Vis Comput (2016) 32:205–216
DOI 10.1007/s00371-015-1191-y

ORIGINAL ARTICLE

Robust motion flow for mesh tracking of freely moving actors

L. Blache[1] · C. Loscos[1] · L. Lucas[1]

Published online: 7 December 2015

Abstract 4D multi-view 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 present a method to derive a time-evolving surface representation from a sequence of binary volumetric data representing an arbitrary motion 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 lack temporal coherence, in terms of structure and topology, as each frame is generated independently. This prevents an easy post-production editing with 3D animation tools. Our goal is to transform this input sequence of independent 3D volumes into a single dynamic structure, directly usable in post-production. 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 multi-view datasets, show that our method outperforms other approaches based on optical flow when considering robustness over several frames.

Keywords Multi-view reconstruction · Motion flow · Dynamic mesh · Voxel matching · Mesh animation

✉ L. Blache
ludovic.blache@univ-reims.fr

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

1 Introduction

The entertainment industry increasingly relies on a mix of real pictures and computer-generated images. While advanced technologies such as motion capture or matte painting are widely used, new methods of *multi-view reconstruction* are now emerging. These methods are based on a *virtual cloning* system that uses a set of multi-viewpoint cameras in an indoor studio to generate an animated 3D model of an actor's performance, without the need for the traditional markers used ubiquitously in motion capture techniques. Most of these multi-view reconstruction studios use a *model-free* approach that generates a 3D object for each frame of the multi-view video sequence. The resulting series of static poses are not well suited for subsequent editing as they are devoid of any temporal coherency or known correspondences. Several projects have been developed to achieve this temporally consistent multi-view reconstruction (e.g., [1,23,28]).

This paper fits in the RECOVER3D project [16] which purpose is to adapt this new technology for TV production. In this context, a 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 [10], producing a sequence of 3D volumes over time (see Fig. 1). We assume that the input is a binary 3D array that represents a sequence of poses generated by this reconstruction process along with the colors captured by the video. These volumes are usually transformed into a sequence of 3D textured meshes, successively loaded for the rendering of each frame. In this constrained industrial framework, our goal is to introduce a dynamic representation of the captured character, that adds a temporally consistent description of the scene. Our ultimate goal is to generate a single, time-evolving triangle mesh representing the motion of the actor in the

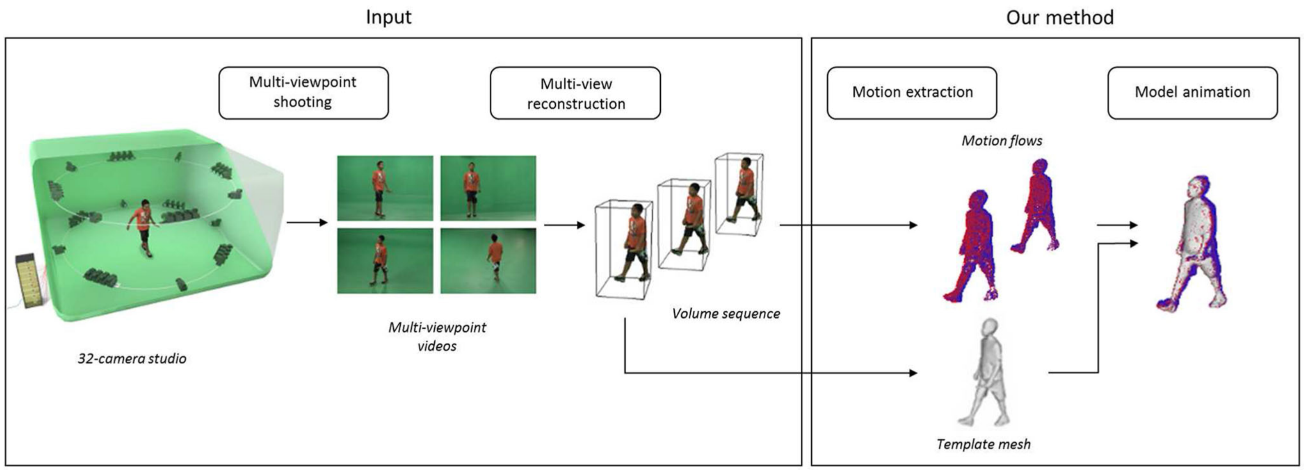


Fig. 1 An overview of our production process. Our method focuses on the motion flow computation and the mesh animation

capture sequence. As the production teams need a practical animated mesh for scene's compositing, we want to create a single, temporally consistent, animated model following the character's motion. This technology should be used to perform the reconstruction of various types of scenes, involving actors wearing costumes and accessories, or even animals. This requirement prevents the use of most existing methods that assume rigidly articulated models. This project is oriented towards broadcast and TV markets, that is why rapid computations are preferred to go towards short production deadlines.

To answer these requirements, 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. We extract the scene motion by computing a 3D motion flow from the input 3D volume sequence. The particularity of our method is to combine two different types of computations with a back and forth approach: a Euclidean distance transform [20] and a choice of complementary criteria (proximity, orientation, and color) that permit to discriminate voxel matching. After the motion flow is filtered, we use it to match a chosen template mesh (one of the sequence frames) to the subsequent meshes by pairs of frames, regularized using a mass-spring system in an iterative approach, in order to create a unique mesh, animated over time. This method works on generic datasets, whatever the shape of the reconstructed object or character.

In Sect. 2, a brief overview of recent advances in model tracking is given. In Sect. 3, our approach is explained, giving details on the object's representation (Sect. 3.1), the motion extraction (Sects 3.2–3.4) and the mesh animation process (Sect. 3.5). Results are then presented in Sect. 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. Multi-view reconstruction methods are usually separated into two main approaches: model-based and model-free.

