Eurographics Workshop on 3D Object Retrieval (2012)
M. Spagnuolo, M. Bronstein, A. Bronstein, and A. Ferreira (Editors)

Kinematic skeleton extraction based on motion boundaries for 3D dynamic meshes

Halim Benhabiles[1], Guillaume Lavoué[2], Jean-Philippe Vandeborre[3,4] and Mohamed Daoudi[3,4]

[1]Le2i UMR CNRS 6303, University of Bourgogne, France
[2]Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
[3]LIFL (UMR Lille1/CNRS 8022), University of Lille 1, France
[4]Institut TELECOM; TELECOM Lille 1, France

Abstract

This paper presents a precise kinematic skeleton extraction method for 3D dynamic meshes. Contrary to previous methods, our method is based on the computation of motion boundaries instead of detecting object parts characterized by rigid transformations. Thanks to a learned boundary edge function, we are able to compute efficiently a set of motion boundaries which in fact correspond to all possible articulations of the 3D object. Moreover, the boundaries are detected even if the parts linked to an object's articulation are immobile over time. The different boundaries are then used to extract the kinematic skeleton.
Experiments show that our algorithm produces more precise skeletons compared to previous methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling —Curve, surface, solid, and object representations

1. Introduction

With the recent progress of acquisition systems, and the increasing of the processor calculation power, the use of dynamic surfaces in the multimedia domain has become important. This kind of data can be created by a designer using animation softwares, or obtained from a scanner, or a scientific simulation. Generally, the dynamic surface is represented as a sequence of 3D meshes (ie. a sequence of frames) with constant connectivity, and time-varying geometry (the position of vertices changes over time). Similarly to static meshes, dynamic meshes require to be pre-processed before being used in a given application. They mainly need a structural representation such as a skeleton which is a key ingredient for their modeling and editing. Few existing works in the literature propose to extract a kinematic skeleton for these dynamic meshes [dATTS08, SY07, AKcPT04]. They basically make use of motion-based geometric segmentation methods [ABH*10,LWC06]. These methods seek to decompose the dynamic mesh into rigid parts by exploiting the temporal information. They assume that the vertices of such parts are characterized by a uniform motion with a single rigid transformation along the sequence. Once the parts are determined, an articulated skeleton is computed.

The main drawback of this kind of methods is the fact that they are limited to exploit only the temporal information for extracting the different parts of the shape. Indeed, such method cannot be suited to detect the relevant (ie. functional) parts of the shape which are immobile along the sequence of meshes. Thus, using the resulting computed skeleton from this kind of methods will not allow to apply any deformation on these latter parts for instance. Tierny *et al.* [TVD08] proposed to represent the 3D object by a set of level lines (or contours) and then keep only those that maximize an edge-length deviation function over time. This latter function allows to capture the articulations of the object. Indeed, the edges located at these articulations undergo a significant variation (edges are stretched) during the movement of the parts linked to them. However, once again this function fails to capture the articulations when the parts linked to them are immobile over time.

In this paper we propose a precise kinematic skeleton extraction method for dynamic meshes. The method is based on computing a set of motion boundaries, that correspond to the articulations of the object, for the whole sequence of meshes. To this end, a set of interest regions is computed independently for each frame of the sequence. The calcu-

