Globally Optimal Spatio-temporal Reconstruction from Cluttered Videos

Ehsan Aganj, Jean-Philippe Pons, Renaud Keriven

Globally optimal spatio-temporal reconstruction from cluttered videos

Ehsan Aganj[1], Jean-Philippe Pons[2], and Renaud Keriven[1]

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

Abstract. We propose a method for multi-view reconstruction from videos adapted to dynamic cluttered scenes under uncontrolled imaging conditions. Taking visibility into account and being based on a global optimization of a true spatio-temporal energy, it offers several desirable properties: no need for silhouettes, robustness to noise, independent from any initialization, no heuristic force, reduced flickering results, etc. Results on real-world data proves the potential of what is, to our knowledge, the only globally optimal spatio-temporal multi-view reconstruction method.

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. Many of these techniques rely on the visual hull concept[16], among which [1, 3, 15]. Computationally efficient, they suffer from several limitations: they provide an approximate reconstruction; this one has to be a closed surface; and, above all, silhouettes have to be segmented in the videos, practically limiting the method to controlled condition capture with a known background. This latest limitation may be lifted when prior knowledge about the geometry is available: free-form deformation of a template body model [2, 11, 15], Laplacian deformation of a laser scan of the initial pose [4, 5], etc. Yet, these methods are unable to recover genuine geometric details such as facial expressions and clothing folds and wrinkles. An exception might be the method proposed by Furukawa et al. [6]. Yielding visually impressive results, this method does not rely on global optimization and handles the occlusion problem via heuristics.

1.1 Our approach

In this paper, we propose a method for multi-view reconstruction from videos adapted to dynamic cluttered scenes under uncontrolled imaging conditions. Taking visibility into account and being based on a global optimization of a true spatio-temporal energy, it offers several desirable properties.

Starting from work by Labatut et al. [17], our method might be seen as its spatio-temporal extension. It is based on modeling an evolving three-dimensional surface as a four-dimensional surface [1, 18, 7]. More precisely, we first generate a quasi-dense 3D point cloud of the scene at each time step and merge the result into a 4D point cloud. This process is conducted in a lenient manner, thus possibly retaining many false matches. Then, we build an adaptive decomposition of the 3D+time space by computing the 4D Delaunay triangulation of this cloud. Finally, we label the Delaunay pentatopes as empty or occupied thus generating a 4D surface. Graph-cut based, this assignment is globally optimal and compatible with the visibility in input images. Optionally but not necessarily, the 3D surfaces corresponding to each time step might be obtained considering 3D slices.

1.2 Contributions

Our method has several significant advantages. First, it is not based on visual hulls:

- The videos do not have to be taken under controlled conditions. The background might be cluttered.
- It can handle closed as well as open scenes. For example, it can simultaneously recover the walls and (potentially moving!) furnitures of an indoor scene and a complete reconstruction of subjects seen from all sides in the input images.

Second, it is based on a global optimum:

- It is robust and does not depend on some initialization.
- It exploits visibility information to guide the position of the surface. As a result, it avoids the minimum cut solution from being an empty surface. Hence it exonerates from the usual techniques proposed so far to solve this problem (ballooning term, silhouette information, photo-flux, etc.). Moreover, this visibility information is not enforced as a hard constraint but integrated in the very optimization framework, hence yielding robustness to outliers.

Finally, and mainly, compared to the independent frame-by-frame computations of [17], it profits from the 4D representation:

- Regularization is handled both in space and time, yielding more robustness to noise both in geometry and in visibility reasoning.
- Points extracted at one given time step transfer information to the surrounding time steps. As a result, more details are obtained at each time step.
- Flicking artifacts in synthesized views are reduced, as consecutive 3D slices have similar geometry and connectivity by construction.
- The temporally continuous representation, which is defined at any time, optionally enables interpolation of objects shape between consecutive frames.

The output of our method might be use for different purposes: as a 4D compact representation; as a list of consecutive 3D meshes; as an initialization for variational spatio-temporal stereovision methods [7].

The remainder of this paper is organized as follows. Section 2 gives some background on the different techniques needed in our approach. In Section 3, we describe in detail the different steps of our algorithm. Section 4 discusses numerical experiments that demonstrate the potential of our approach for reconstructing cluttered scenes from real-world data.

