

# Skeleton Extraction from a Mesh for Easy Skinning Animation

Martin Madaras  
KAI, Fac. of Mathematics  
Physics and Informatics  
Comenius University  
Mlynska dolina, 842-48  
Bratislava, Slovakia

Roman Ďurikovič  
KAI, Fac. of Mathematics  
Physics and Informatics  
Comenius University  
Mlynska dolina, 842-48  
Bratislava, Slovakia  
durikovic@fmph.uniba.sk

Tomáš Ágošton  
KAI, Fac. of Mathematics  
Physics and Informatics  
Comenius University  
Mlynska dolina, 842-48  
Bratislava, Slovakia

Tomoyuki Nishita  
Graduate School of Frontier  
Science  
The University of Tokyo  
5-1-5 Kashiwanoha,  
Kashiwa-shi, Chiba 277-8561,  
Japan  
nis@is.s.u-tokyo.ac.jp

## ABSTRACT

This paper proposes the extraction of a skeleton and skinning weights from a given mesh, describes how to store computed data in Collada 1.5 and use it for an animation. Firstly, the mesh is contracted using constrained Laplacian smoothing in a few iterations. Few vertices from the contracted mesh are chosen as control points. Multiple edges are removed and vertices that are very close to each other are merged using a greedy algorithm minimizing the energy. The greedy selection is applied repeatable until we have the tree structure with edges corresponding to bones. We also propose the automatic assignment of the skinning weights that determine the rigid or soft mesh deformations. In the postprocessing stage the user can inspect the skeleton by previewing skinning deformations, make desired changes and export the skeleton to Collada 1.5. Transformation matrices used in a hierarchical skeleton tree are not transformed to joint's local transformation frame, so they are immediately compatible with majority of animation software and libraries.

## Categories and Subject Descriptors

I.3.5 [Computational Geometry and Object Modeling]: Boundary representations; I.3.7 [Three-Dimensional Graphics and Realism]: Animation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HC'10, Dec. 8 – 10, 2010, Aizu-Wakamatsu, Japan.  
Copyright 2010 University of Aizu Press.

## General Terms

Theory, Algorithms

## Keywords

skeleton extraction, mesh contraction, Collada 1.5, mesh skinning

## 1. INTRODUCTION

A frequently used approach for animation and modification of 3D models is based on creating articulated hierarchical structures - skeletons. Skinning data as a skeleton tree and weights can be either assigned manually or computed from an input mesh. The first option is most often chosen by artists, although sometimes it is unnecessary and time consuming. The skeleton has to be created (or imported from templates), rigged into the mesh and influence weights have to be set. Skilled artists are able to create and rig the skeleton in a short time, but sometimes they have to make a lot of rigging adjustments during the skinning process. In this paper we present how to automatically compute the hierarchical skeleton and skinning weights from an input mesh and use them in a skinning animation using Collada 1.5 as export format between our application and a graphic animation software such as 3D Studio Max, Blender or Maya. Our application also provides a way how to examine the computed skeleton before exporting. The skeleton can be inspected by applying skinning deformations using direct kinematics. The interface also allows to dynamically add or remove a bone or a branching if the user thinks some changes are needed.

## 2. RELATED WORK

Numbers of algorithms have been proposed to compute a skeleton from the mesh geometry. There are three main groups of algorithms: volumetric methods, example based methods and geometric methods. In this paper we focus on

geometric methods, they are the most suitable for meshes, because there is no conversion needed. The geometric methods work directly on polygon meshes. The most widely used geometric methods are Reeb graph based methods [2], Voronoi diagram based [6] and Laplacian smoothing based methods [1].

Laplacian smoothing based methods work directly on the mesh geometry. The main idea of this approach is to apply a well defined filter on mesh vertices. These methods solve the Laplacian system with different weights to constrain the global smoothness and the volume preservation.

## 2.1 Skinning

Many types of mesh deformations can be performed by a skeleton-driven deformation. However, there are types of mesh deformations such as wrinkles, skin folds and another non-bone-driven deformations, where skeleton-driven deformation is not a sufficient option. Examples of non-bone-driven deformation methods are surface based methods [9, 11] and volume based methods [12]. Unfortunately, these methods are not suitable for a real time animation of high resolution meshes in present. Because of its efficiency and simple GPU implementation the most popular skeleton-driven method still remains linear blend skinning (LBS), also known as skeleton subspace deformation. Some real time skinning works have focused on improving the LBS by inferring the character articulation from multiple meshes.