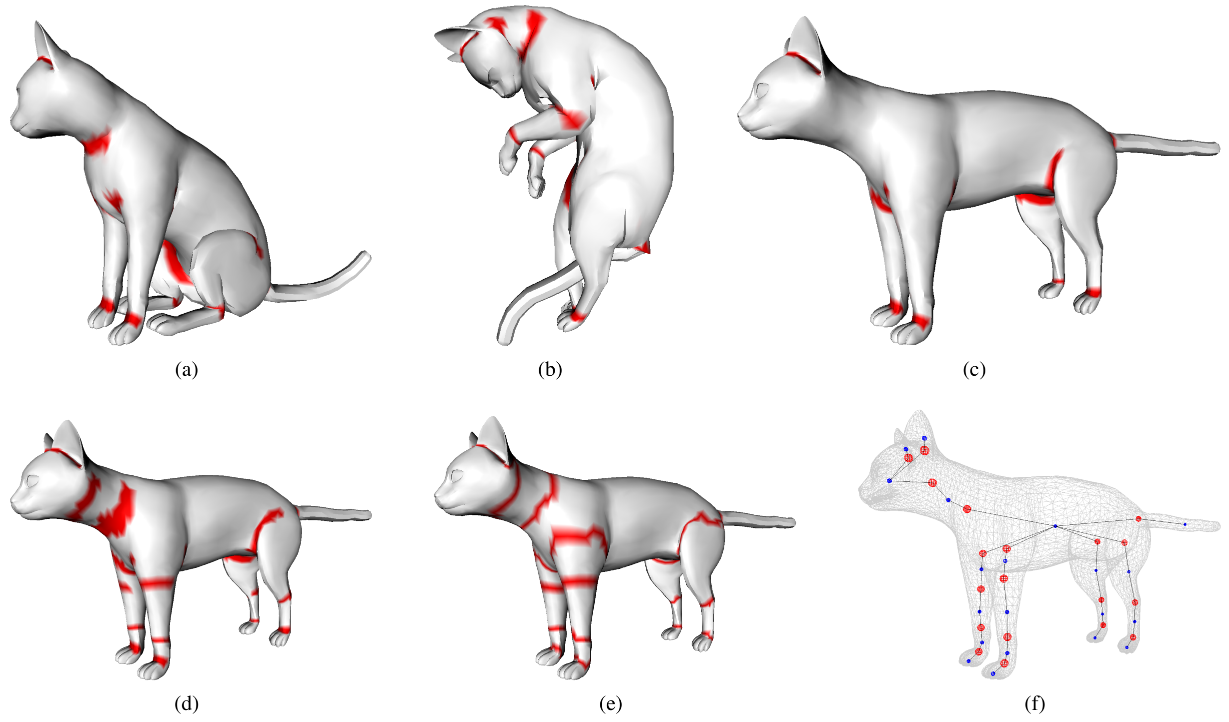

Figure 1: *Overview of our kinematic skeleton extraction method: extracting interest regions for each frame (a,b,c), merging the regions (d), computing boundaries (e), computing skeleton (f).*

lation of these regions is based on the geometry. Then, all the regions computed over the sequence are gathered into one single set to create a unique segmentation for the whole dynamic mesh. To summarize, we obtain a unique segmentation for the whole dynamic mesh that is based both on motion and geometric features. Finally, the skeleton is extracted from this latter segmentation. The main contribution of this work is the following. We present a hybrid technique which combines both static and dynamic information to identify all possible articulations of the object even if its parts are immobile along the sequence of meshes. Moreover, with this approach, the computed motion boundaries (and thus the skeleton nodes) are very precise.

2. Overview of the approach

Given an input dynamic mesh, our method begins by extracting a set of interest regions along the different frames (in red, in figure 1(a,b,c)) and then merging them (figure 1(d)). This step is described in section 3. The set of extracted interest regions along the sequence provides an information about all possible articulations of the object. Next, the interest regions are transformed into thin, closed and smooth contours (figure 1(e)) using a post-processing pipeline described in section 4. Finally, these contours are used to compute the skeleton (figure 1(f)). This latter step is described in section 5.

3. Extracting and merging interest regions

To extract the set of interest regions on a given frame of the sequence, we use a boundary edge function defined in [BLVD11]. This function is a weighted combination of a set of geometric criteria, which is learned in an offline step with the AdaBoost [FS97] classifier using a ground-truth of manually segmented models. This framework was recently introduced in [BLVD11] for segmentation of static meshes. The classifier takes as input a training dataset (we used the princeton segmentation benchmark [CGF09]) and generates the boundary edge function. The training dataset is composed of a set of feature vectors F_E computed for each edge of the ground-truth of static meshes. A feature vector F_E of a given edge contains a set of 33 geometric criteria such as curvature [KvD92], shape diameter function [SSCO08], and is associated with its proper class label L so that $L = +1$ if the edge is a boundary (according to the manual segmentations of the mesh containing this edge) and $L = -1$ if the edge is not a boundary. Once the learning is done, the classifier produces the boundary edge function. This function is a weighted combination of the set of geometric criteria. It takes as input a feature vector from any given edge and outputs a signed scalar value whose sign will provide the estimated classification of the edge (positive for boundary and

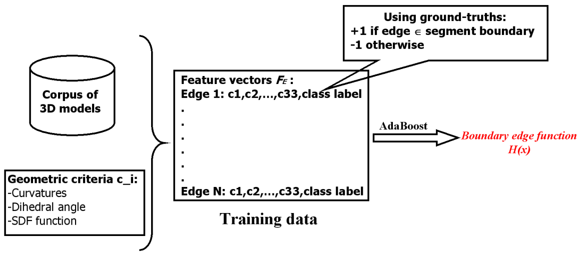

Figure 2: *Learning the boundary edge function.*

negative for non-boundary). Figure 2 illustrates the learning step for the edge function construction.

