Motion-based mesh segmentation using augmented silhouettes

Stefano Marras · Michael M. Bronstein · Kai Hormann · Riccardo Scateni · Roberto Scopigno

Abstract

Motion-based segmentation, the problem of detecting rigid parts of an articulated three-dimensional shape, is an open challenge that has several applications in mesh animation, compression, and interpolation. We present a novel approach that uses the visual perception of the shape and its motion to distinguish the rigid from the deformable parts of the object. Using two-dimensional projections of the different shape poses with respect to a number of different view points, we derive a set of one-dimensional curves, which form a superset of the mesh silhouettes. Analysing these augmented silhouettes, we identify the vertices of the mesh that correspond to the deformable parts, and a subsequent clustering approach, which is based on the diffusion distance, yields a motion-based segmentation of the shape.

Citation Info

Journal
Graphical Models
Volume
74(4), July 2012
Pages
164–172
Note
Proceedings of GMP

1 Introduction

Shape segmentation refers to the problem of partitioning a three-dimensional object into meaningful parts. Although many approaches are described in the literature, it remains a challenging problem, also because the notion of "meaningful parts" is somewhat ambiguous. Typically, algorithms identify the different parts of an object by first classifying the elements of the mesh (faces, edges, or vertices) using some specific metric (dihedral angles, geodesic or diffusion distances, etc.) and then partitioning the mesh using, for example, k-means, hierarchical, or fuzzy clustering.

Katz and Tal [13] introduce the *all-pair shortest paths* (APSP) concept and the idea of clustering for mesh segmentation. This approach has been improved by using hierarchical clustering [2, 10] or k-means clustering [6], while other techniques use random walks [14] or mesh scissoring [16]. For a detailed survey of these techniques, we refer the reader to [24].

1.1 Prior work

The particular setting of the segmentation problem considered in this paper is *motion-based segmentation*, where a set of different poses (deformations) of a single shape is given and the goal is to partition the mesh into parts that move coherently over the different poses. Such segmentation problems have important applications in computer vision [8, 11, 19, 22, 23], graphics [9, 12, 15], mechanical engineering, and biomechanics [1, 4].

A common assumption is that the shape is articulated, that is, it consists of parts that move in a piecewise rigid manner. James and Twigg [12] solve the problem of identifying the motion of different parts of an articulated object by first estimating the trajectory described by each face of the mesh over time and then clustering faces that are characterized by similar movements. Lee et al. [15] present a similar approach, with main differences in the use of a dual graph of the object and a different way to represent and encode face trajectories. While both methods perform a clustering step on the faces of the object, de Aguiar et al. [9] are the first to work with the mesh vertices. They analyse how each vertex moves with respect to its neighbours, compute an *affinity matrix*, and finally use this matrix to decompose the shape. Rosman et al. [22] represent motion as a group-valued function on the shape and determine a segmentation by using a total variation-like functional.

1.2 Main contribution

The inspiration for this work comes from two sources: the recent work of Wuhrer and Brunton (see Section 2.1) on segmentation of dynamic meshes using clustering of dihedral angles of adjacent faces in corresponding poses [28] and the classical result, that the three-dimensional structure (up to a reflection) of a rigid body can be found from only three frames, which goes back to Ullmann [18, 26, 27]. In particular, this allows to unambiguously recover the 3D motion of an object from multiple 2D projections.

We use a similar reasoning to argue that piecewise rigid motions of a surface can be captured by considering a set of curves on the surface. The variation of angles between edges of a curve allows us to assign parts of the

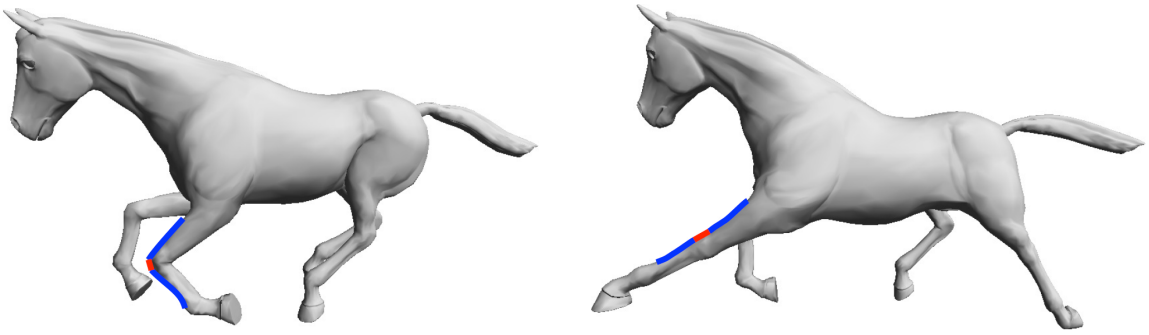


Figure 1: Motivation for our approach: shown are two poses of the horse shape and a curve passing through two articulated parts (upper and bottom part of the leg) connected by a joint (knee). The curve can be easily segmented into rigid parts (blue).

curve to rigid parts of the shape. Merging this information from several curves can then be used to obtain a segmentation of the entire shape. The required 1D analysis significantly alleviates the computational complexity of the algorithm.

2 Background

Let $\mathcal{K} = (X, E, T)$ be an abstract simplicial complex with a set of n nodes $X = \{x_1, \dots, x_n\}$, a set of edges E, and a set of triangles T, such that any edge $e = (i, j) \in E$ between nodes x_i and x_j has exactly two adjacent triangles $t, t' \in T$. Any mapping $\Phi\colon X \to \mathbb{R}^3$ that associates with each node $x_i \in X$ a 3D vertex v_i then yields a triangle mesh $M = (\Phi, \mathcal{K})$ with *geometry* $V = \Phi(X) = \{v_1, \dots, v_n\}$ and *topology* $\mathcal{K}$.

