Dense and Accurate Spatio-Temporal Multi-View Stereovision IMAGINE Research Report 09-43

Jérôme Courchay, Jean-Philippe Pons, Renaud Keriven, Pascal Monasse

Dense and Accurate Spatio-Temporal Multi-View Stereovision IMAGINE Research Report 09-43

Jérôme Courchay[1], Jean-Philippe Pons[2], Renaud Keriven[1], and Pascal Monasse[1]

[1] IMAGINE
École des Ponts ParisTech
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77455 Marne-la-Vallée cedex 2 - France
[2] CSTB
290, route des Lucioles
BP 209
06904 Sophia Antipolis Cedex

Abstract. In this report, we describe a novel method to simultaneously and accurately estimate the 3D shape and 3D motion of a dynamic scene from multiple-viewpoint calibrated videos. We follow a variational approach in the vein of previous work on stereo reconstruction and scene flow estimation. We adopt a representation of a dynamic scene by an animated mesh, i.e. a polygonal mesh with fixed connectivity whose time-varying vertex positions sample the trajectories of material points. Interestingly, this representation ensures a consistent coding of shape and motion by construction. Our method accurately recovers 3D shape and 3D motion by optimizing the positions of the vertices of the animated mesh. This optimization is driven by an energy function which incorporates multi-view and inter-frame photo-consistency, smoothness of the spatio-temporal surface and of the velocity field. Central to our work is an image-based photo-consistency score which can be efficiently computed and which fully handles projective distortion and partial occlusions. We demonstrate the effectiveness of our method on several challenging real-world dynamic scenes.

1 Introduction

In recent years, several methods for automatic generation of complete spatio-temporal models of dynamic scenes from multiple videos have been proposed [1–15]. In particular, the most recent ones have proven effective for full-body marker-less motion capture, yielding visually impressive results. However, when taking a closer look at the aforementioned techniques, it becomes apparent that very few of them achieve a desirable *coupled, dense and accurate 3D shape and 3D motion estimation*.

Accurate 3D shape. Many recent techniques still produce an approximate geometry: free-form deformation of a template body model [2, 11, 15], visual hull [1, 3, 15], Laplacian deformation of a laser scan of the initial pose [4, 5]. These methods are unable to recover genuine geometric details such as facial expressions and clothing folds and wrinkles.

Accurate 3D motion estimation is crucial in some applications like motion transfer and time interpolation. Also, a coarse motion estimation precludes the enforcement of temporal consistency constraints during coupled shape and motion estimation. However, in most existing performance capture techniques, 3D scene flow [16], i.e. the dense 3D motion field of the scene, is not accurately estimated. Often, it is interpolated from sparse 3D correspondences [3, 4, 12]. Some methods do not address 3D motion estimation whatsoever: [7] uses a four-dimensional level set representation which, beyond its very high computational and memory requirements, does not encode 3D correspondence. [10, 15] produce animated meshes but, despite appearances, the underlying 3D correspondences are purely artifactual.

Coupled 3D shape and 3D motion estimation allows to exploit their redundancy, and has long been recognized [17] as a desirable way to improve their performance. However, most marker-less motion capture methods fail to integrate spatio-temporal consistency constraints. In [3, 12, 13], shape is computed independently in each time frame, prior to motion estimation. In [9], shape and motion are estimated sequentially, not simultaneously. In [5], an initial mesh is propagated by 3D scene flow, under silhouette constraints, but without any stereo cues; as a result, this method suffers from temporal drift. The latter is circumvented in [4] by substituting sparse 3D correspondences for dense 3D scene flow, but neither shape or motion are accurate enough to allow enforcing spatio-temporal consistency. In [1, 7], a certain degree of spatio-temporal coherence is obtained through four-dimensional representations, but as these representations do not encode temporal correspondence, they cannot exploit inter-frame matching constraints. In [14], shape and motion are estimated simultaneously using a plane-sweep carving algorithm in a 6D space, but this approach has a very high computational and memory cost, is limited to two frames, and is unable to enforce the smoothness of the recovered shape and motion.

Thus, to our knowledge, two methods [6, 8] achieve this highly desirable coupled, dense and accurate 3D shape and 3D motion estimation. In [8], shape and motion are represented through the detail coefficients of a time-varying subdivision surface. The latter coefficients are estimated by simultaneously optimizing

multi-view and inter-frame photo-consistency. However, the non-linearity of the chosen multi-resolution representation makes this optimization intricate. Also, the required motion initialization relies on the spatio-temporal derivatives of the input images, thereby making it applicable mainly to slowly-moving Lambertian scenes under constant illumination.

[6] is the only work to date which can handle complex real-world dynamic scenes. Despite the effectiveness of this method, we believe that the expansion framework used does not allow to take into account the full visibility depending on occluding patch not computed yet.

In this paper, we propose a novel method to simultaneously and accurately estimate the 3D shape and 3D motion of a dynamic scene from multiple-viewpoint videos. First, we follow a **variational approach** in the vein of previous work on stereo reconstruction and scene flow estimation [9, 17, 19–22]. None of these methods fits our applications in their current state: most are limited to a single time-varying depth map of the scene [17, 19–22], while others do not enforce spatio-temporal consistency constraints [9, 20].

Second, we adopt a representation of a dynamic scenes by an **animated mesh**, i.e. a polygonal mesh with fixed connectivity whose time-varying vertex positions sample the trajectories of material points. Interestingly, this representation ensures a consistent coding of shape and motion by construction. It is widely used in computer graphics, especially in computer animation. It is also popular for performance capture from video [3–6, 10, 11, 15] or from time-varying point clouds [23, 24] (the latter being obtained from video or from fast 3-D scanning hardware).

Our method accurately recovers 3D shape and 3D motion by optimizing the positions of the vertices of the animated mesh. This optimization is driven by an energy function which incorporates multi-view and inter-frame photo-consistency, smoothness of the spatio-temporal surface and of the velocity field. Central to our work is an image-based photo-consistency score which can be efficiently computed and which fully handles projective distortion and partial occlusions, in the spirit of [9].

The rest of this research report is organized as follows. In Section 2, we describe in detail the discrete geometric representation, the variational formulation, the energy function and the associated minimization procedure which constitute our approach. In Section 3, we discuss implementation aspects and we demonstrate the effectiveness of our method on several challenging real-world dynamic scenes.

2 Our Approach

2.1 Discretize then optimize

An overwhelming majority of variational methods in this area [9, 17, 19, 20, 22] and more generally in computer vision, rely on an *optimize then discretize* approach: an energy functional depending on a continuous infinite-dimensional

spatio-temporal representation is considered, the gradient of this energy functional is computed analytically, then the obtained evolution flow is discretized.

In contrast, we adopt a *discretize then optimize* approach: we define an energy function depending on a discrete finite-dimensional spatio-temporal representation, and we use standard non-convex optimization tools. The benefits of this approach have long been recognized in mesh processing, but have seldom been demonstrated in computer vision [25–27]. Thus, the choice of an adequate discrete spatio-temporal representation is crucial in our work.

