Template-Based Reconstruction of Surface Mesh Animation from Point Cloud Animation

Sang Il Park and Seong-Jae Lim

In this paper, we present a method for reconstructing a surface mesh animation sequence from point cloud animation data. We mainly focus on the articulated body of a subject — the motion of which can be roughly described by its internal skeletal structure. The point cloud data is assumed to be captured independently without any inter-frame correspondence information. Using a template model that resembles the given subject, our basic idea for reconstructing the mesh animation is to deform the template model to fit to the point cloud (on a frame-by-frame basis) while maintaining inter-frame coherence. We first estimate the skeletal motion from the point cloud data. After applying the skeletal motion to the template surface, we refine it to fit to the point cloud data. We demonstrate the viability of the method by applying it to reconstruct a fast dancing motion.

Keywords: Point cloud animation, mesh animation, character animation.

Manuscript received Nov. 25, 2013; revised June 2, 2014; accepted June 24, 2014.

The research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012 R1A1A1014702), and this work was supported by the IT R&D Program of MSIP/IITP (10047093, 3D Content Creation and Editing Technology Based on Real Objects for 3D Printing).

Sang Il Park (sipark@sejong.ac.kr) is with the Department of Digital Contents, Sejong University, Seoul, Rep. of Korea.

Seong-Jae Lim (corresponding author, sjlim@etri.re.kr) is with the SW·Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

I. Introduction

The acquisition of dynamically varying geometry data from a deforming object has been an active research area in computer graphics. Conventional methods to capture such a time-varying shape are mostly based on vision techniques. Using synchronized multiple cameras in a controlled studio, the 3D surface data at each instance are captured by exploiting various stereo matching methods. The details of the process can differ according to the choice of hardware setups and implementations. However, the representation of their resulting data usually shares the same form, which is the three-dimensional point cloud. The point cloud has primarily been used for capturing a static object. However, it also facilitates the representation of animation data by concatenating the resulting point clouds, captured individually at each frame, in temporal order.

Although the point cloud representation has proven useful for storing and rendering with tremendous geometric details, it is originally intended to be used for a static object. When it comes to a dynamic object, several issues may arise — for example, the compression of large-size data for efficient memory use and the construction of inter-frame temporal coherence for effective post-processing (pose-editing or re-texturing).

On the other hand, polygonal surface representations, such as triangular meshes, have been more popular for animation purposes, in which an animation is represented by the trajectories of vertices while keeping the connectivity between such vertices unchanged. Moreover, there are already many research results available detailing how to effectively compress and manipulate such mesh animation data [1]–[2]. Thus, the option to use point cloud data for animation purposes is

http://dx.doi.org/10.4218/etrij.14.0113.1181

proving increasingly more attractive.

In this paper, we present a method for converting point cloud animation data into mesh animation data with constant connectivity. We take the point cloud animation sequences without the inter-frame coherency as an input. Our problem is different from the previous work [3]–[4] for generating mesh animation from images captured from synchronized multiple video cameras, in which it is assumed that the silhouette information can be extracted easily without much noise. In our case, however, the point cloud data are already processed and can contain severe noise especially around the region of the occlusion, which is not easily separable. Thus, noise-robust reconstruction is the main issue to be addressed in our method. In addition, the full-body point cloud animation data often lacks details due to the limited number of full-body-shot cameras. It is, therefore, desirable to augment the details of the captured data. Our basic idea for a robust and consistent transformation is to use a template model. The template model is not necessarily to be the same subject to be captured. However, the internal skeletal structure is assumed to be the same. The use of a template model results in several benefits: first, the template model enables us to easily build a temporal coherence by registering its geometric elements to the unstructured point clouds at each frame. Second, we can archive more details to the original captured data. Finally, the template model helps to remove the noises existing in the original input data.

We address three main issues: first, we present a method to adjust the given general template model to better fit the geometry of a particular subject. Second, we recover the motion of the subject and transform it into a mesh animation. Finally, we reduce the visual artifacts of the obtained mesh animation by applying a local deformation filter.