A few solutions to the problem of finding skinning weights were proposed [3], but the methods are either resolution dependent [8] or the weights do not vary smoothly along the mesh [10], causing artifacts with high resolution models.

## 3. MESH CONTRACTION

For contraction of the generated mesh graph we use the contraction algorithm using Laplacian smoothing proposed by [1]. The algorithm does not alter geometry connectivity (final skeleton curve is homotopic to the original mesh), is noise sensitive and works directly on the mesh geometry (the model does not have to be resampled). Geometry contraction removes details from the surface by applying Laplacian smoothing.

### 3.1 Laplacian smoothing

Vertex positions are smoothly contracted along their normals by solving the equation (2). The Laplacian smoothing operator (1) was introduced by [5] for surface smoothing. Laplacian smoothing operator  $L$  is  $n \times n$  square matrix. This operator is applied on  $n$  vertices in vector  $V$  as a filter. Term  $LV$  approximates curvature flow normals, so solving  $LV' = 0$  removes normal components of vertices and contracts the geometry, resulting into a new set of vertices  $V'$ .

$$L_{ij} = \begin{cases} w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} & \text{if } (i, j) \in E \\ \sum_{(i,k) \in E}^k -w_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where:

$E$  – is the set of edges defined in the previous section during a graph conversion process

$\alpha_{ij}, \beta_{ij}$  – are the opposite angles corresponding to the edge  $(i, j)$  [5].

### 3.1.1 Linear equation

Unconstrained solving of this equation contracts the mesh graph into a single point, so the equation is solved in more iterations with carefully chosen weights which control the contractions.  $W_L$  and  $W_H$  are diagonal weighting matrices which control the contraction process. Weighting matrices have to be updated after each iteration to drive the iteration process into a desired state. By increasing  $W_{L,i}$  we can increase the collapsing speed for vertex  $i$  and by increasing  $W_{H,i}$  we increase the attraction weight to attract vertex  $i$  to its current position. All the  $W_{L,i}$  are in the next step multiplied by a predefined constant ( $s_L$ ) and  $W_{H,i}$  are updated in such a way, that the attraction weight is multiplied by a ratio of the change of the area of faces adjacent to the vertex  $i$ . If the area of adjacent faces surrounding vertex  $i$  is smaller, the multiplicative term is higher. Thus, the vertex  $i$  is more attracted into its current position and in the next iteration the geometry is less contracted in the vertex  $i$ .

$$\begin{bmatrix} W_L L \\ W_H \end{bmatrix} V' = \begin{bmatrix} 0 \\ W_H V \end{bmatrix} \quad (2)$$

The iteration process converges when the volume is close to zero. After each iteration, the volume approximation has to be computed. In our implementation we used an approximation algorithm which subdivides the bounding box of the model into an octree structure.

## 4. SKELETON CONSTRUCTION

The contracted mesh graph from the last iteration is simplified, very close vertices are merged and a greedy algorithm is used to select the most important control points. For each control point, the volume of the mesh region the control point represents in the original mesh is computed. The control point which represents the largest volume is chosen as the skeleton root. For more detail refer to [1].

## 5. SKINNING WEIGHTS

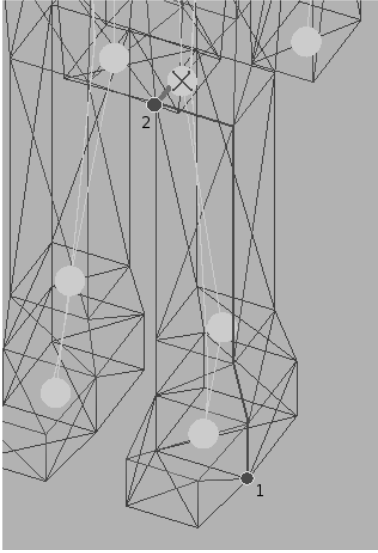
Skinning indices are computed by finding a set of closest control points to each vertex. The geodesic distance is used as a distance measure. A distance between each pair of vertices on the mesh graph from 0th iteration (after conversion from an input mesh) is calculated and stored in matrix  $D$ . It is calculated using Floyd-Warshall algorithm [4], before the mesh graph is contracted. For each control point  $C_k$ , the closest mesh graph vertex is found, for instance  $v_j$ . Then, the resulting geodesic distance (3) between the control point  $C_k$  and the mesh vertex  $v_i$  is computed as a sum of distance between  $v_j$  and  $v_i$  on the mesh graph calculated by Floyd-Warshall algorithm and the euclidean distance between  $C_k$  and  $v_j$ .

