Heterogeneous Deformation Model for 3D Shape and Motion Recovery from Multi-Viewpoint Images

Shohei Nobuhara   Takashi Matsuyama

Graduate School of Informatics, Kyoto University
Sakyo, Kyoto, 606-8501, Japan
E-mail: {nob,tm}@vision.kuee.kyoto-u.ac.jp

Abstract

This paper presents a framework for dynamic 3D shape and motion reconstruction from multi-viewpoint images using a deformable mesh model. By deforming a mesh at a frame to that at the next frame, we can obtain both 3D shape and motion of the object simultaneously. The deformation process of our mesh model is heterogeneous. Each vertex changes its deformation process according to its 1) photometric property (i.e., if it has prominent texture or not), and 2) physical property (i.e., if it is an element of rigid part of the object or not). This heterogeneous deformation model enables us to reconstruct the object which consists of different kinds of materials or parts with different motion models, e.g., rigidly acting body parts and deforming soft clothes or its skins, by a single and unified computational framework.

1. Introduction

In recent years, many studies have been done on 3D shape and motion reconstruction. For static 3D shape reconstruction, several frameworks combining multiple cues such as photometric stereo or silhouette were proposed to accomplish better stability and accuracy. Fua [7] represented object shape by 2.5D triangular mesh model and deformed it based on photometric stereo and silhouette constraint. Cross [6] carved visual hull, a set of voxels given by silhouettes, using photometric property. For 3D motion recovery, Heap [8] proposed human hand tracking from camera images using a given deformable hand model. Bottino [2] tracked 3D human action from multi-viewpoint silhouettes with a known object model. Vedula [18] introduced a framework to compute dense 3D motion flow from optical flows with / without object shape.

The problem we consider in this paper is how we can reconstruct *dynamic* 3D shape from multi-viewpoint images. That is, we focus on how to recover the shape and motion of the object *simultaneously*. A naive method for this problem would be: Step 1. reconstruct 3D shape for each frame, Step 2. estimate 3D motion by establishing correspondences between a pair of 3D shapes at frames t and $t+1$. However, this approach consists of two-stage computational model and it it not so easy to manage each stage to cooperate with the other. We believe that a unified computational model, i.e., simultaneous recovery of 3D shape and motion, is better than the two-stage approach. Toward the simultaneous recovery, for example, Vedula [19] showed an algorithm to recover shapes represented by voxels in 2 frames and per-voxel-correspondence between them simultaneously. Fua [15] proposed a method which uses a soft object model and refine it based on photometric and silhouette constraints.

In [13, 14], a unified computation algorithm using a deformable mesh model [9] was presented. This algorithm represents the shape by a surface mesh model and the motion by translations of its vertices, i.e., deformation from a frame to the next. This deformation based approach can preserve global and local topological structure of the mesh from frame to frame, which provides dense, per-vertex correspondences between two consecutive frames. This is useful for inter-frame 3D data compression [3], motion analysis, and so on. However, this simple per-vertex-deformation can not cope with long term reconstruction or global change of the topological structure of the 3D object shape. This is because 1) per-vertex-deformation approach is purely local, bottom-up and data-driven, and 2) we can not track all the vertices from frame to frame, that is, not all the vertices are *identifiable* or *localizable* on the object surface.

In this paper, we present a framework using *heterogeneous* mesh model to solve these problem. Our heterogeneous deformable mesh model deforms its shape by per-vertex transition governed by a force working at each vertex. In an ordinary mesh deformation model, the forces at each vertex are generated by the same process, and once the forces are given, the vertices deform in a uniform manner, e.g., applying Newton's second law uniformly for each vertex. However, our heterogeneous mesh model changes the force-generation and vertex-deformation rule at each vertex based on its physical and photometric properties. That is, the vertex of our heterogeneous mesh model represents not only the position of the object surface, but also the motion model and the surface characteristic of the object. According to these additional properties of vertices, we control their force and computation process at each iteration step.

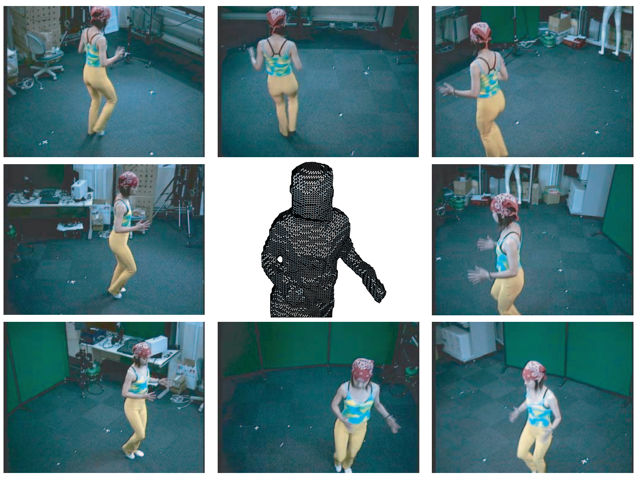

Figure 1. Input images and visual hull

This is an introduction of top-down, or model-driven factors into the deformation process to make it more stable. With our heterogeneous deformable mesh model, we will be able to reconstruct the object which consists of different kind of materials, e.g., rigidly acting body parts and deforming soft clothes or its skins, by a single and unified computational scheme.