2 Background

2.1 Delaunay triangulation

Most of the following definitions are taken from [19]. We also refer the interested reader to some computational geometry textbooks [20, 21]. In the sequel, we call *k-simplex* the convex hull of $k+1$ affinely independent points. For example, a 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle and a 3-simplex is a tetrahedron. In this paper, we will also consider 4-simplices: they are known as *pentachorons* or *pentatopes*. Let $E = \{p_1, \ldots, p_n\}$ be set of points in $\mathbb{R}^d$. The *Voronoi region*, or *Voronoi cell*, denoted by $V(p_i)$, associated to a point p_i is the region of space that is closer from p_i than from all other points in E: $V(p_i) = \{p \in \mathbb{R}^d \ : \ \forall j, \ \|p - p_i\| \leq \|p - p_j\|\}$.

The *Voronoi diagram* of E, denoted by $\mathrm{Vor}(E)$, is the partition of space induced by the Voronoi cells $V(p_i)$. See Figure 1*(a)* for a two-dimensional example of a Voronoi diagram.

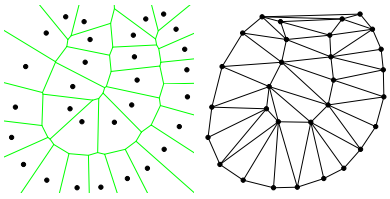

Fig. 1. *(a)* Voronoi diagram of a set of points in the plane. *(b)* Its dual Delaunay triangulation.

The *Delaunay triangulation* $\mathrm{Del}(E)$ of E is defined as the geometric dual of the Voronoi diagram: there is an edge between two points p_i and p_j in the Delaunay triangulation if and only if their Voronoi cells $V(p_i)$ and $V(p_j)$ have a non-empty intersection. It yields a *triangulation* of E, that is to say a partition of the convex hull of E into d-dimensional simplices (i.e. into triangles in 2D, into tetrahedra in 3D, into pentatopes in 4D and so on). See Figure 1*(b)* for a two-dimensional example of a Delaunay triangulation. The fundamental property of the Delaunay triangulation is called the *empty circle* (resp. *empty sphere* in 3D, resp. *empty hypersphere* in higher dimensions) property: in 2D (resp. in 3D, resp. in 4D), a triangle (resp. tetrahedron, resp. pentatope) belongs to the Delaunay triangulation if and only if its circumcircle (resp. circumsphere, resp. circumscribed hypersphere) does not contain any other points of E in its interior.

2.2 Energy minimization by graph cuts

Given a finite directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ with non-negative weights (capacities), and two special vertices, the source s and the sink t, an s-t-cut $\mathcal{C} = (\mathcal{S}, \mathcal{T})$ is a partition of $\mathcal{V}$ into two disjoint sets $\mathcal{S}$ and $\mathcal{T}$ such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$. The cost of the cut is the sum of the capacity of all the edges going from $\mathcal{S}$ to $\mathcal{T}$: $c(\mathcal{S}, \mathcal{T}) = \sum_{(p,q)\in\mathcal{S}\times\mathcal{T}|p\rightarrow q\in\mathcal{E}} w_{pq}$. The minimum s-t-cut problem consists in finding a cut $\mathcal{C}$ with the smallest cost: the Ford-Fulkerson theorem [22] states that this problem is equivalent to computing the maximum flow from the source s to the sink t and many classical algorithms exist to efficiently solve this problem. Such a cut can be viewed as a binary labeling of the nodes: by building an appropriate graph, many segmentation problems in computer vision can be solved very efficiently [23]. More generally, global minimization of a whole class of energy is achievable by graph cuts [24].

Kirsanov and Gortler [25] first proposed to use graph cuts on complexes to globally optimize surface functionals and also developed the idea of using random sparse complexes for their flexibility over regular subdivisions: this differs from the graphs commonly used in computer vision, which are often regular grids in the input images or in the bounding volume of the scene. Our approach similarly relies on a sparse complex-based graph: this graph however directly derives from an adaptive space decomposition efficiently provided by the Delaunay triangulation. Moreover the specifies of our graph construction are quite different and tailored to the multi-view reconstruction problem.

