

Temporally coherent mesh sequence segmentations

Romain Arcila, Cédric Cagniart, Franck Hétroy, Edmond Boyer, Florent Dupont

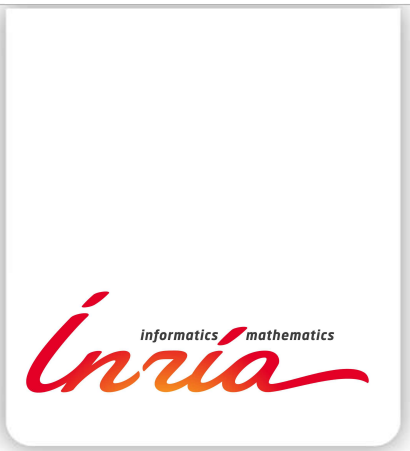
RESEARCH REPORT

N° 7856

January 2012

Project-Team Morpheo

ISSN 0249-6399 ISRN INRIA/RR--7856--FR+ENG



Temporally coherent mesh sequence segmentations

Romain Arcila*†, Cédric Cagniart‡, Franck Hétroy*, Edmond Boyer*, Florent Dupont†

Project-Team Morpheo

Research Report n° 7856 — January 2012 — 17 pages

Abstract: In this report, we consider the problem of fully automatic segmentation of mesh sequences, with or without temporal coherence. More precisely, our goal is to identify model parts that consistently move rigidly over time. We propose a novel framework that incrementally adapts segments along a sequence based on the coherence of motion information within each segment. In contrast to existing approaches, this framework handles meshes independently reconstructed at each time instant, provided that motion cues are available. It allows therefore for meshes with varying connectivity as well as varying topology. Experiments on various data sets in addition to a quantitative evaluation demonstrate the effectiveness and robustness of the approach.

Key-words: mesh sequence, segmentation, mesh tracking, spectral clustering

* Laboratoire Jean Kuntzmann, Inria, Université de Grenoble
† LIRIS, CNRS, Université de Lyon
‡ Technische Universität München

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Segmentations temporellement cohérentes de séquences de maillages

Résumé : Nous considérons dans ce rapport le problème de la segmentation entièrement automatique de séquences de maillages, avec ou sans cohérence temporelle. Plus précisément, notre but est d'identifier les parties d'un modèle qui se déplacent rigidement de manière cohérente au cours du temps. Nous proposons un canevas nouveau pour adapter ces régions de manière incrémentale le long de la séquence, en se basant sur la cohérence de l'information de mouvement dans chaque région. Contrairement aux approches existantes, ce canevas permet de traiter les séquences de maillages reconstruits indépendamment à chaque pas de temps, pourvu que des indicateurs de mouvement soient disponibles. Il permet donc de segmenter des maillages avec changement de connectivité et/ou changement de topologie. Des expériences sur plusieurs jeux de données ainsi qu'une évaluation quantitative démontrent l'efficacité ainsi que la robustesse de cette approche.

Mots-clés : séquence de maillages, segmentation, suivi de maillage, regroupement spectral

1 Introduction

Shape segmentation is the process of partitioning shapes into segments or regions that are semantically consistent with respect to a given criterion. Segmentation interest appears in several application domains where semantic information on shapes, e.g. body parts, are required for interpretation purposes, for instance motion analysis and recognition applications. When considering static 3D models, such process is faced with the issue of finding appropriate criteria to define segments [Sha08]. On the other hand, dynamic or temporal sequences of evolving 3D models overcome this limitation by providing motion information that naturally yield criteria, including rigidity, enabling the identification of segments. This is particularly relevant with multi-view acquisition systems, e.g. [SH07], or animation software, e.g. Blender [ble] for example, that can build temporal mesh sequences of dynamic scenes usually composed of rigid, sometimes articulated, moving parts.

Segmenting mesh sequences is anyway a challenging task. The main reason lies in the inconsistency of the produced meshes that may result from various factors. First, multi-view reconstructions are usually performed independently over time, hence providing meshes with different connectivities and even, occasionally, different topologies. Second, meshes are perturbed by the noise accumulated in the acquisition pipeline. Approaches have been proposed to recover temporally consistent meshes from inconsistent ones, e.g. [VBMP08]. While providing rich information for segmentation over time sequences, they usually require a reference model that introduces an additional step in the acquisition pipeline, hence increasing the noise level. Moreover, the reference model strongly constrains shape evolution to a limited domain.