To reconstruct the shape and motion at frame $t+1$, we assume that we have the shape of the object at frame t and multi-viewpoint images and silhouettes, i.e. visual hull, of the object at frame t and $t+1$. Figure 1 illustrates input images captured by cameras circumnavigating the object (dancing woman) and visual hull reconstructed by our volumetric intersection method proposed in [13] followed by the discrete marching cubes method [11].

Section 2 describes our basic deformable model which enables to integrate several reconstruction cues such as photo-consistency, silhouette, and so on. Section 3 presents how we categorize a vertex and how we implement model-driven deformation into basic deformation process. Section 4 shows experimental results of shape and motion recovery from a multi-viewpoint image sequence.

2. Basic Deformable 3D Mesh Model

As described above, we use a deformable mesh model which deforms its shape from frame t to frame $t+1$. Our deformable mesh model deforms so as to satisfy several constraints corresponding to reconstruction cues [7,13,14]. We represent each constraint as a force working at each vertex and compute how each vertex moves under these forces. Our deformation algorithm consists of the following steps:

Step 1 Set the given object shape at frame t as the initial shape of the mesh model.
Note that the initial mesh at $t=0$ should be given by another method. We used the visual hull of the object as the initial shape at $t=0$ (see Section 4).

Step 2 Deform the model iteratively:

- **Step 2.1** Compute the force working at each vertex respectively.
- **Step 2.2** Move each vertex according to the force.
- **Step 2.3** Terminate if the vertex motions are small enough. Otherwise go back to 2.1 .

Step 3 Take the final shape of the mesh model as the object shape at frame $t+1$.

To realize the shape deformation like SNAKES [10], we can use either energy function based or force based methods. As described above, we employed a force based method. This is firstly, from a computational point of view, because we have too many vertices (for example, the mesh model shown in Figure 1 has about 12,000 vertices) to solve energy function and secondly, from an analytical point of view, because one of the constraints used to control the deformation cannot be represented as any analytical energy function (see below).

We employed the following five constraints to control the frame-to-frame deformation:

1. **Photometric constraint**: a patch in the mesh model should be placed so that its texture, which is computed by projecting the patch onto a captured image at both frame t and $t+1$, should be consistent irrespectively of onto which image it is projected.
2. **Silhouette constraint**: when the mesh model is projected onto an image plane, its 2D silhouette should be coincide with the observed object silhouette at frame $t+1$ on that image plane.
3. **Smoothness constraint**: the 3D mesh should be locally smooth and should not intersect with itself.
4. **Motion flow constraint**: a mesh vertex should drift in the direction of the motion flow of its vicinity.
5. **Inertia constraint**: the motion of a vertex should be temporally smooth and continuous.

In what follows, we describe the forces at each vertex generated to satisfy the constraints.

2.1. Forces at each Vertex

We introduce the following five forces at v to move its position so that the above mentioned five constraints should be satisfied. We denote a vertex at $t+1$, its 3D position, and the set of cameras which can observe that vertex by v, $\boldsymbol{q}_v$, and C_v respectively. For example, $C_v = \{\mathsf{CAM}_2, \mathsf{CAM}_3\}$ in Figure 2.

External Force: $\boldsymbol{F}_e(v)$
First, we define external force $\boldsymbol{F}_e(v)$ to deform the mesh to satisfy the photometric constraint.

$$\boldsymbol{F}_e(v) \equiv \nabla E_e(\boldsymbol{q}_v), \tag{1}$$

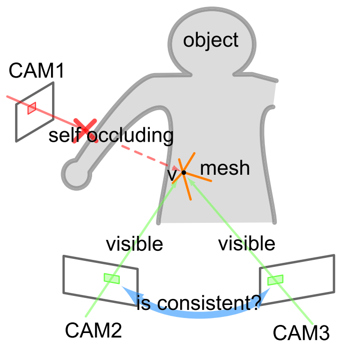

Figure 2. Photometric consistency and visibility

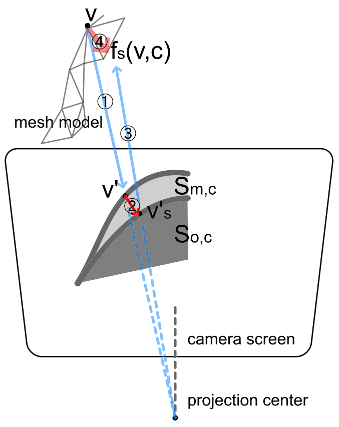

Figure 3. Silhouette preserving force

where $E_e(\boldsymbol{q}_v)$ denotes a "photo-consistency" function [12]. We use the correlation of textures to be mapped around v (Figure 2) :