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) [10] or depth maps (stereo) reconstruction [8]. 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, and animated object. Starck and Hilton [23] 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. Zheng et al. [29] extract a skeleton from each frame of a scanned sequence. They then compute a unique consensus skeleton, matching the successive poses, to derive a time-consistent reconstruction. Nevertheless, this approach is limited to clearly articulated shapes, which is not compatible with our goal. Mitra et al. [17] proposed a method for dynamic registration of scanned surfaces by computing rigid transformations. The authors then propose an extension for deformable bodies, noticing that their transformations can be considered as locally rigid. However, their methods seems sensitive to fast motions and important inter-frame deformations. Tevs et al. [24] match a model through a scanned sequence thanks to set of landmarks correspondences. We can also mention Li et al. [13] who developed a temporally consistent completion of scanned meshes' sequences, using a deformation graph to establish pairwise correspondences between consecutive frames. This registration is based on a non-rigid ICP algorithm [12]. Nevertheless, this method is applied on high definition meshes (despite the scan holes) whereas our visual hull surfaces are less trustworthy.

Model-based multi-view reconstruction approaches use a template model representing an actor—typically, an articulated mesh of a generic human body. De Aguiar et al. [1] use a similar approach with a tetrahedral mesh reconstructed from a 3D scan acquisition. The multi-view reconstruction is then achieved by deforming this template in time according to a set of directives (optical flow or silhouette matching) extracted from the multi-viewpoint videos. Vlasic et al. [28] and Gall et al. [7] employ a predefined skeleton to match the template model with a set of poses defined by silhouettes or visual hulls, before applying local deformation on the mesh to match free-form elements (such as clothes or hair). The main advantage of this family of techniques is the temporally coherent animation they produce. Nevertheless, the use of a template model restricts their generality. Model-based methods are most often limited to a single human model, even if some variants (e.g., Liu et al. [14]) allow the tracking of several actors. Cagniart et al. [5] employ a dynamic surface based on a deformable tile set, initialized with the first frame of the sequence. This mesh is then deformed to match the subsequent poses. Alain et al. [2] uses a similar patch-based parameterization. Following the same kind of approach, we will use the first pose of the sequence as our initial mesh that we then deform based on the rest of the sequence. 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 ...).

Several *motion flow*-based approaches have been proposed to achieve a reconstruction with temporal consistency from a sequence of poses. A mesh-tracking method as proposed by Starck and Hilton [22], Varanasi et al. [26], Furukawa and Ponce [6] or Tung and Matsuyama [25] can match several meshes according to curvature, texture criteria, or photo-consistency, from which one can compute the motion flow describing the movements of an actor between two frames [19]. Motion flow extraction is related to the *scene flow* described by Vedula et al. [27], obtained by merging in the 3D space the optical flows of a multi-camera rig. Anuar and Guskov [3] proposed to compute the 3D optical flow using a voxel-based method. The motion flow can then be used to animate a mesh over time. In our case, however, the visual hull reconstruction usually creates significant artifacts which prevent us from using such a mesh-tracking approach directly. Instead, we can use a volumetric approach to compute a motion flow based on a voxel matching algorithm applied to the input sequence, as proposed by Nobuhara and Matsuyama [18]. A 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.

Discussion

Traditional articulated model-based approaches present discriminating constraints for our application context. Motion flows could be considered limited due to the amplitude of the movements but they can help managing any type of model and work on volumes. We thus decided to explore a motion flow approach and use it to animate a template mesh, generated by our model-free multi-view reconstruction.

3 Our approach

Our input is a sequence of n digital volumes obtained by a silhouette-based reconstruction from multi-view video. The ith volume of this sequence represents the actor at time t_i (see Fig. 1). Our method starts by estimating a 3D motion flow between two consecutive frames by computing a voxel matching based both on local geometry and color between the two corresponding consecutive poses. 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, obtained by a *marching cube* algorithm, 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 Volume description

The reconstructed volumes we use are simple digital volumes, a regular 3D grid of isotropic binary voxels (0 for void voxels and 1 for voxels covering or intersecting the object). Voxels straddling the surface (i.e., those that are not void, yet direct neighbors of void voxels) are assigned a color as well, extracted from the multi-view video frames: the color associated to a voxel is interpolated from the multi-viewpoint images which contains this point. Each surface voxel is then associated with an RGB color (see Fig. 2a). Note that for simplicity of our data structure, we implement a volume as an RGBA array, where the alpha channel is set to 0 for void voxels, 1 for internal voxels, and 0.5 for surface voxels. We then compute another representation of these volumes using a *Euclidean distance transform* (EDT), as described by Saito and Toriwaki [20]. We obtain an unsigned distance volume, represented by a 3D gray-level voxel grid, as shown in Fig. 2b. 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

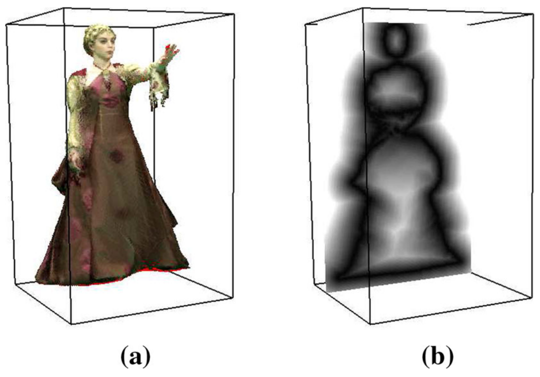


Fig. 2 Volume processing. **a** An example of colored reconstructed volume issued from multi-view reconstruction. **b** A sliced representation of the corresponding EDT. **a** Volume, **b** EDT

a gray-level 3D picture. Thus, we can compute a derivative estimation of this picture. It will be used to compute the normal vectors (see Sect. 3.2.2) and gradient values. To compute the spatial derivative, we use a set of Sobel-like filters which estimate, in a $3 \times 3 \times 3$ window around each voxel, 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.