The remainder of the paper is organized as follows: we first discuss related work in Section II. We then briefly give an overview of our method in Section III. The method consists of three main parts, each of which is described in detail in Section IV. In the last two sections, we provide our experimental results and discuss the advantages and limitations of the method.

II. Related Work

Capturing animation data from a deforming object has been an important problem in computer animation for many years. Here, we briefly summarize the work most related to our own by mainly focusing on the dynamic full-body deformation acquisition. The techniques can be roughly categorized into two classes: template-based methods and non-template-based methods. Our method falls into the former category.

Template-based methods exploit a template model as a prior of the subject to be captured. Template models have been widely used mostly for estimating the correspondences and for augmenting missing information in the captured data. The first full-body dynamic capture of human subjects was done by Sand and others, in 2003 [5]. Although they did not use a geometry model as a template, a predefined skeletal structure together with a primitive deformation model was given to assist in the estimation of the configuration of the subjects. Allen and others built a general template model to estimate the deformation space with respect to different body shapes [6]. In their work, they defined a small set of predefined landmarks on the template model for easily estimating correspondences between captured data. For the same purpose, the predefined landmarks have been commonly used in other methods, such as those in [7]–[9]. However, they were mostly for acquiring static objects or a few sequences of dynamic objects, due to the complexity of their automatic method of building correspondences. In 2008, two interesting works were presented simultaneously by two different groups; their methods having similar capture setups. One was by Vlasic and others [4], while the other was by de Aguiar and others [3]. Both use template models extensively for correspondences and filling missing information. Especially, [3] exploited a lower resolution version of the volumetric template model for effectively tracking fast and complex non-rigid motion. Most recently, Li and others presented a method for capturing a complex dynamic motion only in a single-view setup [10]. They also used their template at a lower resolution so as to achieve robustness and efficiency of captures.

Non-template-based methods do not use models or predefined correspondences. Mitra and others suggested to use a set of frames with dense spatial and temporal data directly to compute the motion of a scanned object [11]. In [12], Sussmuth and others computed a four-dimensional implicit surface approximating the input point cloud animation and reconstructed a polygonal mesh animation in an as-rigid-as-possible manner. Wand and others used a deformable matching based on a statistical optimization for the simultaneous estimation of the shape and the motion [13]. In their following work [14], they improved its efficiency by separately handling the shape and the motion in the optimization. Most recently, Tevs and others presented a method that first detected a small set of landmarks commonly shown in the given point cloud sequences and then extended them to find a dense set of correspondences [15]. Those methods usually assume that the point cloud data have enough details with less noise. However, our data lack details and contain many noises.

III. Overview

We take the sequence of the point cloud from a non-rigidly deforming subject as input data, as shown in Fig. 1, together with a polygonal surface model as a template. Because a human subject is our main interest, we use a general human-surface model with 4,000 vertices, commercially available in [16], for the template. We might exploit a three-dimensional scanner to get the exact, detailed geometry from the subject. However, here, we consider a general scenario in which only the dynamically captured data are available. The template surface model is pre-processed so as to have linear blend skinning (LBS) weights for each vertex; hence, an articulated deformation can be applied. Figure 2(c) shows our initial surface model and the assigned LBS weights, where we segment the whole body into 17 body parts.

The point cloud data consist of a large number of points. For example, the average number of points per frame is about 32,000 in our data set. Dealing with such a large number of points is not efficient when attempting to perform manipulations on them. Thus, we process the point cloud data individually at each frame so that it can be represented in a multi-resolution manner. This process is important, otherwise the following processes would not be feasible because of the required computation time.

To generate surface animation with the given data, our method consists of two separate processes: one for the first frame, and the other for the remaining frames, as illustrated in Fig. 3. We first register the template surface to the point cloud data at the first frame. For an effective registration, the subjects were asked to start each capture session with a pose similar to that of the template model. Additionally, we provide pairs of corresponding features between the template and the point cloud. In our experiments, we define 44 correspondences on the facial features, such as nose, eyes, and mouth, and boney landmarks of the body, such as knees, elbows, shoulders, and so on.