$$E_e(\boldsymbol{q}_v) \equiv \frac{\sum_{c\in C_{v_0}} \left\| p_{v_0,c} - \overline{p_{v_0,v}} \right\|^2 + \sum_{c\in C_v} \left\| p_{v,c} - \overline{p_{v_0,v}} \right\|^2}{N(C_{v_0}) + N(C_v)}, \tag{2}$$

where c denotes a camera in C_v, $N(C_v)$ the number of cameras in C_v, v_0 the initial state of the vertex (the vertex at frame t), $p_{v,c}$ the texture corresponding to v on the image at $t+1$ captured by c, $p_{v_0,c}$ the texture corresponding to v_0 on the image at t captured by c, and $\overline{p_{v_0,v}}$ denotes the average of $p_{v_0,c}$ and $p_{v,c}$. $\boldsymbol{F}_e(v)$ moves v so that its corresponding image textures observed by the cameras in C_v and C_{v_0} become mutually consistent.

Note that for a vertex where $C_v = \emptyset$ and $C_{v_0} = \emptyset$, or in other words, for an invisible vertex, let the external force $\boldsymbol{F_e}(v) = 0$. The positions of invisible vertices are determined by the other forces.

Internal Force: $\boldsymbol{F}_i(v)$

Since $\boldsymbol{F}_e(v)$ may destroy smoothness of the mesh or incur self-intersection, we introduce the internal force $\boldsymbol{F}_i(v)$ at v as a combination of spring and dumper:

$$\boldsymbol{F}_i(v) \equiv \sum_j^n k_j \left(\|\boldsymbol{q}_{v_j} - \boldsymbol{q}_v\| - l_{v,v_j} \right) - d_v \dot{\boldsymbol{q}}_v \tag{3}$$

where v_j denotes neighboring vertices of v, n the number of such vertices, k_j the stiffness of the spring between v and v_j, l_{v,v_j} the natural length the spring, and d_v the damping constant proportion to the velocity of v.

Note that this internal force works so as to make the position of an invisible vertex be interpolated by its neighbors.

Silhouette Preserving Force: $\boldsymbol{F}_s(v)$

To satisfy the silhouette constraint described before, we introduce the silhouette preserving force $\boldsymbol{F}_s(v)$. This is the most distinguishing characteristics of our deformable model and involves nonlinear selective operation based on the global shape of the mesh, which cannot be analytically represented by any energy function.

Figure 3 explains how this force at v is computed, where $S_{o,c}$ denotes the object silhouette at $t+1$ observed by camera c, $S_{m,c}$ the 2D projection of the 3D mesh on the image plane of c, and v' the 2D projection of v on the image plane of c.

1. For each c in C_v, compute the partial silhouette preserving force $\boldsymbol{f}_s(v,c)$ by the following method.
2. If
   (a) v' is located out of $S_{o,c}$ or
   (b) v' is located in $S_{o,c}$ and on the contour of $S_{m,c}$,

   then compute the 2D shortest vector from v' to $S_{o,c}$ (Figure 3 ②) and set its corresponding 3D vector = $\boldsymbol{f}_s(v,c)$ (Figure 3 ④).
3. Otherwise, $\boldsymbol{f}_s(v,c) = 0$.

The overall silhouette preserving force at v is computed by summing up $\boldsymbol{f}_s(v,c)$:

$$\boldsymbol{F}_s(v) \equiv \sum_{c\in C_v} \boldsymbol{f}_s(v,c). \tag{4}$$

Note that $\boldsymbol{F}_s(v)$ works only at those vertices that are located around the object contour generator [6], which is defined based on the global 3D shape of the object as well as locations of image planes of the cameras.

Drift Force: $\boldsymbol{F}_d(v)$

As described in Section 1, we assume that we have silhouette images and a visual hull at each frame. With these visual hulls, i.e., sets of voxels, we can compute rough correspondences between them by the point-set-deformation algorithm [4]. This algorithm gives us the voxel-wise correspondence flow from the voxel set at t to the voxel set at $t+1$. We can represent this flow by a set of correspondence lines:

$$L = \{l_i | \ i = 1,\ldots,N(V)\}, \tag{5}$$

where V denotes the voxel set of the mesh, $N(V)$ the number of voxels in V, and l_i the correspondence line starting