In light of this, we propose an iterative scheme that clusters vertices into rigid segments along sequences using motion information between successive frames. In each frame, rigid segments can be refined by separating parts that present inconsistent motions or, on the contrary, merged when neighboring segments present similar motion. Motion information are estimated by matching meshes at successive instants. The contribution of the approach is threefold:

- it is fully automatic and does not require prior knowledge on the observed shape;
- it handles arbitrary shape evolutions, including changes in topology;
- segmentation can be computed on the fly.

This method can provide two kinds of segmentations: either one segmentation for the whole sequence, or a segmentation which evolves during the sequence, representing the currently displayed motion.

The remainder of the paper is as follows. Section 2 discusses related work. Section 3 outlines the proposed scheme. Sections 4 and 5 presents the matching between two successive time instants and the segmentation algorithm respectively. Section 6 shows comparisons with previous works and an evaluation proposal. Finally, we conclude in section 7.

2 Related Work

Spatio-temporal mesh sequences can be split into two categories:

- *dynamic meshes*, or temporally coherent mesh sequences: all meshes of the sequence have the same number of vertices and connectivity. This means that only Euclidean coordinates of the mesh vertices change over time;
- *unconstrained mesh sequences*: each mesh of the sequence has its own numbers of vertices and faces and its own connectivity. Moreover, global topology changes can happen. Such sequences are sometimes called *3D videos*, or *time-varying meshes*. Figure 1 shows some frames of the Balloon sequence which is a multi-camera reconstructed sequence.

Dynamic meshes are generally created by deforming a reference mesh, using skeleton driven animation system. While different in nature from meshes generated from real images, a significant effort has been devoted to their temporal segmentation. Unconstrained mesh sequences are generally captured from videos and have been a subject of interest over the last years in the computer vision domain. This kind of sequences now tends to be used by both communities [SH07, VBMP08].

Segmentation of a static shape is a common problem in several areas, i.e. texture mapping among others and many solutions have been proposed, see Shamir [Sha08] for a survey. Conversely, segmentation of temporal mesh sequences has received less attention. Several methods have been proposed to compute motion based segmentation of dynamic meshes. Among them, [LWC06, dATTS08] segment a dynamic mesh into rigid components. In particular, de Aguiar [dATTS08] proposes a spectral approach which relies on the fact that the distance between two points is invariant under rigid transformation. In this paper, we consider unconstrained mesh sequences and hence cannot use the invariant proposed by de Aguiar et al.

In contrast to [VBMP08, CBI10] which transform an unconstrained mesh sequence into a consistent mesh sequence, we only use the motion information available between two frames.

A main issue of unconstrained mesh sequences is that there is no direct matching between vertices of consecutive frames. This matching needs to be explicitly computed. This can be done using one of the numerous methods proposed in the literature [KSMH09] for example.

Cuzzolin et al. [CMK*08] compute protrusions (extremities) segmentation on point cloud sequences. Lee et al. [LTA08] propose a segmentation method for unconstrained mesh sequences using an additional skeleton as input. Arcila et al. [AKH*10] propose a framework to segment mesh sequences, but they cannot handle large reconstructed mesh sequences, such as mesh sequences built from multi-camera systems. Franco et al. [FB11] propose such a method for mesh sequences, but it requires as input the desired number of clusters.

In this paper, we propose a fully automatic approach to segment meshes with arbitrary evolution, including topology changes. To the best of our knowledge, no similar method exist in 3d. Interestingly, our approach shares similarities with the recent work of Brox and Malik [BM10] in the case of image segmentation. It should be noted that even if our method aims at segmenting mesh

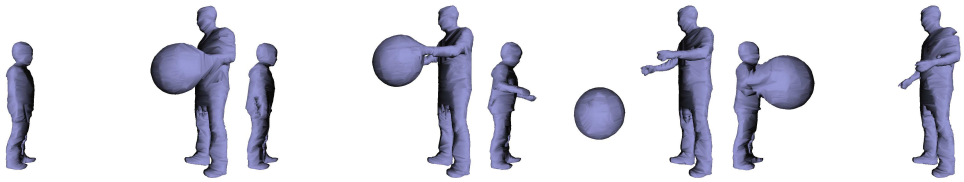


Figure 1: Frame examples from a multi-cam unconstrained mesh sequence (balloon sequence [Bal11].

sequences where meshes have different connectivities, it still performs well on dynamic meshes (see Section 6).

3 Method Outline

Our method takes as input a mesh sequence, where each mesh can have its own connectivity, and with potentially global topology changes. It uses an iterative approach and is designed to work on a small time window, containing a few (typically, 5) meshes. Working on only a few meshes at the same time allows to handle long sequences composed of meshes with a high number of vertices.