After the first-frame registration, we sequentially generate a matching surface model to the point cloud at a given frame by deforming the surface model obtained at the previous frame.

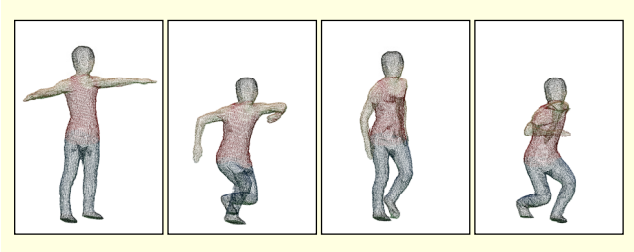

Fig. 1. Excerpts from the input point cloud animation data of the shuffle dance. Each frame consists of about 32,000 points.

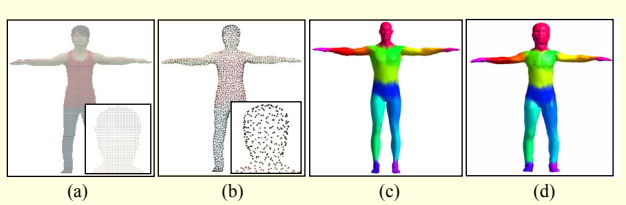

Fig. 2. First-frame registration: (a) point cloud data at first frame with 34,720 points, (b) reduced point cloud data with 2,000 points, (c) given template surface model, and (d) registered surface model of first frame.

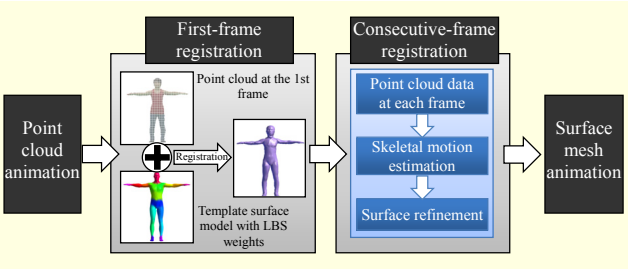

Fig. 3. Overview of system.

This per-frame registration consists of two stages: we first estimate the skeletal structure by computing the joint angles resembling the pose of the given data. The skeletal information is used in the application of the LBS weights to the template model for the generation of an initial estimate to the given frame. Then, we find the best per-vertex deformation of the surface in which the difference between the point cloud and the surface model is minimized while keeping the smoothness of the surface.

Although the temporal coherence between consecutive frames is loosely considered in the skeleton estimation stage, there still exist artifacts of the temporal inconsistency in the resulting mesh animation. By exploiting the estimated skeletal motion, we reduce the artifacts in a post process while reflecting the non-rigid deformation.

IV. Method

1. Data Representation

The point cloud animation data consist of a sequence of the point cloud. Each point cloud is captured individually at each frame and includes a large number of points. Among these points, there exists neither time coherence between frames nor spatial relationship, such as neighboring information at a frame. We denote the point cloud at a given frame t as a set $P^t = \left\{ \mathbf{p}_1^t, \mathbf{p}_2^t, \ldots, \mathbf{p}_{N_p^t}^t \right\}$, where N_p^t is the number of points and $\mathbf{p}_i^t$ is the position of the ith point. The average value of N_p^t in our experiments is about 32,000. For estimating basic

ETRI Journal, Volume 36, Number 6, December 2014
http://dx.doi.org/10.4218/etrij.14.0113.1181

surface properties, such as a normal and curvature, we store the indices of the closest K neighbors for each point based on the Euclidian distance.