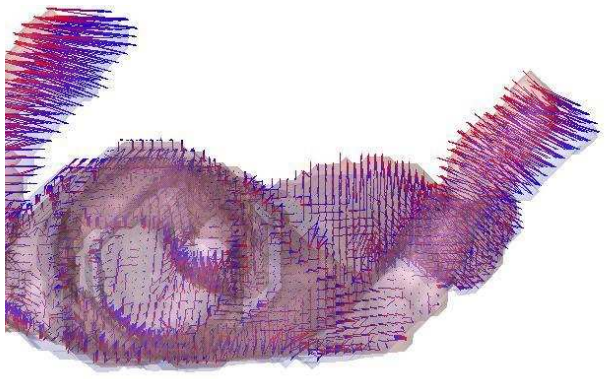

Figure 4. Roughly estimated motion flow lines

from i-th voxel in V. Whereas visual hulls do not represent accurate object shapes, we can use this correspondence as roughly estimated motion flow. Figure 4 shows roughly estimated motion flow lines from frame t to $t+1$. The start and end points of each line are colored blue and red. The slate blue and slate red regions denote the visual hull at t and $t+1$, which are the same as the center column of Figure 7.

Once the motion flow is obtained, we define the potential field $E_d(v)$ with this flow. First, let l_v be the closest correspondence line in L from a vertex v, $\boldsymbol{p}_{l_v,v}$ the closest point on l_v from v, and $\boldsymbol{s}_{l_v}$ the stating point of the correspondence line l_v. Then, we define the potential field as the function of the distance from v to l_v and the distance from $\boldsymbol{s}_{l_v}$ to $\boldsymbol{p}_{l_v,v}$:

$$E_d(\boldsymbol{q}_v) \equiv \|\boldsymbol{s}_{l_v} - \boldsymbol{p}_{l_v,v}\|^2 - \|\boldsymbol{q}_v - \boldsymbol{p}_{l_v,v}\|^2. \tag{6}$$

Finally, we define the drift force $\boldsymbol{F}_d(v)$ at vertex v was the gradient vector of $E_d(\boldsymbol{q}_v)$:

$$\boldsymbol{F}_d(v) \equiv \nabla E_d(\boldsymbol{q}_v). \tag{7}$$

Inertia Force: $\boldsymbol{F}_n(v)$
If we can assume that the interval between successive frames is short enough, we can expect the continuity and the smoothness of the object motion. This assumption tells us that we can predict a vertex location at $t+1$ from its motion history.

We can represent such predictions as a set of prediction lines connecting $\boldsymbol{q}_v$ and $\hat{\boldsymbol{q}}_v$, where $\hat{\boldsymbol{q}}_v$ denotes the predicted location of v. Then we can define the inertia force $\boldsymbol{F}_n(v)$ just in the same way as the drift force $\boldsymbol{F}_d(v)$:

$$\boldsymbol{F}_n(v) \equiv \nabla E_n(\boldsymbol{q}_v), \tag{8}$$

where $E_n(\boldsymbol{q}_v)$ denotes potential field defined based on the set of prediction lines.
Overall Vertex Force: $\boldsymbol{F}(v)$
Finally we define vertex force $\boldsymbol{F}(v)$ with coefficients $\alpha, \beta, \gamma, \delta, \varepsilon$ as follows:

$$\begin{aligned}\boldsymbol{F}(v) \equiv\ & \alpha\boldsymbol{F}_i(v) + \beta\boldsymbol{F}_e(v) + \gamma\boldsymbol{F}_s(v) \\ & + \delta\boldsymbol{F}_d(v) + \varepsilon\boldsymbol{F}_n(v).\end{aligned} \tag{9}$$

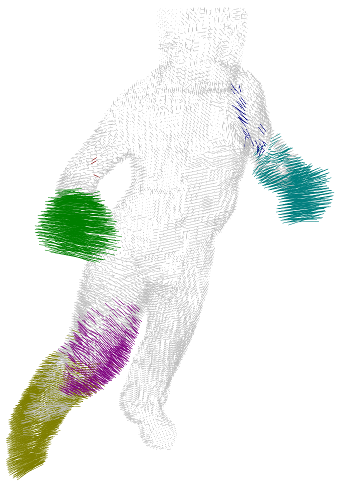

Figure 5. Clustered motion flow lines

2.2. Deformation Computation

Given a force working at each vertex, we can compute how the mesh deforms its shape by solving Newtonian equation of motion:

$$\begin{aligned}\boldsymbol{F}(v) &= m\boldsymbol{a} \\ &= m\ddot{\boldsymbol{q}}_v\end{aligned} \tag{10}$$

with forward Euler integration method or other iterative method [1, 17]. We use a given m for all the vertices and backward Euler integration method for the experiments described below.

3. Heterogeneous Deformation

As described in Section 1, we introduce a heterogeneous deformation process into the basic deformable model in Section 2 based on vertex identifiability and motion model.

