Toward a real-time tracking of dense point-sampled geometry

François Destelle, Céline Roudet, Marc Neveu, Albert Dipanda

TOWARD A REAL-TIME TRACKING OF DENSE POINT-SAMPLED GEOMETRY

F. Destelle, C. Roudet, M. Neveu, A. Dipanda

LE2I Laboratory UMR-CNRS 5158, University of Burgundy

ABSTRACT

In this paper, we address the problem of tracking temporal deformations between two arbitrary densely sampled point-based surfaces. We propose an intuitive and efficient resolution to the point matching problem within two frames of a sequence. The proposed method utilizes two distinct space partition trees, one for each point cloud, which both are defined on a unique discrete space. Our method takes advantage of multi-resolution concerns, voxel adjacency relations, and a specific distance function.

Experimental results obtained from both simulated and real reconstructed data sets demonstrate that the proposed method can handle efficiently the tracking process even for very large point clouds. Moreover, our method is easy to implement and very fast, which provides possibilities for real-time tracking applications.

Index Terms— Computer Vision, 3D Processing

1. INTRODUCTION

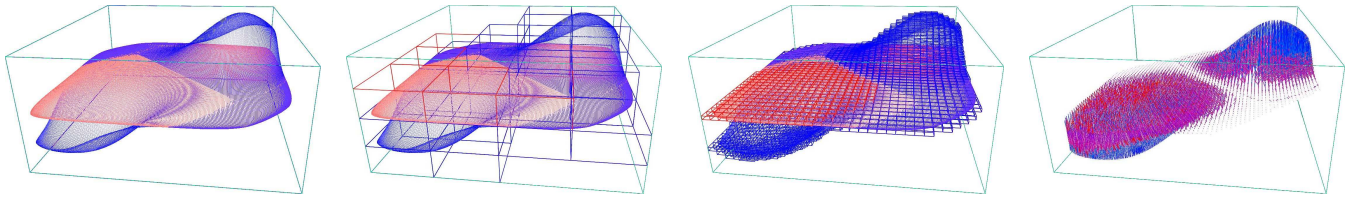
Recovering the temporal evolution of arbitrary deformable objects offers a large variety of applications in computer vision, ranging from the motion capture of human or animal bodies to the analysis of complex dynamic systems. Motion tracking yields temporal correspondences between captures of the moving objects at several instants, because it provides correspondences between the scene captures at several instants. Nowadays, acquisition systems usually produce dense point-sampled representations of an observed scene at high rates.

Existing markerless motion capture methods differ mostly in the prior knowledge of the observed shapes and in the nature of the input data sets. Numerous works focus on motion analysis of specific subjects, namely human bodies [1] or faces [2]. These approaches benefit from a strong prior knowledge by using a reference model, e.g. an articulated skeleton.

In contrast, non knowledge-based methods try to build a displacement field between different frames without any assumption on the observed 3D scene. Among these, point-based approaches consider a sequence of point clouds as input, usually without any other additional information such as photometric measures or shape reference. Recently, many works have focused on a large temporal sequence of captures [3, 4], involving densely sampled time-varying data sets. Applications of these methods can be ranged from data reconstruction to consistent temporal shape tracking. Although especially robust, these techniques are not designed to interactively handle the tracking of large point clouds.

Another kind of method aims to establish a correspondence between only two subsequent frames of a sequence. Within this context, Vedula et al. initially introduced the 3D scene flow [5] as a generalization of the classical two-dimensional optical flow and proposed to apply this method on 3D colored voxels [6]. The motion of each point in a cloud is described with local differential methods, thus limiting the analysis of small displacements between successive frames. De Aguiar et al. [7] proposed a volume decomposition technique based on ellipsoidal shells, each one containing a set of voxels. The tracking method is then assured by a matching of these ellipsoids between two frames. Cmolik et al. [8] presented several morphing techniques applied to point-sampled surfaces. The authors introduced a system based on a spatial clustering of the scene relative to principal component analysis of points contained in each voxel. However, some problematic cases cannot be handled, depending of the object geometry and the space clustering.