Our algorithm alternates between two steps at iteration k:

- Matching between 2 consecutive frames and displacement vectors computation within a time window from frame k.
- Segmentation of frame k and mapping to frame $k+1$.

This method relies on efficient matching and segmentation algorithms, that we describe in section 4 and 5.

We create a motion-based segmentation by clustering vertices into rigid components. Using a simple boolean parameter, we can generate two different types of segmentations:

- *global segmentations*: a single segmentation is sought over the full sequence;
- *time-varying segmentations*: the segmentation of a given frame represents the current motion around this frame. Clusters can appear and disappear along the sequence.

The notations used throughout the article are the following:

- N: number of frames;
- F_k: k^{th} frame that can be composed of several meshes;
- F'_k: k^{th} frame F_k registered to F_{k+1};
- $nv(F_k)$: number of vertices in F_k;
- $v_i^{(k)}$: vertex with index i in F_k;
- $\mathrm{Ng}(v_i^{(k)})$: 1-ring neighbors of vertex $v_i^{(k)}$.

Note that we always use k as the index for a frame, and i, j as the indices for vertices in a frame.

4 Vertex Matching

The objective of this step is, given meshes at frames $F_k, k \in [0..N-1]$, to provide a mapping from vertices $v_i^{(k)}$ to vertices $v_j^{(k+1)}$ for $1 \leq k < N$, and a possibly different mapping from vertices $v_j^{(k+1)}$ to vertices $v_i^{(k)}$. This mapping is further used to propagate segment labels over the sequence. We proceed iteratively according to the following successive stages: first, meshes at frames k and $k+1$ are registered, then displacement vectors and correspondences are estimated. The following subsections detail these stages. Please note that the segmentation algorithm presented further is independent of the matching performed and mainly requires displacement vectors. Thus, an approach based on 3D scene flow (e.g., [SAL*08]) could also be considered.

4.1 Mesh Registration

The matching stage of our approach aims at establishing a dense cross parametrization between pairs of adjacent meshes of the sequence. Among the many available algorithms for this task, we chose to favor generality by casting the problem as the registration of two sets of points and normals. This means that we exclusively use geometric cues to align the two meshes, even when photometric information is available like in the case of meshes reconstructed from multi-camera systems. Thus, our approach also handles the case of software generated mesh sequences.

We implemented the method of Cagniart et al. [CBI10] that iteratively deforms the mesh F_k to fit the mesh F_{k+1}. This approach decouples the dimensionality of the deformation from the complexity of the input geometry by arbitrarily dividing the surface into elements called patches. Each of these patches is associated to a rigid frame that encodes for a local deformation with respect to the reference pose F_k. The optimization procedure is inspired by ICP as it iteratively re-estimates point correspondences between the deformed mesh and the target point set and then minimizes the distance between the two point sets while penalizing non rigid deformations of a patch with respect to its neighbors. Running this algorithm in a coarse-to-fine manner by varying the radii of the patches has proven in our experiments to robustly converge, and to be faster than using a single patch-subdivision level.

4.2 Mapping and Displacement Vectors Extraction

By using the previous step, we get the registered mesh F'_k of the frame F_k on frame F_{k+1}. To create a mapping from F_k to F_{k+1}, we find for each vertex in F'_k the closest vertex in F_{k+1} using Euclidean distance. We also create a mapping from F_{k+1} to F_k by finding for each vertex in F_{k+1} the closest vertex in F'_k. Both mappings are necessary for the subsequent stage of our algorithm (see Section 5.3).

The displacement vector of each vertex $v_i^{(k)}$ in F_k such that $0 \leqslant i < nv(F_k)$ is given as

$$\overrightarrow{DV_i}^{(k)} = v_i'^{(k)} - v_i^{(k)},$$

with $v_i'^{(k)}$ the corresponding vertex in F'_k.

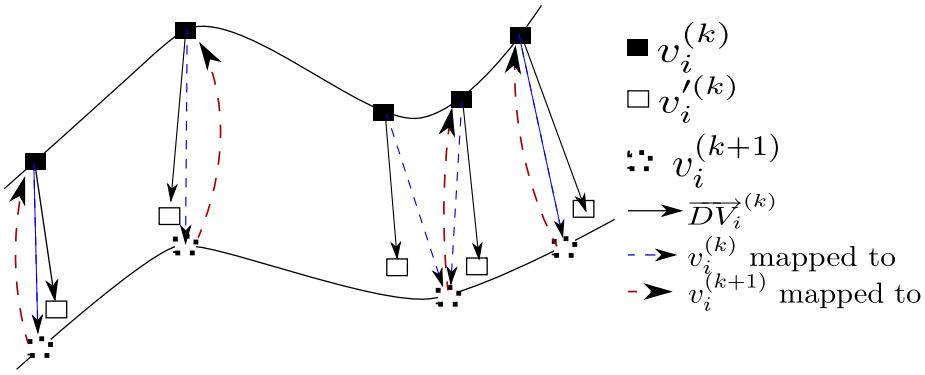


Figure 2: Matching process.

