93(1):65–85, 2004.

[20] Hyewon Seo, Young In Yeo, and Kwangyun Wohn. 3d body reconstruction from photos based on range scan. In *International Conference on Technologies for E-Learning and Digital Entertainment*, pages 849–860. Springer, 2006.

[21] Ke-Li Cheng, Ruo-Feng Tong, Min Tang, Jing-Ye Qian, and Michel Sarkis. Parametric human body reconstruction based on sparse key points. *IEEE transactions on visualization and computer graphics*, 22(11):2467–2479, 2016.

[22] Masoud Z Lifkooee, Ömer M Soysal, and Kazim Sekeroglu. Video mining for facial action unit classification using statistical spatial–temporal feature image and log deep convolutional neural network. *Machine Vision and Applications*, pages 1–17, 2018.

[23] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3d skeletons: A state-of-the-art report. *Comput. Graph. Forum*, 35(2):573–597, May 2016.

[24] Punam K. Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. A survey on skeletonization algorithms and their applications. *Pattern Recognition Letters*, 76:3 – 12, 2016. Special Issue on Skeletonization and its Application.

[25] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007.

[26] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph.*, 28(3):71:1–71:9, 2009.

[27] Qian Zheng, Andrei Sharf, Andrea Tagliasacchi, Baoquan Chen, Hao Zhang, Alla Sheffer, and Daniel Cohen-Or. Consensus skeleton for non-rigid space-time registration. In *Computer Graphics Forum*, volume 29, pages 635–644. Wiley Online Library, 2010.

[28] Doug L James and Christopher D Twigg. Skinning mesh animations. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 399–407. ACM, 2005.

[29] Matteo Colaianni, Michael Zollhoefer, Jochen Suessmuth, Bettina Seider, and Günther Greiner. A pose invariant statistical shape model for human bodies. In *Proceedings of the 5th International Conference on 3D Body Scanning Technologies*, pages 327–336, 2014.

[30] Stefanie Wuhrer, Chang Shu, and Pengcheng Xi. Posture-invariant statistical shape analysis using laplace operator. *Computers & Graphics*, 36(5):410–416, 2012.

[31] T. A. Davis and W. W. Hager. Dynamic supernodes in sparse cholesky update//downdate and triangular solves. *ACM Trans. Math. Software*, 35(4), 2009.

[32] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014.

[33] Scape: Shape completion and animation of people. `https://ai.stanford.edu/~drago/Projects/scape/scape.html`, Dec. 2018.

[34] Human3.6m: 3.6 million 3d human poses and corresponding images. `http://vision.imar.ro/human3.6m/description.php`, Dec. 2018.

[35] Zongyi Xu, Qianni Zhang, and Shiyang Cheng. Multilevel active registration for kinect human body scans: from low quality to high quality. *Multimedia Systems*, 24(3):257–270, 2018.

[36] Caesar: The most comprehensive source for body measurement data. `https://store.sae.org/caesar/`, Dec. 2018.

[37] Shrec'14: Shape retrieval of non-rigid 3d human models. `http://www.cs.cf.ac.uk/shaperetrieval/shrec14/`, Dec. 2018.

[38] The stitched puppet: A graphical model of 3d human shape and pose. `http://stitch.is.tue.mpg.de`, Dec. 2018.

[39] X. Li and S. Iyengar. On computing mapping of 3d objects: A survey. *ACM Computing Surveys*, 47(2):34:1–34:45, 2015.

[40] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707.

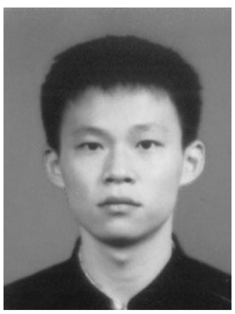
[41] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015.

[42] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016.

[43] Marc Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–198, 2002.

[44] Zhiguang Liu, Liuyang Zhou, Howard Leung, Franck Multon, and Hubert PH Shum. High quality compatible triangulations for planar shape animation. In *Proceedings of the ACM SIGGRAPH ASIA Workshop*, pages 1–8, 2017.

Masoud Zadghorban Lifkooee Received an MSc degree in electrical engineering from University of Guilan in 2013 where he worked on image processing and pattern recognition projects such as sign language recognition. Then, he moved to Louisiana State University to continue his research in 2015 where he worked on some machine learning and machine vision projects such as vehicle classification and facial expression recognition. Now he is currently working with Dr. Xin Li in Louisiana State University department of computer engineering on human body modeling for virtual reality environments and especially to be used in the Cave Automatic Virtual Environment.



Celong Liu received the B.E. degree in Engineering Physics and M.S. degree in Physics from Tsinghua University, Beijing in 2011 and 2014. He is currently a Ph.D. candidate in the School of Electrical Engineering and Computer Science, Louisiana State University. His research interests include computer graphics, computational geometry, and computer vision.

Yongqing Liang received his Bachelor?s degree in computer science and technology from Fudan University in 2017. He is currently pursuing his Ph.D. degree in electrical computer engineer in Louisiana State University, advised by Dr. Xin Li. His research interests include computer vision, computer graphics and machine learning.

Yimin Zhu is Professor and holder of the Pulte Homes Endowed Professorship in the Bert S. Turner Department of Construction Management at Louisiana State University. Dr. Zhu received his Ph.D. degree from the M. E. Rinker, Sr. School of Construction Management at the University of Florida in 1999. He served as Assistant Professor from 2004 to 2010, and Associate Professor and Graduate Program Director from 2010 to 2013 in the School of Construction at Florida International University. He joined the Department of Construction Management at Louisiana State University in 2014. Between 2000 and 2001 he was a visiting professor in the Department of Building Construction at Georgia Institute of Technology.

Xin Li is the Oskar R. Menton associate professor in School of Electrical Engineering and Computer Science at Louisiana State University (LSU). He received his B.Eng. in Computer Science from University of Science and Technology of China in 2003, and Ph.D. in Computer Science from State University of New York at Stony Brook in 2008. He led the Geometric and Visual Computing lab at LSU. His research interests include geometric and visual computing, geometric data modeling and processing, and their applications in graphics, vision, robotics, and forensics. More details can be found at http://www.ece.lsu.edu/xinli.