Suppose we are given N manifold triangle meshes $M_k = (\Phi_k, \mathcal{K})$, $k = 1, \dots, N$ with the same topology, which represent N different poses of some shape. We assume that all poses can be segmented consistently into l *rigid parts* and m *deformable joints*. Then there exists a decomposition of the set X into disjoint sets $S_1, \dots, S_l$ and $J_1, \dots, J_m$, such that the corresponding patches of the meshes M_k with vertices $\Phi_k(S_i)$ and $\Phi_k(J_j)$ are related by suitable rigid or non-rigid transformations, respectively.

The main goal of motion-based segmentation is to find the rigid parts S_i of the shape, using only the information given by the N meshes $M_1, \dots, M_N$. We further stick to the common approach where the joint regions J_j are neglected and their nodes distributed to the nearest rigid part for simplicity.

2.1 Wuhrer–Brunton method

The approach of Wuhrer and Brunton [28] solves the problem as follows. Given the meshes, it computes the *dual graph* $\mathcal{G}$ of the common simplicial complex $\mathcal{K}$, in which a node corresponds to a triangle of the original mesh, and an edge connects two adjacent triangles. Each edge of $\mathcal{G}$ is then weighted by the maximum variation of its corresponding dihedral angles across all poses. In this way, dual edges with small weights correspond to rigid parts of the mesh, while dual edges with large weights correspond to deformable joints. To determine the l rigid parts of the shape, the dual graph $\mathcal{G}$ is partitioned by computing its minimum spanning tree, and then cutting off the dual edges with the l largest weights. The number of segments is usually unknown and can be computed at runtime. The possibility of *over-segmentation* requires to perform an additional merging step in order to obtain reasonably sized segments. The complexity of the first step of the algorithm is $\mathcal{O}(N^2 n + n \log n)$, while the merging step, which consists of the reinsertion of some of the edges into the spanning tree, has $\mathcal{O}(n \log n)$ complexity.

2.2 Motivation for our approach

The main motivation for our approach comes from the following observation. Assume that we are given a curve going through the rigid parts S and S' and a joint J connecting them. Then, if the shape is articulated at this joint, we will observe the two parts of the curve undergoing a rigid transformation. As a result, the curvature of the curve at the joint will change, but will remain constant at the rigid parts (see Figure 1). Given a set of curves that cover all rigid parts of the articulated shape and observing them in different poses, we will be able to segment the shape according to its motion.

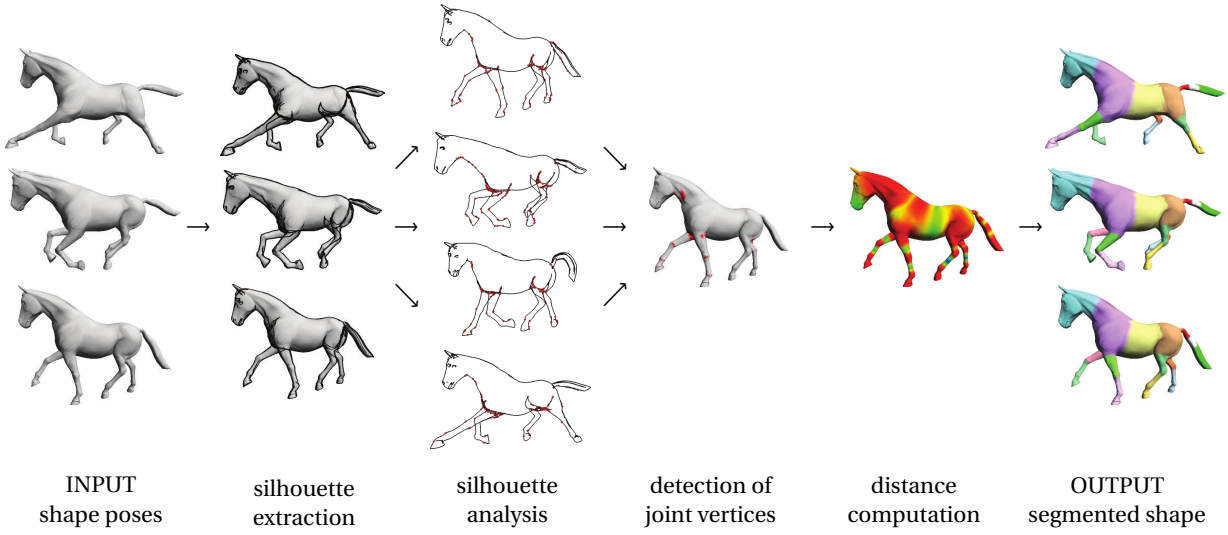


Figure 2: Different stages of the proposed motion-based segmentation algorithm.

3 Silhouette-based segmentation

Based on the motivation outlined above, our approach to shape segmentation from motion can be divided into three main steps, as visualized in Figure 2. First, we extract a set of 1D curves from different poses of the shape. Second, we analyse these 1D curves based on the motion information. Third, we combine the results of the 1D curve analysis to obtain the shape segmentation. In what follows, we describe these three steps in details.