2.2 Animated mesh representation

In our context, animated polygonal meshes present many significant advantages. Compared to unrelated meshes at different time instants, they are more compact, easier to store and to manipulate. They provide a direct access both to the shape of the scene at a given time instant, and to motion trajectories. 3D shape and 3D motion are mutually consistent by construction.

Their fixed topology may be regarded as a limitation, as argued in [12]. We believe that it is not, since the human body has a constant - spherical, if disregarding pierces - topology. It is questionable to treat a character with hands on hips as a genus-2 torus. It should rather be regarded as a topological sphere with some temporary contact regions.

Furthermore, let us mention that our method is not limited to a spherical topology: while the topology of the animated mesh is constant *across time*, we are able to modify it *across our optimization process* using a spatio-temporal version of Delaunay deformable models [28].

2.3 Variational formulation

In the following, we consider a dynamic scene, imaged by N calibrated and synchronized video sequences composed of T frames, and represented by an animated polygonal mesh with K vertices. We note:

- $I_{i,t} : \Omega_i \subset \mathbb{R}^2 \to \mathbb{R}^d$, $i \in \{1..N\}$, $t \in \{1..T\}$ the input images. In practice $d = 1$ for grayscale images and $d = 3$ for color images.
- $\boldsymbol{X} = \{\boldsymbol{x}_{k,t},\, k \in \{1..K\},\, t \in \{1..T\}\}$ the 3D positions of the vertices of the animated mesh at the different time instants,
- $\boldsymbol{X}_t = \{\boldsymbol{x}_{k,t},\, k \in \{1..K\}\}$ the t^{th} temporal slice of the animated mesh.

In the sequel, by a slight abuse of notation, we indistinctly use $\boldsymbol{X}$ and $\boldsymbol{X}_t$ to refer to the animated mesh and to the positions of its vertices.

The energy to minimize with respect to $\boldsymbol{X}$ is composed of a data attachment term, of a regularization term for the spatio-temporal surface and of a regularization term for the velocity field:

$$E(\boldsymbol{X}) = E_D(\boldsymbol{X}) + \lambda_S E_S(\boldsymbol{X}) + \lambda_V E_V(\boldsymbol{X})\ . \tag{1}$$

E_D encourages multi-view and frame-to-frame matching consistency. It is defined as the sum over camera pairs (i, j) and pairs of time frames (t, u) of

a dissimilarity measure between image $I_{i,t}$ and the reprojection of $I_{j,u}$ via the animated mesh. The detailed description of this term is left to Section 2.4.

E_S favors the regularity of the spatio-temporal surface. We use the total area of the animated mesh. The minimization of this term by gradient descent yields a discrete version of the well known mean curvature motion, which we implement as described in [29].

E_V penalizes rapid variations of the velocity field along the animated mesh. It is the total squared L^2 norm over the animated mesh of the gradient of the velocity field. The detailed description of this term is left to Section 2.8.

We minimize the above energy function using a standard gradient descent on the spatio-temporal positions $\boldsymbol{X}$. In order to avoid unwanted local minima, we resort to a multi-resolution and chronological scheme. We first optimize the first three frames of a low-resolution animated mesh using low-resolution versions of input images. Then we initialize an additional time frame by extrapolating 3D position from speed and acceleration of previous frames. We iteratively add time frames, and optimize the sequence using a sliding time window of a few frames, until we reconstruct the whole temporal sequence at low resolution. We repeatedly apply the above procedure with increased image and mesh resolutions, until we reach the desired accuracy.

2.4 Data attachment term

The formal definition of E_D and of its gradient requires some additional notations. The perspective projection performed by camera i is denoted by $\Pi_i : \mathbb{R}^3 \to \mathbb{R}^2$. Our method takes into account the visibility of the surface points. We refer to $\boldsymbol{X}_{i,t}$ as the part of the temporal slice $\boldsymbol{X}_t$ visible in image i. The back-projection of a point of camera i on the animated mesh at frame t is denoted by $\Pi^{-1}_{i,\boldsymbol{X}_t} : \Pi_i(\boldsymbol{X}_t) \to \boldsymbol{X}_{i,t}$.

We also define 3D transport functions $T_{\boldsymbol{X}_t \rightharpoonup \boldsymbol{X}_u}$ that map points in $\boldsymbol{X}_t$ to points in $\boldsymbol{X}_u$. This can be written formally using the linear finite-element representation depicted in Figure 1. For each vertex k of the animated mesh at some time frame t, we define a basis function $\phi_{k,\boldsymbol{X}_t}$ such that (i) $\phi_{k,\boldsymbol{X}_t}(\boldsymbol{x}_{k,t}) = 1$ (ii) $\forall l \neq k$, $\phi_{k,\boldsymbol{X}_t}(\boldsymbol{x}_{l,t}) = 0$ (iii) $\phi_{k,\boldsymbol{X}_t}$ varies linearly inside the triangular facets adjacent to the k^{th} vertex, and cancels outside this ring. We then have at pixel p_i in image i:

$$T_{\boldsymbol{X}_t \rightharpoonup \boldsymbol{X}_u} = \sum_k \boldsymbol{x}_{k,u}\, \phi_{k,\boldsymbol{X}_t} \ . \tag{2}$$

In a simpler way we can say that the back-projection Y_t of pixel p_i lies on a triangular facet f and has barycentric coordinates $\phi_{l,\boldsymbol{X}_t}(Y_t)$ at time t, l being a vertex of f. So the position of this particle at time u is $Y_u = \sum_{l \in f} \boldsymbol{x}_{l,u}\, \phi_{l,\boldsymbol{X}_t}(Y_t)$, that is $Y_u = \sum_{k \in \boldsymbol{X}} \boldsymbol{x}_{k,u}\, \phi_{k,\boldsymbol{X}_t}(Y_t)$ since $\phi_{k,\boldsymbol{X}_t}(Y_t)$ cancels if vertex k is outside facet f.

Finally, we define image transport functions $T_{(i,\boldsymbol{X}_t) \rightharpoonup (j,\boldsymbol{X}_u)}$ which map positions in $I_{i,t}$ to positions in $I_{j,u}$ via the animated mesh:

$$T_{(i,\boldsymbol{X}_t) \rightharpoonup (j,\boldsymbol{X}_u)} = \Pi_j \circ T_{\boldsymbol{X}_t \rightharpoonup \boldsymbol{X}_u} \circ \Pi^{-1}_{i,\boldsymbol{X}_t} \ . \tag{3}$$

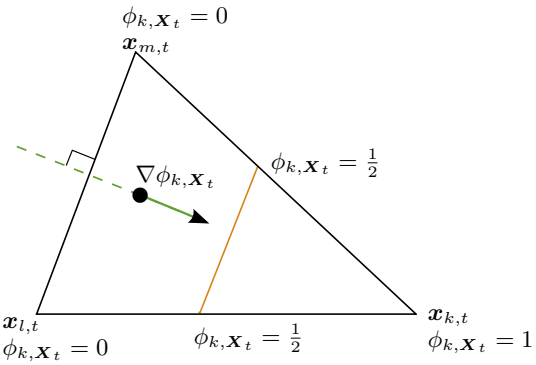

Fig. 1. Finite element representation over a facet (k, l, m) of the animated mesh.