3 Method

Our spatio-temporal reconstruction algorithm consists in four steps: (i) a quasi-dense 3D point cloud is generated for each frame, each point memorizing the two or more images from which it has been triangulated. An spatio-temporal 4D point cloud is obtained by adding time as the fourth dimension to all the 3D points; (ii) the Delaunay triangulation of the 4D point cloud is computed; (iii) the Delaunay pentatopes (full-dimensional simplices in $\mathbb{R}^4$) are labeled inside or outside the spatio-temporal object minimizing some energy, a 4D oriented surface is then extracted as the set of 4D tetrahedra lying between inside and outside pentatopes; (iv) the 3D surface at a given time is obtained by intersecting this 4D hyper-surface with a temporal plane.

3.1 4D point cloud generation, 4D Delaunay triangulation

Given multiple video sequences of a dynamic scene, we first make a dense 3D point cloud for each time instant. Let I_k^t, $k \in \{1, \cdots, n\}$, $t \in [0, T]$ denote the input video sequence. For each image we extract interest points $x_{k,i}^t$ of several types (in practice Harris points and DOGs) without any scale information and with thresholds such that their number is high enough. For each image pair in a time instant t, $(I_k^t, I_{k'}^t)$ whose visual fields intersect, for all $(x_{k,i}^t, x_{k',j}^t)$ of the same type verifying the epipolar constraint up to a certain error (due to point

extraction but also to calibration), we triangulate the corresponding 3D point $X^t_{kk',ij}$. Let H be the homography from I^t_k to $I^t_{k'}$ induced by the plane at $X^t_{kk',ij}$ normal to the optical ray of $x^t_{k,i}$, we evalute the normalized cross correlation (NCC) between a given neighborhood of $x^t_{k,i}$ and its image by H around $x^t_{k',j}$. Then for each $x^t_{k,i}$, we keep the m best point $x^t_{k',j}$ according to NCC, only if their NCC is higher than a given threshold, and add the corresponding $X^t_{kk',ij}$ to the point cloud at time instant t (in practice $m = 1$ or 2). The final 3D point cloud is then obtained by merging close 3D points, so that a point of the cloud comes from possibly more than two images.

Now we construct a "global" spatio-temporal point cloud by regarding time as a fourth dimension, and treating it similarly to the three spatial dimensions. At first sight, this is questionable since space is not homogenous to time regarding physical units. We obtain physical homogeneity of our 4D space by considering a scaling factor v between space and time dimensions. This scaling factor is homogeneous to a speed, and can be interpreted as a reference displacement per time unit. The global point cloud is obtained by taking a 4D point $(X^t_i, vt) \in \mathbb{R}^4$ for the point X^t_i generated from the input images in time t. At the end, we compute the 4D Delaunay triangulation of the spatio-temporal cloud storing in each vertex the list of the views and keypoints from which it has been generated.

3.2 4D hyper-surface extraction

In this step we compute a four-dimensional representation of the dynamic scene by extracting a 4D mesh from the Delaunay triangulation of the point cloud. This is done by labeling Delaunay pentatopes as inside or outside of the spatio-temporal scene. The final oriented hypersurface is then extracted as the set of facets between inside and outside pentatopes. The exact nature of these "facets" deserves clarification: they are tetrahedra with 4D coordinates, so they are indeed *simplicial* pieces of a hypersurface in $\mathbb{R}^4$. Now we make a graph of neighbor pentatopes which we will use to find the optimal label assignment. For that, we take Delaunay pentatopes as vertices and we add edges between every two pentatopes which are neighbor via a two or three dimensional face. In addition, we add a link between each vertex of the graph and the sink and the source. A globally optimal label assignment is then efficiently found by applying the graph cuts optimization method on this graph.

In the sequel, we note S the surface to be reconstructed. As discussed above, S is a union of 4D Delaunay facets. In order to find an optimal solution satisfying both spatial and temporal constraints, we minimize an energy composed of two terms, one dealing with visibility, and the other dealing with spatial and temporal smoothness,

$$\mathrm{E}(S) = \mathrm{E}_{\mathrm{vis}}(S) + \mathrm{E}_{\mathrm{smooth}}(S) \tag{1}$$

In the rest of this section, we give the exact definition of these two terms and we describe how they can be implemented in the graph cuts framework.

