Eurographics Workshop on 3D Object Retrieval (2014)
B. Bustos, H. Tabia, J.-P. Vandeborre, and R. Veltkamp (Editors)

3D Volume Matching for Mesh Animation of Moving Actors

L. Blache, C. Loscos, O. Nocent and L. Lucas

CReSTIC-SIC, University of Reims Champagne-Ardenne, France

Abstract

4D multiview reconstruction of moving actors has many applications in the entertainment industry and although studios providing such services become more accessible, efforts have to be done in order to improve the underlying technology to produce high-quality 4D contents. In this paper, we enable surface matching for an animated mesh sequence in order to introduce coherence in the data. The context is provided by an indoor multi-camera system which performs synchronized video captures from multiple viewpoints in a chroma key studio. Our input is given by a volumetric silhouette-based reconstruction algorithm that generates a visual hull at each frame of the video sequence. These 3D volumetric models differ from one frame to another, in terms of structure and topology, which makes them very difficult to use in post-production and 3D animation software solutions. Our goal is to transform this input sequence of independent 3D volumes into a single dynamic volumetric structure, directly usable in post-production. These volumes are then transformed into an animated mesh. Our approach is based on a motion estimation procedure. An unsigned distance function on the volumes is used as the main shape descriptor and a 3D surface matching algorithm minimizes the interference between unrelated surface regions. Experimental results, tested on our multiview datasets, show that our method outperforms approaches based on optical flow when considering robustness over several frames.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

This paper fits in the RECOVER3D project [LSI*13] which context is an integrated virtual video system for the broadcast and motion picture markets using multiview reconstruction. The innovation brought by this project aims at freeing the creation of video images from classic material constraints linked to multi-camera shooting, thanks to a new *virtual cloning* system of actors and scenes based on smart 3D video capture, natively delivering 3D models. Data are generated from captures in a multiview studio, as illustrated in figure 1. This set of multi-viewpoint cameras (*cyber dome*) generates, for each frame, the digital transcription of the scene in three dimensions using a volumetric *visual hull* algorithm [Lau94], producing a sequence of 3D volumes over time. These volumes are usually transformed into a sequence of 3D textured meshes, successively loaded for the rendering of each frame. Our goal is to introduce a dynamic representation of the character, freeing ourselves from this static, temporally inconsistent description of the scene. We want to create a single, temporally consistent, animated model following the character's motion. Our long-term goal is an approach as generic as possible, allowing us to work on various types of scenes: one or several actors, dressed freely and manipulating accessories, containing close-up shots. These constraints require the consideration of a method which is not limited to rigid motion recovery.

To reach this goal, we developed a new method which uses a feature-based volume tracking to identify the actor's motions and then apply a surface matching algorithm. The input of our method is a sequence of 3D volumes generated independently one to another. We extract the scene motion by computing a 3D motion flow from these volumes. The particularity of our method is to combine two different types of computations with a back and forth approach: a Euclidean distance transform [ST94] and a choice of complementary criteria (proximity, orientation and color) that permit to discriminate voxel matching. After the motion flow is filtered, it is used to match a chosen template mesh (one of the sequence frames) to the sub-sequent meshes by pairs of frames, regularized using a mass-spring system in an it-

DOI: 10.2312/3dor.20141052

erative approach, in order to create a unique mesh that is animated over time. This method works on generic datasets, whatever the shape of the reconstructed object or character.

In section 2, a brief overview of recent advances in model tracking is given. In section 3, our approach is explained, giving details on the object's representation (3.1), the motion extraction (3.2) and the mesh animation process (3.4). Results are then presented in section 4 showing the quality of the motion retrieval and its robustness over several frames.

2. Previous work

This section gives a brief overview of the existing techniques for acquiring a 4D model of moving actors. Multiview reconstruction methods are usually separated into two main approaches: model-based and model-free.

Model-based reconstruction approaches use a predefined *template* model representing an actor, which is most of the time an articulated mesh of a generic human body, or obtained by another reconstruction method like a 3D scan of the actor, as in [dAST*08]. The multiview reconstruction over time is then proceeded by animating this template, following the movements of the actor during the sequence. The model is moved according to a set of directives (optical flow, silhouette matching ...) extracted from the videos. In [VBMP08] and [GSDA*09], a skeleton is fitted to the model to enable the animation. Local deformations are then performed on the mesh in order to match non-rigid motions (like clothes or hair). The advantage of these methods is that they produce temporally consistent animations. The main problem of this kind of approaches is the very strong assumption about the scene's content. Most of the generic models limit the reconstruction to a single human shape, even if some methods, like [LSG*11], allow to represent several actors. The template model is most of the time limited in its representation to a set of possible clothes (dresses, for example, bring failure), or require to be prepared during a complex manual step before the multiview acquisition. These approaches are too restrictive for our goal because we do not want to make assumptions about the reconstructed actors. Skeleton-based approaches, especially, could lead to strong limitations if the reconstruction is proceeded on actors wearing loose costumes (dresses or coats for example) or accessories (bags, hats ...).