With these notations in hand the reprojection of image j at time u in image i at time t via the animated mesh writes $I_{j,u} \circ T_{(i,\boldsymbol{X}_t) \rightharpoonup (j,\boldsymbol{X}_u)}$. This is illustrated in Figure 2.

The data attachment term is the sum over oriented camera pairs (i, j) and oriented pairs (t, u) of time frames of a dissimilarity measure M between image $I_{i,t}$ and the above defined reprojection of $I_{j,u}$ via the animated mesh. The dissimilarity is computed only over the region of image plane i where both images are defined, i.e. after discarding semi-occluded regions. This image region writes $\mathit{\Pi}_i \left(\boldsymbol{X}_{i,t} \cap T_{\boldsymbol{X}_u \rightharpoonup \boldsymbol{X}_t}(\boldsymbol{X}_{j,u})\right)$.More clearly, pixel p_i in image i is visible in both images, if its back-projection lies on the surface at time t, and this point on the surface once transported at time u is visible (nore occluded, nore outside the image frame) in image $I_{j,u}$. This visible image region is computed before each optimization step on graphics hardware. For conciseness, we will omit it in the equations below:

$$E_D(\boldsymbol{X}) = \sum_{i,j} \sum_{t,u} M \left[I_{i,t} \ , \ I_{j,u} \circ T_{(i,\boldsymbol{X}_t) \rightharpoonup (j,\boldsymbol{X}_u)} \right] \ . \tag{4}$$

We now compute the partial derivative of this energy term with respect to the variation of a single position $\boldsymbol{x}_{k,t}$ of the animated mesh. First, we note that the only oriented pairs of time frames affected by such a variation are (u, t) and (t, u), $u \in \{1..T\}$. Second, when the animated mesh moves, the reprojected image changes. Hence the partial derivative of E_D involves the derivative of the similarity measure M with respect to its second argument, denoted by $\partial_2 M$.

Using the chain rule, and after some index manipulations, we get:

$$\begin{aligned} \frac{\partial E_D}{\partial \boldsymbol{x}_{k,t}} = & \sum_{i,j} \sum_{u} \\ & \int_{\Omega_i} \partial_2 M \left[I_{i,t} \ , \ I_{j,u} \circ T_{(i,\boldsymbol{X}_t) \rightharpoonup (j,\boldsymbol{X}_u)} \right] DI_{j,u} \frac{\partial T_{(i,\boldsymbol{X}_t) \rightharpoonup (j,\boldsymbol{X}_u)}}{\partial \boldsymbol{x}_{k,t}}(\boldsymbol{p}_i) \, d\boldsymbol{p}_i \\ & + \int_{\Omega_j} \partial_2 M \left[I_{j,u} \ , \ I_{i,t} \circ T_{(j,\boldsymbol{X}_u) \rightharpoonup (i,\boldsymbol{X}_t)} \right] DI_{i,t} \frac{\partial T_{(j,\boldsymbol{X}_u) \rightharpoonup (i,\boldsymbol{X}_t)}}{\partial \boldsymbol{x}_{k,t}}(\boldsymbol{p}_j) \, d\boldsymbol{p}_j \ , \end{aligned} \tag{5}$$

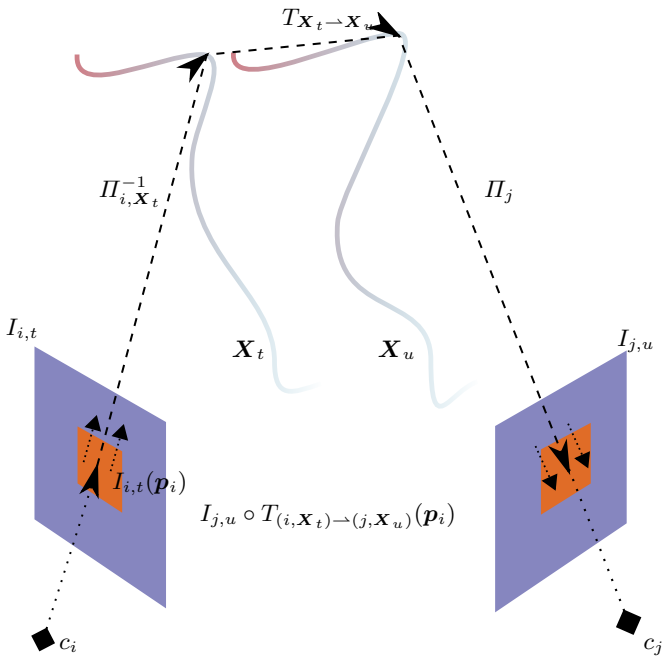

Fig. 2. Reprojection of image j at time u in image i at time t via the animated mesh.

where $DI_{\cdot,\cdot}$ denotes the Jacobian matrices of the input images. For conciseness, we have omitted the points where the latter are evaluated in the above equation.

As regards the quantities $\frac{\partial T_{\dots}}{\partial \boldsymbol{x}_{k,t}}$, we can make several observations. First, they are purely geometric, i.e. independent of image data. Second, they cancel outside the ring of triangular facets adjacent to the k^{th} vertex. Hence, despite appearances, integration is performed only over the visible projection of this ring in the different images, not over the full image domains. Third, these quantities involve the normal of the triangular facet visible at pixel p_i, and the barycentric coordinate of $\boldsymbol{x}_{k,t}$ in this facet. Complete expressions can be obtained using a non trivial geometric reasoning. here below the detailed numerical computation, but also an additional intuitive solving are proposed. The numerical solving, mainly consist in computing how barycentric coordinates change for a small perturbation of the surface.

2.5 Data energy gradient flow

Back-projection on X of the pixel p_i in image i writes:

$$\Pi^{-1}_{i,\boldsymbol{X}_t}(p_i) = \sum_k \boldsymbol{x}_{k,t}\, \phi_{k,\boldsymbol{X}_t}$$

After a small perturbation δX of surface, then the back projected point vary along the optical ray $\boldsymbol{d}_i$, then one get:

$$\Pi^{-1}_{i,\boldsymbol{X}_t+\delta X_t}(p_i) = \sum_k (\boldsymbol{x}_{k,t} + \boldsymbol{\delta x_{k,t}})\ \phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}$$

And α being unknown:

$$\sum_k \boldsymbol{x}_{k,t}\,\phi_{k,\boldsymbol{X}_t} + \alpha \boldsymbol{d}_i = \sum_k (\boldsymbol{x}_{k,t} + \boldsymbol{\delta x_{k,t}})\ \phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}$$

In an other way,

$$\alpha \boldsymbol{d}_i = \sum_k \boldsymbol{x}_{k,t}\ (\phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t} - \phi_{k,\boldsymbol{X}_t}) + \sum_k \boldsymbol{\delta x_{k,t}}\,\phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}$$