$$\text{gd}(i, k) = D[i, j] + d(C_k, v_j) \quad (3)$$

where:

$d(C_i, v_k)$  – is euclidean distance between the mesh vertex  $v_j$  and  $k^{\text{th}}$  control point  $C_k$

Weights (4) are assigned in a way that weight sum for each vertex is equal to 1.0. Fractions are constructed in a way that weights are indirectly dependent on the geodesic distance. This construction guarantees that the closer control points will have greater influence over mesh vertices than the further ones. The geodesic distance is a real-value function



**Figure 1:** This figure shows computation of the geodesic distance between the control point marked with the cross and the mesh vertex (1) on the bottom right. The path between vertices (1) and (2) is the shortest path on the mesh calculated by Floyd-Warshall algorithm and the line between vertex (2) and the cross is the euclidean distance.

defined on the mesh surface. Because the function varies smoothly along the mesh, the resulting weights are fluently distributed over the mesh regions.

$$\text{weight}(i, k) = \frac{1}{\text{gd}(i, k)} \sum_{k' \in S} \left( \frac{1}{\text{gd}(i, k')} \right) \quad (4)$$

where:

$\text{gd}(i, k)$  – is geodesic distance between the mesh vertex  $v_i$  and  $k^{\text{th}}$  control point  $C_k$

$S$  – is the set of control point indices controlling the vertex  $v_i$

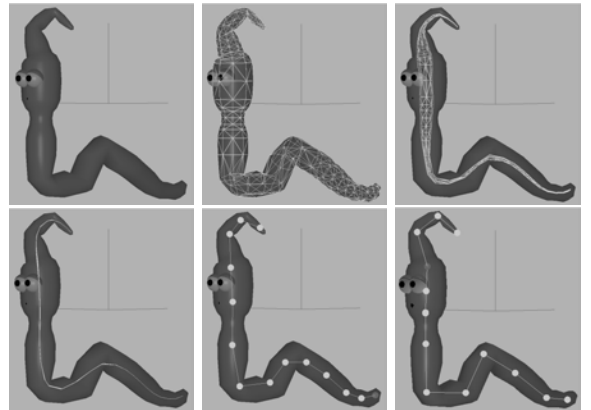
Floyd-Warshall algorithm has time complexity  $O(n^3)$ , so it takes quite a long time on models with higher number of vertices. To optimize that time, we can use a downsampled mesh for this computation. For the downsampling, we used previously mentioned QEM simplification method [7]. The downsampled mesh preserves the mesh branching, tunnels and important vertices.

## 6. DEPENDENCY ON RESOLUTION AND DIMENSIONS

As we have previously mentioned, each term of Laplacian coordinates  $\delta_i = -4A_i k_i n_i$  is proportional to the area of adjacent faces surrounding the vertex  $i$ , denoted as  $A_i$ . If we want to contract all the types of models with the same weight system, the solution has to satisfy two conditions. The first condition is that the contraction of two models can not be dependent on their dimensions. The Laplacian operator is suitable for this case, because if the coordinates are propor-

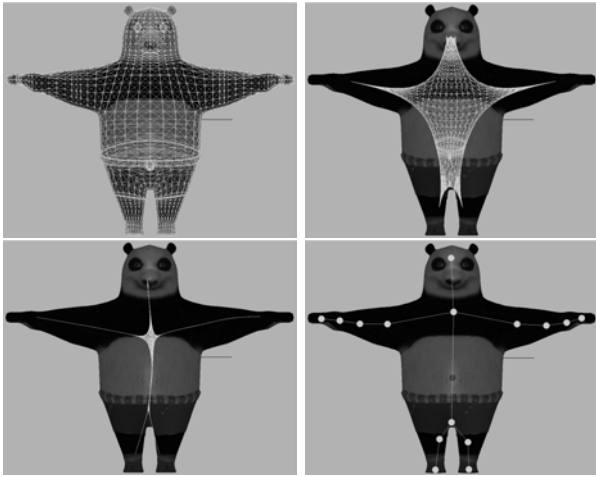
tional to  $A_i$  then the model vertices are contracted more if the dimension is higher and contracted less if the dimension is lower. The second condition is that the contraction can not be dependent on the resolution of the model. Models representing the same object have to be contracted equally, regardless of the resolution (if they are downsampled or not). This condition is not satisfied by the operator, because the contraction forces from the Laplacian operator are smaller for denser models. This can be observed, because the area of adjacent faces surrounding each vertex is smaller for denser models, but the dimensions stay the same. The solution how to update the contraction weight proposed by [1] satisfies the second condition, but violates the first condition. To satisfy both conditions we are looking for such a term  $t$  that the resulting Laplacian coordinates  $W_L t L V$  will be approximately the same for denser and sparser models while the first condition will not be broken. Such a term has to be a function indirectly proportional to the average face area of the model, denoted as  $A$ . With this term we can fulfil the second condition, but we will break the first one, because for models with small dimensions this term will be high and they will be over-contracted. We can keep the first condition by making the term directly proportional to the sum of face areas over the model, denoted as  $S$ . Thus, the term should be approximately  $t \approx \frac{S}{A}$ , this results to  $t \approx \#vertices$ . We initially set  $W_{L,i} = W_{L,i} t$ , so the resulting Laplacian coordinates  $W_{L,i} L V$  will not depend on the resolution of the model and length of the edges. We have found the term  $t = \#vertices$  as a good value to modify the input contraction weight.