The use of the boundary edge function on each frame of the dynamic mesh leads to produce a set of regions (figure 1(a,b,c)), each of which is represented by a set of connected edges of the mesh. These regions, only computed using geometry, all correspond to possible articulations of the object. This result was expected since the boundary function is learned from human manual segmentations. Indeed people tend to segment an object by defining boundaries that separate its different functional parts [HS97]. However, if we consider only one given frame, we can notice that we never detect all possible articulations, indeed it depends on the pose of the animated object on the frame. Hence, we propose to merge all regions along the sequence, in order to cover all possible articulations that appear over time. In other words, this merging allows to exploit the temporal information. Note that the merging operation makes use of the *OR* operator applied on the boundary edge function from the different frames, and does not require any processing since the dynamic mesh is represented by a sequence of meshes with constant connectivity (vertex to vertex correspondence).

4. Post-processing pipeline

The extracted interest regions cannot be used directly for computing the skeleton. Consequently, they need to be processed. To this end, we use the processing pipeline introduced for static mesh segmentation in [BLVD11] that comprises three stages. In the first stage, for each interest region (a set of connected edges), a thinning algorithm is applied. This latter algorithm allows to thin the interest region to a piecewise linear contour. Next, each open contour is completed to form a closed boundary around a specific part of the object (figure 1(e)). The principle is to find the weighted shortest path between the two endpoints of the contour while minimizing a cost function based on the boundary edge function. At this step we have created a set of closed contours. Finally, each contour is optimized, in term of smoothness and precision, using a snake movement based also on the boundary edge function. In other terms the contours are enforced to be closed as much as possible to the articulations of the object.

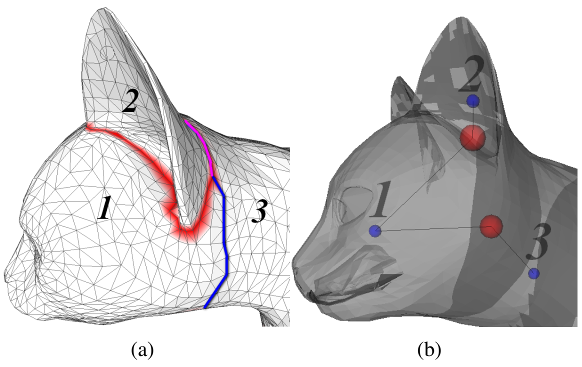

Figure 3: *Example of three adjacent segments (a), and the resulting skeleton (b).*

5. Skeleton computation

To compute the skeleton, we use a simple algorithm which takes as input the dynamic mesh together with the set of closed boundaries and gives as output a structure composed of a set of edges and points that represents the kinematic skeleton (figure 1(f)).

The algorithm begins by computing the centroids of both boundaries and segments (ie. regions separated by the boundaries), then it connects each boundary centroid (red points in figure 1(f)) with its two adjacent segment centroids (blue points in figure 1(f)). Two segments are adjacent if they share the same boundary edges or a part of them. However, it is possible that a part of a given boundary be shared between more than two segments. Figure 3(a) illustrates an example in which three segments are adjacent since they share a common part of their boundaries (see the magenta part in the figure). In this latter case, each boundary is connected with the two segments that share its maximum common part as illustrated in figure 3(b).

6. Experiments and results

Figure 4 shows some dynamic surfaces and their extracted kinematic skeletons. One can notice that the obtained skeletons describe efficiently the dynamic meshes thanks to the motion boundaries (in red). Indeed, each boundary correspond to an articulation of the object. Moreover, the rigid parts are easily captured since they are limited by motion boundaries. Consequently, these rigid parts can undergo motions by simple rotations, and thus edit the dynamic shape without any difficulty.

Figure 5 shows a visual comparison between our algorithm and those from the state-of-the-art applied on the horse model. The figure clearly shows that the algorithm from Lee *et al.* [LWC06] fails to detect the different articulations of the legs although these latter ones are characterized by motions. The algorithm from Aguira *et al.* [dATTS08] overcomes this