Now as $\sum_k \boldsymbol{x}_{k,t}\ (\phi_{k,\boldsymbol{X}_t+\delta X_t} - \phi_{k,\boldsymbol{X}_t})$ is a vector lying on the surface one get that is dot product with facet normal is null, so:

$$\alpha = \sum_k \phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}\ \frac{\boldsymbol{N}\cdot\boldsymbol{\delta x_{k,t}}}{\boldsymbol{N}\cdot\boldsymbol{d}_i}$$

So by noticing $\boldsymbol{K}_{k,t} = \boldsymbol{\delta x_{k,t}} - \frac{\boldsymbol{N}\cdot\boldsymbol{\delta x_{k,t}}}{\boldsymbol{N}\cdot\boldsymbol{d}_i}\boldsymbol{d}_i$ we have the following equations:

$$\sum_k (\boldsymbol{x}_{k,t} + \boldsymbol{K}_{k,t})\ \phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t} = \sum_k \boldsymbol{x}_{k,t}\,\phi_{k,\boldsymbol{X}_t}$$

Now as back-projection lies on the facet f we have $\sum_{k\in f}\phi_{k,\boldsymbol{X}_t} = 1$, so one can notice that:

$$\phi_{1,\boldsymbol{X}_t}(\boldsymbol{x}_{1,t} - \boldsymbol{x}_{3,t}) + \phi_{2,\boldsymbol{X}_t}(\boldsymbol{x}_{2,t} - \boldsymbol{x}_{3,t}) + \boldsymbol{x}_{3,t} = \sum_k \boldsymbol{x}_{k,t}\,\phi_{k,\boldsymbol{X}_t}$$

And if we consider small displacements, the back-projected point still lies on the facet f and so one can write the vector equality:

$$\sum_{k=1,2} \phi_{k,\boldsymbol{X}_t}(\boldsymbol{x}_{k,t} - \boldsymbol{x}_{3,t}) = \sum_{k=1,2} \phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}\left[(\boldsymbol{x}_{k,t} - \boldsymbol{x}_{3,t}) + (\boldsymbol{K}_{k,t} - \boldsymbol{K}_{3,t})\right]$$

Our objective is to compute the unknowns $\phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}$ which allows to retrieve back-projected particle position on the mesh after perturbation. So, to find $\phi_{2,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}$ we just have to compute the cross product of left part and right part of equation 6 with the vector $(\boldsymbol{x}_{1,t} - \boldsymbol{x}_{3,t}) + (\boldsymbol{K}_{2,t} - \boldsymbol{K}_{3,t})$ to obtain finally with a first order limited development $\phi_{2,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}$ and in a more general way $\phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t}$ with a symetric formulation:

$$\phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta} X_t} = \phi_{k,\boldsymbol{X}_t} + \frac{[(\boldsymbol{x}_{k-1,t} - \boldsymbol{x}_{k+1,t}) \times (\sum_v \phi_{v,\boldsymbol{X}_t}\,\boldsymbol{K}_{v,t})]\cdot\boldsymbol{N}}{[(\boldsymbol{x}_{k,t} - \boldsymbol{x}_{k+1,t}) \times (\boldsymbol{x}_{k-1,t} - \boldsymbol{x}_{k+1,t})]\cdot\boldsymbol{N}}$$

At time u the position variation after and before perturbation is:

$$\boldsymbol{\Delta X} = \sum_k \left(\boldsymbol{x}_{k,u} + \boldsymbol{\delta x_{k,u}}\right)\, \phi_{k,\boldsymbol{X}_t+\boldsymbol{\delta}X_t} - \sum_k \boldsymbol{x}_{k,u}\, \phi_{k,\boldsymbol{X}_t}$$

So by extending the formulation of $\boldsymbol{K}_{k,t}$ in function of $\boldsymbol{\delta x_{k,t}}$ and with simple reformulation given that $\boldsymbol{a}\cdot(\boldsymbol{c}\times\boldsymbol{b}) = -\boldsymbol{b}\cdot(\boldsymbol{c}\times\boldsymbol{a})$ one can easily check we get:

$$\begin{aligned}\boldsymbol{\Delta X} = &\textstyle\sum_k \boldsymbol{\delta x_{k,u}}\, \phi_{k,\boldsymbol{X}_t} \\ &+ \textstyle\sum_k \left(\sum_v \boldsymbol{\delta x_{v,t}}\, \phi_{v,\boldsymbol{X}_t}\right) \cdot \frac{\boldsymbol{N}\times(\boldsymbol{x}_{k-1,t}-\boldsymbol{x}_{k+1,t})}{[(\boldsymbol{x}_{k,t}-\boldsymbol{x}_{k+1,t})\times(\boldsymbol{x}_{k-1,t}-\boldsymbol{x}_{k+1,t})]\cdot\boldsymbol{N}}\, \boldsymbol{x}_{k,u} \\ &- \textstyle\sum_k \left(\sum_v \boldsymbol{\delta x_{v,t}}\, \phi_{v,\boldsymbol{X}_t}\right) \cdot \frac{[(\boldsymbol{x}_{k-1,t}-\boldsymbol{x}_{k+1,t})\times\boldsymbol{d}_i]\cdot\boldsymbol{N}}{[(\boldsymbol{x}_{k,t}-\boldsymbol{x}_{k+1,t})\times(\boldsymbol{x}_{k-1,t}-\boldsymbol{x}_{k+1,t})]\cdot\boldsymbol{N}}\, \frac{\boldsymbol{N}}{\boldsymbol{N}\cdot\boldsymbol{d}_i}\, \boldsymbol{x}_{k,u}\end{aligned}$$

Still with triple product formula $\boldsymbol{b}(\boldsymbol{a}\cdot\boldsymbol{c}) = \boldsymbol{c}(\boldsymbol{a}\cdot\boldsymbol{b}) + \boldsymbol{a}\times(\boldsymbol{b}\times\boldsymbol{c})$, and by denoting $\boldsymbol{V_{k,t}} = \frac{\boldsymbol{N}\times(\boldsymbol{x}_{k-1,t}-\boldsymbol{x}_{k+1,t})}{[(\boldsymbol{x}_{k,t}-\boldsymbol{x}_{k+1,t})\times(\boldsymbol{x}_{k-1,t}-\boldsymbol{x}_{k+1,t})]\cdot\boldsymbol{N}}$ we get:

$$\begin{aligned}\boldsymbol{\Delta X} = &\textstyle\sum_k \boldsymbol{\delta x_{k,u}}\, \phi_{k,\boldsymbol{X}_t} \\ &+ \left(\textstyle\sum_k \left[\boldsymbol{V_{k,t}}\right]_\times \left[\boldsymbol{x}_{k,u}\right]_\times + \left(\boldsymbol{V_{k,t}}\cdot\boldsymbol{x}_{k,u}\right) Id\right) \left(\textstyle\sum_v \boldsymbol{\delta x_{v,t}}\, \phi_{v,\boldsymbol{X}_t}\right) \\ &- \left(\textstyle\sum_k \boldsymbol{V_{k,t}}\cdot\boldsymbol{d}_i\, \boldsymbol{x}_{k,u}\right) \left(\textstyle\sum_v \frac{\phi_{v,\boldsymbol{X}_t}\boldsymbol{\delta x_{v,t}}\cdot\boldsymbol{N}}{\boldsymbol{N}\cdot\boldsymbol{d}_i}\right)\end{aligned}$$