Model-free methods do not use a template mesh and are supposed to be more generic. The most commonly used are based on visual hull (silhouettes) or depth maps (stereo) reconstruction. The main problem is that these approaches compute a static reconstruction of the scene at each frame of the multi-viewpoint videos. Thus, they obtain a sequence of static 3D objects which represent the successive actors' poses, but without any consistency in term of structure or topology. To be used for animation, these sequences need to be processed and transformed into a single, temporally consistent, animated object. Starck and Hilton [SH07b] proposed a model-free method based on visual hull and stereo reconstruction. A spherical parameterization is operated on the object. This restricts the process to work only on single closed surfaces. Cagniart *et al.* [CBI10] create a dynamic patch-based mesh from the first frame and then deform it according to the poses described in each frame. Li *et al.* [LLV*12] use mesh correspondences to enhance high-resolution scan sequences with hole-filling and temporal consistency. Another common way to establish a temporal consistency is to match the successive meshes. These *mesh-tracking* methods compute a matching between the vertices of two meshes according to curvature or color criteria [SH07a] [VZBH08] [TM10]. This tracking can be used to compute a *motion flow* which describes the movements of the character between two frames [PLBF11]. This motion flow can also be computed by a *scene flow* method. A scene flow, as introduced by Vedula *et al.* [VBR*99], is the 3D equivalent of optical flow, computed by merging the optical flows of a multi-viewpoint context. It is often used for motion tracking applications. Anuar and Guskov [AG04] use a method that adapts optical flow to 3D discrete space, to compute the motion directly in the 3D reconstruction sequence. The motion flow can then be used to animate a mesh. In the case of visual hull reconstruction, the meshes may contain too many inconsistencies (holes and changes in topology between frames) to proceed a robust matching. Therefore a volumetric approach is more appropriate, like the method proposed by Nobuhara and Matsuyama [NM04] which computes a motion flow by matching a volumetric silhouette-based reconstruction and then uses it to animate a template mesh. The motion estimation is performed by matching the voxels of reconstructed discrete volumes. The template is obtained by a *marching cube* triangulation of the first frame volume. However, the motion flows computed in this method are simply obtained by matching each voxel to the closest one in another frame, thus producing motion vectors which lack accuracy.

3. Our Approach

Our input is a sequence of discrete volumes obtained by a preliminary reconstruction stage, from a set of multiview video sequences. It represents the character's pose at each video frame (see figure 1). Our method starts by computing a 3D motion flow between two consecutive frames. At this stage we work on the reconstructed volumes. In the next step we use these flows to animate a dynamic mesh model. The reconstructed mesh at the first frame is used as the initial template model. By deforming it at each frame according to the estimated flows, we deduce a character's animation.

3.1. Volumes description

The reconstructed volumes we use are simple binary digital volumes, a 3D grid of voxels defined by binary values

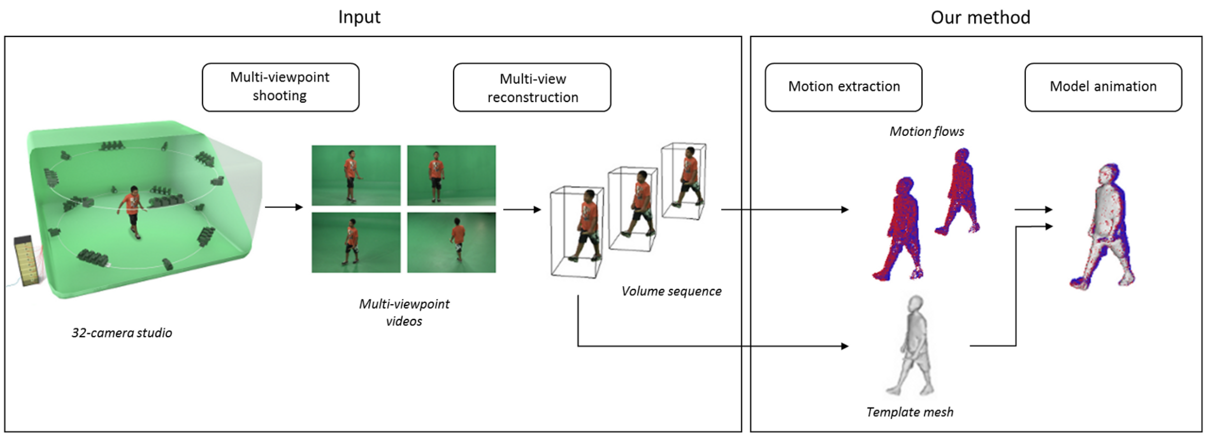

Figure 1: *An overview of our production process. Our method focuses on the motion flow computation and the mesh animation.*