3.2 Voxel matching

Given two consecutive volumes V^n and V^{n+1} which correspond to frames t_n and t_{n+1}, our goal is to compute a matching $V^n \rightarrow V^{n+1}$ 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 Sect. 3.2.1), an orientation criterion (see Sect. 3.2.2) and a colorimetric criterion (see Sect. 3.2.3). ω_p, ω_n and ω_c are weighting terms, fixed by the user. In our experimentations we used $\omega_p = 0.3$, $\omega_n = 0.45$, and $\omega_c = 0.25$. 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 immerse the binary volume V^0 in the EDT grid of V^1, so that the EDT value associated to each surface voxel of V^0 represents its distance to the next pose at time t_1. This distance is used to automatically define a *search radius* which corresponds to the maximum amplitude of the motion. 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 Fig. 4, top). Thus, we ensure that each surface voxel in V^n and V^{n+1} is associated to at least one vector.

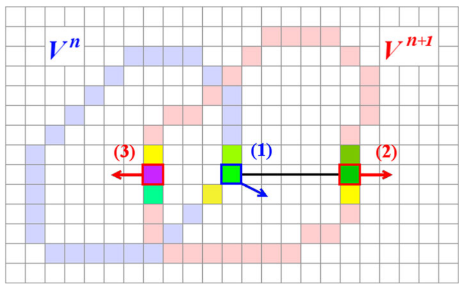


Fig. 3 Voxel matching between two consecutive volumes. The voxel (*1*) from the V^n volume matches better the voxel (*2*) from the V^{n+1} volume than the voxel (*3*). The neighboring voxels are represented with their *colors*. Normal vectors are figured by *arrows*

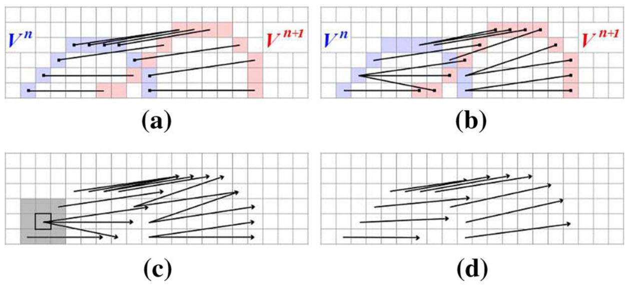


Fig. 4 *Top* forward (**a**) and backward (**b**) matching between the two volumes. *Bottom* Gaussian filter (in *gray*) applied to the raw vector fields (**c**) and final motion field (**d**)

3.2.1 Proximity criterion

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

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

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 Fig. 5b).

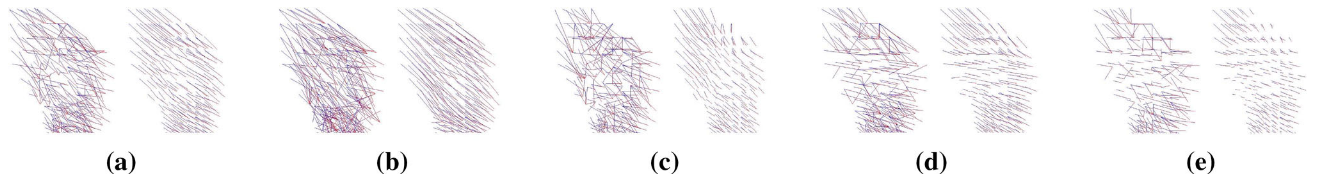


Fig. 5 Influence of the matching criteria. Results, before and after regularization, of the left hand's voxel matching: **a** with $\omega_p = 0.3$, $\omega_n = 0.45$, and $\omega_c = 0.25$, **b** without proximity criterion ($\omega_p = 0$), **c** without orientation criterion ($\omega_n = 0$), **d** without colorimetric criterion ($\omega_c = 0$), and **e** with all weights set to 1

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 Fig. 5c, this criterion penalizes the matching of two voxels which belong to back facing surfaces. For example, in Fig. 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:

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

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 around v_i^n and v_j^{n+1}, respectively,

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

if v_{i+k}^n belongs to the surface. This constraint favors the matching of two voxels which belong to close color blocks corresponding to the same object's part (see Fig. 5d).

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 Fig. 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 Prediction

Considering that, at a time t_i, the motion flow between t_i and $t_i + 1$ should be a continuity of the previous motion vectors (between t_{i-1} and t_i), we can improve our voxel matching algorithm once the motion vectors between the first two frames of the sequence has been estimated. Each surface voxel at t_i uses the motion vector previously estimated between t_{i-1} and t_i to predict the position of the matching voxel in t_{i+1}. We then search for the best matching voxel in t_{i+1} using a noticeably reduced radius around the predicted position, offering a highly efficient speedup of 60 % compared to the brute force algorithm. This prediction is repeated for the next matching phases throughout the whole sequence.

3.5 Mesh animation

We construct an initial (template) mesh based on the first volume of the sequence by extracting it using a *marching cubes* algorithm over the alpha value. We clean up the resulting triangle mesh through edge collapse simplification to ensure that each triangle is non-degenerate, and will thus not create numerical artifacts in our subsequent tracking.