That is,

$$\begin{aligned}\boldsymbol{\Delta X} = &\textstyle\sum_k \boldsymbol{\delta x_{k,u}}\, \phi_{k,\boldsymbol{X}_t} \\ &+ \left(\textstyle\sum_k \left[\boldsymbol{V_{k,t}}\right]_\times \left[\boldsymbol{x}_{k,u}\right]_\times + \left(\boldsymbol{V_{k,t}}\cdot\boldsymbol{x}_{k,u}\right) Id\right) \left(\textstyle\sum_v \boldsymbol{\delta x_{v,t}}\, \phi_{v,\boldsymbol{X}_t}\right) \\ &- \left(\textstyle\sum_k \left[\boldsymbol{V_{k,t}}\right]_\times \left[\boldsymbol{x}_{k,u}\right]_\times + \left(\boldsymbol{V_{k,t}}\cdot\boldsymbol{x}_{k,u}\right) Id\right) \boldsymbol{d}_i \left(\textstyle\sum_v \frac{\phi_{v,\boldsymbol{X}_t}\boldsymbol{\delta x_{v,t}}\cdot\boldsymbol{N}}{\boldsymbol{N}\cdot\boldsymbol{d}_i}\right)\end{aligned}$$

By denoting the deformation matrix relative to the facet

$$D_f\left(t,u\right) = -\sum_k \left[\boldsymbol{V_{k,t}}\right]_\times \left[\boldsymbol{x}_{k,u}\right]_\times - \left(\boldsymbol{V_{k,t}}\cdot\boldsymbol{x}_{k,u}\right) Id$$

One have the simple formula:

$$\boldsymbol{\Delta X} = \sum_v \boldsymbol{\delta x_{v,u}}\, \phi_{v,\boldsymbol{X}_t} + D_f\left(t,u\right) \sum_v \phi_{v,\boldsymbol{X}_t} \left(-\boldsymbol{\delta x_{v,t}} + \boldsymbol{d}_i\, \frac{\boldsymbol{\delta x_{v,t}}\cdot\boldsymbol{N}}{\boldsymbol{N}\cdot\boldsymbol{d}_i}\right)$$

The variation of energy corresponding to a surface perturbation $\epsilon\boldsymbol{\delta}$ is:

$$\frac{dE_D}{d\epsilon} = \sum_{i,j} \sum_{t,\mu} \sum_{x\in\Omega_i} \frac{\partial M}{\partial 2}(t,\mu)\, DI_{j,\mu}\, D\Pi_j \boldsymbol{\Delta X}(\boldsymbol{x},\boldsymbol{\mu}) \tag{6}$$

Point remaining fixed in first image we only consider $\frac{\partial M}{\partial 2}(t,\mu)$ the similarity partial derivative according to the second term. This similarity partial derivative calculation will be described more deeply in the next section. By rearranging

correctly those different terms and sums, and using the domain of non semi-occluded points in image $I_{i,t}$ whose back-projection lies in f that we write in a naive manner,

$$\Omega_{f,t} = \Omega_i \cap \Pi_i(f_t)$$

one can easily check that we get:

$$\frac{dE_D}{d\epsilon} = \sum_t \sum_v \langle \nabla E_D(v,t), \boldsymbol{\delta x_{v,t}} \rangle$$

With the three following gradients:

$$\boldsymbol{\nabla E_D(v,t)} = \sum_{i,j} \sum_u \sum_{f \ni v} \boldsymbol{\delta E_1} + \boldsymbol{\delta E_2} + \boldsymbol{\delta E_3}$$

and,

$$\begin{cases} \boldsymbol{\delta E_1} = \sum_{x \in \Omega_{f,t}} \frac{\phi_{v,\boldsymbol{X}_t}}{\boldsymbol{N} \cdot \boldsymbol{d}_i} \left[\frac{\partial M}{\partial 2}(t,u)\, DI_{j,u}\, D\Pi_j\, D_f(t,u)\, \boldsymbol{d}_i \right] \boldsymbol{N} \\ \boldsymbol{\delta E_2} = -\sum_{x \in \Omega_{f,t}} \phi_{v,\boldsymbol{X}_t} \left[\frac{\partial M}{\partial 2}(t,u)\, DI_{j,u}\, D\Pi_j\, D_f(t,u) \right]^T \\ \boldsymbol{\delta E_3} = \sum_{x \in \Omega_{f,u}} \phi_{v,\boldsymbol{X}_u} \left[\frac{\partial M}{\partial 2}(u,t)\, DI_{j,t}\, D\Pi_j \right]^T \end{cases}$$

2.6 Similarity measure

We have chosen here to use the Normalized Cross Correlation which is still one of the most popular stereovision matching measure.

Concerning the similarity gradient equation (6) in previous section, the detailed equation is a bit more involved since the correlation variation in the window Ω_{x_k} centered in x_k will depend on x_k but also on all other pixels inside the correlation window.:

$$\frac{dE_D}{d\epsilon} = \sum_{i,j} \sum_{t,\mu} \sum_{x \in \Omega_i} \sum_{x_k \in \Omega_x} \frac{\partial M_{\Omega_{x_k}}}{\partial 2}(x,t,\mu)\, DI_{j,\mu}\, D\Pi_j \boldsymbol{\Delta X(x,\mu)}$$

Where $\frac{\partial M_{\Omega_{x_k}}}{\partial 2}(x,t,\mu)$ corresponds to the gradient at point x of the correllation in the window centered in x_k this can be formulated in a simpler manner by integrating:

$$\frac{dE_D}{d\epsilon} = \sum_{i,j} \sum_{t,\mu} \sum_{x \in \Omega} \underbrace{\sum_{x_k \in \Omega_x} \frac{\partial M_{\Omega_{x_k}}}{\partial 2}(x,t,\mu)}_{\frac{\partial M}{\partial 2}(\mu,t)}\ DI_{j,\mu}\, D\Pi_j \boldsymbol{\Delta X(x,\mu)}$$

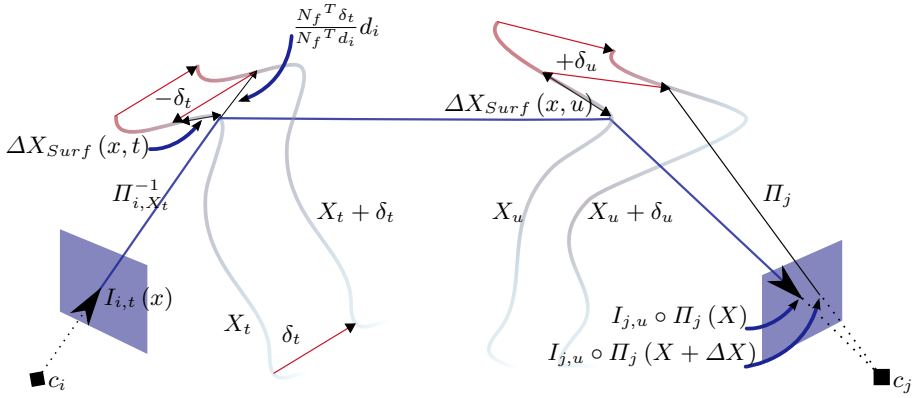

Fig. 3. Intensity variation in image j for a space-time surface perturbation δ

2.7 Intuitive gradient flow

If we add a spatio-temporal perturbation δX, if we denote by $\boldsymbol{d_i}$ the optical ray from camera center to back-projected point, and $\boldsymbol{N}$ the normal to the surface, the new back-projected point will be:

$$\Pi^{-1}_{i,(\boldsymbol{X}_t+\epsilon\delta X_t)^t}(x) = \Pi^{-1}_{i,\boldsymbol{X}_t}(x) + \frac{\boldsymbol{N}\cdot\boldsymbol{\delta}}{\boldsymbol{N}\cdot\boldsymbol{d_i}}\boldsymbol{d_i}$$