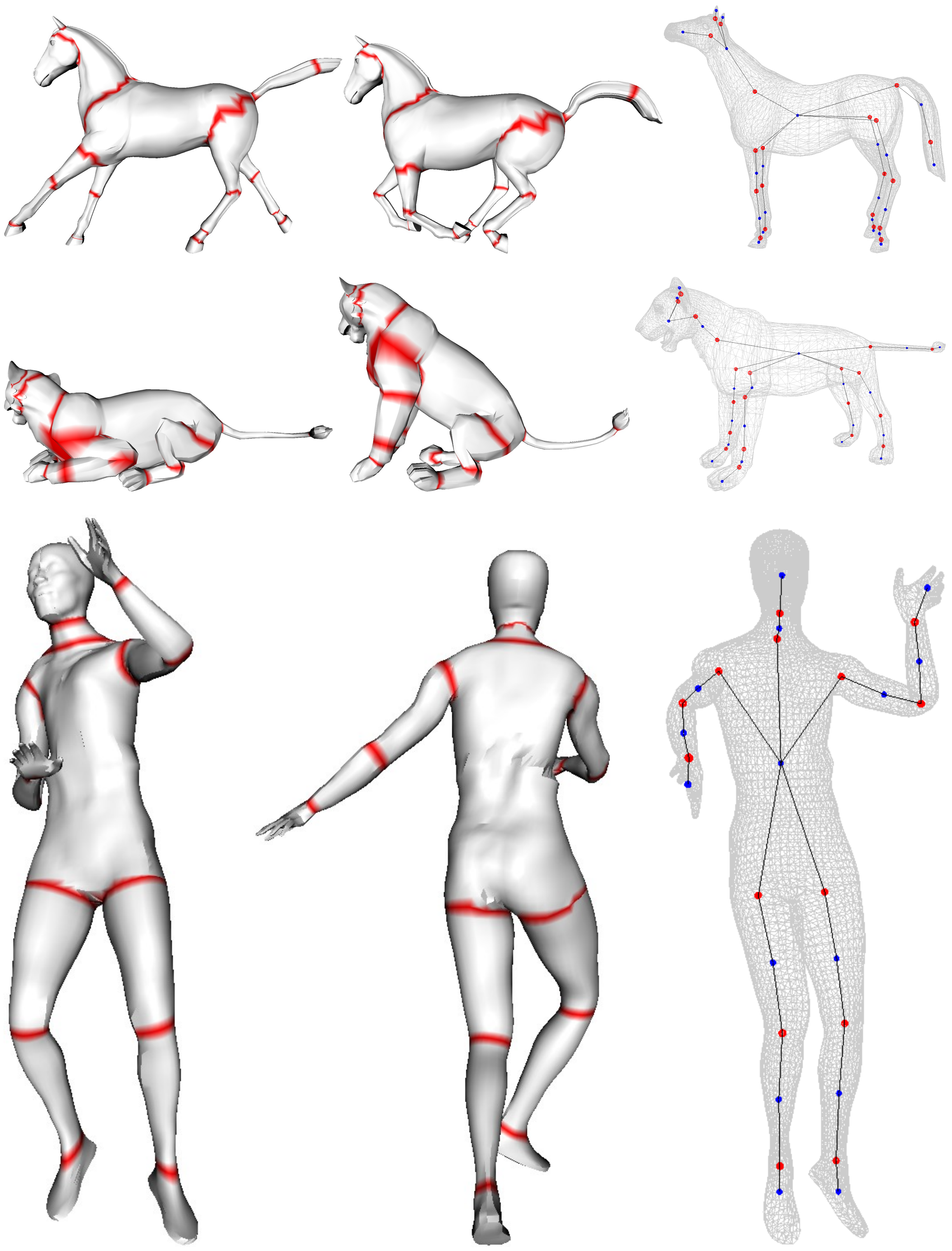

Figure 4: *For each row, dynamic surfaces and their corresponding kinematic skeletons.*

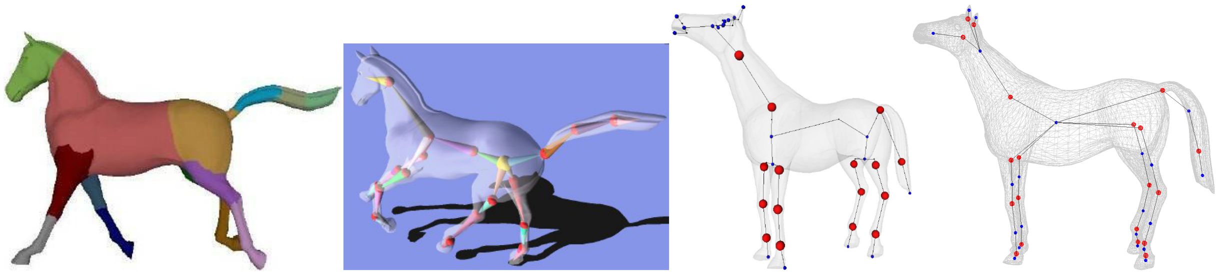

Figure 5: *From left to right: segmentation obtained by Lee* et al. *[LWC06], skeleton obtained by: Aguira* et al. *[dATTS08], Tierny* et al. *[TVD08], and our algorithm.*

drawback, but it fails to detect the articulations of immobile parts of the model such as the ears. The results obtained by the algorithm from Tierny *et al.* [TVD08] and our algorithm are better since they detect both kinds of articulations (mobile and immobile). However, our skeleton is slightly more accurate in terms of position of motion nodes (red points) thanks to the boundary edge function.