Such a large number of points slow down basic operations on the points, such as searching. It also makes it hard to reflect global features. Thus, for better efficiency and effectiveness, we construct a separate lower-resolution version of the data in a multi-resolution manner based on the method of [17] as follows: we first cluster a set of points such that they are close to each other as well as close to the 3D plane approximating the point distribution in the cluster. In this way, we can get a bigger cluster on the planar region and a smaller cluster around a higher curvature region. Then, by picking the point, for each cluster, closest to its center as a representative, we build a lower-resolution version of the point cloud. We denote this smaller version at frame t as $\hat{P}^t = \left\{\hat{\mathbf{p}}_1^t, \hat{\mathbf{p}}_2^t, \ldots, \hat{\mathbf{p}}_{\hat{N}_p^t}^t\right\}$ with $\hat{N}_p^t$ points. Our average value of $\hat{N}_p^t$ is about 2,000. Figure 2(b) shows the reduced point cloud at the first frame. We use the lower-resolution version when a rough estimate is adequate, such as in the early stages of the optimization, and use the full version when greater accuracy is required.

The template surface model M is composed of the vertices and their connectivity information. For simplicity of explanation, the template model is assumed to be a triangular mesh. Then, it can be represented as a set of vertex positions and a set of vertex index triples representing a triangle. We denote the position of the ith vertex as $\mathbf{v}_i$, $1 \le i \le N_v$, where N_v is the number of vertices in the mesh, and a triple for triangle j as $\mathbf{t}_j = \left\{t_j^1, t_j^2, t_j^3\right\}$. Each vertex i is assigned with a set of LBS weight values, $\alpha_{i,s}$, $1 \le s \le N_s$, where $\alpha_{i,s}$ is the weight value of vertex i for body part s, N_s is the total number of body parts, and $\sum_{s=1}^{N_s} \alpha_{i,s} = 1$. In our experiments, N_s is 17, as shown in Fig. 2(c), where different colors are used to indicate different parts.

2. First-Frame Registration

We register the template surface model M to the point cloud P^1 at the first frame by deforming the surface. We adopt the optimization framework of Allen and others [2]: we first find the global transformation of M to have the best fit with P^1. The global transformation is defined as a combination of translation and rotation followed by scaling. After the global matching, we optimize the local deformation at each vertex. We represent the local deformation at each vertex as a 3 × 4 affine transformation matrix, which we denote as $\mathbf{A}_i$ for vertex i.

For an effective matching, we assign pair-wise feature correspondences between selected vertices in M and points in P^1. We denote the set of the feature vertex indices of the surface and its corresponding set for the point cloud as K_j^M and K_j^P, respectively, where $1 \le j \le N_c$, and N_c is the total number of correspondences. We select 44 feature pairs for the correspondence in our experiments. Note that this assignment is done just once for the first frame.

The global matching is performed by considering only these corresponding pairs. The global translation is easily found by computing the displacement between the center of the chosen vertices and that of the corresponding points. The global rotation is obtained by using the method of Horn [18] for the absolute orientation problem. Finally, we apply the least squares fitting to find the global scaling value. We denote the position of vertex i after the global transform as $\tilde{\mathbf{v}}_i$.

The optimization for the local deformation is to find the best affine transform of every vertex that minimizes the combination of the following three error terms: the first is the data error E_d for measuring gaps between all the points and the surface, the second is the smoothness error E_s for keeping the original details of the surface, and last is E_f, which is for reflecting the user-given corresponding features during optimization. Please note that we mostly follow the concept of the three error terms from [6]. However, we customize the data error term to be applicable to our point cloud data. We give a detailed description of the aforementioned error terms in the following subsections.