(0 for void voxels and 1 for voxels covering or intersecting the object). We then compute another representation of these volumes by using a *Euclidean distance transform* (EDT), as described by Saito and Toriwaki [ST94]. We obtain an unsigned distance volume, represented by a 3D grey-level voxel grid, as shown in figure 2. Each voxel is associated to a positive value which corresponds to the Euclidean distance to the closest boundary of the object. This volume description could be considered as a grey-level 3D picture. Thus, we can compute a derivative estimation of this picture. It will be used to compute the normal vectors (see section 3.2.2) and gradient values. To compute the spatial derivative, we use a set of Sobel-like filters which estimate around each voxel, in a $3 \times 3 \times 3$ window, the EDT variations for each spatial axis. A temporal derivative is also computed on the same neighborhood by the differences of the values between two consecutive frames.

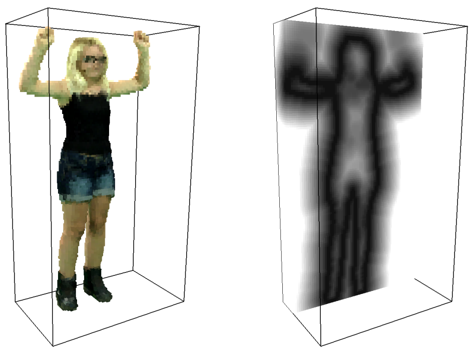

Figure 2: *Left: an example of colored reconstructed volume. Right: a sliced representation of the corresponding EDT.*

The last available information is color which can be extracted from the multiview video frames. We use it to texture the original volume. Each surface voxel is then associated with an RGB color (see figure 2, left).

3.2. Voxel matching

Given two consecutive volumes V^n and V^{n+1} which correspond to frames n and $n+1$, our goal is to compute a matching $V^n \to V^{n+1}$ representing the scene flow. We define as *surface voxels* the voxels which belong to the object and have at least one void voxel in their direct neighborhood. These surface voxels are characterized by an RGB color and a surface's normal vector. We want to match each surface voxel $v_i^n \in V^n$ to another surface voxel $v_j^{n+1} \in V^{n+1}$ minimizing the following distance function:

$$D(v_i^n, v_j^{n+1}) = \omega_p \delta_{i,j} + \omega_n \varphi_{i,j} + \omega_c \sigma_{i,j} \quad (1)$$

where $\delta_{i,j}$, $\varphi_{i,j}$ and $\sigma_{i,j}$ correspond respectively to a proximity criterion (see section 3.2.1), an orientation criterion (see section 3.2.2) and a colorimetric criterion (see section 3.2.3). ω_p, ω_n and ω_c are weighting terms, fixed by the user. In our experimentations we used $\omega_p = 1$, $\omega_n = 5$ and $\omega_c = 10$. These criteria allow to match the voxels which correspond to the same part of the surface, identified by an orientation and a texture. In case of large motions, the color is the most invariant feature. The proximity should only be a discriminating characteristic when several voxels satisfy the other terms of the distance function.

We define a *search radius* which corresponds to the maximum amplitude of the motion. Thus, this radius strongly depends on the dataset and must be defined by the user. For each surface voxel v_i^n we look through the surface voxels of V^{n+1} contained in this neighborhood and we select the voxel v_j^{n+1} which corresponds to the smallest result of the function (1). Figure 3 shows an example of voxel matching. The positions of voxels v_i^n and v_j^{n+1} define a 3D vector. This vector is added to a vector field at the v_i^n position. This vector field is represented by the same structure as the voxel grid. Each square could contain one or several vectors. The same operation is repeated, looking this time, for each v_j^{n+1}, for the matching surface voxel v_i^n. The resulting vectors are added to the vector field at v_i^n position. This backward pass allows us to find a part of the motion which could have been ignored by the forward matching process (see figure 4, top). Thus, we ensure that each surface voxel in V^n and V^{n+1} is associated to at least one vector.

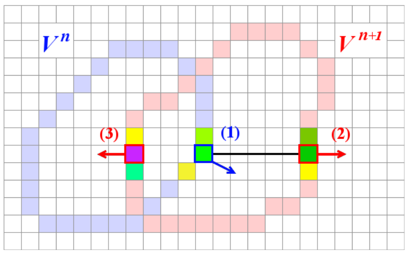

Figure 3: *Voxel matching between two consecutive volumes. The voxel (1) from the V^n volume matches better the voxel (2) from the V^{n+1} volume than the voxel (3). The neighboring voxels are represented with their colors. Normal vectors are figured by arrows.*

3.2.1. Proximity criterion

The proximity criterion corresponds to the Euclidean distance between the two voxels:

$$\delta_{i,j} = \left\| \mathbf{p}_j^{n+1} - \mathbf{p}_i^n \right\|$$

with $\mathbf{p}_i^n$ and $\mathbf{p}_j^{n+1}$ being the 3D positions of v_i^n and v_j^{n+1}. This criterion allows us, if several voxels satisfy the other criteria, to select the closest one (see figure 8(b)).