## 7. RESULTS



**Figure 2:** The contraction and the extraction of the skeleton in few iterations from higher resolution geometry and its comparison to manually rigged skeleton by an artist. (Top-left) An input model. (Top-middle) The converted mesh graph. (Top-right) The mesh graph after 2 iterations. (Bottom-left) The mesh graph after the last iteration. (Bottom-middle) The extracted skeleton. (Bottom-right) The skeleton rigged by an artist.

We have tested the framework on a wide range of different models. We have achieved good results with both high resolution and also with low resolution models. The more vertices the model has, the better skeleton can be computed,



**Figure 3: Another example of higher resolution geometry.** The extracted skeleton has sparser nodes at the core parts. This feature can be observed, because many faces at core parts are contracted into the same region. The points are also moved towards the mesh boundary of the limbs, because of the shifting of the control points to the center of its local mesh.

but the algorithm takes more time. Contraction of geometry can be seen in Figure 2 and 3. The model of a worm in Figure 2 was published with a manually rigged skeleton (Bottom-right image). After the comparison we can conclude that the extracted skeleton is very close to the manually rigged one. After all, geometry and skinning data can be exported into Collada 1.5 *.dae* file and transferred into an external application to create animations.

## 8. CONCLUSION

In this paper we propose a framework for extracting a skeleton and skinning data from the geometry mesh using an iterative mesh contraction. The approach begins with converting the geometry into a mesh graph. This graph is iteratively contracted and a greedy algorithm is applied to choose the most important subset of vertices. In the next step, these vertices are converted into a hierarchical skeleton. Important skinning data such as indices and weights which are controlling the influence of the skinning process over vertices are computed as well. When the data is computed, our framework also provides a way to inspect the skeleton, simulate the deformation process with skinning algorithm implemented on GPU and allow the user to change the skeleton branching if it is needed.

The extracted skeletons have sparser nodes at the core parts of the model. This feature can be observed, because many faces at core parts are contracted into the same region. Computed skeletons are independent of the size and resolution of the models. The approach is insensitive to noise, but works only for closed mesh models with 2D manifold connectivity.

## 9. ACKNOWLEDGMENTS

This research was partially supported by a VEGA 1/0662/09 2009-2011 project a Scientific grant from Ministry of Edu-

cation of Slovak Republic and Slovak Academy of Science, and The Tokyo University research grant.

## 10. REFERENCES

- [1] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. Skeleton extraction by mesh contraction. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, 2008.
- [2] G. Aujay, F. Hétroy, F. Lazarus, and C. Depraz. Harmonic skeleton for realistic character animation. In *Symposium on Computer Animation*, pages 151–160, 2007.
- [3] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 72, 2007.
- [4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*, pages 558–565. MIT Press and McGraw-Hill, first edition, 1990.
- [5] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH 99*, pages 317–324, 1999.
- [6] T. K. Dey and J. Sun. Defining and computing curve-skeletons with medial geodesic function. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 143–152, 2006.
- [7] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.
- [8] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 954–961, 2003.
- [9] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 479–487, 2005.
- [10] L. Wade and R. E. Parent. Fast, fully-automated generation of control skeletons for use in animation. In *CA '00: Proceedings of the Computer Animation*, page 164, 2000.
- [11] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 644–651, 2004.
- [12] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph laplacian. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 496–503, 2005.