In this paper, we present a general point cloud tracking method without any prior knowledge on the observed 3D scene. We propose a voxel-based method suited to the efficient tracking of deformable shapes over small-time step captures. This framework is designed to handle the tracking of arbitrary dense point clouds. We show in Figure 1 an overview of the main steps of our algorithm. The system expects two point clouds as input, which are acquired from a dynamic 3D scene. In the first step, the point clouds are embedded in a common voxel discrete space. We consider two regular octrees as space partition structures. Then, correspondences between voxels are established by means of a dynamic programming method, the voxel matching algorithm (Section 2). We eventually refine this flow using a regularization step (Section 3) which employs simple and efficient algorithms. Finally, we handle a point correspondence assignment step (Section 4), related to the ICP algorithm [9], in order to generate a dense scene flow that describes the movement of each point of the sampled surface over time.



Common space | Level 2 | Level 5 | 3D Scene flow

Fig. 1. Major steps of our tracking algorithm computed from the red point cloud (a plane) to the blue one (a wave) : we first generate the common discrete space. We then make use of a multiresolution regular voxel grid (illustrated for the levels 2 and 5) on which we compute a discrete scene flow to obtain a dense point-to-point matching which can describe complex deformations.

2. OUR VOXEL MAPPING ALGORITHM

The first step of our method is the discretization of the observed scene. In contrast with previous works [7, 8] we propose to build a common and regular voxel space. This unique discrete space allows to benefit from consistent adjacency relations and the application of several filtering processes described in Section 3.

Notations Let $\mathcal{N}_i, i \in [\![1, N]\!]$, be the point cloud sequence, where each point cloud contains n_i points in $\mathbb{R}^3$. Considering two particular clouds $\mathcal{N}_1$ and $\mathcal{N}_2$, we aim to estimate a path from the former to the latter. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two regular octrees defined respectively by the voxel sets $\mathcal{U}_n$ and $\mathcal{V}_n$ for each resolution n, where the root cells $\mathcal{U}_0$ and $\mathcal{V}_0$ correspond to the bounding box of the point sets $\mathcal{N}_1$ and $\mathcal{N}_2$. We further embed the cloud $\mathcal{N}_1$ into the first space partition $\mathcal{O}_1$, and $\mathcal{N}_2$ into $\mathcal{O}_2$. These octrees are said regular regarding their subdivision process, we use the center of a cell as the cut point. As a result, the voxels of each octree share the same geometric sites but they do not contain the same data set. Now consider a subdivision level $s \in [\![0, S]\!]$, and let $(\mathcal{U}_s, \mathcal{V}_s)$ be the level s non-empty cells of $(\mathcal{O}_1, \mathcal{O}_2)$. We define the level s voxel mapping as $\mathcal{M}_s : \mathcal{U}_s \rightarrow \mathcal{V}_s$. The maximum level S is kept user defined, we have fixed $S = 6$ for all our tests.

2.1. Voxel Distance Function

We define the mapping $\mathcal{M}_s$ by minimizing a distance measure between the two voxel sets $(\mathcal{U}_s, \mathcal{V}_s)$. Data voxelization implies the mapping of a surface defined locally by the points contained in each voxel. In order to describe a piece of surface, we compute in each voxel a tangent plane that approximates the point cloud. This estimation is performed by a momentum analysis. In this framework, the comparison of two piecewise linear surfaces contained in the voxels $(u, v) \in (\mathcal{U}_s, \mathcal{V}_s)$ is equivalent to the evaluation of two couples (point-vector). For a couple of voxels (u, v), we consider two centers of mass (g_u, g_v) and two tangent plane normals (n_u, n_v). Our distance measure is defined as $\Delta(u_s, v_s) = \delta(u_s, v_s) + \varphi(u_s, v_s)$, where:

$$\begin{cases} \delta(u_0, v_0) = & \| \overrightarrow{g_0 g_0} \| \\ \delta(u_s, v_s) = & | \, \| \overrightarrow{g_u g_v} \| - \delta(u_{s-1}, v_{s-1}) \, | \\ \varphi(u_s, v_s) = & 1 - | \, n_u \, . \, n_v \, | \end{cases} \quad (1)$$

We added an important hierarchical flow penalization: a flow computed for a subdivision level s is not supposed to highly deviate from the coarser level $s-1$ flow. Aside this voxel-based hierarchical concern, this measure is related to several others described in [9].