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. The result is too irregular to be used (see Fig. 6), details due to non-rigid deformation (such as cloth folds or hair) will be missed. Moreover, mesh quality may also degrade over time as large deformation happens, making mesh regularization desirable. Thus, we once again

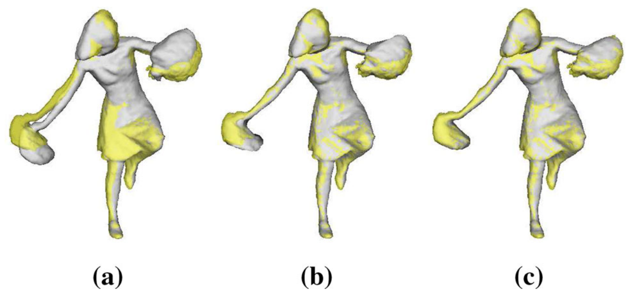


Fig. 6 Results of the mesh animation process. Template mesh (in *gray*) in initial pose (**a**). The same mesh after the application of the motion vectors towards the next visual hull in *yellow* (**b**). The final result after mesh regularization (**c**)

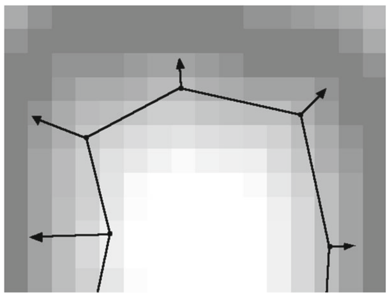


Fig. 7 Silhouette fitting. Vertices are moved normal to the surface, according to the EDT values

apply a regularization algorithm, this time to obtain a regular mesh which corresponds to the pose defined by the visual hull. We thus compute local vertex displacements based on both fitting accuracy and regularization as follows.

Regularization
We regularize the mesh by applying spring-like forces to favor equi-length edges:

$$f_r(p_i) = \sum_{j \in N(i)} (\|p_i - p_j\| - \bar{r}_i) \frac{p_i - p_j}{\|p_i - p_j\|}$$

where p_j is a vertex from the one-ring neighborhood of p_i, while the rest length $\bar{r}_i$ is set to the current average length of the edges adjacent to p_i. We use only the tangential component of the resulting vector.

Silhouette fitting
Using the EDT, we also push each vertex toward the visual hull surface (see Fig. 7) by adding the following force:

$$f_s(p_i) = \mathbf{n}_{\mathbf{p_i}} \cdot \mathrm{EDT}(p_i)$$

with $\mathbf{n}_{\mathbf{p_i}}$ and $\mathrm{EDT}(p_i)$ being the normal vector and the EDT value at p_i, respectively.

We integrate the sum of these two forces over 250 time steps by updating position and velocity of each vertex (assumed to be all of unit mass) using a simple Runge–Kutta explicit integrator to make the integration trivially parallelizable. Weighting the two forces further allows the user to control regularization vs shape fitting depending on the noise present in the volume sequence.

4 Results

We tested our method on several datasets obtained through volumetric visual hull reconstruction. The *girl* dataset contains simple motions, with a woman slowly moving her arms. The visual hull volume has a 73 × 132 × 43 voxels resolution and is reconstructed for 30 frames. The first mesh, used as a template, contains 11,912 vertices. The *cheerleader* and *astronaut* sequences both contain 25 volumes, with an average 180 × 270 × 170 voxel resolution. The template mesh contains 19,234 vertices for the *cheerleader* sequence and 8048 for the *astronaut* sequence. These three datasets come from an indoor studio shoot using a 24-camera rig. The *astronaut* contains a nearly rigid but large motion of the arms. The *cheerleader* is more challenging due to the free-moving shapes of the pom-pom, the skirt and the large motions of the arms. The *dancer* sequence was generated using the multi-viewpoint images provided by the GrImage platform[1] with an average 150 × 100 × 300 voxels resolution for 30 frames. This dataset presents a quick dancing motion, with large movements of the arms and a dress, with a lower quality of the visual hull. We also tested our method on the *capoeira* sequence, using the multi-view videos described in [1] to generate a 30 frames sequences. This last sequence contains fast motions, especially for the legs, which lead to large inter-frame displacements. 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 Fig. 9). Figure 8 presents the results for the three sequences produced by our multi-view studio. The motion flow between two frames is computed in an average time of 50 and 100 s for the *cheerleader* (Fig. 8a) and *astronaut* (Fig. 8c) sequences, respectively. For the *girl* sequence (Fig. 8a), the motion is computed in less than 10 s. The motion estimation is performed in 60 s per frame with the *dancer* sequence (Fig. 8d). Due to the fast inter-frame motion in the *capoeira* sequence, the motion flow computation took an average of 200 s (Fig. 8e).

We compared our approach with our own implementations of two 3D-adapted optical flow algorithms as presented in [4]:

[1] http://4drepository.inrialpes.fr/.

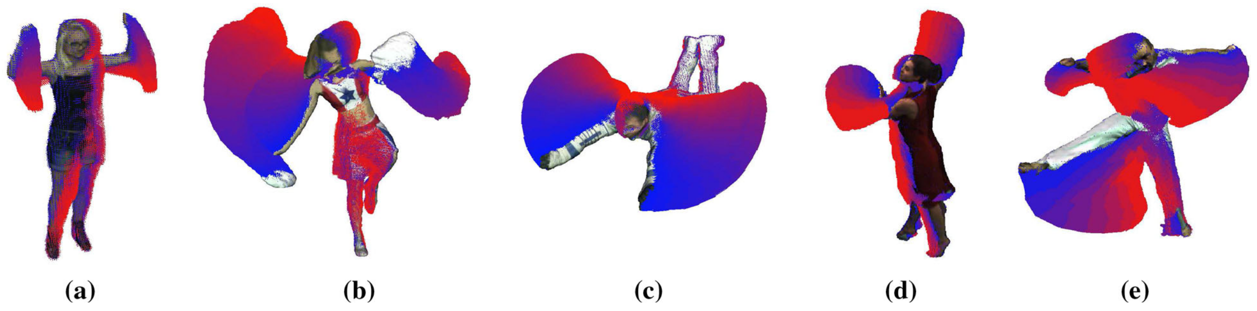


Fig. 8 Accumulated motion flows through several frames of five test sequences (motion vectors are oriented from *blue* to *red*). **a** *Girl*, **b** *Cheerleader*, **c** *Astronaut*, **d** *Dancer*, **e** *Capoeira*