Visibility term The visibility term that we propose for a spatio-temporal scene is a careful extension of the static case proposed in [17]. The idea of their work is

that if a point belongs to the final surface then it should be visible in the views from which it has been triangulated. This yields to the penalization of all the facets intersecting the ray between the point and the cameras from which it has been generated. In the dynamic case the same argument holds. A point which belongs to the final hypersurface should be visible in all its generating views. Consequently, all 4D pentatopes which intersect a 4D ray emanating from the point to the camera center of one of its generating views should be labeled as outside, and the pentatope behind the point should be labeled as inside. We remark that the spatio-temporal center of a camera at a given frame is its 3D center positioned in the temporal plane of that frame. Similarly to the static case, in order to make an energy which can be minimized via graph cuts, we take the number of intersections of a ray with the oriented hypersurface as the visibility term. At this point, there are several important remarks to be made.

First, the visibility of a point at a given frame is defined only in the temporal plane corresponding to that frame. Therefore, the rays between the point and its generating views lie completely in the temporal plane which passes through that point.

Second, a 4D facet of the Delaunay triangulation passes generally through several consecutive frames. As a consequence, each intersection of a ray with a facet is considered as a penalizing "vote" for the facet. The final vote is then computed as the sum of all votes coming from different frames intersecting the facet. This is an important property since it makes a global visibility vote on every 4D facet taking in account the temporal coherence.

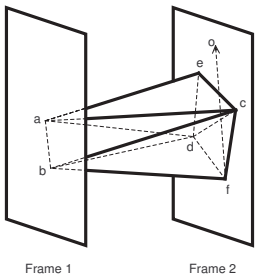

Fig. 2. The tetrahedra *acde* and *bcdf* have a one-dimensional intersection (the segment *cd*), but they should be connected in the graph (refer to text for details).

Third, contrarily to the static case presented in [17] where edges of the graph are only "full dimensional facets" (3D triangles) between Delaunay simplices, in the dynamic case, in addition to these facets we add an edge between every two pentatopes which are not neighbors via a full-dimensional facet (4D) but via a 3D facet. Figure 2 shows an example of this situation. For simplicity reasons we consider a lower dimensional scene: points are on 2D planes, time is the third dimension and the spatio-temporal object is extracted from the 3D Delaunay triangulation of the point cloud. Points a and b are on frame 1, and points c,d,e

and f are on frame 2. Point o is the center of a camera from which f has been generated. The tetrahedra $acde$ and $bcdf$ have a one-dimensional intersection (the segment cd), but they should be connected in the graph since the ray fo intersects cd and therefore a penalization term should be added between them. It is important to note that despite the 3D intersection of the ray with the face acd, no penalization term should be added between the tetrahedra $abcd$ and $bcdf$. That is because $abcd$ does not appear in the static representation of the scene on frame 2.

The intersections of the ray with the triangulation can be computed in the four-dimensional space handling carefully the situation discussed above. However, as a ray always lies completely in a temporal plane, we propose to find these intersections more easily by intersecting the 3D ray with the 3D intersection of the triangulation with the temporal plane. Obviously only the pentatopes which make a full-dimensional (4D) temporal intersection should appear in the temporal slice. In this case, a 3D facet intersected by a ray will correspond to an edge of the graph, and the unnecessary intersections discussed in the example above will be omitted by definition.

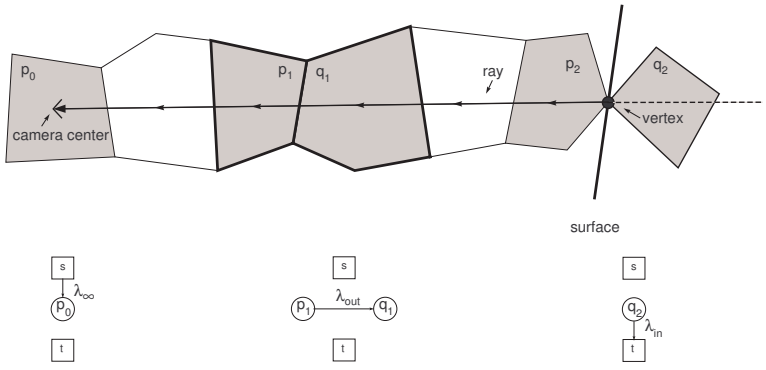

Fig. 3. Top: A 3D slice of the 4D triangulation. A ray emanating from a vertex to a camera center intersects 3D cells. Bottom: The corresponding visibility-related energy term that penalizes the number of intersections with the ray and the edge weights of the crossed pentatope in the graph.