3.1. Vertex Categorization

Vertex Identifiability As is well known, we can not expect that all the points on the object surface have prominent texture and can be recovered by stereo method. Hence not all the vertices of the mesh model are *identifiable*, and the external force $\boldsymbol{F}_e(v)$, which put a vertex on the real object surface based on texture correlation, will not work at such vertices. That is, we assume that we can categorize the vertices into two types:

Ca-1 a vertex with prominent texture which should *lead* its neighbors, or

Ca-2 others which should be *led* by its neighbors.

We regard a vertex as identifiable if it has consistent and prominent textures in visible cameras. Here, we introduce an identifiability-scoring function $I(v)$ as follows:

$$I(v) \equiv E_e(\boldsymbol{q_v}) \times \operatorname{argmin}\left\{ \min_{c \in C_{v_0}} \nabla p_{v_0,c},\ \min_{c \in C_v} \nabla p_{v,c} \right\}, \quad (11)$$

where $E_e(\boldsymbol{q_v})$ denotes the correlation of textures of v (see Equation (2)), $\nabla p_{v_0,c}$ and $\nabla p_{v,c}$ the derivatives of the texture of v_0 and v on camera c respectively. With this function $I(v)$, we compute the identifiability for each vertex, and label as **Ca-1** (identifiable) if the score exceeds a certain threshold, and as **Ca-2** if not.

Motion Model On the other hand, as described in the definition of the drift force $\boldsymbol{F}_d(v)$ and the inertia force $\boldsymbol{F}_n(v)$, we have roughly estimated motion flow from frame t to $t+1$. By clustering this flow, we can estimate which vertices are moving together, i.e., rigidly (Figure 5). With this clustering result, we assume that we can categorize the vertices into another two types:

Cb-1 an element of a rigid part of the object and should move together with others in the same part, or

Cb-2 a vertex corresponding to a part of object surface under free deformation.

We cluster the motion flow represented as the set of lines (Figure 4) as follows:

Step 1 Eliminate lines which are too short to be regarded as a part of structured motion. We used the average length between neighboring vertices as the threshold of this elimination.

Step 2 Group the resulting lines into subsets based on geodesic distances between the starting points of each line. With geodesic distance on visual hull surface, we can prevent the merge of two sets which are close in Euclidean distance but not topologically.

Step 3 Group each subset into clusters based on the direction of each line.

Step 4 Label each line as

Rigid motion flow if it is an element of a cluster,

Deformation flow otherwise.

In Figure 5, each colored lines denote a rigid motion flow, and gray lines denote deformation flow. We label a vertex on a rigid motion flow as **Cb-1** and others as **Cb-2**.

3.2. Heterogeneous Deformation Algorithm

With these two categorizations, we add following steps in the basic deformation process.
For each vertex,

- if it categorized as **Ca-1**, let the force of the vertex diffuse to those of neighbors so that it lead its neighbors,
- and / or if categorized as **Cb-1**, make the springs of the vertex stiff [5, 16] to move together with others.

We define the diffusion process as follows.

1. Let v be a vertex of type **Ca-1**, v_j a neighboring vertex of v.
2. For each v_j, modify the force $F(v_j)$:

$$\begin{aligned} \boldsymbol{F}(v_j) &= \omega \boldsymbol{F}(v) + (1-\omega)\boldsymbol{F}_{\text{prev}}(v_j), \\ \omega &= e^{-D_g(v,v_j)}, \end{aligned} \quad (12)$$

   where $\boldsymbol{F}_{\text{prev}}(v_j)$ denotes the original force at v_j, and $D_g(v,v_j)$ the geodesic distance between v and v_j.

Note that for a vertex of type **Ca-2** $\wedge$ **Cb-2**, a vertex without prominent texture or not a part of a rigid part, its position is interpolated by the internal force $\boldsymbol{F}_i(v)$, and a vertex of type **Ca-1** $\wedge$ **Cb-1**, a vertex with prominent texture and a part of a rigid part, deforms so as to lead the rigid part which the vertex belongs to.

Applying these steps, our heterogeneous deformation process is modified as follows:

Step 1 Set the given object shape at frame t as the initial shape of the mesh model.

Step 2 Compute roughly estimated motion flow for the drift force $\boldsymbol{F}_d(v)$ and the inertia force $\boldsymbol{F}_n(v)$.

Step 3 Categorize the vertices based on the motion flow:

Step 3.1 By clustering the estimated motion flow, label the vertex whether **Cb-1**: it is an element of a rigid part, or **Cb-2**: it is not.

Step 3.2 Make the springs of vertices labeled as **Cb-1** stiff.

Step 4 Deform the model iteratively:

Step 4.1 Compute forces working at each vertex respectively.

Step 4.2 For a vertex whose identifiability $I(v)$ exceeds a certain threshold, that is, for a vertex labeled as **Ca-1**, let the force of it diffuse to those of neighbors.

Step 4.3 Move each vertex according to the force.

Step 4.4 Terminate if the vertex motions are small enough. Otherwise go back to 2.1 .