3.1 Augmented silhouette extraction

Though in principle our approach can work with any set of curves, we use a specific data structure that contains information about the perceptively relevant edges of the shape, referred to as the *augmented silhouette*. The choice of working with a 1D silhouette-based structure may seem problematic, due to the fact that silhouettes are often unstable and subject to abrupt changes, but it is also true that these changes encode the relevant motion information. Moreover, in order to deal with the changes in a stable way, we create a correspondence map between the silhouette edges and the mesh elements.

Given K view points $\mathcal{V}_1,\ldots,\mathcal{V}_K$, the augmented silhouette $\mathcal{S}_{j,k}=(X,E_{j,k})$ of mesh M_j with respect to view point $\mathcal{V}_k$ is a graph with nodes X and edges $E_{j,k}\subset E$. An edge $(i_1,i_2)\in E$ is an element of $E_{j,k}$ if and only if the corresponding mesh edge $[v_{j,i_1},v_{j,i_2}]$ is *perceptively relevant*, that is, exactly one of the two adjacent faces is visible from view point $\mathcal{V}_k$. Since we do not take occlusion into account, the augmented silhouette usually contains more than just the silhouette edges of M_j with respect to $\mathcal{V}_k$, hence the name (see Figure 3).

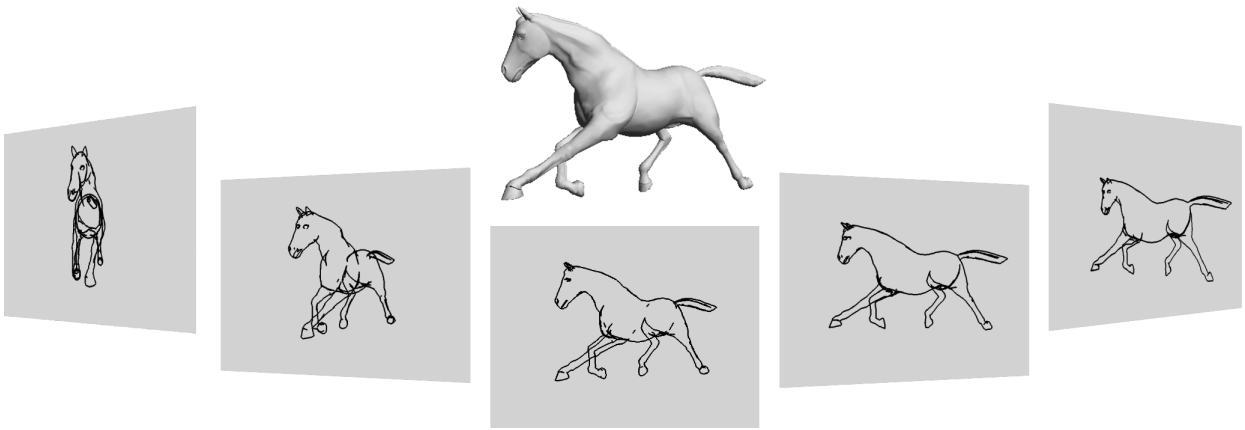


Figure 3: Examples of several augmented silhouettes extracted from a 3D shape of the horse for different view points.

The classification of edges into perceptually relevant and irrelevant is simple and can be done in parallel for different edges and for different view points. We first process all triangles of M_j in parallel and determine the signs of the dot products between each triangle normal and the viewing directions of the view points $\mathcal{V}_k$. Then we process all edges of M_j, compare the signs of the adjacent triangles, and classify an edge as perceptively relevant if and only if the signs are different.

In our experiments we use $K = 25$ view points, with positions taken from a subdivided icosahedron that circumscribes all poses and viewing directions towards the centre of this icosahedron. Moreover, we use only K' randomly chosen augmented silhouettes from all KN possibilities and found a value of K' between 50 and 100 to be sufficient for correctly identifying the rigid and deformable parts.

Typically, each augmented silhouette contains only $O(\sqrt{n})$ edges and nodes, so the overall size of all silhouettes is on the order of $K'\sqrt{n}$, which is significantly less than the Nn vertices from all N meshes, because $K' < KN \ll N\sqrt{n}$, especially for large meshes.

3.2 1D curve analysis

The goal of the second step is to identify the vertices where the deformation occurs and to label them as *significant vertices*. In order to do so, we consider each of the K' selected augmented silhouettes from the first step separately.

Suppose that x_i is a node with neighbouring nodes x_{i_1} and x_{i_2} in such an augmented silhouette $\mathcal{S}_{j,k}$. All three nodes have corresponding 3D vertices $v_{l,i} = \Phi_l(x_i)$, $v_{l,i_1} = \Phi_l(x_{i_1})$, $v_{l,i_2} = \Phi_l(x_{i_2})$ in each of the N meshes M_l and related 2D vertices $w_{l,i}$, w_{l,i_1}, w_{l,i_2} in the projection with respect to view point $\mathcal{V}_k$. We then compute the angles

$$\alpha_l = \sphericalangle(w_{l,i_1} - w_{l,i}, w_{l,i_2} - w_{l,i})$$

between the two edges incident to w_{l_i} in the 2D projections of the N poses and assign the maximal difference

$$\max_{l=1,\ldots,N} \alpha_l - \min_{l=1,\ldots,N} \alpha_l$$

of these angles as a *deformation weight* to the node x_i. In this way, the nodes which are likely to correspond to joint vertices of the shape are the ones with the largest weights.

This computation is not carried out for nodes x_i with less than two neighbours in $\mathcal{S}_{j,k}$, and for nodes with more than two neighbours we compute the difference between the maximum and the minimum of all angles formed by any pair of adjacent edges.