A. Data Error

The main purpose of the registration is to make the shape of the template model resemble the given point cloud. We can measure the closeness of this matching by computing the distances from each point to the closest surface. Because our template model is a triangular mesh, we first find the closest triangle to each point from the point cloud. For the closest triangle $\mathbf{t}_{c_i}$ to the point $\mathbf{p}_i^1$, we can also compute the closest position on the triangle and represent it in its barycentric coordinate $\left(w_{c_i}^1, w_{c_i}^2, w_{c_i}^3\right)$ with the three vertices of the triangle. Then, the data error becomes

$$E_d = \sum_{i=1}^{N_p^1} \left\| \mathbf{p}_i^1 - \sum_{k=1}^{3} w_{c_i}^k \left(\mathbf{A}_{t_{ci}^k} \cdot \tilde{\mathbf{v}}_{t_{ci}^k} \right) \right\|^2 ,$$

where N_p^1 is the number of the points in the point cloud at the first frame.

B. Feature Error

This error is designed to reduce the distance between the corresponding pairs, which guides the resulting surface to have a contextual matching to the point cloud. This term is

especially necessary when the shape of both the template and the point cloud are not initially close enough to each other. By denoting the vertex index of the *j*th pair of the correspondences as K_j^M and the index of the point as K_j^P, the error is defined as follows:

$$E_f = \sum_{j=1}^{N_c} \left\| \mathbf{A}_{K_j^M} \cdot \tilde{\mathbf{v}}_{K_j^M} - \mathbf{p}^1_{K_j^P} \right\|^2 .$$

C. Smoothness Error

The local details can be preserved if a vertex and its connected neighboring vertices undergo a similar local deformation to each other. Thus, we measure the local change by comparing the difference in the affine transform matrices of the neighboring vertices as follows:

$$E_s = \sum_{i=1}^{N_v} \left(\sum_{k=1}^{d_i} \left\| \mathbf{A}_i - \mathbf{A}_{i,k} \right\|_F^2 \right),$$

where $\| \cdot \|_F$ is the Frobenius norm, $\mathbf{A}_{i,k}$ is the affine matrix of the *k*th 1-ring neighbor of vertex *i*, and d_i is the number of the 1-ring neighbors.

D. Optimization

Considering the above three error terms, the total error E is the weighted combination of them. That is,

$$E = \omega_d E_d + \omega_f E_f + \omega_s E_s ,$$

where ω_d, ω_f, and ω_s are the respective weight values. We run the optimization based on the conjugate gradient method. Initially, we assign weights only to ω_f and ω_s so that the model is registered globally first. After the convergence, we perform the optimization again with equal weights for all three terms.

3. Consecutive-Frame Registration

We use the registered model for the first frame as the new template model for the rest of the frames because the model is already specific to the subject. Our registering process consists of three subprocesses. We first estimate the skeletal motion for all the frames, which provides a good initial estimate for the surface. Then, we optimize the local deformation to fit to the point cloud data. Finally, we reduce the noise from the resulting surface animation by using inter-frame coherency.

A. Skeletal Motion Estimation

From the first frame registration, we estimate the skeletal structure of the subject, including the joint positions and length of bones. Because the skeletal motion is used for the rough initial estimate and for the later local registration, it is not required to be very accurate. For this reason, we assume that the skeletal structure is approximated from the given LBS weight values, by which the number of bones is the same as the number of body parts, and whereby the joint positions are located at the center of the boundary between the neighboring body parts.

While keeping the length of the bone constant, we estimate the joint configuration for each frame, which is parameterized with the position of the root joint and rotations of all the joints. Starting from the second frame, we sequentially find the best joint configuration by altering one joint from the previous frame such that after applying LBS to the template surface, the distance between the point cloud and the surface should be minimized while keeping a minimal change to the previous frame. We exploit a gradient-based method for the optimization.

B. Surface Refinement

With the skeleton pose at a given frame, we first apply LBS to the template surface model to get the initial surface for the optimization. Then, we find the best local deformation by minimizing the error term, in a manner similar to that given in Section IV-2, with the exception of the correspondence term, which is only available at the first frame.

$$E = \omega_d E_d + \omega_s E_s .$$

C. Post Noise Reduction

Our process of registration is basically a per-frame process, in which each frame is considered independently except for the estimation of the skeletal motion. Thus, inevitably it can lose inter-frame consistency, which results in a jerkiness in the motion of each vertex. Visually, this severely degrades the quality of the resulting animation. To remedy the artifacts, we present a noise reduction process.