At the end of this stage, we get the mappings and displacement vectors between each pair of successive frames (see Figure 2).

5 Motion-based Segmentation

Our algorithm proceeds iteratively over a time sequence. First, in the case of a time-varying segmentation, neighboring clusters, in the current segmentation at time k, that present similar motions are merged (Section 5.2). Then, the current segmentation is refined using the displacement vectors at time k (Section 5.1). Finally, the segmentation is remapped onto the mesh at $k+1$ (Section 5.3).

Features of the approach are:

- it can process a sequence with numerous frames, where each frame can have a huge number of vertices. It only requires few frames in memory at a time. For instance, results shown on Figure 3 correspond to a sequence of 300 frames with approximately 15,000 vertices per frame;

- it can control cluster creation and keep track of the transformations.

In the case of a global segmentation (Section 5.4), for which no cluster merging is allowed, the segmentation computed on the last frame is the finest one. This segmentation is then mapped back onto the whole sequence.

Note that we use a time window, 5 frames from k to $k+4$ in our experiments, to compute displacements at each frame k. The segmentation of a frame is performed using a spectral clustering approach applied on the graph of vertices of the frame. Edges in this graph correspond to edges on the mesh. An edge is weighted by the distance between the transformations of its 2 vertices. This is in contrast to [dATTS08] where Euclidean distances between vertices are considered. Intuitively transformation distances better constrain rigidity than Euclidean distances and our experiments confirm this fact.

Our segmentation algorithm produces, by construction, connected clusters since the atomic operations over clusters are: splitting a cluster in connected clusters through spectral clustering, and merging neighboring clusters.

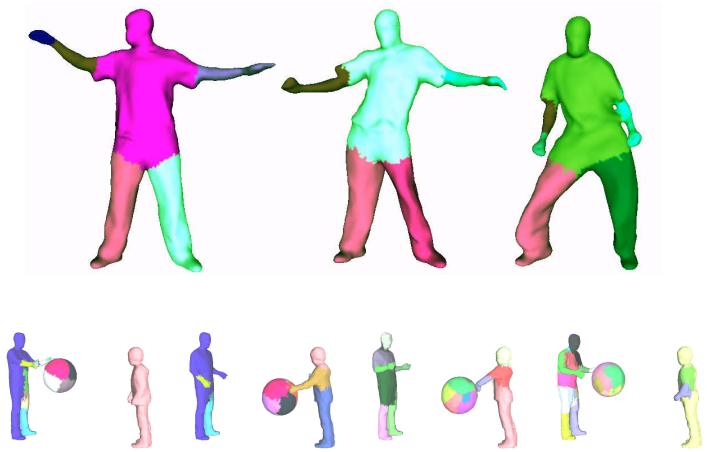


Figure 3: First row: time-varying segmentation generated by our algorithm on the dancer sequence [SH07]. First frames are decomposed into 6 segments, then the right arm and right hand clusters are merged since they move the same way. Finally, this cluster is split again. Note that we can handle topology changes (in the last frames, the left arm is connected to the body). Second row : time-varying segmentation of a sequence with 15,000 vertices per mesh.

5.1 Spectral Clustering

Using mappings and displacement vectors computed as previously explained, we build a graph whose edges are weighted according to rigid transformation similitude between their two endpoints. The normalized Laplacian matrix associated with this graph is computed and the clustering is performed on this matrix.

5.1.1 Graph Construction

To compute the spectral clustering, the weighted adjacency matrix of the graph associated to the vertex transformations between two consecutive frames is build. The nodes of this graph are the vertices of the frame at time k. To construct this graph, we first compute the rigid transformation which maps each vertex $v_i^{(k)}$ of the frame F_k, and its one-ring neighborhood $\mathrm{Ng}(v_i^{(k)})$, onto the frame F'_k . This transformation is computed using Horn's method that estimates a 4×4 matrix representing the best rigid transformation between 2 point clouds. A transformation matrix $T_i^{(k)}$ is therefore associated to each vertex $v_i^{(k)}$. To compute the weights of the graph $W^{(k)}$ edges, we use the following expression [MSZ94]:

$$w_{ij}^{(k)} = \begin{cases} \frac{1}{\|\log(T_i^{(k)-1} T_j^{(k)})\|^2} & \text{if } i \neq j, \\ 0 & \text{if } i = j. \end{cases} \quad (1)$$

As demonstrated in [MSZ94], this distance is the most meaningful one between rigid transformations in the sense that it corresponds to distances on the man-

ifold of rigid transformations $SE(3)$. We comment on the consequences of this choice in section 6.3.2.