3.2.2. Orientation criterion

The orientation criterion measures the difference between the normal vectors of the two voxels:

$$\varphi_{i,j} = 1 - \mathbf{n}_i^n \cdot \mathbf{n}_j^{n+1}$$

with $\mathbf{n}_i^n$ and $\mathbf{n}_j^{n+1}$ being respectively the normal vectors at v_i^n and v_j^{n+1}. As illustrated in figure 8(c), this criterion penalizes the matching of two voxels which belong to back facing surfaces. For example, in figure 3, the voxel (1) is matched with voxel (2) which normal vector has a closer orientation.

3.2.3. Colorimetric criterion

The colorimetric criterion is similar to a *block matching* algorithm, as used for motion estimation in digital video processing. We compare the colorimetric difference between two voxels as well as between their direct neighborhoods:

$$\sigma_{i,j} = \left\| v_j^{n+1} - v_i^n \right\|_{RGB} + \left\| B_j^{n+1} - B_i^n \right\|_{RGB}$$

B_i^n and B_j^{n+1} are the blocks which correspond to the surface voxels contained in a neighborhood of fixed size b:

$$B_i^n = \sum_{k=1}^{b} v_{i+k}^n$$

if v_{i+k}^n belongs to the surface. This constraint favours the matching of two voxels which belong to close color blocks corresponding to the same object's part (see figure 8(d)).

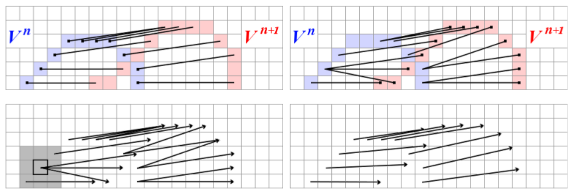

Figure 4: *Top: forward and backward matching between the two volumes. Bottom: Gaussian filter (in grey) applied to the raw vector fields (left) and final motion field (right).*

3.3. Motion regularization

The voxel matching step results in a 3D vector field which should describe the motion of the volumetric object between V^n and V^{n+1}. However, several inconsistent matches remain and the global motion is too irregular to be used. That is why a *smoothing* step is performed to get a coherent motion flow, as shown in figure 4 (bottom). We apply a Gaussian filter on the initial vector field. For each surface voxel, we compute a single vector which is an average, weighted by Gaussian coefficients, of all the vectors in a defined neighborhood. Thus, we obtain a smooth 3D motion field where each surface voxel is associated with a single motion vector. This filtering operation cleans the irrelevant vectors and regularizes the vector set to produce a coherent motion description where each surface voxel is associated to a single motion vector. The size of this filter depends on the dimension of the volumes and must be defined by the user. In our case, we perform a single filtering iteration, but for high resolution volumes, the filter can also be applied several times to enhance the smoothing effect.

3.4. Mesh animation

In the animation step, the template mesh is immersed in the motion field and we apply to each vertex the translation defined by the closest vector. Because the result is too irregular to be used (see figure 10), we once again apply a regularization algorithm, this time to obtain a regular mesh which corresponds to the pose defined by the visual hull. We consider the mesh as a mechanical mass-spring system. Each vertex is submitted to a set of forces including:

- **spring force**: Each incident edge applies a force on the vertex, to equalize the edges' length. This force tends to regularize the vertices distribution.
- **smoothing force**: A regularization operator, applies a Laplacian smoothing (*umbrella operator*) [KCVS98] which tends to smooth the surface of the mesh.
- **matching force**: The EDT distance field derivative (see section 3.1) brings each vertex closer to the object's surface.

We use a *local* Euler integration scheme to resolve this system: for each vertex, we apply a semi implicit resolution algorithm, with a *fixed neighborhood* (we do not change the position of the other vertices). This operation is applied on each vertex, that corresponds to one *global* iteration. We apply as many global iterations as necessary.

4. Results

Results were tested on two datasets acquired with a dome similar to the one illustrated in figure 1. The *girl* dataset contains simple motions, with a woman slowly moving her arms. The visual hull volume has a $73 \times 132 \times 43$ voxels resolution and is reconstructed for 30 frames. The *boy* dataset is more complex, with a young man walking with relatively loose clothes, thus with a movement showing large displacements (due to faster motion and lower acquisition frequency). The reconstructed volume has a $89 \times 129 \times 69$ resolution, and the sequence contains 10 frames. All timings were done on a 64 bit Intel Core i7 CPU 2.20 GHz.