From our observations, the artifacts are more obvious when the motion of a vertex is different from that of its neighboring vertices. Borrowing an idea from multi-resolution signal processing, we divide the motion of a vertex into two levels: one is the global motion, and the other is the local motion. Because of the assumption of the articulated body for the subject, we define the global motion for each vertex as the motion resulting solely from the LBS-weighted skeletal motion. Then, the local motion can be defined as the difference between the computed motion and the global motion. Given the position $\mathbf{v}_i^t$ of the *i*th vertex of the resulting surface at frame *t*, we denote the position of the vertex obtained by only applying the skeletal motion to the template model as $\mathbf{v}_i^{\text{global}\,t}$.

http://dx.doi.org/10.4218/etrij.14.0113.1181

Then, the local position $\mathbf{v}_i^{\text{local}^t}$ is computed as follows:

$$\mathbf{v}_i^{\text{local}^t} = \mathbf{v}_i^t - \mathbf{v}_i^{\text{global}^t}.$$

Because the global motion is smooth due to the smooth skeletal motion, we perform the temporal smoothing operation on the local position. We use a Gaussian filter with a kernel size of five frames. By doing this, we keep the global motion unchanged and reduce the locally abrupt movement of the vertices.

V. Experimental Results

We implemented the algorithm using C++ with Windows 7. The experiments were performed on a PC with an Intel i7 CPU at 3.2 GHz with 8 GB of memory. We applied our method to reconstruct the fast dancing motion of the actor shown in Fig. 1. The total number of frames is 300, and each frame consists of about 35,000 points. Our template surface model, as shown in Fig. 2, is made up of 4,000 vertices. The resulting mesh animation is given in Fig. 4. As explained in Section IV, we first estimate the skeletal motion, as shown in the middle row of Fig. 4. After applying the skeletal motion to the template surface, we refine it to fit to the point cloud, as shown in the bottom row of Fig. 4. The average processing time taken for each frame is about 20 seconds. The resulting movie files can be found at http://dasan.sejong.ac.kr/~sipark/pointcloud/. Because of the use of a template model, our method can reduce the noises existing in the original data, originating from the visual occlusions, during the capture session. As shown in Fig. 5, there exist artifacts near the left armpit, which are removed in our reconstructed mesh animation. Additionally, the template surface model augments the details to the reconstructed animation. For example, our resulting mesh animation has detailed geometries around the fingers and the ears.

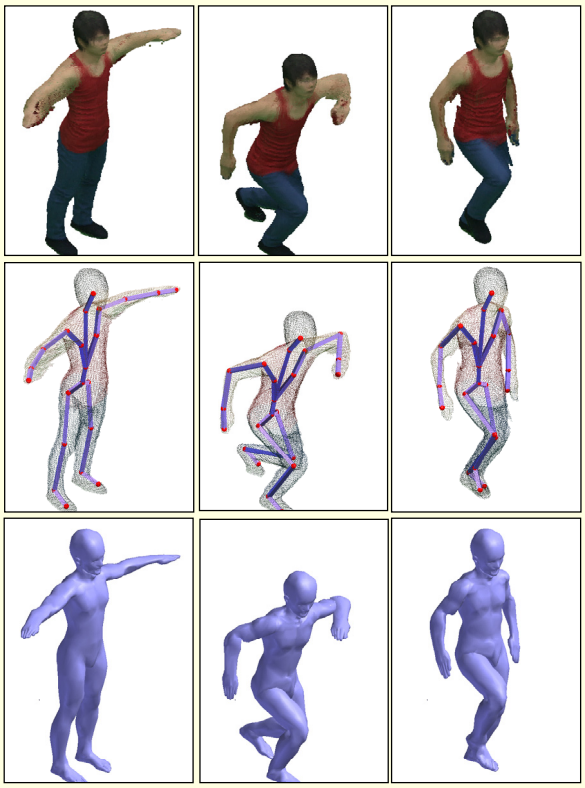

Fig. 4. Our experimental results. Top row: input point cloud data; middle row: estimated skeletal poses; bottom row: reconstructed mesh animation.