Previous calculation has been introduced in [9]. Notice that in this formula point on image i remains fixed, since we move along optical ray. Which is necessary in our case, as we integrate over the image pixel grid. Now we want to know the position of back-projection after perturbation not only at time t (as done above) but also at other time u. At this time we get a position but this is not a particle that could be tracked, just coordinates. So we want to identify from which particle, on mesh before perturbation, this position comes from. Then we aim to follow this particle over time.

We can see on Figure 3 the global particle tracking process. Our back-projection of x at time t on a weakly perturbated surface, comes from particle X that had at time t and before perturbation, following coordinates:

$$X(t) = \Pi^{-1}_{i,\boldsymbol{X}_t}(x) + \frac{\boldsymbol{N}\cdot\boldsymbol{\delta}}{\boldsymbol{N}\cdot\boldsymbol{d_i}}\boldsymbol{d_i} - \boldsymbol{\delta}$$

We aim to write equations in the mesh so as functions of vertices, the back-projected point writes:

$$\Pi^{-1}_{i,\boldsymbol{X}_t}(x) = \sum_k \boldsymbol{x}_{k,t}\,\phi_{k,\boldsymbol{X}_t}$$

So perturbation at back-projected point at time t, lying on facet f is:

$$\boldsymbol{\delta} = \sum_{v\in f} \boldsymbol{\delta x_{v,t}}\,\phi_{v,\boldsymbol{X}_t}$$

The particle X writes in the discrete mesh:

$$X(t) = \Pi^{-1}_{i,\boldsymbol{X}_t}(x) + \sum_{v \in f} \phi_{v,\boldsymbol{X}_t} \left(\frac{\boldsymbol{N} \cdot \boldsymbol{\delta x_{v,t}}}{\boldsymbol{N} \cdot \boldsymbol{d_i}} \boldsymbol{d_i} - \boldsymbol{\delta x_{v,t}} \right)$$

So if we consider the variation of position along surface between the point X_1 back-projected on surface before perturbation and the particle X at time t we get:

$$\begin{aligned} \boldsymbol{\Delta X_{Surf}}(x,t) &= X(t) - \Pi^{-1}_{i,\boldsymbol{X}_t}(x) \\ &= \textstyle\sum_{v \in f} \left(\phi_{v,\boldsymbol{X}_t} \frac{\boldsymbol{N} \cdot \boldsymbol{\delta x_{v,t}}}{\boldsymbol{N} \cdot \boldsymbol{d_i}} \boldsymbol{d_i} - \phi_{v,\boldsymbol{X}_t} \boldsymbol{\delta x_{v,t}} \right) \end{aligned}$$

We denote by $D_f(t,u)$ the intrinsic deformation matrix of facet f from time t to time u. This deformation matrix maps a vector at time t to the vector at time u accordingly with the facet deformation. So, The new variation of position at time u is:

$$\boldsymbol{\Delta X_{Surf}}(x,u) = \textstyle\sum_{v \in f} \phi_{v,\boldsymbol{X}_t} D_f(t,u) \left(\frac{\boldsymbol{N} \cdot \boldsymbol{\delta x_{v,t}}}{\boldsymbol{N} \cdot \boldsymbol{d_i}} \boldsymbol{d_i} - \boldsymbol{\delta x_{v,t}} \right)$$

Now one can easily compute the variation of position between the coordinate of back-projected point at time u and the position of tracked particle X once perturbated with δX_u:

$$\begin{aligned} \boldsymbol{\Delta X}(x,u) &= \boldsymbol{\Delta X_{Surf}}(x,u) + \textstyle\sum_{v \in f} \phi_{v,\boldsymbol{X}_t} \boldsymbol{\delta x_{v,u}} \\ \boldsymbol{\Delta X}(x,u) &= \textstyle\sum_{v \in f} \phi_{v,\boldsymbol{X}_t} \left[D_f(t,u) \left(\frac{\boldsymbol{N} \cdot \boldsymbol{\delta x_{v,t}}}{\boldsymbol{N} \cdot \boldsymbol{d_i}} \boldsymbol{d_i} - \boldsymbol{\delta x_{v,t}} \right) + \boldsymbol{\delta x_{v,u}} \right] \end{aligned}$$

As one can see, this intuitive reasonning leads to the same result than in section 2.5, but before we have provided the exact values of the deformation matrix $D_f(t,u)$.

2.8 Velocity field regularization term

The velocity field is unambiguously encoded by the animated mesh $\boldsymbol{X}$. Specifically, it is a continuous and piecewise linear vector field $\boldsymbol{X}_t \to \mathbb{R}^3$ defined by

$$\boldsymbol{v}_{\boldsymbol{X},t}(\boldsymbol{x}) = T_{\boldsymbol{X}_t \rightharpoonup \boldsymbol{X}_{t+1}}(\boldsymbol{x}) - \boldsymbol{x} \ , \tag{7}$$

or equivalently by

$$\boldsymbol{v}_{\boldsymbol{X},t} = \sum_k (\boldsymbol{x}_{k,t+1} - \boldsymbol{x}_{k,t}) \, \phi_{k,\boldsymbol{X}_t} \ . \tag{8}$$

The velocity field regularization term writes:

$$E_V(\boldsymbol{X}) = \sum_t \int_{\boldsymbol{X}_t} \|\nabla \boldsymbol{v}_{\boldsymbol{X},t}(\boldsymbol{x})\|^2 d\boldsymbol{x} \ . \tag{9}$$

To simplify this expression, we use the fact that $\nabla\phi_{k,\boldsymbol{X}_t}$ is constant in each triangular facet f of $\boldsymbol{X}_t$ and equals $\frac{\boldsymbol{h}_{k,f}}{\|\boldsymbol{h}_{k,f}\|^2}$, where $\boldsymbol{h}_{k,f}$ is triangle's height going through vertex k. A_f being the area of f, the energy term becomes:

$$E_V(\boldsymbol{X}) = \sum_t \sum_{f \in \boldsymbol{X}_t} A_f \left\| \sum_{k \in f} \frac{\boldsymbol{h}_{k,f}}{\|\boldsymbol{h}_{k,f}\|^2} (\boldsymbol{x}_{k,t+1} - \boldsymbol{x}_{k,t}) \right\|^2 . \tag{10}$$

If we neglect the variation of $\boldsymbol{h}_{k,f}$ with respect to vertex displacement, the partial derivatives $\frac{\partial E_V}{\partial \boldsymbol{x}_{k,t}}$ of this energy term can now easily be derived.