Step 5 Take the final shape of the mesh model as the object shape at frame $t+1$.

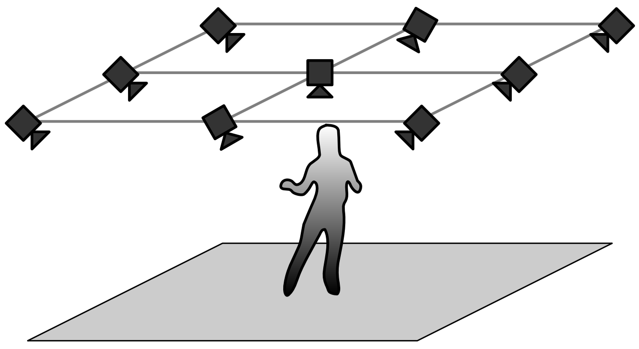
Figure 6. Camera arrangement

4. Experimental Results

Figure 7, 9, and 8 illustrate the inter-frame deformation through eight successive frames. The columns of Figure 7 show, from left to right, the captured images, the visual hulls generated by the frame-wise discrete marching cubes method, and the mesh models deformed frame by frame, respectively. In these two figures, colored areas of the mesh denote rigid parts of the object estimated by the clustering at **Step 3**. Note that the visual hull in frame t was used as the initial shape for our deformation. In this experiment, we used 9 cameras circumnavigating the object (Figure 6). Captured multi-viewpoint videos are not completely synchronized and include motion blur. The mesh models consist of about 12,000 vertices and 24,000 triangles, and the processing time per frame is about 20 minutes by PC (Xeon 3.0GHz). We used fixed coefficients $\alpha = 0.2, \beta = 0.2, \gamma = 0.2, \delta = 0.3, \varepsilon = 0.1$ given a priori.

From these results, we can observe:

- Our deformable mesh model can follow the partially-rigid object motion smoothly. In Figure 7 and 9, we can observe that the arms of the object are labeled as rigid regions.
- During its dynamic deformation, our mesh model preserves both global and local topological structure and hence we can find corresponding vertices between any pair of frames for all vertices. Figure 9 illustrates this topology preserving characteristic. That is, the top-left mesh denotes a part of the initial mesh obtained by applying the marching cubes to the visual hull at t. The lower light arrow stands for our deformation process, where any parts of the mesh can be traced over time. Aligned along the upper dark arrow, on the other hand, are parts of the meshes obtained by applying the marching cubes to each visual hull independently, where no vertex correspondence can be established because the topological structures of the meshes are different.
- In Figure 8, we can observe that the vertices corresponding to the right lower arm and hand of the object were labeled into different groups (green and blue) at frame $t+7$ because the arm and the hand moved to different directions. This tells that vertices labeled as a single rigid part may be separated into two or more parts through successive frame recovery, and such vertices labeled into different parts may be grouped together in future frame. That is, we can find new joint of the object through the recovery, and we should introduce a deformation model which can learn where is a rigid part or a joint, and can utilize it to accomplish more robust reconstruction.

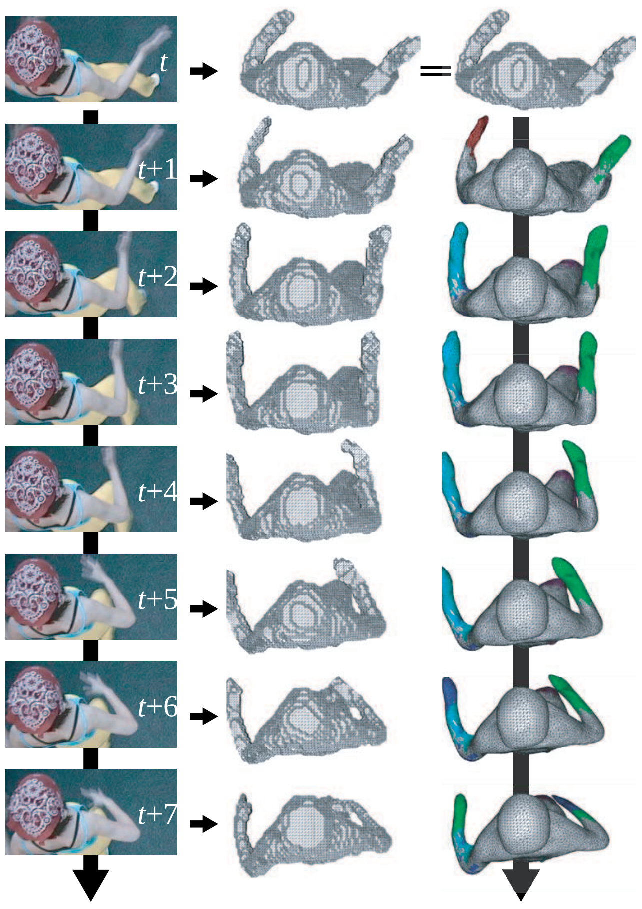

Figure 7. Successive deformation results

We applied the inter-frame deformation for a long series of frames and observed the following problem:

- In the dancing lady video sequence, the overall topological structure changes depending on her dancing poses; sometimes hands are attached to her body. The current mesh model cannot cope with such topological changes. For example, the "hand" of the mesh stick its "waist."