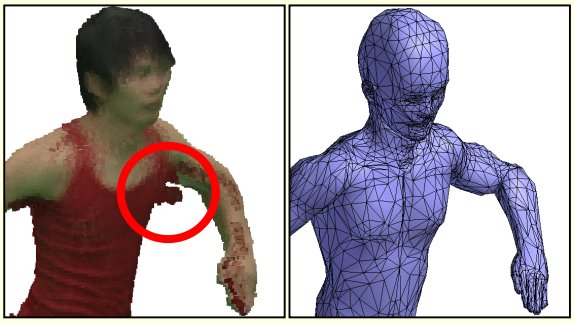

Fig. 5. Close-up view. Input point cloud data consisting of noise due to occlusions during the capture. However, our reconstruction reduces the noise through the use of a template model.

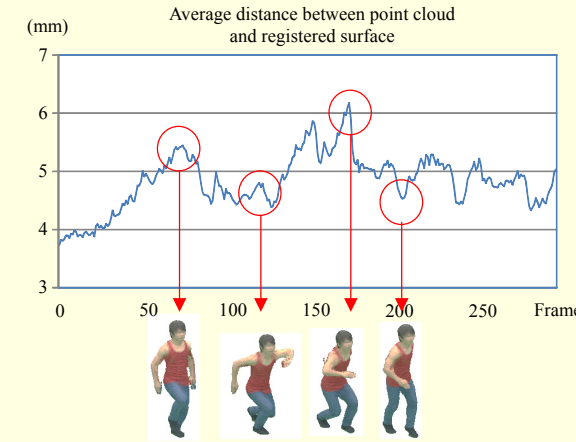

Fig. 6. Reconstruction error.

Figure 6 shows the reconstruction error for all frames. The error is measured as the average distance between the point cloud and the reconstructed surface. The errors range from 3.7 mm to 6.2 mm.

VI. Discussion

In this paper, we present a method for converting point cloud

animation data into mesh animation data. Our main idea is to use a template surface model with an internal skeletal structure for tracking and estimating the time-varying geometry of the articulated subject. We demonstrate the viability of the method by applying it to a fast dancing motion.

Our method has several limitations. First, our method sequentially finds the skeletal motion by advancing the frames one at a time. Thus, if any failures happen in a frame, then the error can be accumulated in the following frames. To fix this problem, one possible solution is to allow the user to adjust the joint configuration interactively during the process, as Vlasic and others did in [4]. Second, the smoothness error term in the optimization may prevent local deformations of the surface. This is fine when noise exists in the input data. However, it can also remove desired deformations, such as wrinkles on clothes. Finally, we assume that the template mesh and its skeletal structure are given by the user. However, for more usability, it would be desirable if the system could estimate those data from the point clouds so that it can be applied to an arbitrary subject; for example, animals with different skeletal structures [19].

References

[1] D.L. James and C.D. Twigg, "Skinning Mesh Animations," *ACM Trans. Graph.*, vol. 24, no. 3, July 2005, pp. 399–407.

[2] S. Kircher and M. Garland, "Editing Arbitrarily Deforming Surface Animations," *ACM Trans. Graph.*, vol. 25, no. 3, July 2006, pp. 1098–1107.

[3] E. de Aguiar et al., "Performance Capture from Sparse Multiview Video," *ACM Trans. Graph.*, vol. 27, no. 3, 2008, pp. 98:1–98:10.

[4] D. Vlasic et al., "Articulated Mesh Animation from Multi-view Silhouettes," *ACM Trans. Graph.*, vol. 27, no. 3, 2008, pp. 97:1–97:9.