After assigning a deformation weight to each node, we further analyse the augmented silhouette and identify the *significant* nodes by iterative adaptive clustering. We first find the minimum and maximum of all deformation weights and choose them as seeds s_0, s_1 of two clusters C_0, C_1, respectively. We then measure, for each node of the graph, the distance to s_0 and s_1 and assign it to the closest cluster. In our case, the distance between two nodes is computed by simply taking the absolute value of the difference between their deformation weights. We then recompute the seeds s_0, s_1 by averaging the deformation weights of the nodes of each cluster. The process is repeated until the clusters do not change any more, which usually happens after 6 to 7 iterations. At the end, cluster C_1 contains the significant vertices of the augmented silhouette. This process does not require any parameter and leads to better results than using a simple threshold.

Note that a node $x \in X$ may be classified differently in each of the K' augmented silhouettes: either as significant, as insignificant, or not classified at all, if it has less than two neighbours.

As we compute N angles for each pair of adjacent edges in each augmented silhouette, the overall complexity of computing all deformation weights is on the order of $K'N\sqrt{n} < K^2N\sqrt{n}$, which is significantly less than the $O(N^2n)$ operations needed in the algorithm of Wuhrer and Brunton to compute the weights for the edges of the dual graph. The additional computational cost for the clustering processes is negligible, as it is only on the order of $K'\sqrt{n}$.

3.3 2D surface clustering

The final step takes into account the analysis of all the K' augmented silhouettes from the previous step. For each node $x \in X$, we count the number of times that it has been classified as significant and call this number the *saliency* of the node. In this way, we can identify the nodes in X that correspond to the non-rigid joint regions of the shape by simply thresholding on the saliency and selecting the nodes that appear to be significant in at least a certain percentage of all augmented silhouettes, usually 10% to 20%. Thresholding has the effect of smoothing out noise which is introduced by poor silhouettes: a high threshold leads to less salient vertices and

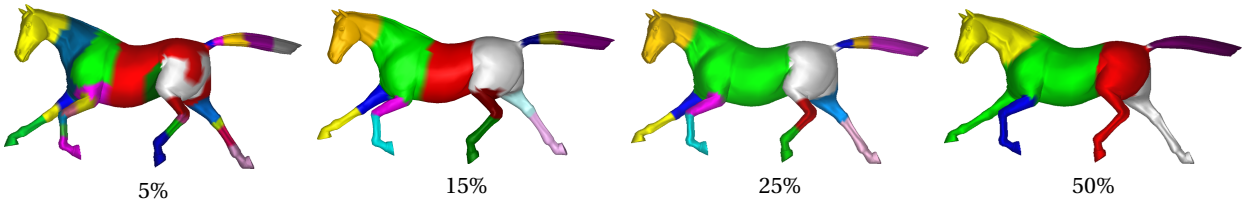


Figure 4: Segmentations obtained with different values of the saliency threshold.

fewer segments, while a small threshold leads to more salient vertices and therefore to over-segmentation; see Figure 4.

The final segmentation is found by first computing the distances of all nodes to these selected nodes and then growing clusters from the local maxima of this distance function.

To determine the distance between two nodes x and x', we consider the corresponding 3D vertices v and v' in one of the meshes and compute the *diffusion distance* [3, 7]

$$d(x,y)=\sum_{i>0} f(\lambda_i)\big(\phi_i(x)-\phi_i(y)\big)^2, \tag{1}$$

where $f(\lambda)$ is some *filter* function, ϕ_i are the eigenfunctions, and λ_i the eigenvalues of the discretized Laplace-Beltrami operator Δ. In our experiments, we use $f(\lambda)=e^{-2\lambda t}$, with good results, but other choices are possible. For example, using $f(\lambda)=\lambda^{-1}$ or $f(\lambda)=\lambda^{-2}$ in (1) would give the commute time distance, or the biharmonic distance [17], respectively. As this distance is intrinsic and invariant to isometric shape deformations, it does not matter which of the N meshes we consider for the actual computation.

In our experiments, we use the *cotangent weight* discretization [20] of the form $\Delta f = A^{-1}Wf$, where f is a function defined on the vertices of the mesh, represented by a vector of size N, A is a diagonal matrix of size $N\times N$ that contains the area elements of each vertex, and W is a $N\times N$ zero-mean matrix with elements

$$w_{ij}=\begin{cases} -(\cot\beta_{ij}+\cot\gamma_{ij})/2, & (i,j)\in E,\\ -\sum_{k\neq i} w_{ik}, & i=j,\\ 0, & \text{otherwise.}\end{cases}$$

Here, β_{ij} and γ_{ij} denote the angles opposite the edge $[v_i,v_j]$ in the two triangles sharing this edge. The eigenfunctions and eigenvalues of Δ are found by solving the generalized eigendecomposition problem $W\phi_i=\lambda_i A\phi_i$. The parameter t controls the scale of the diffusion distance and is selected based on the shape diameter. Since the coefficients $e^{-2\lambda_i t}$ decay fast, in practice it is enough to use the first 5 to 10 eigenpairs to accurately approximate the diffusion distance (1).

As can be seen in Figure 2 (fifth column), this isolates connected subsets of X that are characterized by similar distance values and correspond to the rigid parts of the shape. We then perform a region growing clustering in order to build the different segments of the shape, picking as seeds the local maxima of the distance function and adding all connected vertices with a distance value less than some threshold. The clustering stops when at least 80% of the vertices have been clustered. The remaining vertices are finally assigned to the nearest cluster. The resulting segmentation is shown in Figure 2 (sixth column).