4.1. Evaluation of the motion flow reconstruction

When testing the motion flow on these datasets, we obtain a satisfying motion field due to the regularization step, where each surface voxel is associated to a displacement vector (see figure 5). Figure 7 (left) presents the results for full sequence on the *girl* dataset, for which, the motion between two frames is computed in less than 10 seconds. We used a 3-voxel search radius and a single regularization iteration. Figure 7 (right) shows the tracking of the *boy* dataset. We used a 10-voxel search radius and the motion computation step took 65 seconds.

We compared our approach with our own implementations of two 3D-adapted optical flow algorithms as presented in [BT04] : the first one is based on the Lucas and Kanade method [LK81] like the method described in [AG04], and the second one on the variational approach by Horn and Schunck [HS81]. Our tests show that for similar settings, the Lucas-Kanade approach is faster (less than 5 seconds for *girl*, 50 seconds for *boy*) but displacement vectors are not oriented correctly (see an example of results in figure 6 (left) for a zoom-in on the girl's upper body). It was expected as this kind of image warping approach is not well suited for large displacements. One common improvement to avoid this problem would be to implement a coarse-to-fine computation. The Horn-Schunck algorithm is significantly slower (5 minutes for *girl*, 10 minutes for *boy*) and does not give convincing results with displacement distances not corresponding to the actual movement (see figure 6 (right)). The Euclidean distance volume, used as a 3D picture, does not seem to be a good enough information to compute a consistent motion information. Despite of its high algorithmic complexity, our voxel matching method provides a better representation of the motion. While it is mostly only possible to evaluate visually the motion flows, a quantitative evaluation was performed on the mesh itself (see section 4.2) which confirms our observations on the flows.

4.1.1. Discussion on the chosen parameters

Figure 8 shows the influence of the three criteria (proximity, orientation, color) for voxel matching, defined by weights ω_p, ω_n and ω_c (see Eq.(1)), fixed by the user. Figure 8(b) shows that without the proximity criterion ($\omega_p = 0$), most of the matched voxels are too distant, even if the search radius is adapted to the motion. The matching could associate two voxels which seems identical but does not correspond to the same part of the surface. The same problem appears if the orientation criterion's weight (ω_n) is set to zero. As illustrated in figure 8(c), most of the voxels are matched with another voxel which is close but corresponds to a backfacing surface. Figure 8(d) shows the lack of precision in the matching computed without colorimetric criterion ($\omega_c = 0$). The efficiency of this criterion increases when the volume is highly textured (*i.e.*, there are lots of variations in the voxels' colors). At last, figure 8(e) shows that these criteria do not have the same influence, depending on the dataset used, and most of the time, different weights are chosen by datasets. The method presented by [NM04], which uses only Euclidean distance (means $\omega_n = \omega_c = 0$) is expected to be even less efficient than results shown with $\omega_n = 0$ or $\omega_c = 0$. It is really the combination of the three criteria that improves the quality of the matching process.

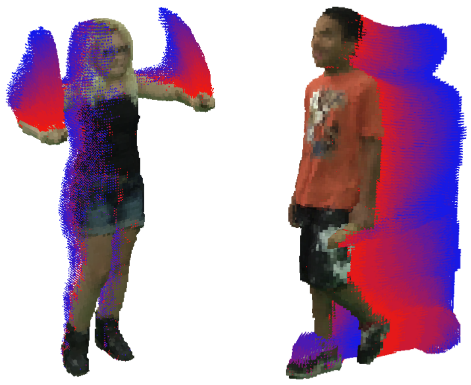
Figure 7: *Accumulated motion flows through several frames of the girl (left) and boy (right) sequences.*

4.2. Qualitative evaluation of the mesh animation

After the application of motion vectors' translations, the template mesh (see figure 10, left) is altered (see figure 10, middle). After our iterative regularization, we obtain a smooth and regular mesh which matches the pose at each frame (see figure 10, right). Several results are represented in figure 11. The *girl* mesh processing between two frames took around 50 seconds. We used 100 global iterations and 50 local iterations. The first mesh, used as a template,

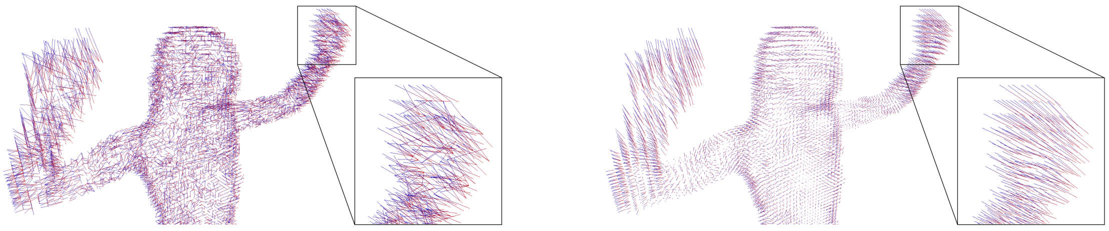

Figure 5: *Motion field regularization. Left: result of the voxel matching step. Right: vector field after regularization (vectors are oriented from blue to red).*