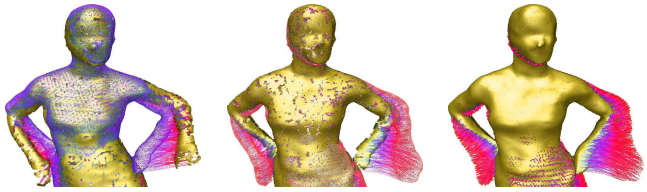
2.2. Discrete Scene Flow

We aim to build a set of vectors $\mathcal{X}_s$ which defines the voxel-to-voxel flow: $\mathcal{X}_s : \{x_u = \overrightarrow{g_u g_v} \mid (u, v) \in \mathcal{M}_s\}$, considering the center of mass of each voxel. Our approach is based on a bipartite graph resolution, thus we choose to apply a slight variation of the well-known Hungarian algorithm.
As stated beforehand, this algorithm is based on hierarchical concerns and voxel adjacency relations. In the first step, the two root cells u_0 and v_0 are paired: $\mathcal{X}_0 = \{\overrightarrow{g_{u_0} g_{v_0}}\}$. Then, for an octree subdivision level s, each cell u_s needs to be mapped with a cell v_s. The candidate cells v_s we chose are defined by a *constant* spatial area in the octree $\mathcal{O}_2$. This area is composed of two sets in $\mathcal{V}_s$:
1. Adjacent neighbors of u_s in $\mathcal{O}_2$, thus among 27 cells v_s.
2. Assuming that u_{s-1} has been mapped with v_{s-1}, we consider v^1_{s-1} as the 1-neighborhood of v_{s-1}. The second set is composed by the cells issued from the subdivision of v^1_{s-1}.
This set of candidates relies heavily on the octree hierarchy. This concern allows the definition of a constant research area for any subdivision level s, implying a very low computational complexity. Moreover, this search area can be far away from the current voxel u_s, thus a high subdivision level s does not limit our method to low amplitude deformations.

3. SCENE FLOW REGULARIZATION

Evaluation and Filtering It is a difficult problem to evaluate the overall quality of a scene flow. We propose to use the gradient magnitude as a local quality measure related to the vector flow homogeneity (or regularity). This gradient $G(u)$ is computed like the classical image gradient operator for our 3D vector field $x_u \in \mathcal{X}$,

$$G(u) = \; x_u - \frac{1}{|u^1|} \sum_{w \in u^1} x_w \quad , \quad (2)$$



(a) (b) (c)

Fig. 2. Our algorithm implies that every cell of the first octree (red color) is mapped with a cell of the second one (blue color). (a) Reverse mapping algorithm is a mirrored version of (b). (c) The two coupled flows form a complete bipartite correspondence between two frames.

where the voxels u^1 constitute the 1-neighborhood of u. This value highlights locally the sites where a mapping vector acts as an outsider, denoting a probable failure of our algorithm. Hence we propose to detect and filter these sites of high gradient magnitude. The filtering threshold can be determined automatically, but we assume in this work that it is kept user defined.

Diffusion Process The result of our algorithm is not supposed to find a map for every cell u_s. This concern is significant, since our algorithm, casted on an octree subdivision level s, relies on the results obtained for the precedent level $s-1$, see Section 2.1. Our diffusion process allows to assure the mapping of each cell of a subdivision level before progressing to the next one. It is defined by a simple operation: for each cell u_s for which our algorithm has not found any map, we search the best candidate w_s among the cells mapped with its 1-neighborhood which minimize $\Delta(u_s, w_s), (u_s^1, w_s) \in \mathcal{M}_s$.