The time complexity of this segmentation step is $O(n^2)$, due to the pairwise distance computation between vertices, which is quite heavy for large meshes. However, the following section explains how the runtime can be drastically reduced thanks to parallelization of the algorithm on the GPU.

4 Efficient implementation

Our algorithm is highly parallelizable and can be efficiently implemented on SIMD-type processors. In our implementation, we parallelize part of the algorithm on the GPU, using the CUDA architecture [21], while other parts are parallelized on multi-core CPUs.

In the first stage of the algorithm, the extraction of the silhouettes is parallelized: a number of parallel CUDA threads is launched to test the visibility of faces and edges of the mesh. Furthermore, each viewpoint can be processed independently of the others. In the second stage, the analysis of the curves can be distributed on multiple processing units (for example, multiple CPU cores), since each silhouette is examined separately from

	n	N	K	silhouette extraction	1D analysis	2D clustering	total
cat	7207	8	50	1.02	0.51	0.773	2.3
horse	8431	10	50	0.898	0.55	0.643	2.1
elephant	42321	10	100	7.602	4.081	4.58	16.3
armadillo	165954	5	125	79.36	32.446	32.149	144

Table 1: Different shapes used in our experiments and runtime (in sec) of different stages of the algorithm.

the others. In the third stage, the computation of the diffusion distance between a vertex v and all other vertices in V is executed in separate threads. We use this feature to speed up the computation of the minimum distance from the selected vertices of V: we compute the distance between each selected vertex and the other vertices in V and then pick, for each vertex, the minimum distance value. If s is the number of selected vertices, we need to update the minimum distance of each vertex s times, and the update is performed in parallel for each vertex in V. Using the same logic, we accelerate also the final part of the algorithm, that is, assigning significant vertices to the nearest cluster.

5 Results

In this section, we present several results of our algorithm and comparisons with other methods. The tests have been performed on an Intel QuadCore Q9550 2.83GHz with 4GB onboard memory and using CUDA parallelization on an NVIDIA GeForce GTX-260 graphics card. We use the LEMON graph library to store the augmented silhouettes. Shapes are provided by the Aim@Shape repository and by the Sumner shape dataset [25]. Execution times of the algorithm are summarized in Table 1.

5.1 Qualitative evaluation

Figure 5 shows a comparison between our method, the method of Rosman et al. [22], and the results from Wuhrer and Brunton [28], applied to the same input poses of the horse model with approximately 8K vertices. The segmentation is computed using 50 silhouettes, randomly picked from 10 poses. The segmentation is consistent with the shape articulations and correctly captures the rigid parts of the object, except for the hoofs. The boundaries of the rigid parts are generally smoother than the ones detected by the other algorithms. We also do not detect small patches, as Rosman et al. [22], since they do not correspond to any visually perceived motion.

Figure 6 shows the cat model, segmented using 50 different silhouettes and 8 input poses. In both tests, the runtime of the algorithm is slightly lower than [28], but the difference becomes more significant with the increase of the number of vertices of the input shapes, especially thanks to the massive parallelization. In the elephant model (see Figure 7) with approximately 40K vertices and segmented using 100 silhouettes, the legs

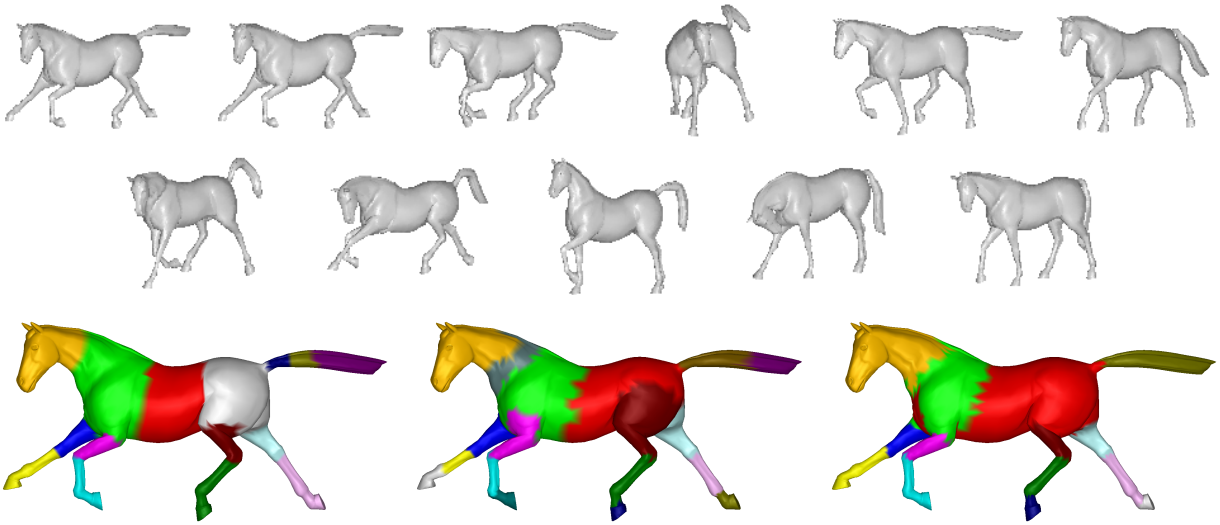


Figure 5: Visual comparison between our method (left), Rosman et al. [22] (middle), and Wuhrer and Brunton [28] (right).