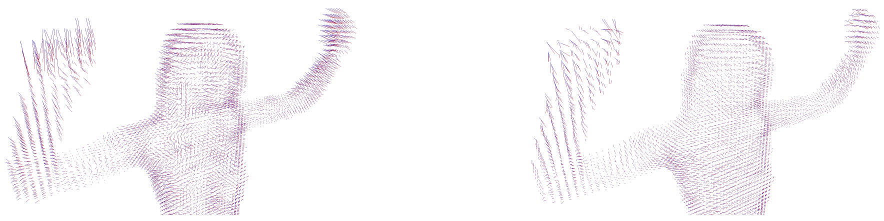

Figure 6: *Left: result for the Lucas-Kanade method. Right: result for the Horn-Schunck method.*

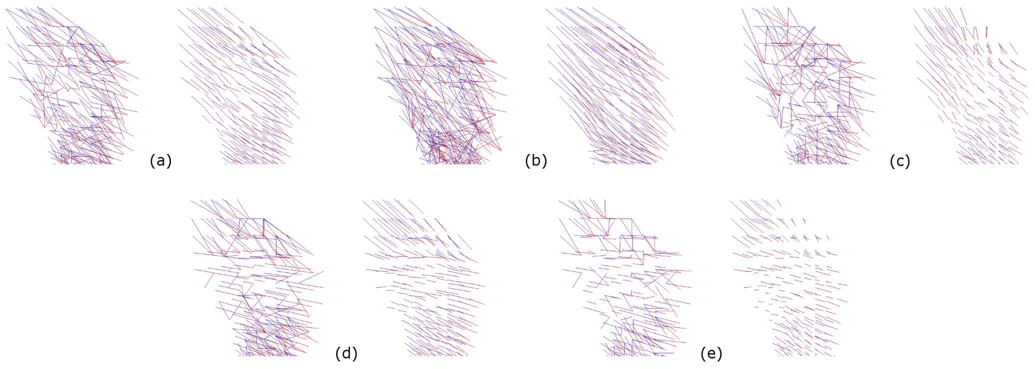

Figure 8: *Influence of the matching criteria. Results, before and after regularization, of the left hand's voxel matching: (a) with* $\omega_p = 1$*,* $\omega_n = 5$ *and* $\omega_c = 10$*, (b) without proximity criterion (*$\omega_p = 0$*), (c) without orientation criterion (*$\omega_n = 0$*), (d) without colorimetric criterion (*$\omega_c = 0$*), and (e) with all weights set to 1.*

		Average EDV			Average Hausdorff distance		
dataset	frame	Voxel Matching	Lucas-Kanade	Horn-Schunck	Voxel Matching	Lucas-Kanade	Horn-Schunck
girl	2	1.147	1.146	1.145	0.00157	0.00157	0.00157
	4	1.154	1.155	1.156	0.00157	0.00157	0.00156
	6	1.146	1.148	1.147	0.00167	0.00167	0.00168
	…	…	…	…	…	…	…
	14	1.145	1.148	1.148	0.00184	0.00184	0.00186
	…	…	…	…	…	…	…
	24	1.145	1.142	1.139	0.00175	0.00177	0.00196
	…	…	…	…	…	…	…
	28	1.144	1.134	1.130	0.00191	0.00206	0.00219
boy	1	1.131	1.126	1.117	0.00196	0.00202	0.00329
	2	1.126	1.123	1.111	0.00226	0.00224	0.00392
	3	1.130	1.132	1.113	0.00244	0.00240	0.00426

Table 1: *Mesh matching measurement*

contains 11912 vertices. For the *boy* dataset, we used 120 global iterations, and the template mesh contains 15646 vertices. The mesh processing took 80 seconds. To measure the matching quality of the deformed template and the target pose, we used two different metrics:

- **The Euclidean distance volume (EDV)**, which is the distance between each vertex of the deformed template mesh with the corresponding voxel. Its minimum is 1 for a vertex belonging to the voxel.
- **The Hausdorff distance**, which is the distance between the deformed template and a mesh obtained by visual hull reconstruction of the same frame (this value is computed with respect to the diagonal of the bounding box).