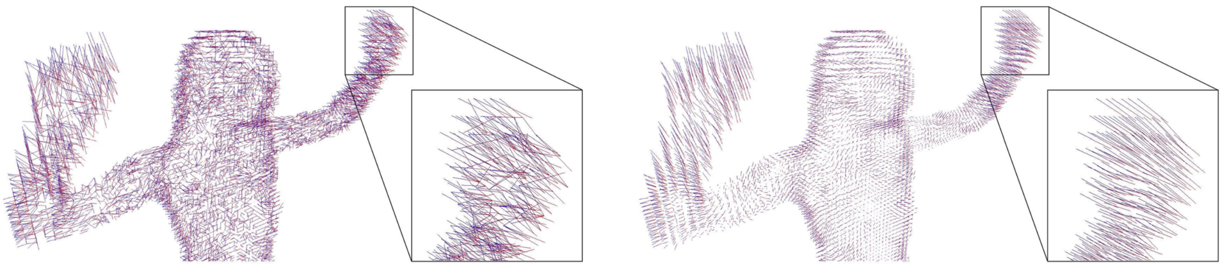


Fig. 9 Motion field regularization. *Left* result of the voxel matching step. *Right* vector field after regularization (vectors are oriented from *blue* to *red*)

the first one is based on the Lucas and Kanade method [15] and the second one on the variational approach by Horn and Schunck [9]. For comparison, the method described in [3] is using a similar approach to the Lucas–Kanade version, applied on a discrete distance function, as our EDT. Our tests show that for similar settings, the Lucas–Kanade approach is faster (less than 5 s for *girl*) but displacement vectors are not oriented correctly (see an example of results in Fig. 10a 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 min on the same dataset) and does not give convincing results with displacement distances not corresponding to the actual movement (see Fig. 10b). With the other datasets, these limitations of our Horn–Schunck implementation are increased, due to the higher resolution of the volumes. We thus compared our motion flow computing with the Lucas–Kanade flows. Results are presented in the Table 1 for three datasets. With the other sequences, the motion's amplitude between two consecutive frames is too large to get a consistent flow with the Lucas–Kanade implementation. On the other hand, our back and forth voxel matching ensures that the motion flow covers the whole displacements, whatever their amplitude.

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 Sect. 4.3) which confirms our observations on the flows.

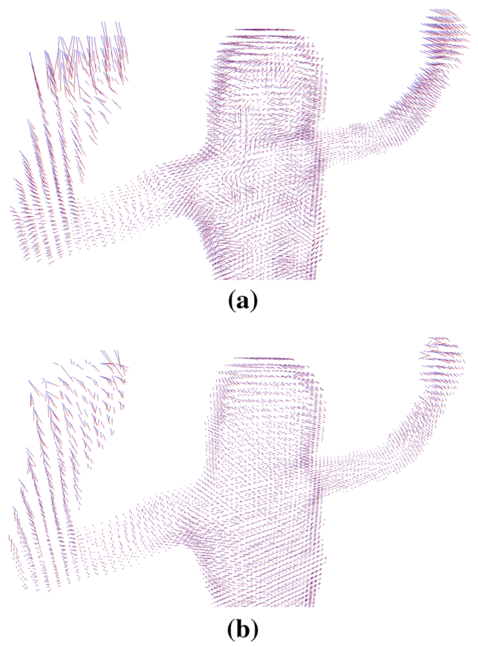


Fig. 10 Result of the two 3D optical flow computation between two frames of the *girl* sequence. **a** Lucas–Kanade, **b** Horn–Schunck

Table 1 Mesh matching measurement (average Hausdorff distance) between animated mesh and visual hull of the target pose

Frame	Girl		Dancer		Cheerleader	
	Lucas–Kanade	Voxel matching	Lucas–Kanade	Voxel matching	Lucas–Kanade	Voxel matching
2	0.00157	0.00157	0.007325	0.003321	0.002224	0.001772
3	0.00157	0.00157	0.008143	0.003091	0.002180	0.001846
4	0.00157	0.00157	0.007863	0.003061	0.002558	0.001914
5	0.00160	0.00160	0.006692	0.002173	0.002914	0.002244
6	0.00167	0.00167	0.006649	0.002682	0.002564	0.002196
7	0.00172	0.00173	0.006525	0.002569	0.002709	0.002054
8	0.00171	0.00173	0.007404	0.002431	0.002872	0.002150
…	…	…		…	…	…
12	0.00175	0.00177		0.003065	0.003166	0.001854
…	…	…		…	…	…
16	0.00180	0.00183		0.007644	0.003232	0.002061
…	…	…		…	…	…
20	0.00171	0.00174		0.005482	0.003198	0.002085
…	…	…		…	…	…
24	0.00177	0.00175		0.003628	0.002930	0.001945
…	…	…		…		
28	0.00206	0.00191		0.002933		

4.2 Discussion on the chosen parameters

The method presented by [18], which uses only Euclidean distance (meaning, in our case, $\omega_n = \omega_c = 0$) is less efficient than the results we obtain with our multiple criteria approach. Figure 5 shows the influence of the three criteria (proximity, orientation, and color) for voxel matching, defined by weights ω_p, ω_n, and ω_c (see Eq. (1)), fixed by the user. Figure 5b shows that without the proximity criterion ($\omega_p = 0$), most of the matched voxels are too distant. 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 Fig. 5c, most of the voxels are matched with another voxel which is close but corresponds to a backfacing surface. Figure 5d 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, Fig. 5e 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. These results show that this voxel-based approach can be made drastically more robust for visual hulls if one considers orientation and texture of the voxels for matching and proper filtering. It is really the combination of the three criteria that improves the quality of the matching process.