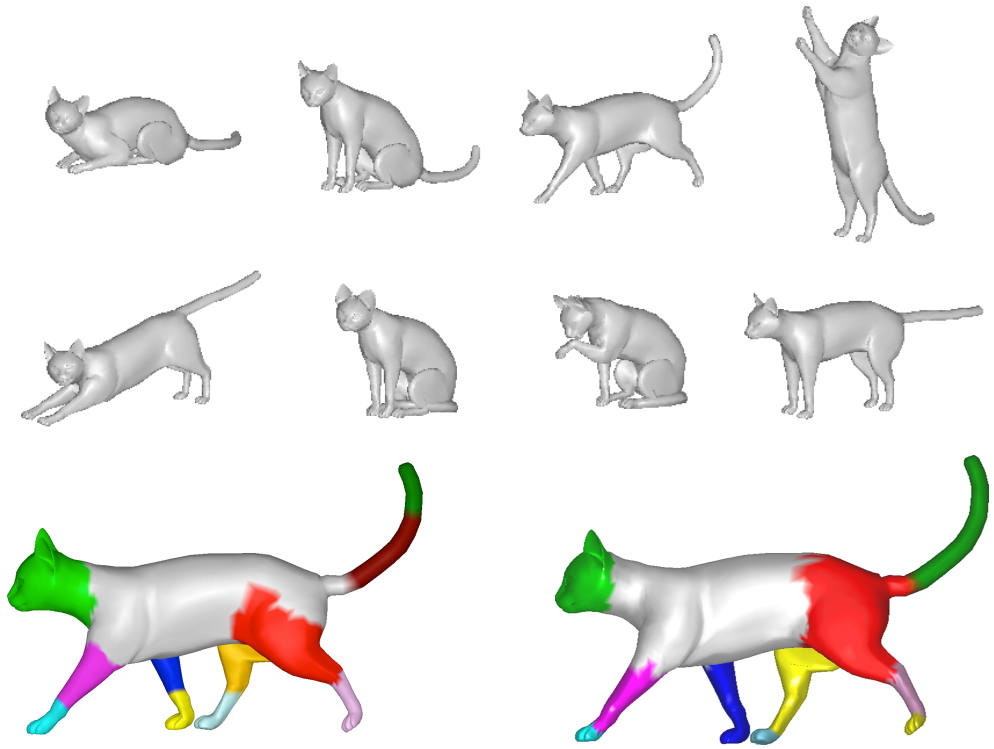


Figure 6: Segmentation of the cat shape using our method (left) and the one by Wuhrer and Brunton [28] (right).

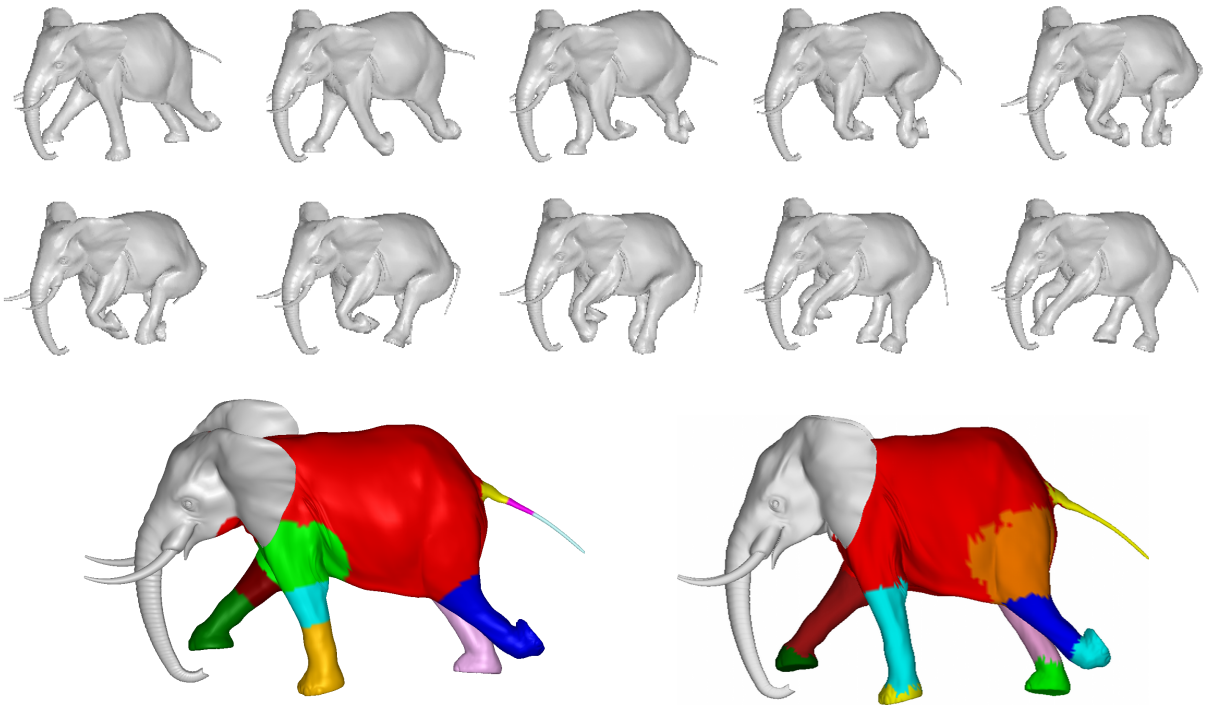


Figure 7: Segmentation of the galloping elephant shape using our method (left) and the one by Wuhrer and Brunton [28] (right).