The reverse Mapping Application of our algorithm implies a state where every voxel of $\mathcal{O}_1$ is mapped with a voxel of $\mathcal{O}_2$. Besides this concern, we need to assure that each voxel of $\mathcal{O}_2$ is reached by at least one mapping vector. Unless this assumption, we face problematic cases where some parts of the target shape could not be described by a deformation of the initial one. To tackle this problem without adding much complexity to our method, we propose to mirror the mapping process, see Figure 2. Once is performed our first mapping step, we apply the same voxel mapping algorithm from $\mathcal{O}_2$ to $\mathcal{O}_1$. In consequence, our double vector flow implies that several voxels $u_s \in \mathcal{U}_s$ can be mapped with a single voxel $v_s \in \mathcal{V}_s$, and several voxels v_s can be mapped with a unique voxel u_s.

4. MOTION TRACKING

Once our voxel mapping algorithm has built the global voxel-to-voxel vector mapping $\mathcal{X}$, the final step of our framework is to define the dense point-to-point 3D scene flow $\mathcal{M}_p$. Our system uses an efficient approach involving a local geometric warping by a noniterative quaternion transform [10]. For each voxel, we consider the local point set orientation given by the momentum analysis computed in Section 2.1. We project each point from a voxel u_S on the local basis oriented by the point set of its associated voxel v_S, $(u_S, v_S) \in \mathcal{M}_S$. From this point, we apply the same Hungarian algorithm we used in Section 2.2 to map each point of the couple (u_S, v_S). This step of our algorithm considers only the last octree subdivision level S, thus we can easily afford this naive mapping approach since we assume that u_S does not contain a large amount of points.

5. RESULTS AND DISCUSSION

We present here an application of our tracking algorithm framework throughout the temporal up-sampling of point cloud sequences. Such application can be used to realize slow motions for a dynamic capture, or to enhance the overall fluidity of an animation. We used both computer generated scenes and real reconstructed scan data sets. In Figure 3, our input data set is composed of two frames of the sequence *Samba* from the MIT CSAIL group: the frames 120 and 124. The morphing between these two frames was handled by a linear interpolation of the first point cloud along the dense 3D scene flow produced by our algorithm. We show here four incremental morphing steps from 0% (first frame) to 100% (second frame). In Figure 4, we present the same kind of morphing on the synthetised model *Hand*. Results show visually plausible oversampled frames, although some defects can be seen. In Figure 3, the red square highlights an important global topology change. Our system handled these cases by the creation of two *holes* in the point cloud, where each *frontier* is morphed to the other. In Figure 4, the red squares show that, despite a correct behavior of our technique, a simple linear interpolation cannot handle well the large non-rigid deformations.

We present a quantitative evaluation in Table 1. In order to evaluate the quality of our tracking, we use the Hausdorff distance between each oversampled frame taken at 50% morphing and a ground truth. In Figure 3 the ground truth is the known frame 122, and we have generated the second one in Figure 4. In our tests, the mean error is inferior to one percent of the model height. Our framework achieves very high computational performances compares to the previous point-based and mesh-based tracking methods. All reported running times are for a C implementation running on a 2.40GHz Intel Xeon processor. We stand that simple code optimizations, e.g. involving parallel computing, can orient our framework toward a real-time application.

Our algorithm is highly flexible, easy to implement, and can handle relatively large non-rigid deformations. The extreme generality of this method presents inherent limitations, but it can profitably orient a more complex tracking framework, e.g. a skeleton-based approach. Our future works will focus on this concern, the use of our method as a fast preprocessing step to a more robust and complete tracking method.