4.3 Qualitative evaluation of the mesh animation

After the application of motion vectors' translations, the template mesh (see Fig. 6a) is altered (see Fig. 6b). After our iterative regularization, we obtain a smooth and regular mesh which matches the pose at each frame (see Fig. 6c). The *girl* mesh processing between two frames took around 50 s. Mesh adjustment (see Sect. 3.5) was applied with weights 0.6 and 0.4 for regularization vs. fitting. We used 50 iterations for the numerical integration step. The first mesh, used as a template, contains 11,912 vertices. We applied a maximum of 200 iterations during the mesh regularization step for the *cheerleader astronaut* sequences, with an average computing of 100 s. For the *dancer* sequence, for which the template mesh contains 7843 vertices, the mesh animation is processed in 60 s. Results from these sequences are presented in Fig. 11, demonstrating robustness of our approach in light of the coarseness of the input volumes. Our two-steps mesh animation allows non-rigid deformations and will thus remove the potential geometric features present at the initial time which should not be kept *as is* in time (for instance, a wrinkle on a dress).The *cheerleader* dataset shows that shape of the pom-poms is correctly adjusted after the global deformation phase (see Fig. 6b, c).

To measure the matching quality of the deformed template and the target pose, we used the average Hausdorff distance as metric, which represents the distance between the deformed template and a mesh obtained by visual hull

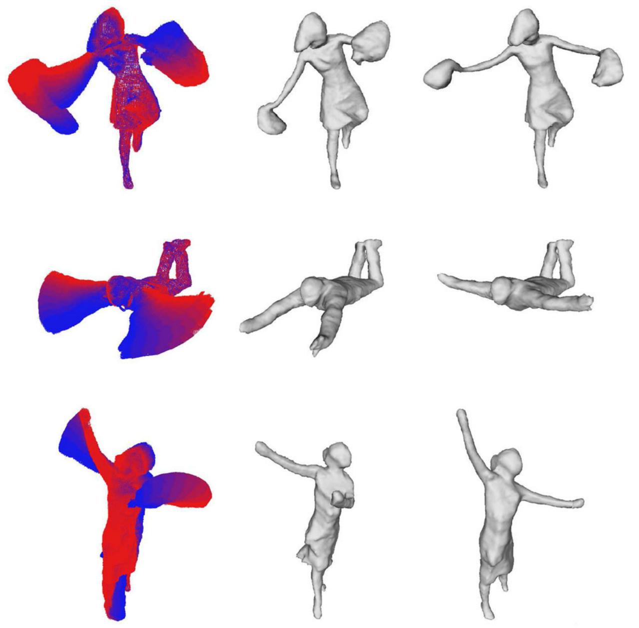


Fig. 11 Result of the mesh animation for the *Cheerleader*, *Astronaut* and *Dancer* sequences. *From left to right* the extracted motion flow volume (motion vectors oriented from *blue* to *red*), the initial mesh, and the result of the mesh animation

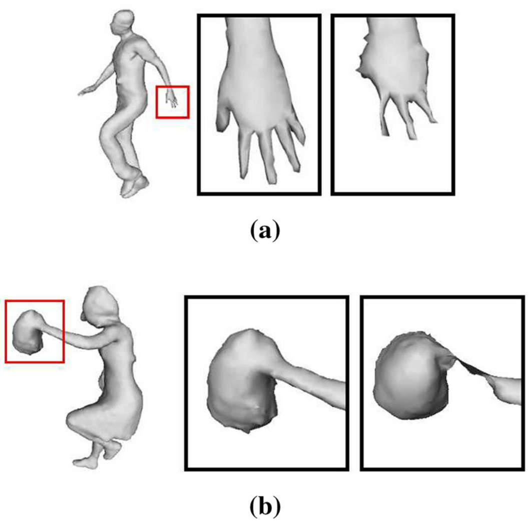


Fig. 12 Stress case. **a** Example of small details where the mesh slowly degrades over time by lack of dense enough information. **b** Other kind of mesh drift occurs when the motion flow does not correctly matches the new pose of the actor. Here is an example with the *cheerleader* dataset, animated with the 3D Lucas–Kanade motion flow

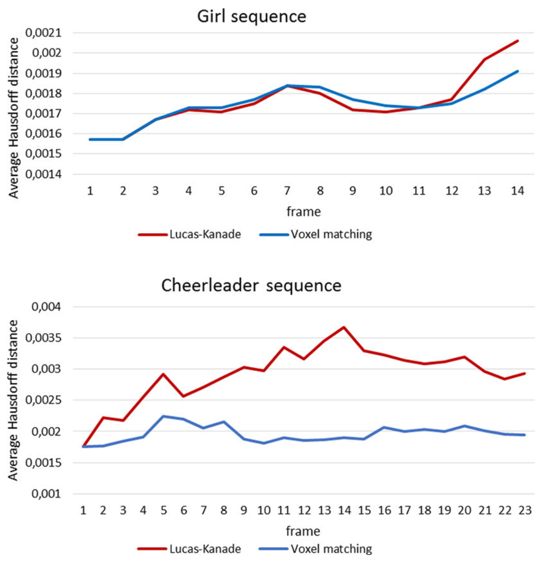


Fig. 13 Evolution of the average Hausdorff distance during the *girl* and *cheerleader* sequences

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 flow with the same mesh regularization parameters. The Table 1 presents the evolution of this matching metric during three sequences. We can see that after several frames, the Hausdorff distance increases for the Lucas–Kanade approach whereas it stays stable with our method. These divergences correspond to the inconsistencies that appear in the mesh after several frames, due to a bad motion flow. If the Lucas–Kanade approach and ours give similar results for the first poses, the results differ significantly from the real visual hull after several frames. With Lucas–Kanade, the mesh animation stays robust for 23 frames of the *girl* sequence. With our voxel matching approach, we obtain consistent results during the complete sequence. With the *dancer* dataset, the shape matching, using Lucas–Kanade, starts to produce inconsistent results after only three frames. Similar results are obtained with the *cheerleader* where the mesh animation becomes inconsistent after five frames (see Fig. 12b). Whereas with our motion flow, the Hausdorff distance stay stable (see Fig. 13). The *girl* dataset is the only one where the mesh matching stay robust during the whole sequence using the 3D optical flow approach, even if the visual mesh consistency is quickly lost, because of the low motion's amplitude. With the other sequences, the 3D Lucas–Kanade implementation fails to maintain a consistent mesh animation after a few frames. The consistency of the matching of the matching between the animated mesh and the visual hulls is also measured by comparing the color of the voxels that correspond to each vertex during the sequence. We compute the distance in the colorimetric space between the closest voxels of a vertex in two consecutive frames and repeat this for each pair of frames through the whole sequence. The resulting values are normalized with respect to the maximum color difference (between black and white). The average difference is 0.042 for the *dancer* dataset (with a standard deviation of 0.059) and 0.039 for the *cheerleader* sequence (with a