We should remark that the 3D intersection of a 4D Delaunay triangulation with a plane is a polyhedron which contains generally cells with more than four vertices. Figure 3 shows the 3D object and a ray intersecting the cells. The vertices p_0, p_1, q_1, p_2 and q_2 shown in Figure 3(bottom) are the vertices of the graph which correspond to the pentatopes whose temporal intersections make the cells p_0, p_1, q_1, p_2 and q_2 shown in Figure 3(top) respectively. Different visibility terms are added. The cell containing the camera should be labeled as outside: a term λ_∞ is added to the edge from source to p_0. A 3D facet crossed by the ray from inside to outside should be penalized: a term λ_{out} is added to the edge from p_1 to q_1. The cell behind the origin of the ray should be labeled as

inside: a term λ_{in} is added to the edge from q_2 to the sink. The positive weights λ_{in}, λ_{out} and λ_{∞} take into account the confidence in the reconstructed vertex yielding the ray. By summing up these visibility terms over all the frames, we make a complex distribution of "vote" for each pentatope taking in account the time coherence.

Spatio-temporal smoothness In order to take in account both spatial smoothness and temporal continuity, we propose to minimize the area of the 4D surface in $\mathbb{R}^4$. This yields to the sum of volumes over all the 4D tetrahedra between inside and outside pentatopes,

$\text{E}_{\text{smooth}}(S) = A(S) = \int_S \text{d}S = \sum_{T \in S} A(T)$ where S is the 4D surface to be reconstructed, T is a 4D tetrahedra, and $A(T)$ is the volume of the tetrahedra T in $\mathbb{R}^4$. Minimizing this term encourages smoothness in time and in space. As in the static case, this is trivially minimized in the graph cuts framework: for each pair of pentatopes (sharing a tetrahedra T) represented by vertices p and q in our graph, a term $w = A(T)$ is added the edge $p \rightarrow q$ and to its opposite edge $q \rightarrow p$.

3.3 3D surface extraction

The output 4D mesh cannot be used directly for rendering. Fortunately, the 3D scene at a given time instant is easily obtained by intersecting this 4D mesh with a temporal plane. This task can be performed very efficiently, even in real-time on GPUs (Graphics Processor Units), since it reduces to a *marching tetrahedra* algorithm [26] on the tetrahedra of the 4D mesh, with the temporal coordinate of vertices used as the scalar field for isocontouring. It produces one triangle or one quad per boundary tetrahedron intersected by the selected temporal plane.

Fig. 4. Some images of the "Trousers" dataset.

4 Experimental results

We have implemented our method using *CGAL* (Computational Geometry Algorithms Library, homepage: www.cgal.org) [27]. CGAL defines most data structure and algorithms needed in our method. For the computation of 4D Delaunay triangulation we have used the Quickhull algorithm library (QHull) [28] (homepage: www.qhull.org).

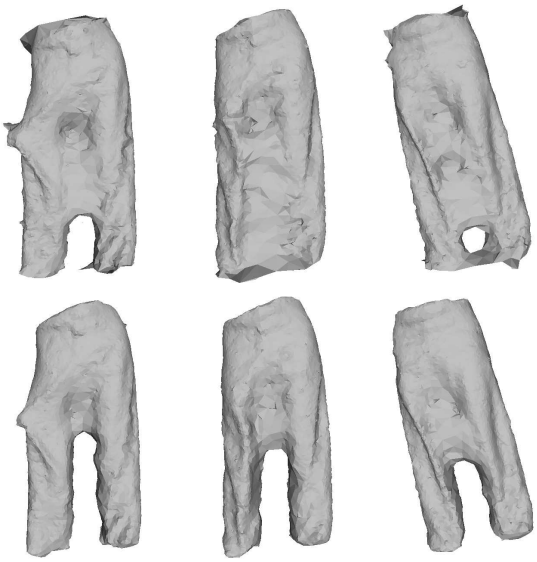

Fig. 5. A comparison between our method and the method of [17] applied independly in each frame. Top: 3D meshes obtained by the method of [17]. Bottom: corresponding 3D slices of the 4D representation obtained by our method.