5.1.2 Laplacian Matrix and Clustering

Using the weighted adjacency matrix $W^{(k)}$, we build the normalized Laplacian matrix $L_{rw}^{(k)}$ as follows and use the well-known Shi-Malik normalized spectral clustering algorithm [SM00] to segment the graph.

$$D_{ii}^{(k)} = \sum_{j \in \mathrm{Ng}(v_i^{(k)})} w_{ij^{(k)}}. \tag{2}$$

$$L^{(k)} = D^{(k)} - W^{(k)}. \tag{3}$$

$$L_{rw}^{(k)} = D^{(k)-1} L^{(k)} = I^{(k)} - D^{(k)-1} W^{(k)}. \tag{4}$$

This method assumes the number K of clusters to be known. We compute K using the eigen gap method (see section 6.3.1 for a discussion on the eigen gap method).

5.2 Merging

In the case of a time-varying segmentation, we merge at each time step neighboring clusters with similar motions before applying the spectral clustering. To this aim, the distance between the transformations of neighboring clusters is thresholded using T_{merge}, where the transformation of a cluster is estimated over all its vertex displacements using the Horn's method.

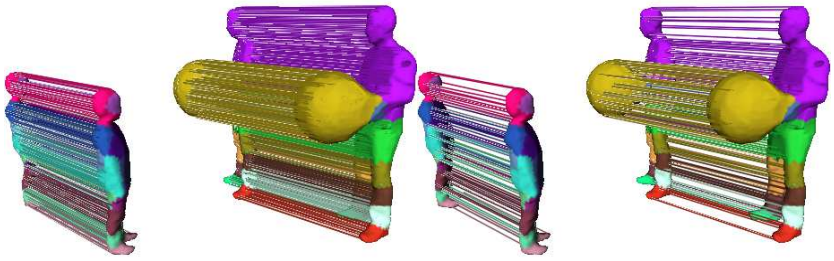
The merging starts with the cluster with the minimal residual error, and stops when this process cannot be applied anymore. The residual error for a cluster corresponds to the average distance, for all points $v_i^{(k)}$ of the cluster, between the point $v_i^{(k+1)}$ and the location of $v_i^{(k)}$ after the rigid transformation computed for the cluster is applied. An issue that arises from having two different thresholds for the cluster merging and splitting operations is that two clusters which are merged can then be split right afterwards. To avoid this situation, we apply spectral clustering on the set of the two clusters before merging them in practice. If they are split during clustering, then we do not effectively merge them.

5.3 Mapping onto the next frame

The segmentation is computed at each time step on the current frame F_k. Labels are then mapped onto the frame F_{k+1} using the bi-directional mapping defined in Section 4.2. Clusters are first transferred using the mapping from frame F_k to F_{k+1}. Then for all unmatched vertices in F_{k+1}, we use the mapping from F_{k+1} to F_k. Clusters which are mapped on 2 different meshes are split.

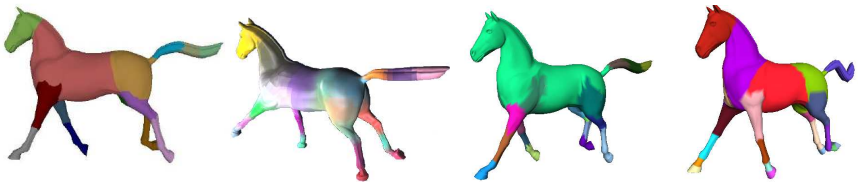
5.4 Global Segmentation

In order to get a time-coherent global segmentation over the whole sequence, the segmentation obtained for the last frame need to be transferred to all previous



(a) (b)

Figure 4: Result of vertex matching. (a) Full display. (b) Partial display.



[LWC06]. [dATTS08]. [AKH*10]. With our method.

Figure 5: Segmentation results on a dynamic mesh.

frames. To this purpose, we apply the mapping process described in the previous section in reverse order, from last to first frame. For each pair of successive frames (F_k, F_{k+1}) we first remap using the vertex matching from F_{k+1} to F_k, then for all remaining points, we use the vertex matching from F_k to F_{k+1}.