[5] P. Sand, L. McMillan, and J. Popovic, "Continuous Capture of Skin Deformation," *ACM Trans. Graph.*, vol. 22, no. 3, July 2003, pp. 578–586.

[6] B. Allen, B. Curless, and Z. Popovic, "The Space of Human Body Shapes," *ACM Trans. Graph.*, vol. 22, no. 3, July 2003, pp. 587–594.

[7] D. Anguelov et al., "Recovering Articulated Object Models from 3D Range Data," *Proc. Conf. Uncertainty Artif. Intell.*, Banff, Canada, 2004, pp. 18–26.

[8] D. Anguelov et al., "Scape: Shape Completion and Animation of People," *ACM Trans. Graph.*, vol. 24, no. 3, July 2005, pp. 408–416.

[9] A.M. Bronstein, M.M. Bronstein, and R. Kimmel, "Generalized Multidimensional Scaling: A Framework for Insometry-Invariant Partial Surface Matching," *Proc. National Academy Sci.*, vol. 103, no. 5, 2006, pp. 1168–1172.

[10] H. Li et al., "Robust Singleview Geometry and Motion Reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, Dec. 2009, pp. 174:1–175:10.

[11] N.J. Mitra et al., "Dynamic Geometry Registration," *Proc. Eurographics Symp. Geometry Process.*, Barcelona, Spain, 2007, pp. 173–182.

[12] J. Sussmuth, M. Winter, and G. Greiner, "Reconstructing Animated Meshes from Time-Varying Point Clouds," *Proc. Symp. Geometry Process.*, Copenhagen, Denmark, 2008, pp. 1469–1476.

[13] M. Wand et al., "Efficient Reconstruction of Non-rigid Shape and Motion from Real-Time 3D Scanner Data," *ACM Trans. Graph.*, vol. 28, no. 2, Apr. 2009, pp. 15:1–15:15.

[14] M. Wand et al., "Reconstruction of Deforming Geometry from Time-Varying Point Clouds," *Proc. Symp. Geometry Process.*, Barcelona, Spain, 2007, pp. 49–58.

[15] A. Tevs et al., "Animation Cartography – Intrinsic Reconstruction of Shape and Motion," *ACM Trans. Graph.*, vol. 31, no. 2, Apr. 2012, pp. 12:1–12:15.

[16] CGHuman, CGCharacter, 2013. Accessed Aug. 26, 2013. http://www.cgcharacter.com/cghuman.html

[17] M. Pauly, L. Kobbelt, and M.H. Gross, "Multi-resolution Modeling of Point-Sampled Geometry," ETH CS Technical Report, no. 373, 2002.

[18] B. Horn, "Closed-Form Solution of Absolute Orientation Using Unit Quaternions," *J. Opt. Soc. America*, vol. 4, no. 4, Apr. 1, 1987, pp. 629–642.

[19] M. Sung, "Fast Motion Synthesis of Quadrupedal Animals Using a Minimum Amount of Motion Capture Data," *ETRI J.*, vol. 35, no. 6, Dec. 2013, pp. 1029–1037.

Sang Il Park received his PhD degree in computer science from the Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in 2004. After working for about two years as a postdoctoral fellow at Carnegie Mellon University, Pittsburgh, PA, USA, and then at the National Institute of Advanced Industrial Science and Technology, Tokyo, Japan, he is currently affiliated with the Department of Digital Contents, Sejong University, Seoul, Rep. of Korea, as an assistant professor. His primary research interests are synthesizing character animation and visually simulating natural phenomena.

Seong-Jae Lim received his PhD degree in information and communication engineering from the Gwangju Institute of Science and Technology, Gwangju, Rep. of Korea, in 2006. From 2004 to 2005, he was with the University of Pennsylvania, Philadelphia, USA, as a visiting scholar at the Medical Image Processing Laboratory. He is currently a senior researcher of the Creative Content Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea. His primary research interests are sensor-based human modeling, deformation, and animation.