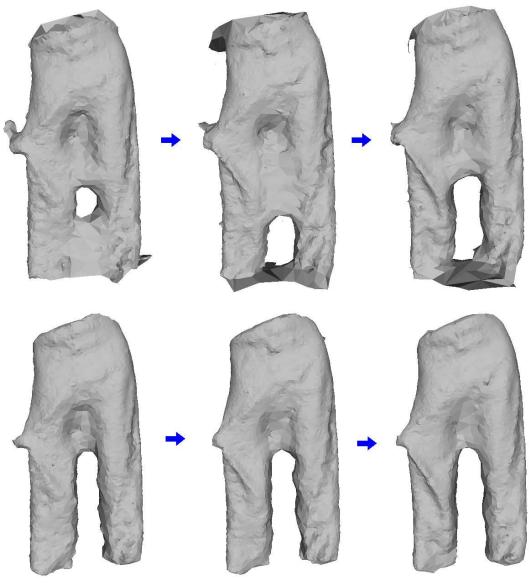

Fig. 6. The first three consecutive frames of the Trousers dataset reconstructed by (Top): the method of [17] (Bottom): our method. The frame-by-frame reconstruction of the method [17] makes no temporal continuity. In contrast, our method reconstructs correctly the trouser, avoids flicking artifacts and provides a much more continuous motion.

In our first experiment, we have tested our method on the first 60 frames of the "Trousers" sequence which is courtesy of R. White, K. Crane and D.A. Forsyth [29]. The sequence is acquired by 8 cameras at a 640×480 resolution. Figure 4 shows some images of this dataset. Despite their highly textured images, it is in fact a quite challenging dataset because of the very fast and complex motion of cloth and folds and severe self occlusion in various parts of the video. Regarding to an approximate size of the object we have chosen a spatio-temporal scaling factor $v = 30$ (space unit/frame). Figure 5 shows a comparison between our method and the method of [17] applied independly in each frame. We have compared the 3D meshes output by their method for some frames of the sequence with the corresponding 3D slices of our 4D spatio-temporal scene. A total number of 793769 points have been generated for the initial point cloud. To provide a fair comparison, we have used the same point clouds in both methods.

We observe that the method of [17] fails to separate correctly the two trouser legs when there is not enough distance between them. In addition, as shown in figure 6, their frame-by-frame reconstruction makes no temporal continuity. In contrast, relying on a global optimization, our method reconstructs correctly the trouser, avoids flicking artifacts and provides a much more continuous motion. This perfectly illustrates the capability of our approach to take advantage of temporal coherence in order to obtain more detailed and more continuous result. Please see supplemental material for a video illustrating better the result of our method. The computational times of our method and the method of [17] for this experiment are 210 and 112 minutes respectively on a standard workstation. However, the most expensive part of our method is the computation of the 4D Delaunay triangulation. Fortunately, this can be strongly reduced using an optimized 4D Delaunay code.

In a second experiment, we have tested our method on the first 40 frames of the "Dancer" dataset which was made available to us by the 4Dviews company (http://4dviews.com). It is acquired by 14 calibrated and synchronized video cameras. Figure 7(top) shows some images of this dataset. The result shows that despite the lowly textured parts of the images, our method makes a correct 4D representation of the dancer. Figure 7(bottom) shows some 3D slices extracted from the 4D object.

Finally, we should remark that in order to have better visualization and to make better comparisons we have smoothed the results of our experiments. However, the output of our method might be used as a 4D compact representation, as a list of consecutive 3D meshes or as an initialization for variational spatio-temporal stereovision methods.

5 Discussion and Conclusion

We have presented a new method for multi-view reconstruction from videos adapted to dynamic cluttered scenes under uncontrolled imaging conditions. The main idea of our method is to regard time as the fourth dimension, and to extract a hyper-surface from the 4D Delaunay triangulation of the input points as the