To cope with this kind of topological structure changes, we should employ more model-driven approach which can 1) prevent global self-intersection, and 2) learn the motion model of each part of the object through the recovery.

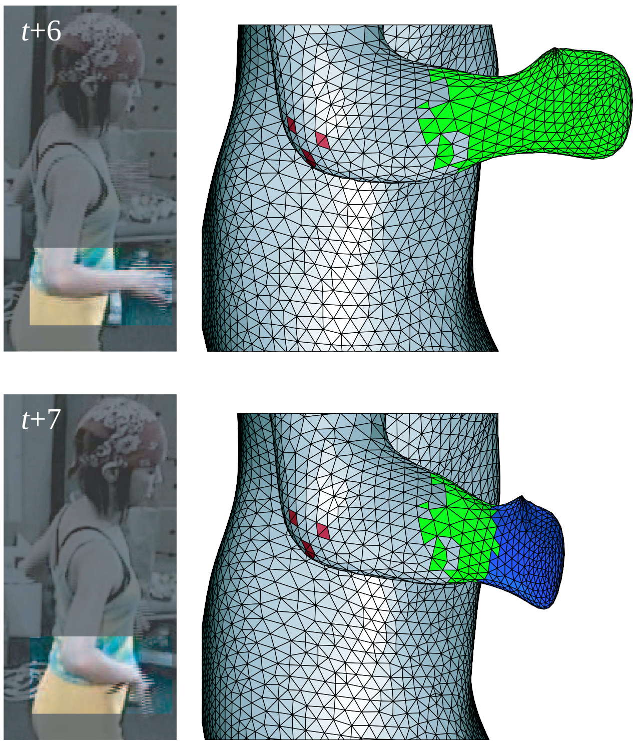

Figure 8. Successive deformation results (detailed, side view)

5. Conclusion

In this paper, we proposed a computational framework using a heterogeneous deformable mesh model to reconstruct dynamic 3D shape, i.e., full 3D shape and motion simultaneously. We believe that a unified recovery approach is better than the two-stage approach which recovers shapes first and then recovers motions from the shapes.

With our heterogeneous deformation, we can reconstruct the object which consists of different kinds of materials, e.g., rigidly acting body parts and deforming soft clothes or its skins, by single model and unified computational scheme.

Our computational framework will enable us to develop more effective 3D motion analysis, 3D data compression, and so on.

Acknowledgements

This work is supported by the Grant-in-Aid for Scientific Research 133224051. We are grateful to Real World Computing Partnership, Japan for allowing us to use their multi-viewpoint video data.

References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. *Proc. of SIGGRAPH 1998*, 32:43–54, 1998.

[2] A. Bottino and A. Laurentini. A silhouette-based technique for the reconstruction of human movement. *Computer Graphics and Image Processing*, 83:79–95, 2001.

[3] H. Briceño, P. Sander, L. McMillan, S. Gortler, and H. Hoppe. Geometry videos: A new representation for 3d animations. In *Proc. of ACM Symposium on Computer Animation*, pages 136–146, 2003.

[4] D. Burr. A dynamic model for image registration. *Computer Graphics and Image Processing*, 15:102–112, 1981.

[5] J. Christensen, J. Marks, and J. T. Ngo. Automatic motion synthesis for 3d mass-spring models. *The Visual Computer*, 13(1):20–28, 1997.

[6] G. Cross and A. Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture, 2000.

[7] P. Fua and Y. G. Leclerc. Using 3-dimensional meshes to combine image-based and geometry-based constraints. In *Proc. of European Conference on Computer Vision*, volume 2, pages 281–291, 1994.

[8] T. Heap and D. Hogg. Towards 3d hand tracking using a deformable model. In *Proc. of 2nd International Conference on Automatic Face and Gesture Recognition*, pages 140–145, 1996.

[9] J. Isidoro and S. Sclaroff. Stochastic mesh-based multiview reconstruction. In *Proceedings of 3D Data Processing, Visualization, and Transmission*, pages 568–577, Padova, Italy, July 2002.

[10] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[11] Y. Kenmochi, K. Kotani, and A. Imiya. Marching cubes method with connectivity. In *Proc. of 1999 International Conference on Image Processing*, pages 361–365, Kobe, Japan, Oct. 1999.

[12] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *Proc. of International Conference on Computer Vision*, pages 307–314, Kerkyra, Greece, Sept. 1999.