3 Numerical Experiments

3.1 Implementation aspects

The computation of image reprojections via the animated mesh and of the data attachment energy's gradient are the most expensive parts of our algorithm. Hence, they are implemented on GPU using the OpenGL API and the Cg shading language.

In all our experiments, we choose the opposite of normalized cross correlation as the image dissimilarity measure M, in order to accommodate moving shadows and time-varying lighting conditions. We use a small correlation window of 9×9 pixels. The storage of the animated mesh and the computation of spatio-temporal smoothing terms are based on the C++ Computational Geometry Algorithms Library (CGAL)[3].

Also, in order to limit the computational expense, the summation in Equation 4 is restricted to pairs of neighboring cameras pairs of neighboring time frames in a sliding time window, without any noticeable degradation of the results.

The resolution of the mesh is controlled by a lower and an upper edge length thresholds, that are applied to the whole time sequence: an edge is bisected if it is longer than the upper threshold in *at least one* time frame; an edge is collapsed if it is shorter than the lower threshold in *all* time frames.

The topology of the mesh is automatically corrected when needed by applying Delaunay deformable models [28] to the coordinates of the animated mesh at a reference time frame. The user chooses a reference frame that reflects the actual topology of the scene: e.g a pose with arms and legs slightly apart for human motion.

3.2 Experimental results

The "Pants" dataset is composed of 8 cameras. It is courtesy of R. White, K. Crane and D.A. Forsyth [30]. We have successfully applied our algorithm to the first 60 frames of this dataset. Due to the high image resolution, four multi-resolution scales have been necessary to obtain the accurate spatio-temporal reconstruction shown in Figure 4.

[3] `http://www.cgal.org/`

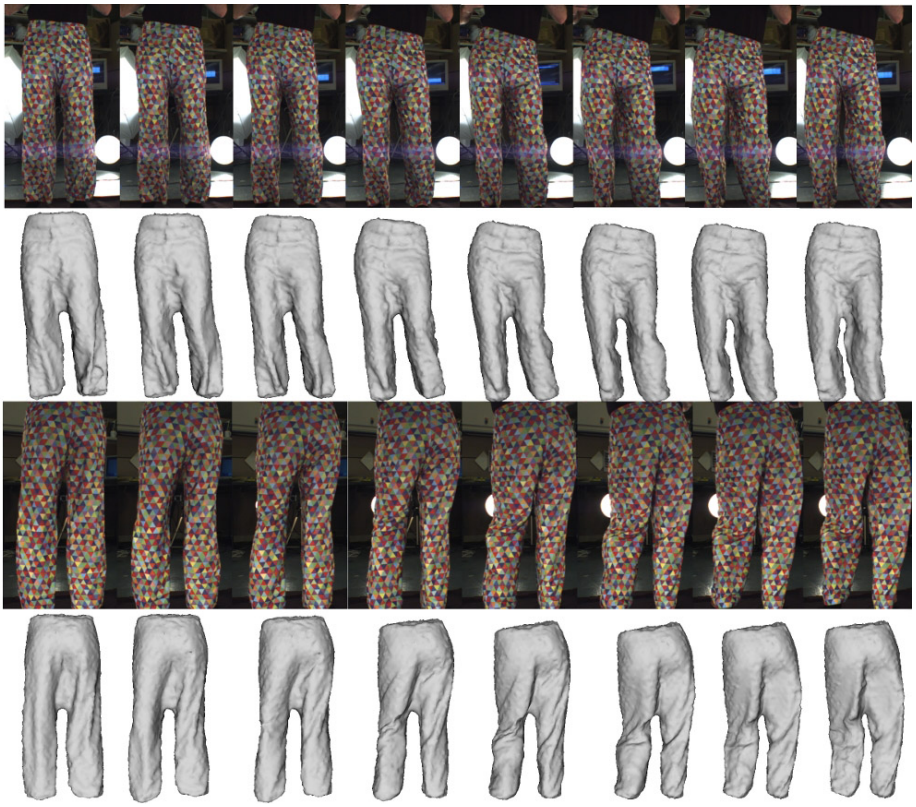

Fig. 4. Our results on the "Pants" dataset. See text for more details.

Figure 5 demonstrates the superiority of our spatio-temporal approach compared to a frame-by-frame multi-view stereovision method [9], on the "Pants" dataset. The improvements are three-fold: (i) our approach exploits speed and acceleration to make better initial guesses of the subsequent time frames, thus being less prone to unwanted local minima (ii) thanks to the enforcement of temporal coherence, our approach is less likely to fail in regions with low photo-consistency evidence (iii) our approach simultaneously and consistently estimates 3D shape and 3D scene flow.

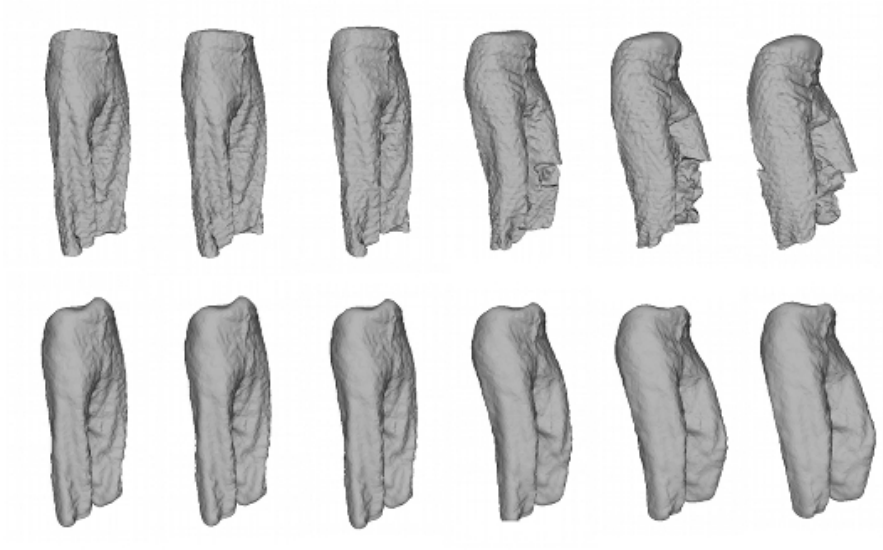

Fig. 5. Comparison between a frame-by-frame multi-view stereovision approach (top) and our spatio-temporal approach (bottom) on the "Pants" dataset. See text for details.

The "Dancer" dataset was made available to us by the 4Dviews company[4]. It was acquired by 14 calibrated and synchronized video cameras. We have applied our algorithm to the first 10 frames of this dataset. To bootstrap our multi-resolution and chronological optimization procedure, we have used a standard stereo-vision algorithm at the first time frame. The obtained reconstruction after processing three multi-resolution levels is displayed in Figure 6. We insist on the fact that we have not used silhouette information in our algorithm and that stereovision on such a dataset is particularly challenging: because it was design for visual hull based techniques, many parts of the subject are textureless.