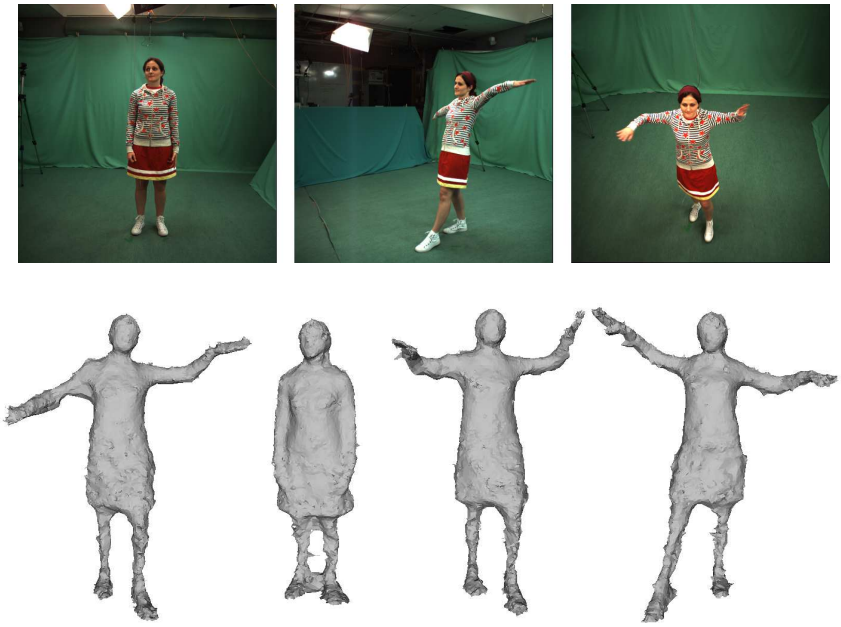

Fig. 7. Top: Some images of the "Dancer" dataset. Bottom: Some 3D slices extracted from the 4D representation of the "Danser" dataset, obtained by our method.

spatio-temporal representation of the scene. This is done by labeling Delaunay pentatopes as empty or occupied. A globally optimal assignment is efficiently found using graph cuts. We have validated our method on real video sequences. Our results prove the potential of what is, to our knowledge, the only globally optimal spatio-temporal multiview reconstruction method.

References

1. Aganj, E., Pons, J.P., Ségonne, F., Keriven, R.: Spatio-temporal shape from silhouette using four-dimensional Delaunay meshing. In: ICCV. (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., T.C.R.C.T.S., Seidel, H.: Dense correspondence finding for parametrization-free animation reconstruction from video. In: CVPR. (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: CVPR. (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: CVPR. Volume 1. (2004) 350–355

8. Neumann, J., Aloimonos, Y.: Spatio-temporal stereo using multi-resolution subdivision surfaces. IJCV **47**(1–3) (2002) 181–193
9. J.-P.Pons, R.K., Faugeras, O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. IJCV **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. TVCG **13**(4) (2007) 663–674
12. Varanasi, K., Zaharescu, A., Boyer, E., Horaud, R.P.: Temporal surface tracking using mesh evolution. In: ECCV. Volume 2. (2008) 30–43
13. Vedula, S., Baker, S., Kanade, T.: Image-based spatio-temporal modeling and view interpolation of dynamic events. ACM TOG **24**(2) (2005) 240–261
14. Vedula, S., Baker, S., Seitz, S., Kanade, T.: Shape and motion carving in 6D. CVPR (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. Laurentini, A.: The visual hull concept for silhouette-based image understanding. IEEE Trans. Pattern Anal. Mach. Intell. **16**(2) (1994) 150–162
17. Labatut, P., Pons, J.P., Keriven, R.: Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In: ICCV. (2007)
18. Cheung, G.K.M., Baker, S., Kanade, T.: Shape-from-silhouette across time part i: Theory and algorithms. IJCV **62**(3) (2004) 221–247
19. Boissonnat, J.D., Oudot, S.: Provably good sampling and meshing of surfaces. Graphical Models **67** (2005) 405–451
20. Boissonnat, J.D., Yvinec, M.: Algorithmic Geometry. Cambridge University Press (1998)
21. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry, Algorithms and Applications. SpringerVerlag (1997)
22. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press (1962)
23. Greig, D.M., Porteous, B.T., Seheult, A.H.: Exact maximum a posteriori estimation for binary images
24. Kolmogorov, V., Zabih, R.: What energy functions can be minimizedvia graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. **26**(2) (2004) 147–159
25. Kirsanov, D., Gortler, S.J.: A discrete global minimization algorithm for continuous variational problems. harvard computer science technical report: Tr-14-04. Technical report, Cambridge, MA (07/2004 2004)
26. Guéziec, A., Hummel, R.: Exploiting triangulated surface extraction using tetrahedral decomposition. TVCG **1**(4) (1995) 328–342
27. Boissonnat, J.D., Devillers, O., Teillaud, M., Yvinec, M.: Triangulations in CGAL. In: Annual Symposium on Computational Geometry. (2000) 11–18
28. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software **22** (1996) 469–483
29. White, R., Crane, K., Forsyth, D.: Capturing and animating occluded cloth. In: SIGGRAPH. (2007)