[13] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation and high fidelity visualization for 3d video. *Computer Vision and Image Understanding*, page in print, 2004.

[14] S. Nobuhara and T. Matsuyama. Dynamic 3d shape from multi-viewpoint images using deformable mesh models. In *Proc. of 3rd International Symposium on Image and Signal Processing and Analysis*, pages 192–197, Rome, Italy, Sept. 2003.

[15] R. Plänkers and P. Fua. Articulated soft objects for multi-view shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1182–1187, 2003.

[16] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In W. A. Davis and P. Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154. Canadian Human-Computer Communications Society, 1995.

[17] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proc. of SIGGRAPH 87*, pages 205–214, July 1987.

[18] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Proceedings of the 7th International Conference on Computer Vision*, volume 2, pages 722 – 729, Sept. 1999.

[19] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6d. In *Computer Vision and Pattern Recognition (CVPR)*, June 2000.

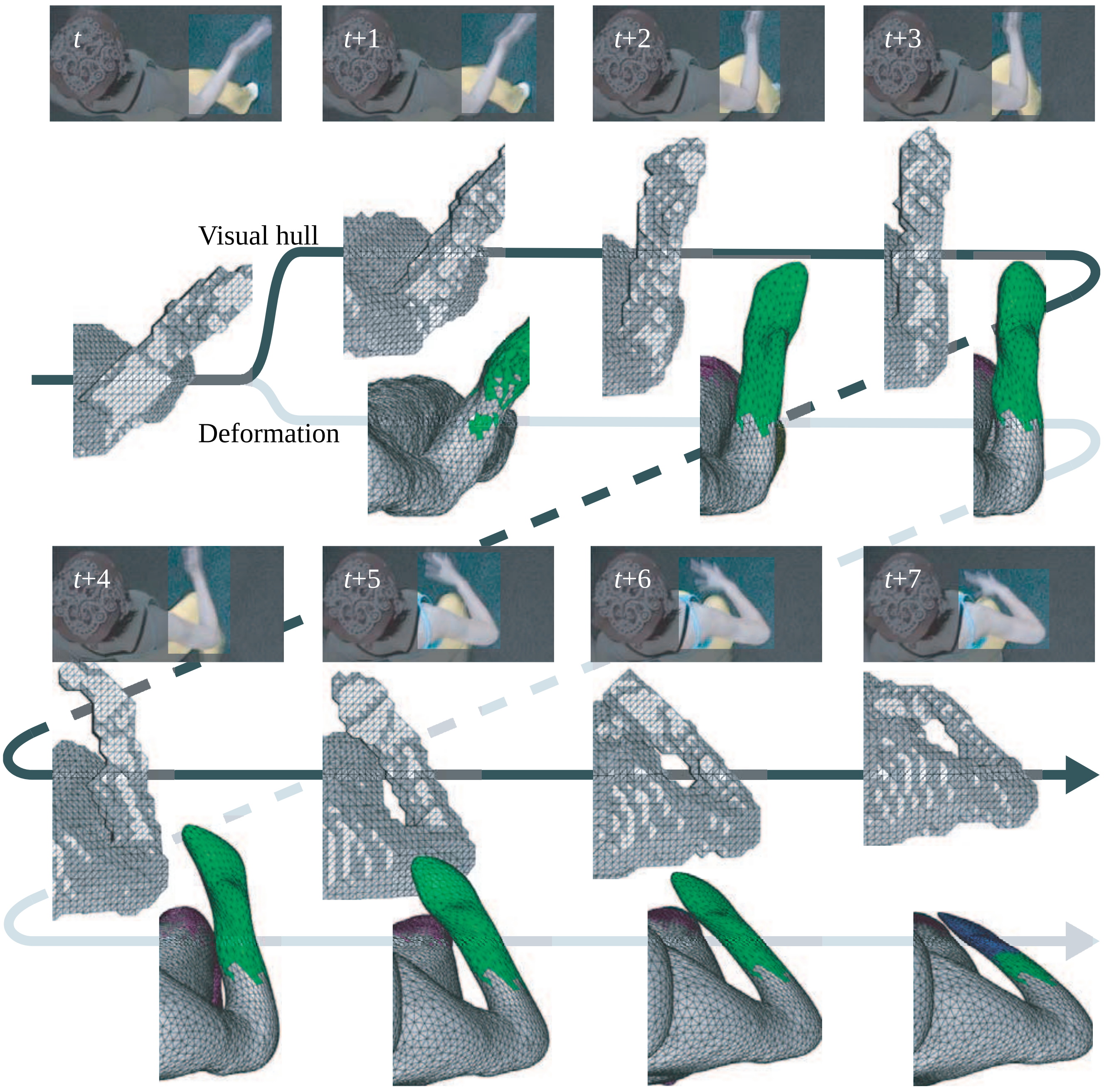

Figure 9. Successive deformation results (detailed)