6 Experiments and Discussion

6.1 Results and Comparisons with Previous Works

6.1.1 Vertex Matching

The vertex matching computation is an important step since our segmentation algorithm relies on it. Figure 4 shows the result of vertex matching between two successive frames of an unconstrained mesh sequence. Computation time is about 30 seconds for two frames with approximately 7000 vertices each. [AKH*10] used another matching method which provides similar results, but the proposed method outperforms their matching algorithm which takes about 13 minutes to complete computation with the same data. Note that outliers in the matching are not explicitly taken into account in the segmentation, however their influence is limited by the threshold on the cluster size that tends to force their merges with neighboring clusters.

6.1.2 Segmentation of Dynamic Meshes

As shown in Figure 5, our method yields visually similar results to previous methods dedicated to dynamic meshes, i.e. temporally coherent sequences. Note

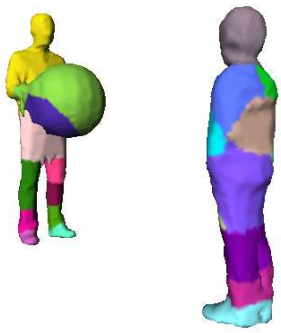


Figure 6: Global segmentation result on the balloon sequence.

that each cluster is connected: even if some clusters in Figure 5 share the same color, they represent different clusters.

6.1.3 Global Segmentation of Unconstrained Mesh Sequences

To the best of our knowledge, [AKH*10] is the only previous method which computes global segmentation of unconstrained mesh sequences. Our method shares some similarities with their framework; however, our segmentation algorithm allows for better cluster evolutions by enabling both merge and split operations and by using a spectral clustering method for that purpose. On dynamic meshes such as the horse sequence (see Figure 5), our segmentation results are closer to the presented results in [dATTS08] and in [LWC06]. The improvement over [AKH*10] illustrates the benefit of the spectral clustering.

6.1.4 Time-Varying vs. Global Segmentations

Figures 3 shows time-varying segmentations computed on the Balloon and Dancer sequences. Figure 6 shows a global segmentation computed on the Dancer sequence. By construction, global segmentations contains more clusters than time-varying segmentations since no merging process is applied. Thresholds for both time varying and global segmentation of the Dancer sequence share the same value, except for the eigen gap threshold which is slightly lower in the time varying segmentation case. In general, threshold values for a sequence are easily found in a few trials. The computation time of the segmentation between 2 frames of the Dancer sequence is approximately 3 minutes with a (not optimized) Matlab implementation. Additional results appear in the accompanying video.

6.2 Quantitative Evaluation

A quantitative and objective comparison of segmentation methods is an ill-posed problem since there is no common definition of what an optimal segmentation should be in the general case. The segmentation evaluation has been recently addressed in the static case using ground truth (i.e. segmentations defined by humans) [BVLD09]. In the dynamic mesh case, none of the previously cited

articles proposes an evaluation of the obtained segmentations. We propose the following framework to evaluate a mesh sequence segmentation method.

6.2.1 Optimal Segmentation

The *optimal* segmentation of a mesh animation (dynamic mesh or unconstrained mesh sequence) into rigid components can be guessed when the motion is known. This is, for instance, the case with skeleton-based mesh animations, as created in the computer graphics industry. In this case, each mesh vertex of the sequence is attached to at least one (usually, no more than 4) "joints" of the animation skeleton, with given weights called *skinning weights*. These joints are organized in a hierarchy, which is represented by the "bones" of the skeleton that are, therefore, directed. For our evaluation, we attach each vertex to only one joint among the related joints, the furthest in the hierarchy from the root joint. If this joint is not unique, we keep the one with the greatest skinning weight. Each joint has its own motion, but several joints can move together in a rigid manner. For a given frame, we can therefore cluster joints of the animation skeleton into *joint sets*, each joint set representing a different motion. We now define as an *optimal cluster* the set of vertices related to joints in the same joint set. Since we know the motion of each joint, we exactly know, for each frame, what are the optimal clusters.

This definition can be applied in the general case of unconstrained mesh sequences, provided that each vertex of each frame can be attached to a joint. However, we only tested it in the more convenient case of a dynamic mesh.

6.2.2 Error Criteria

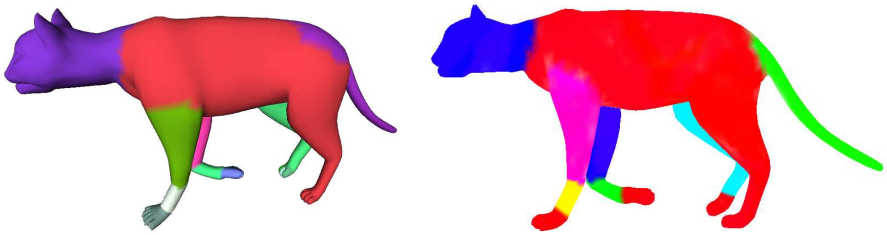
We propose the following criteria in order to evaluate a given segmentation with respect to the previously defined optimal segmentation:

- *Frame Assignment Error* (FAE): for a given frame, the ratio of vertices which are not assigned to the correct cluster. This includes the case of clusters which are not created, or which are wrongly created;
- *Vertex Assignment Error* (VAE, in the case of dynamic meshes): for a given vertex, the ratio of frames in which the vertex is not assigned to the correct cluster.