[4] http://4dviews.com

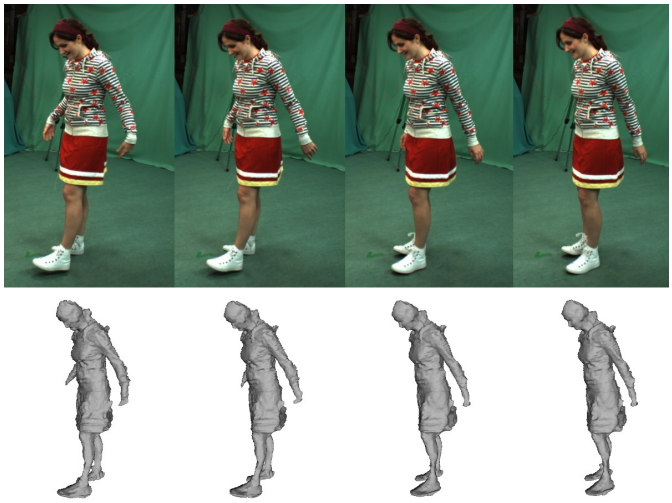

Fig. 6. Our results on the "Dancer" dataset. See text for more details.

4 Conclusion

We have presented a novel variational approach to dense and accurate 3D shape and motion reconstruction from multi-view video sequences. Our method leverages the benefits of the animated mesh representation, of image-based photo-consistency, of discrete geometric optimization and of GPU computation. We have validated our algorithm on two challenging real datasets, and obtained results that rival state-of-the-art techniques.

References

1. Aganj, E., Pons, J.P., Ségonne, F., Keriven, R.: Spatio-temporal shape from silhouette using four-dimensional Delaunay meshing. In: IEEE International Conference on Computer Vision. (2007)
2. Ahmed, N., de Aguiar, E., Theobalt, C., Magnor, M., Seidel, H.P.: Automatic generation of personalized human avatars from multi-view video. In: Proc. VRST'05. (2005) 257–260
3. Ahmed, N., Theobalt, C., Rössl, C., Thrun, S., Seidel, H.P.: Dense correspondence finding for parametrization-free animation reconstruction from video. In: IEEE Conference on Computer Vision and Pattern Recognition. (2008)
4. de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.P., Thrun, S.: Performance capture from sparse multi-view video. In: ACM SIGGRAPH. (2008)
5. de Aguiar, E., Theobalt, C., Stoll, C., Seidel, H.P.: Marker-less deformable mesh tracking for human shape and motion capture. In: IEEE Conference on Computer Vision and Pattern Recognition. (2007)
6. Furukawa, Y., Ponce, J.: Dense 3D motion capture from synchronized video streams. In: IEEE Conference on Computer Vision and Pattern Recognition. (2008)

7. Goldlücke, B., Magnor, M.: Space-time isosurface evolution for temporally coherent 3D reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume 1. (2004) 350–355
8. Neumann, J., Aloimonos, Y.: Spatio-temporal stereo using multi-resolution subdivision surfaces. The International Journal of Computer Vision **47**(1–3) (2002) 181–193
9. Pons, J.P., Keriven, R., Faugeras, O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. The International Journal of Computer Vision **72**(2) (2007) 179–193
10. Starck, J., Hilton, A.: Surface capture for performance based animation. IEEE Computer Graphics and Applications **27**(3) (2007) 21–31
11. Theobalt, C., Ahmed, N., Lensch, H., Magnor, M., Seidel, H.P.: Seeing people in different light-joint shape, motion, and reflectance capture. IEEE Transactions on Visualization and Computer Graphics **13**(4) (2007) 663–674
12. Varanasi, K., Zaharescu, A., Boyer, E., Horaud, R.P.: Temporal surface tracking using mesh evolution. In: European Conference on Computer Vision. Volume 2. (2008) 30–43
13. Vedula, S., Baker, S., Kanade, T.: Image-based spatio-temporal modeling and view interpolation of dynamic events. ACM Transactions on Graphics **24**(2) (2005) 240–261
14. Vedula, S., Baker, S., Seitz, S., Kanade, T.: Shape and motion carving in 6D. IEEE Conference on Computer Vision and Pattern Recognition (2000)
15. Vlasic, D., Baran, I., Matusik, W., Popović, J.: Articulated mesh animation from multi-view silhouettes. ACM Transactions on Graphics **27**(3) (2008)
16. Vedula, S., Baker, S.: Three-dimensional scene flow. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(3) (2005) 475–480
17. Zhang, Y., Kambhamettu, C.: Integrated 3D scene flow and structure recovery from multiview image sequences. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume 2. (2000) 674–681
18. Huguet, F., Devernay, F.: A variational method for scene flow estimation from stereo sequences. In: IEEE International Conference on Computer Vision. (2007)
19. Pons, J.P., Keriven, R., Faugeras, O., Hermosillo, G.: Variational stereovision and 3D scene flow estimation with statistical similarity measures. In: IEEE International conference on Computer Vision. Volume 2. 597–602
20. Wedel, A., Rabe, C., Vaudrey, T., Brox, T., Franke, U., Cremers, D.: Efficient dense scene flow from sparse or dense stereo data. In: European Conference on Computer Vision. (2008) 739–751
21. Zhang, Y., Kambhamettu, C.: On 3D scene flow and structure estimation. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume 2. (2001) 778–785
22. Süßmuth, J., Winter, M., Greiner, G.: Reconstructing animated meshes from time-varying point clouds. Computer Graphics Forum (Proc. Eurographics Symposium on Geometry Processing) **27**(5) (2008) 1469–1476
23. Wand, M., Jenke, P., Huang, Q., Bokeloh, M., Guibas, L., Schilling, A.: Reconstruction of deforming geometry from time-varying point clouds. In: Eurographics Symposium on Geometry Processing. (2007) 49–58
24. Delaunoy, A., Prados, E., Gargallo, P., Pons, J.P., Sturm, P.: Minimizing the multi-view stereo reprojection error for triangular surface meshes. In: British Machine Vision Conference. (2008)

25. Slabaugh, G.G., Unal, G.B.: Active polyhedron: Surface evolution theory applied to deformable meshes. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume 2. (2005) 84–91
26. Vu, H.H., Keriven, R., Labatut, P., Pons, J.P.: Towards high-resolution large-scale multi-view stereo. In: IEEE Conference on Computer Vision and Pattern Recognition. (2009)
27. Pons, J.P., Boissonnat, J.D.: Delaunay deformable models: Topology-adaptive meshes based on the restricted Delaunay triangulation. In: IEEE Conference on Computer Vision and Pattern Recognition. (2007)
28. Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.P.: Interactive multi-resolution modeling on arbitrary meshes. In: International Conference on Computer Graphics and Interactive Techniques. (1998) 105–114
29. White, R., Crane, K., Forsyth, D.: Capturing and animating occluded cloth. In: SIGGRAPH. (2007)