We tested the whole process with motion vectors obtained by our method (voxel matching) and by 3D optical flows (Lucas-Kanade and Horn-Schunck) with the same mesh regularization parameters. Results are shown in table 1. The av-

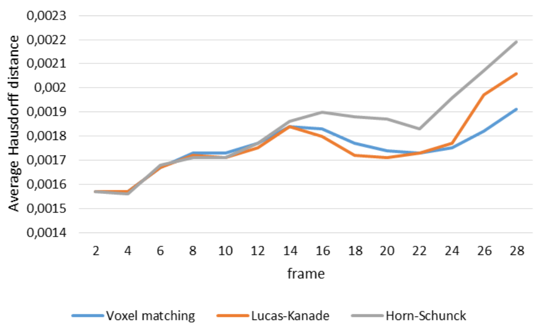

Figure 9: *Evolution of the average Hausdorff distance during the girl sequence.*

erage values obtained for the EDV are given on the left column. As expected, the volumetric distance stays stable because the mesh regularization method tends to push the vertices according to the distance volume gradient. The Hausdorff distance is more significant because it really measures the distance between the transformed template mesh and the target pose. For both datasets, the Horn-Schunck gives the worse results. The Lucas-Kanade approach and ours give similar results for the first frames. However, the results differ significantly from the real visual hull after several frames. With Horn-Schunck vectors, the results become inconsistent after 13 frames. With Lucas-Kanade, it stays robust for 23 frames. With our voxel matching approach, we obtain consistent results during the complete sequence, as demonstrated by the graph in figure 9.

While our method shows to be robust for the full sequence of the *girl* data, it is not for the *boy* sequence. This dataset is more complex in its type of movement, and there are significant changes in topology which appear frequently (fusion of the hands and arms with the torso for example). As shown in figure 11 (bottom), some mesh details are not properly recovered, like the stick in the left hand. The validity duration of the mesh template thus depends on the geometry topology changes rather than on the number of frames. When large topological changes occur, a new pose should be used as a new template, and the whole processing started again to continue the animation. We expect this limitation to vary depending on the volumetric resolution of the input.

4.3. Limitations and future work

In order to restrict the number of topology errors, our goal is to proceed the reconstruction of the first frame, which is used as template mesh, with a model's pose that limits ambiguities and using a high quality visual hull method, enhanced with stereo-based voxel carving. However, the changes in the topology of objects that could appear during the sequences are not well supported and may result in inconsistent motions. Our future implementations will have to integrate an adaptive shape model which could deal with these topology

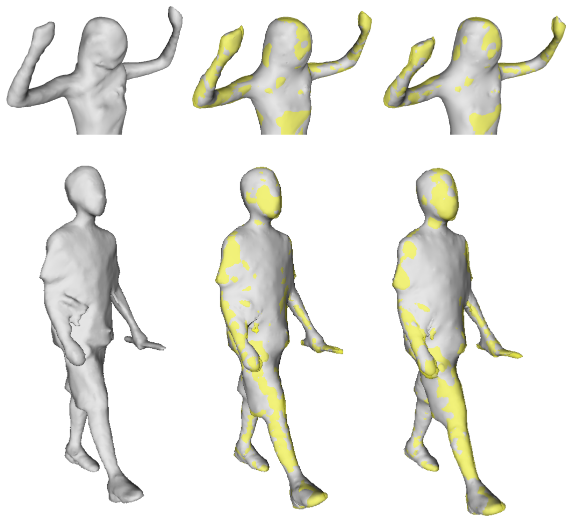

Figure 11: *Result of the mesh deformation for the girl (top) and the boy (bottom) sequences for three consecutive frames. Left: the initial pose used as template model. Middle and Right: the two following frames. The deformed template is in grey and the visual hull is superposed in yellow.*

modifications, as in the method proposed by Letouzey and Boyer [LB12] for example. Another issue is the number of parameters which have to be fixed by the user (weighting coefficients for voxel matching and mesh regularization, Gaussian filter radius, and number of iterations) and that may not be robust for all the sequence. These problems prevent us from computing efficiently an animation from long and complex sequences. The last limitation is the computation time, which could be reduced by the use of GPGPU technologies. Notice that all the processing times concern a simple CPU implementation. We currently do not use any kind of parallelization.

5. Conclusion

Our method allows us to compute a voxel matching for motion flow estimation. This correspondence is established without *a priori* knowledge about the nature of the volumes, except that they are of course supposed to represent the same object and belong to the same sequence. Our mesh deformation process, associated with a vertex regularization step, leads the mesh from the first frame to the pose defined by the next frame's reconstruction, providing a temporally coherent evolution. Our future work will focus on the identification of the changes occurring in the topology during the sequence. It could be argued that working on volumetric input could lead to approximations. However, this allows us to keep the input as generic as possible to later be able to transfer the motion flow to more precise modeling.

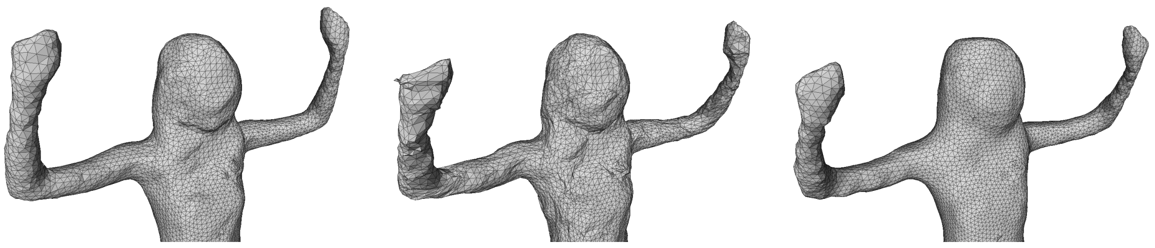

Figure 10: *Results of the mesh animation process. Left: template mesh in initial pose. Middle: same mesh after the application of the motion vectors. Right: final result after mesh regularization.*

6. Acknowledgments

We would like to thank our partner XD Productions. This work has been carried out thanks to the support of the RECOVER3D project, funded by the *Investissements d'Avenir* program of the French government, managed by DGCIS.