are clearly separated from the rest of the body which is almost static. The motion of the front legs is visible and well captured, while the hind legs move rigidly, as also detected by Wuhrer and Brunton [28]. The runtime for

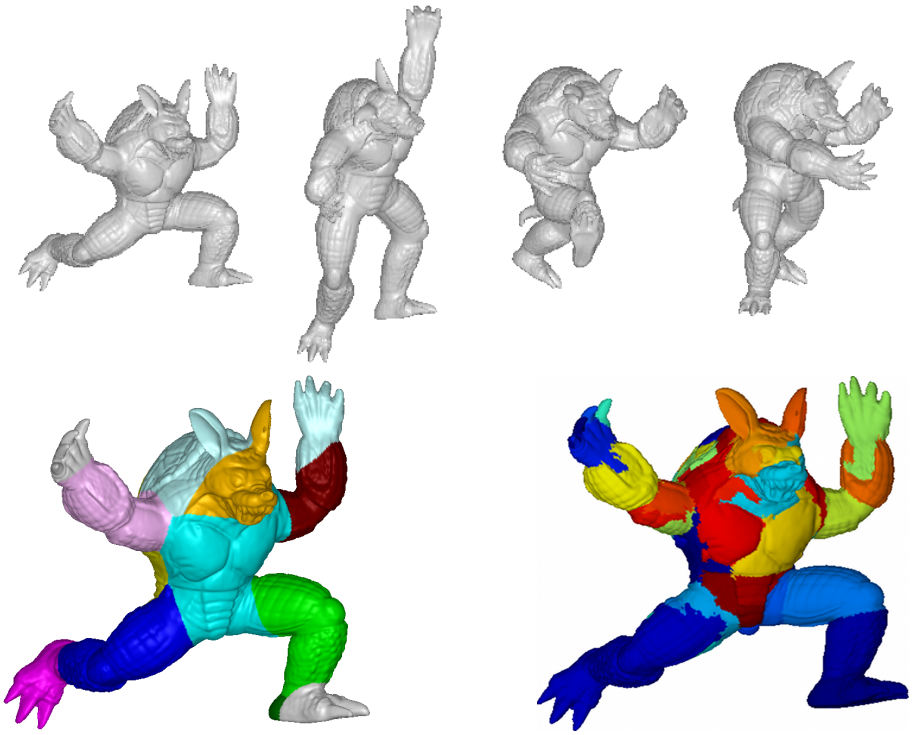


Figure 8: Segmentation of the armadillo shape using our method (left) and the one by Wuhrer and Brunton [28] (right).

this test is 16.3 secs, and thus significantly smaller than the 46 secs needed by Wuhrer and Brunton to obtain a comparable segmentation.

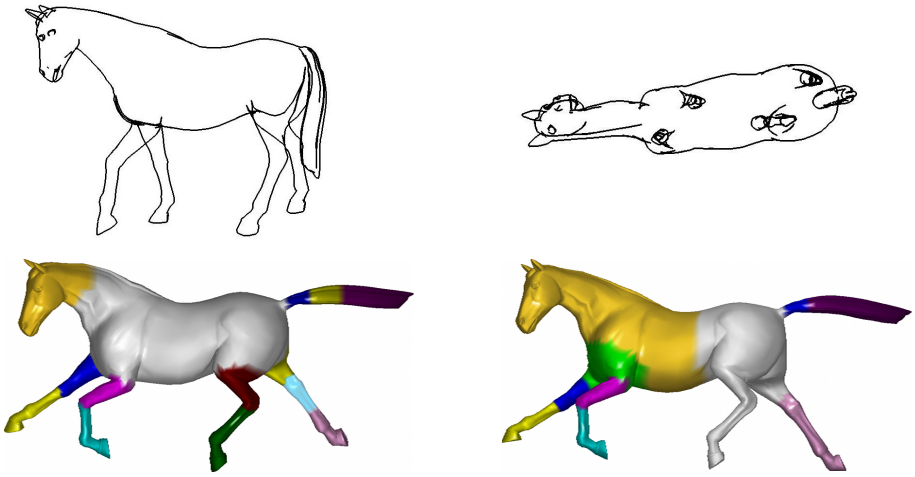
The largest test case is the armadillo model (see Figure 8) with 166K vertices. Segmentation is performed using 5 input poses and capturing 125 silhouettes. In this sequence, some of the rigid parts are not clearly detected. The motion of the left knee, for example, is not captured, because the surface deformation is distributed over a large number of vertices and we cannot distinguish clearly where the motion occurs during the analysis of the augmented silhouettes. A similar situation can be found for example in the horse case, where the deformation related to the hooves is distributed across different vertices. On the other hand, our approach is still able to capture the general structure of the object without resulting in over-segmentation as in the method of Wuhrer and Brunton [28] and it runs in 144 secs, compared to their 349 secs.

5.2 Quantitative evaluation

In order to provide a more objective comparison between our algorithm and the method by Wuhrer and Brunton [28] in terms of quality of the results, we follow the approach proposed in [5]. For each model we selected a sequence of poses and created some "ground-truth" by performing several, different, manual motion-based segmentations of the shapes. We then computed the *Hamming distance* between each ground-truth and a reference segmentation, and finally determined the average of the distances. Note that we adapted the formulation proposed in [5] to work with vertices. The evaluation results related to our method and [28] are presented in Table 2. As one can notice, our method has better or at least equal quality, with the advantage of being faster. However, none of the methods is able to achieve a satisfactory segmentation of the Armadillo model.

	our method	Wuhrer–Brunton [28]
cat	**0.16**	0.19
horse	**0.17**	0.19
elephant	0.13	0.13
armadillo	0.34	**0.28**

Table 2: Segmentation accuracy, measured as the Hamming distance from the ground-truth segmentation.



side (0.21) top (0.25)

Figure 9: Two segmentations of the horse (second row) from single silhouettes (first row).