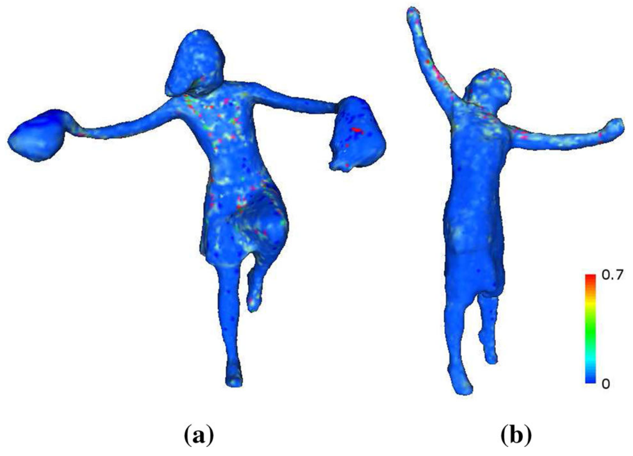


Fig. 14 Representation of the texture matching metric on the animated mesh for the *cheerleader* (**a**) and *dancer* sequences. Each vertex is associated with a *color* that represents the colorimetric difference of the closest voxels in two consecutive frames. A good matching corresponds to a small value. The minimum value is 0 (*blue*) and maximum is 0.7 (*red*)

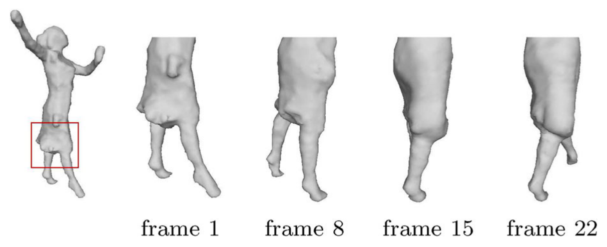


Fig. 15 Example of free-moving clothes' tracking in the *dancer* sequence

standard deviation of 0.058). These values shows that, despite the regularization of the mesh, the moving vertices stay globally associated with the same parts of the surface during the animation. The maximum differences are logically observed in the regions animated by the largest motions, as shown in Fig. 14.

Our mesh animation method leads to an adaptation of the template during the sequence, avoiding some of the model-based inconveniences, as in [1], where the tracked model retains some of the surface details (clothing folds) from the initial pose during the whole sequence. The cheerleader dataset shows that shape of the pom-poms is correctly adjusted after the global deformation phase (see Fig. 6b, c). With the *dancer* sequence, we show that the mesh correctly tracks the shape of the moving dress (see Fig. 15). This type of animation would be hardly recovered with an articulated model tracking approach. However, in this extreme case, our local regularization step degrades the details of the hands as they are of a size too close to the size of a grid element. While a decrease of the regularization factor can mitigate this issue, thin features cannot be well preserved. A local optimization with subgrid accuracy (i.e., super-resolution) could solve this issue, most likely at the cost of a significantly increased computational complexity. 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.

4.4 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 modifications, as in the method proposed by Letouzey and Boyer [11], for example. As presented in Fig. 12, our method is still sensitive to thin details recovering. Figure 12a shows an example on a synthetic dataset that contains thin details such as fingers. As described in Sect. 4.3, the animated mesh may become inconsistent if the motion flow does not properly matches the successive actor's poses. These problems often occurs when we use the 3D Lucas–Kanade approach (as presented in Fig. 12b) instead of our voxel matching algorithm, but can also occur after several frames even using our method, especially due to large displacements between two frames. A mesh animation based on an *ARAP* deformation [21], guided by the motion vectors, could ensure a better conservation of the mesh structure during the sequence and lead to more robust results. The datasets on which we tested our method where around 30 frames long. We wish to work on longer sequences in our future improvements. Another issue is the number of parameters which have to be fixed by the user or empirically determined (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

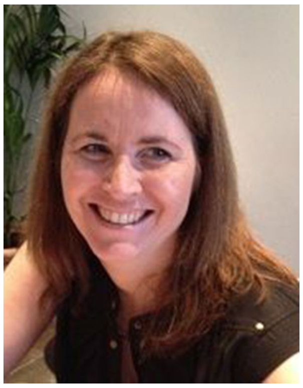
Our method allows us to compute a voxel matching for motion flow estimation. This correspondence is established without a priori knowledge about the nature of the volumes, except that they are of course supposed to represent the same object and belong to the same sequence. Note that the motion flow may be used, as a descriptor of the actor's movements, in other applications such as interaction of the reconstructed character with its virtual environment. 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.

Acknowledgments We would like to thank our industrial partner XD Productions (Paris). This work has been carried out thanks to the support of the RECOVER3D project, funded by the *Investissements d'Avenir* program, managed by DGCIS. Some of the captured performance data were provided courtesy of the research group 3D Video and Vision-based Graphics of the Max-Planck-Center for Visual Computing and Communication (MPI Informatik/Stanford) and Morpheo research team of INRIA and laboratoire Jean Kuntzmann (Grenoble University).