References

[AG04] ANUAR N., GUSKOV I.: Extracting animated meshes with adaptive motion estimation. In *9th International Workshop on Vision, Modeling and Visualization (VMV)* (2004), pp. 63–71. 2, 5

[BT04] BARRON J., THACKER A.: *Tutorial: Computing 2D and 3D Optical Flow*. Tech. Rep. 2004-012, Tina Memo, 2004. http://www.tina-vision.net/docs/memos/2004-012.pdf. 5

[CBI10] CAGNIART C., BOYER E., ILIC S.: Probabilistic deformable surface tracking from multiple videos. In *Proceedings of the 11th European conference on Computer vision: Part IV (ECCV)* (2010), pp. 326–339. 2

[dAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 98:1–98:10. 2

[GSDA*09] GALL J., STOLL C., DE AGUIAR E., THEOBALT C., ROSENHAHN B., SEIDEL H.-P.: Motion capture using joint skeleton tracking and surface estimation. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (VPR)* (2009), pp. 1746–1753. 2

[HS81] HORN B., SCHUNCK B.: Determining optical flow. *Artificial Intelligence 17*, 1-3 (1981), 185–203. 5

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* (1998), pp. 105–114. 4

[Lau94] LAURENTINI A.: Visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 2 (1994), 150–162. 1

[LB12] LETOUZEY A., BOYER E.: Progressive shape models. In *CVPR - Computer Vision and Patern Recognition - 2012* (June 2012), IEEE, pp. 190–197. 7

[LK81] LUCAS B. D., KANADE T.: An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop* (1981), pp. 121–130. 5

[LLV*12] LI H., LUO L., VLASIC D., PEERS P., POPOVIĆ J., PAULY M., RUSINKIEWICZ S.: Temporally coherent completion of dynamic shapes. *ACM Transactions on Graphics 31*, 1 (2012). 2

[LSG*11] LIU Y., STOLL C., GALL J., SEIDEL H.-P., THEOBALT C.: Markerless motion capture of interacting characters using multi-view image segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2011), pp. 1249–1256. 2

[LSI*13] LUCAS L., SOUCHET P., ISMAËL M., NOCENT O., NIQUIN C., LOSCOS C., BLACHE L., PRÉVOST S., REMION Y.: Recover3d: A hybrid multi-view system for 4d reconstruction of moving actors. In *4th International Conference on 3D Body Scanning Technologies* (Nov. 2013), pp. 219–230. 1

[NM04] NOBUHARA S., MATSUYAMA T.: Heterogeneous deformation model for 3D shape and motion recovery from multi-viewpoint images. In *Proceedings - 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)* (2004), pp. 566–573. 2, 5

[PLBF11] PETIT B., LETOUZEY A., BOYER E., FRANCO J.-S.: Surface flow from visual cues. In *16th International Workshop on Vision, Modeling and Visualization (VMV)* (Oct. 2011), pp. 1–8. 2

[SH07a] STARCK J., HILTON A.: Correspondence labelling for wide-timeframe free-form surface matching. In *Proceedings of the IEEE International Conference on Computer Vision* (2007). 2

[SH07b] STARCK J., HILTON A.: Surface capture for performance-based animation. *IEEE Computer Graphics and Applications 27*, 3 (2007), 21–31. 2

[ST94] SAITO T., TORIWAKI J.-I.: New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition 27*, 11 (1994), 1551–1565. 1, 3

[TM10] TUNG T., MATSUYAMA T.: Dynamic surface matching by geodesic mapping for 3D animation transfer. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), pp. 1402–1409. 2

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. In *SIGGRAPH'08: International Conference on Computer Graphics and Interactive Techniques* (2008). 2

[VBR*99] VEDULA S., BAKER S., RANDER P., COLLINS R., KANADE T.: Three-dimensional scene flow. In *Proceedings of the IEEE International Conference on Computer Vision* (1999), vol. 2, pp. 722–729. 2

[VZBH08] VARANASI K., ZAHARESCU A., BOYER E., HORAUD R.: Temporal surface tracking using mesh evolution. In *10th European Conference on Computer Vision (ECCV)* (2008), vol. 5303 of *Lecture Notes in Computer Science (LNCS)*, pp. 30–43. 2