5.3 Limitations

A significant limitation of the algorithm is related to the quality of each viewpoint, that is, the amount of deformation that can be seen from that specific viewpoint. Figure 9 shows two examples of a segmentation obtained from just one silhouette, one taken from the top of the object, and one from the side. The side view better captures the motion of legs and tail, so it is possible to approximatively detect each leg, the tail, and the rear part of the body from just one silhouette. However, the top view is not able to capture correctly every leg of the horse, and also detects a patch that does not correspond to a rigid part.

6 Conclusions

In this paper, we introduced a method for motion-based 3D shape segmentation. Our method is based on extracting 1D silhouette curves from the shape, segmenting them into rigid parts, and then merging the 1D segmentation information to obtain the shape segmentation. The method is computationally efficient and highly parallelizable on graphics hardware. Our experiments show a segmentation quality comparable with state-of-the-art methods at lower computational complexity.

However, several improvements are still possible. First, the algorithm will benefit from a smarter choice of the view points, in order to be able to capture the entire motion of the shape with only a minimal set of silhouettes. Alternatively, one could assign to each silhouette a weight, related to the motion captured. In this way, it is possible to discard redundant or noisy data. The algorithm should also be able to detect the deformation even if it is spread across a number of connected vertices instead of a single, deforming vertex. Finally, different distance measures, such as geodesics, could be used instead of the diffusion distance, especially if the computation becomes too expensive.

Acknowledgements

This work was supported by the SNF under project number 200021-134639 and by the 3D-COFORM consortium. The authors are grateful to G. Rosman and A. Brunton for providing comparison to their methods and would like to thank the anonymous reviewers for their valuable comments.

References

[1] E. J. Alexander, C. Bregler, and T. P. Andriacchi. Non-rigid modeling of body segments for improved skeletal motion estimation. *Computer Modeling in Engineering & Sciences*, 4(3):351–364, 2003.

[2] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, Mar. 2006.

[3] M. M. Bronstein and A. M. Bronstein. Shape recognition with spectral distances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):1065–1071, May 2011.

[4] A. B. Carman and P. D. Milburn. Determining rigid body transformation parameters from ill-conditioned spatial marker co-ordinates. *Journal of Biomechanics*, 39(10):1778–1786, 2006.

[5] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3):Article 73, 12 pages, Aug. 2009. Proceedings of SIGGRAPH.

[6] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, Aug. 2004. Proceedings of SIGGRAPH.

[7] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426–7431, May 2005.

[8] A. I. Comport, É. Marchand, and F. Chaumette. Complex articulated object tracking. In F. J. Perales and B. A. Draper, editors, *Articulated Motion and Deformable Objects*, volume 3179 of *Lecture Notes in Computer Science*, pages 189–201. Springer, 2004.

[9] E. de Aguiar, C. Theobalt, S. Thrun, and H.-P. Seidel. Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum*, 27(2):389–397, Apr. 2008. Proceedings of Eurographics.

[10] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 49–58, 2001.

[11] S. Hauberg, S. Sommer, and K. S. Pedersen. Gaussian-like spatial priors for articulated tracking. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6311 of *Lecture Notes in Computer Science*, pages 425–437. Springer, 2010.

[12] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics*, 24(3):399–407, July 2005. Proceedings of SIGGRAPH.

[13] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, July 2003. Proceedings of SIGGRAPH.

[14] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Rapid and effective segmentation of 3D models using random walks. *Computer Aided Geometric Design*, 26(6):665–679, Aug. 2009.

[15] T.-Y. Lee, Y.-S. Wang, and T.-G. Chen. Segmenting a deforming mesh into near-rigid components. *The Visual Computer*, 22(9):729–739, Sept. 2006. Proceedings of Pacific Graphics.

[16] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design*, 22(5):444–465, July 2005.

[17] Y. Lipman, R. M. Rustamov, and T. A. Funkhouser. Biharmonic distance. *ACM Transactions on Graphics*, 29(3):Article 27, 11 pages, June 2010.

[18] D. Marr. *Vision*. MIT Press, 2010.

[19] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *Proceedings of Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[20] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, Mathematics and Visualization, pages 35–57. Springer, 2003.

[21] nvidia. *NVIDIA CUDA Compute Unified Device Architecture – Programming Guide, Version 1.0*, 2007.

[22] G. Rosman, M. M. Bronstein, A. M. Bronstein, A. Wolf, and R. Kimmel. Group-valued regularization framework for motion segmentation of dynamic non-rigid shapes. In A. M. Bruckstein, B. M. t. Haar Romeny, A. M. Bronstein, and M. M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 725–736. Springer, 2011.

[23] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 406–420. Springer, 2010.

[24] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, Sept. 2008.

[25] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, Aug. 2004. Proceedings of SIGGRAPH.

[26] S. Ullman. Filling-in the gaps: The shape of subjective contours and a model for their generation. *Biological Cybernetics*, 25(1):1–6, Mar. 1976.

[27] S. Ullman. On visual detection of light sources. *Biological Cybernetics*, 21(4):205–211, Dec. 1976.

[28] S. Wuhrer and A. Brunton. Segmenting animated objects into near-rigid components. *The Visual Computer*, 26(2):147–155, Feb. 2010.