6.2.3 Results

We tested our algorithm on a walking cat skeleton-based animation (see Figure 7 and the accompanying video). We get a time-varying segmentation with a FAE up to 17%, for the worst frame. Wrongly assigned vertices correspond to the cat skin around joints and to a wrong subdivision in cat paw, i.e. in the less rigid areas.

In the case of global segmentations, and if we do not take into account matching issues, the FAE is the same for all frames. For the cat sequence, the FAE is also 17%. The VAE varies between 0% and 100%, and is only relevant as a relative criterion to compare vertices. On the cat sequence vertices in highly rigid areas (paws, tail, body) are always assigned to the correct cluster, while some vertices around joints can be assigned to the same neighboring cluster for all frames.



(a) (b)

Figure 7: Result on a skeleton-based synthetic animation. (a) Computed time-varying segmentation. (b) Optimal time-varying segmentation, for the same frame.

6.3 Discussion

6.3.1 User Defined Parameters

We use three parameters to drive the segmentation: the number of clusters, the minimum cluster size and the merge threshold. The eigen gap method is used for each frame to determine the number of clusters to create, but is not always reliable. To overcome this problem and to avoid the creation of many subclusters, we use the minimum cluster size parameter. In all of our experiments, we have arbitrarily set this parameter to 5%. This value works well enough for reconstructed mesh sequence as meshes in these sequences are uniform. This threshold is used for instance on the balloon sequence to ensure that the ball is not split into many subclusters as it is hard to find a rigid transformation on the ball. It improves computation time but also avoids small cluster creation around articulations, which are usually not rigid. Finally, the merge threshold is used to decide if two clusters represent the same motion (see section 5.2). It is closely related to the eigen gap value, since if these thresholds are not set correctly, the algorithm can merge two clusters which are split by the spectral clustering right afterwards.

6.3.2 Distance Function

We have tried our algorithm with three distance functions to compute distances between rigid transformations:

1. Residual errors between clusters;
2. Frobenius norm;
3. Logarithm of matrices (see Equation 1).

In our experiments, the residual errors between clusters are really difficult to set correctly. The logarithm of matrices provides the best result but if the computation time is a hard constraint, then the Frobenius norm yields also quite good results. Actually on dynamic meshes, both yield the same results. However, on unconstrained mesh sequences logarithm based distance yields better results, i.e. more regular clusters, than the Frobenius distance.

6.3.3 Current Limitations

We are currently aware of three limitations in the proposed algorithm:

- our method clearly depends on the the quality of the matching process. Important errors in matching computation can lead to wrong results;
- clusters which are wrongly subdivided are transferred to the following frames, meaning that errors on an early frame in the sequence can affect the whole segmentation. This issue is less important on time-varying segmentation since clusters will be merged later. Nevertheless, on global segmentations, it can lead to wrong results. Such errors are generally due to error in the matching process;
- segmentation can slightly drift: this is due to the fact that we only consider 2 frames when matching.

As shown in our quantitative evaluations, vertices which are wrongly assigned to a cluster are located near articulations. Vertices in rigid regions are generally correctly clustered.

7 Conclusion

We have presented a segmentation method which takes as input a mesh sequence, either reconstructed from multi-view videos or created using an animation software. It produces a global segmentation or a time-varying segmentation into rigid components and handles topological changes. The method is fast and yields visually consistent results. It uses a few parameters which can be easily set. We have also proposed a framework for quantitative evaluations of rigid segmentation methods, and we have shown that our method behaves favorably with respect to existing approaches. Future works include the use of a bayesian framework to track and predict clusters. We also plan to improve the vertex assignments around articulations to clusters and optimize our implementation.

8 Acknowledgements

The Dancer sequence is courtesy of University of Surrey [SH07]. The Balloon sequence is courtesy of Inria Grenoble [Bal11]. The Cat sequence is courtesy of Inria Grenoble [AHLD07].

References

[AHLD07] Aujay G., Hétroy F., Lazarus F., Depraz C.: Harmonic skeleton for realistic character animation. In *Symposium on Computer Animation, SCA 07, August, 2007* (2007), Eurographics, pp. 151–160.