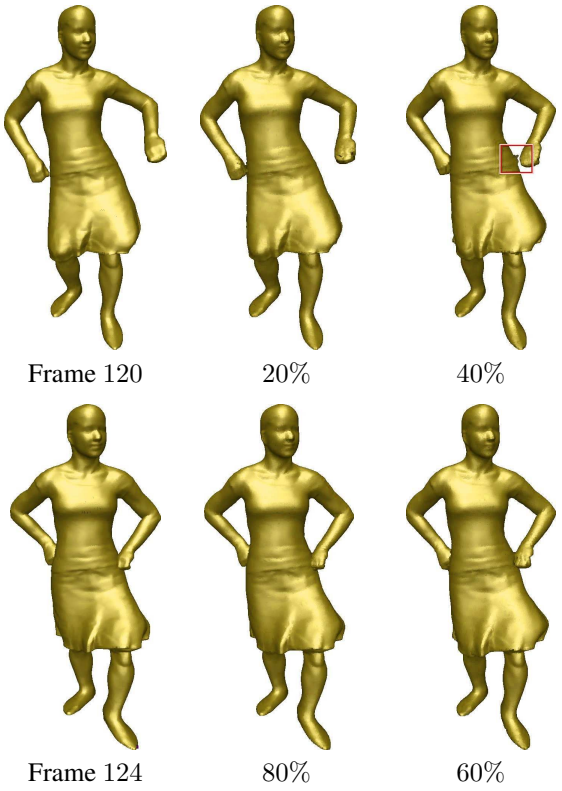


Fig. 3. MIT CSAIL model Samba: up-sampling from the frame number 120 to the frame 124 by 20% morphing increments.

		Mapping		Error	
Model	Size	Voxels	Points	Max	Mean
Samba	215588	260	370	5.3%	0.83%
Hand	634694	430	640	7.7%	0.98%

Table 1. Performance benchmark of several scene tracking displayed in Figures 3,4. Values are in milliseconds.

6. ACKNOWLEDGMENTS

This work was supported by the Regional Council of Burgundy within the PARI SSTIC5 project.

7. REFERENCES

[1] Thomas B. Moeslund and Erik Granum, "A survey of computer vision-based human motion capture," *Computer Vision Image Understanding*, vol. 81, pp. 231–268, March 2001.

[2] Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly, "Realtime performance-based facial animation," *ACM Transactions on Graphics (Proceedings SIGGRAPH 2011)*, August 2011.

[3] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling, "Efficient reconstruction of non-rigid shape and motion from real-time 3D scanner data," *ACM Transactions on Graphics*, vol. 28, no. 2, pp. 15:1–15:15, 2009.

[4] Will Chang and Matthias Zwicker, "Global registration of dynamic range scans for articulated model reconstruction," *ACM Transactions on Graphics, to appear*, vol. 30, no. 3, 2011.

[5] Sundar Vedula, Simon Baker, Peter Rander, Robert T. Collins, and Takeo Kanade, "Three-dimensional scene flow," in *International Conference on Computer Vision*, 1999, pp. 722–729.

[6] Sundar Vedula, Simon Baker, Steven Seitz, and Takeo Kanade, "Shape and motion carving in 6d," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2000, pp. 592–598.

[7] Edilson de Aguiar, Christian Theobalt, Marcus Magnor, Holger Theisel, and Hans-Peter Seidel, "M^3: Marker-free model reconstruction and motion tracking from 3d voxel data," in *Proceedings of Pacific Graphics 2004*, 2004, pp. 101–110.

[8] Ladislav Cmolìk and Miroslav Uller, "Point cloud morphing," in *Proceedings of the 7th Central European Seminar on Computer Graphics*, 2003, pp. 97–105.

[9] Szymon Rusinkiewicz and Marc Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, june 2001.

[10] Shinji Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 376–380, April 1991.

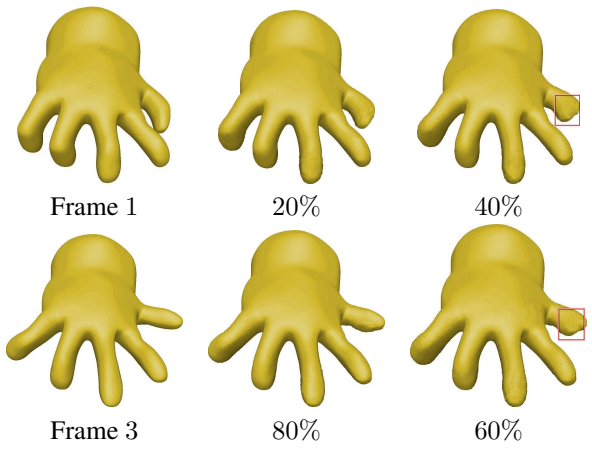


Fig. 4. Generated frames of the Hand: an open surface model. We morphed the frame 1 to the frame 3, and kept the frame 2 as a ground truth for the distance evaluation test.