References

1. de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.P., Thrun, S.: Performance capture from sparse multi-view video. ACM Trans. Graph. **27**(3), 98:1–98:10 (2008)
2. Allain, B., Franco, J.S., Boyer, E., Tung, T.: On mean pose and variability of 3D deformable models. In: European Conference on Computer Vision, ECCV 2014, pp. 284–297 (2014)
3. Anuar, N., Guskov, I.: Extracting animated meshes with adaptive motion estimation. In: International Workshop on Vision, Modeling and Visualization (VMV), pp. 63–71 (2004)
4. Barron, J., Thacker, A.: Tutorial: Computing 2D and 3D optical flow. In: Technical Report 2004-012, Tina Memo (2004)
5. Cagniart, C., Boyer, E., Ilic, S.: Probabilistic deformable surface tracking from multiple videos. In: 11th European Conference on Computer Vision (ECCV). Lecture Notes in Computer Science, vol. 6314, pp. 326–339 (2010)
6. Furukawa, Y., Ponce, J.: Dense 3D motion capture from synchronized video streams. In: IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008, pp. 1–8 (2008)
7. Gall, J., Stoll, C., de Aguiar, E., Theobalt, C., Rosenhahn, B., Seidel, H.P.: Motion capture using joint skeleton tracking and surface estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1746–1753 (2009)
8. Hilton, A., Starck, J.: Multiple view reconstruction of people. In: Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT), pp. 357–364 (2004)
9. Horn, B., Schunck, B.: Determining optical flow. Artif. Intell. **17**(1–3), 185–203 (1981)
10. Laurentini, A.: Visual hull concept for silhouette-based image understanding. IEEE Trans. Pattern Anal. Mach. Intell. **16**(2), 150–162 (1994)
11. Letouzey, A., Boyer, E.: Progressive shape models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 190–197 (2012)
12. Li, H., Adams, B., Guibas, L.J., Pauly, M.: Robust single-view geometry and motion reconstruction. ACM Trans. Graph. **28**(5), 175:1–175:10 (2009)
13. Li, H., Luo, L., Vlasic, D., Peers, P., Popović, J., Pauly, M., Rusinkiewicz, S.: Temporally coherent completion of dynamic shapes. ACM Trans. Graph. **31**(1), 2:1–2:11 (2012)
14. 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 Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1249–1256 (2011)
15. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of Imaging Understanding Workshop, pp. 121–130 (1981)
16. 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, pp. 219–230 (2013)
17. Mitra, N.J., Flory, S., Ovsjanikov, M., Gelfand, N., Guibas, L., Pottmann, H.: Dynamic geometry registration. In: Symposium on Geometry Processing, pp. 173–182 (2007)
18. Nobuhara, S., Matsuyama, T.: Heterogeneous deformation model for 3D shape and motion recovery from multi-viewpoint images. In: Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT), pp. 566–573 (2004)
19. Petit, B., Letouzey, A., Boyer, E., Franco, J.S.: Surface flow from visual cues. In: International Workshop on Vision, Modeling and Visualization (VMV), pp. 1–8 (2011)
20. Saito, T., Toriwaki, J.I.: New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. Pattern Recognit. **27**(11), 1551–1565 (1994)
21. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP, pp. 109–116 (2007)
22. Starck, J., Hilton, A.: Correspondence labelling for wide-timeframe free-form surface matching. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1–8 (2007)
23. Starck, J., Hilton, A.: Surface capture for performance-based animation. IEEE Comput. Graph. Appl. **27**(3), 21–31 (2007)
24. Tevs, A., Berner, A., Wand, M., Ihrke, I., Bokeloh, M., Kerber, J., Seidel, H.P.: Animation cartography—intrinsic reconstruction of shape and motion. ACM Trans. Graph. **31**(2), 12:1–12:15 (2012)
25. Tung, T., Matsuyama, T.: Dynamic surface matching by geodesic mapping for 3D animation transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1402–1409 (2010)
26. Varanasi, K., Zaharescu, A., Boyer, E., Horaud, R.: Temporal surface tracking using mesh evolution. In: 10th European Conference on Computer Vision (ECCV). Lecture Notes in Computer Science, vol. 5303, pp. 30–43 (2008)
27. Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. Proc. IEEE Int. Conf. Comput. Vis. (ICCV) **2**, 722–729 (1999)
28. Vlasic, D., Baran, I., Matusik, W., Popović, J.: Articulated mesh animation from multi-view silhouettes. ACM Trans. Graph. **27**(3), 97:1–97:9 (2008)
29. Zheng, Q., Sharf, A., Tagliasacchi, A., Chen, B., Zhang, H., Sheffer, A., Cohen-Or, D.: Consensus skeleton for non-rigid space-time registration. Comput. Graph. Forum **29**(2), 635–644 (2010)

Ludovic Blache obtained a Master's degree at the University of Aix-Marseille in 2011. He is currently a PhD candidate at the University of Reims Champagne-Ardenne, studying dynamic mesh animation applied on multi-view reconstruction. His research interests include geometric modeling and computer graphics.



Céline Loscos is a professor in computer science at the University of Reims Champagne-Ardenne, at the CReSTIC laboratory. She obtained her PhD degree at the University Joseph Fourier in Grenoble in 1999. She then joined University College London as a lecturer until 2007. She was a full time researcher at the Girona Graphics Group in Spain until 2010. Her research topics include HDR imaging, illumination simulation, modeling and relighting for mixed reality.

Laurent Lucas received PhD in computer science from the University of Reims Champagne-Ardenne (URCA) in 1995. He currently leads the SIC research group and is also in charge of the virtual reality platform of the URCA. Full professor since 2003 in the Computer Sciences Department of the Graduate Institute of Technology, his research interests include visualization and co-operation between image processing and computer graphics particularly in 3DTV and theirs applications. He has co-authored more than 50 peer-reviewed international publications in these areas. His current research focuses on 3D vision and he is in charge of the RECOVER3D project.