[AKH*10] Arcila R., Kartik B., Hétroy F., Denis F., Dupont F.: A framework for motion-based mesh sequence segmentation. In *WSCG* (2010).

[Bal11] Balloon sequence, Retrieved april 2011. `http://4drepository.inrialpes.fr/`.

[ble] Blender. `http://www.blender.org/`.

[BM10] Brox T., Malik J.: Object segmentation by long term analysis of point trajectories. In *ECCV* (2010).

[BVLD09] Benhabiles H., Vandeborre J.-P., Lavoué G., Daoudi M.: A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3D-models. In *SMI* (2009).

[CBI10] Cagniart C., Boyer E., Ilic S.: Iterative deformable surface tracking in multi-view setups. In *3DPVT* (2010).

[CMK*08] Cuzzolin F., Mateus D., Knossow D., Boyer E., Horaud R.: Coherent laplacian 3-d protrusion segmentation. In *CVPR* (2008).

[dATTS08] de Aguiar E., Theobalt C., Thrun S., Seidel H.: Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum (Eurographics Proceedings) 27*, 2 (2008).

[FB11] Franco J.-S., Boyer E.: Learning Temporally Consistent Rigidities. In *IEEE CVPR* (2011), pp. 1241–1248.

[KSMH09] Knossow D., Sharma A., Mateus D., Horaud R.: Inexact matching of large and sparse graphs using laplacian eigenvectors. In *7th Workshop on Graph-based Representations in Pattern Recognition* (2009).

[LTA08] Lee N., T.Yamasaki, Aizawa K.: Hierarchical mesh decomposition and motion tracking for time-varying-meshes. In *ICME* (2008).

[LWC06] Lee T.-Y., Wang Y.-S., Chen T.-G.: Segmenting a deforming mesh into near-rigid components. *The Visual Computer 22*, 9 (2006).

[MSZ94] Murray R. M., Sastry S. S., Zexiang L.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., 1994.

[SAL*08] Sharf A., Alcantara D., Lewiner T., Greif C., Sheffer A., Amenta N., Cohen-Or D.: Space-time surface reconstruction using incompressible flow. *ACM Transactions on Graphics (SIGGRAPH Asia Proceedings) 27*, 5 (2008).

[SH07] Starck J., Hilton A.: Surface capture for performance based animation. *IEEE Computer Graphics and Applications* (2007).

[Sha08] Shamir A.: A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6 (2008).

[SM00] Shi J., Malik J.: Normalized cuts and image segmentation. *PAMI 22*, 8 (2000).

[VBMP08] Vlasic D., Baran I., Matusik W., Popović J.: Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics (SIGGRAPH Proceedings)* (2008).

Contents

1 Introduction **3**

2 Related Work **4**

3 Method Outline **5**

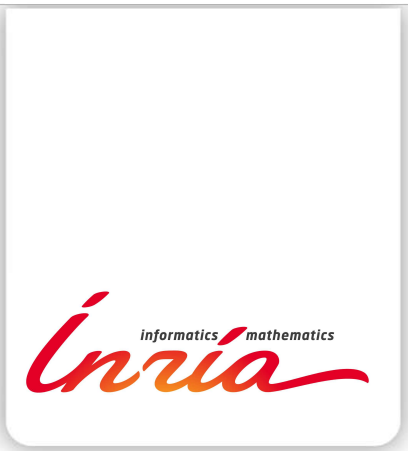
4 Vertex Matching **6**
4.1 Mesh Registration 6
4.2 Mapping and Displacement Vectors Extraction 6

5 Motion-based Segmentation **7**
5.1 Spectral Clustering 8
5.1.1 Graph Construction 8
5.1.2 Laplacian Matrix and Clustering 9
5.2 Merging 9
5.3 Mapping onto the next frame 9
5.4 Global Segmentation 9

6 Experiments and Discussion **10**
6.1 Results and Comparisons with Previous Works 10
6.1.1 Vertex Matching 10
6.1.2 Segmentation of Dynamic Meshes 10
6.1.3 Global Segmentation of Unconstrained Mesh Sequences . 11
6.1.4 Time-Varying vs. Global Segmentations 11
6.2 Quantitative Evaluation 11
6.2.1 Optimal Segmentation 12
6.2.2 Error Criteria 12
6.2.3 Results 12
6.3 Discussion 13
6.3.1 User Defined Parameters 13
6.3.2 Distance Function 13
6.3.3 Current Limitations 14

7 Conclusion **14**

8 Acknowledgements **14**



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399