The whole process for computing the kinematic skeleton runs at reasonable time (in seconds, and only few minutes in the worst case). Table 1 summarizes the running time for all models presented in this paper. The experiments were carried out on a 3 GHz Intel(R) Core(TM) 2 Duo CPU with 4 Gb memory. One can notice that the running time is more important when the number of frames increases. This is due to the fact that the edge function is computed for all edges of each frame. However, the computation process is done independently from a frame to another. Moreover, it is done independently from an edge to another. Thus, the process can be easily parallelized which will decrease the running time considerably.

Dynamic-mesh	Vertices	Frames	Skeleton (s.)
Cat	7207	10	37
Dance	7061	201	411
Horse	8431	49	204
Lion	7207	10	28

Table 1: *Computation time for kinematic skeleton extraction of some dynamic meshes.*

7. Conclusion

In this paper, we presented an automatic method that allows to extract kinematic skeletons for dynamic surfaces. The method is based on motion boundary extraction. We have shown through some experiments that our method allows to extract precise skeleton represented by a set of motion nodes that correspond to all articulations of the object. Morover, the skeleton includes rigid nodes which correspond to object's parts characterized by uniform motion. Hence the dynamic object can be edited easily by applying simple rotations on these latter nodes.

For future work, we would like to generalize our method for dynamic shapes with variable connectivity. In this latter case, merging the interest regions extracted along the sequence of a given shape is not obvious. Indeed, we need to pre-process the sequence in such way that each frame has a vertex to vertex correspondence with the next one.

ACKNOWLEDGMENTS

This work has been supported by the ANR (Agence Nationale de la Recherche, France) through MADRAS project (ANR-07-MDCO-015).

References

[ABH*10] ARCILA R., BUDDHA S. K., HÉTROY F., DENIS F., DUPONT F.: A framework for motion-based mesh sequence segmentation. In *International Conference on Computer Graphics, Visualization and Computer Vision, WSCG* (2010). 1

[AKcPT04] ANGUELOV D., KOLLERHOI-CHEUNG D., PRAVEEN P., THRUN S. S.: Recovering articulated object models from 3d range data. In *20th Conference on Uncertainty in Artificial Intelligence* (2004), pp. 18–26. 1

[BLVD11] BENHABILES H., LAVOUÉ G., VANDEBORRE J.-P., DAOUDI M.: Learning boundary edges for 3d-mesh segmentation. *Computer Graphics Forum - Eurographics Association 30*, 8 (2011), 2170–2182. 2, 3

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3d mesh segmentation. *ACM Transactions on Graphics (SIGGRAPH) 28(3)* (2009). 2

[dATTS08] DE AGUIAR E., THEOBALT C., THRUN S., SEIDEL H.-P.: Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum - Eurographics Association 27*, 2 (2008), 389–397. 1, 3, 5

[FS97] FREUND Y., SCHAPIRE R. E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*, 1 (1997), 119–139. 2

[HS97] HAUFFMAN D., SINGH M.: Salience of visual parts. *Cognition 63* (1997), 29–78. 3

[KvD92] KOENDERINK J. J., VAN DOORN A. J.: Surface shape and curvature scales. *Image Vision Comput. 8(10)* (1992), 557–565. 2

[LWC06] LEE T.-Y., WANG Y.-S., CHEN T.-G.: Segmenting a deforming mesh into near-rigid components. *Visual Computer 22* (2006), 729–739. 1, 3, 5

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *Visual Computer 24*, 4 (2008), 249–259. 2

[SY07] SCHAEFER S., YUKSEL C.: Example-based skeleton extraction. In *Proceedings of the fifth Eurographics Symposium on Geometry Processing* (2007), pp. 153–162. 1

[TVD08] TIERNY J., VANDEBORRE J.-P., DAOUDI M.: Fast and precise kinematic skeleton extraction of 3d dynamic meshes. In *19th IEEE International Conference on Pattern Recognition (ICPR-2008)* (2008), pp. 1–4. 1, 5