Adjusting Animation Rigs to Human-Like 3D Models

Jorge E. Ramirez and Antonio Susin and Xavier Lligadas

Universitat Politècnica de Catalunya
Barcelona, Spain
jramirez@lsi.upc.edu,toni.susin@upc.edu,xavi@lsi.upc.edu
http://www.lsi.upc.edu/~moving/

Abstract. In the animation process of a human-like 3D model, a skeleton must be specified to define the model's surface deformation of its limbs. Nowadays the skeleton specification is hand made and very time consuming task. In this paper we propose a novel semi-automatic method for rigging a 3D model in an arbitrary pose using a skeleton defined in an animation datafile with no specific initial pose. First a skeleton is extracted from the voxelizated model, this skeleton is refined and transformed into a tree-data structure. Because the 3D model can be in an arbitrary pose, user interaction is required to select the five limbs correspondence (head, hands and feet), and finally a skeleton taken from an animation data file or a external source is adjusted to the geometric skeleton.

Key words: skeleton driven animation, rig adjustment, skeletonization, thinning, voxelization,skinning,animation.

1 Introduction.

In skeleton driven animation one of the most time consuming tasks is the rig process. The rig process places a set of controls (joints) that interconnected by artificial bones (links) specify which parts of the 3D model must be rotated and translated to produce the desired motion (skeleton binding). Nowadays, the rig process is done manually, and it is created by placing the joints in the character's medial axis where an articulation should be. The number of joints that a skeleton will have depends directly on the animator and the chosen animation technique. If it concerns to a hand-made animation the number of joints used in a skeleton will be defined entirely by the animator. If the technique used is motion capture the number of joints used in a skeleton will depend on the number of captured joints.
The rigging process is tedious and time consuming, to reduce this time we have developed a human assisted method that allows an easy reuse of predefined skeletons that can be taken from motion captured files or previous character animations and adjust it to an arbitrary 3D model. Our method extracts a skeleton using a thinning process over a previously voxelizated 3D model. In this

paper we define a ***geometric skeleton*** as the obtained skeleton after finishing the thinning process, and a ***logic skeleton*** as the one created by an animator or taken from a motion capture file. A logic skeleton could be used in different 3D models if joints parameters were properly adjusted, this adjustment can be done manually or automatically. In this paper we present a method to adjust a logic skeleton to a geometric skeleton. While other approaches are constrained to an initial pose (T-pose or anthropometric pose), which makes the adjustment easier, our method is not pose constrained and we deal with models without a specific initial pose. Nowadays, these kind of models are becoming popular because they can be obtained from scanners or vision systems.

1.1 Related work.

Our work is initially based on the skeleton extraction, in 2D this problem was solved using hexagonal sampling [11] as an alternative to the classic square sampling.
In the 3D case we can find several thinning algorithms ([2],[1],[8],[7]) based on removing voxels from the surface of the voxelizated model until only a skeleton remains. In [10] the Euclidean distance and the Discrete medial surface is used to extract a 3D skeleton. A penalized algorithm [9] based in a modified dijkstra method is used for skeleton extraction, and a hybrid method [3] use a modified version of the thinning algorithm mixed with force fields to refine the process. For the automatic rigging [4] and [12] propose two different approaches. In [4] a predefined skeleton is embedded into the model's medial surface. A new method to extract the skeleton is proposed in [12], where using two 3D silhouettes and the mid points of the internal edges of a Delaunay triangulation, a skeleton is estimated.

1.2 Method overview.

The main idea of our method is to use an existing logic skeleton, and adjust it to an arbitrary human-like model. The geometric skeleton creation process is based on the method described in [5]. A geometric skeleton is the mapping to a tree data structure of the skeleton obtained after the thinning process is applied over the voxelizated closed mesh.
There are two main advantages of representing a geometric skeleton as a tree data structure:

1. ***Fast and easy traversal over all the skeleton:*** When the thinning procedure has been applied to the model, we define a node for each obtained voxel. All the operations (coordinate transforms, neighborhood and classification of the nodes) done over the voxelizated space are applied and stored in a data structure.
2. ***Allow us to perform operations over nodes:*** Modify or delete a node or an entire set of nodes (loops).

The chosen data structure to represent our geometric skeleton is a n-ary tree.

2 Creation process.

To create a geometrical skeleton we use a modification of the traversal algorithm described in [5]. Basically we create a node of the tree each time the algorithm is traversing a new voxel. As starting point we choose a random voxel from the thinned model.

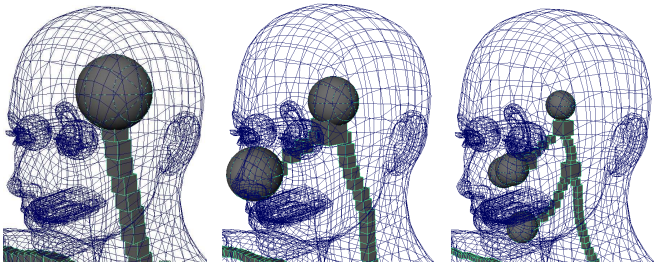
Node classification. In [5] the voxels or points of the skeleton where classified by their neighborhood, in this paper we are going to use this classification for the nodes of the geometric skeleton. The nodes are classified as:

- ***Flow nodes:*** Nodes with two neighbors, these nodes represent tubular segments of the skeleton, usually limbs (legs, arms, neck, etc.). In our tree data structure this kind of nodes will have one child and one parent.
- ***Connection nodes:*** These nodes will have a number of neighbors greater than two. They usually represent a solid-rigid part of the model, like the hips or the chest. In our tree data structure this can be a node with or without parent and more than one child.
- ***End nodes:*** Nodes with only one neighbor. These nodes will represent the final section of a limb, like the hands, feet, or the head. In our data structure this will be a node with only one neighbor, its parent.

2.1 Geometric skeleton post-processing.

Once the geometric skeleton is created, we apply a post-process to refine it. This post-process will have the following steps:

1. ***Deletion of loops and redundant nodes:*** The result of the thinning process over a voxelizated model is a set of voxels that represents a skeleton. Usually, this set has voxels which could be noisy or redundant nodes (voxels which can not be removed because of their topology condition [2]). We must have in mind that the size of the voxel in our space can change the number of details and noise in the geometric skeleton. If the voxel size is small, the thinning algorithm tends to introduce more voxels as end nodes, this will generate more branches in the geometric skeleton (fig. 1) .
2. ***Root node adjust:*** Because we use a random voxel as starting point in the creation of a geometric skeleton, the root node must be adjusted. Only connection nodes can be root nodes. The main reason for this is that in practically all the animation formats, the hips are taken as the center of mass for translations and rotations, and it can be considered as a solid-rigid. If the root node in the geometric skeleton is not a connection node, the nearest connection node is assigned as the root, and the geometric skeleton tree is balanced to the new root node.The assigned connection node is the first approximation to the model's hip, the appropriate assignment will be done in a posterior step.



(a) Voxel size at 1% of height. (b) Voxel size at 0.65% of height. (c) Voxel size at 0.4% of height.

Fig. 1: *Extracted skeleton at different voxel sizes.*

3. ***Skeleton smoothing:*** A smoothing step is mandatory because in a voxelizated space, changes of position between nodes of the skeleton in the same neighborhood are mainly produced in the edges of the voxel. This lead to undesirable artifacts if this data is used to calculate direction changes between two voxels. By changing the position of the voxels from edge to face neighborhood a smooth transition is granted. We use a window based method as our smoothing process.

3 Segments.

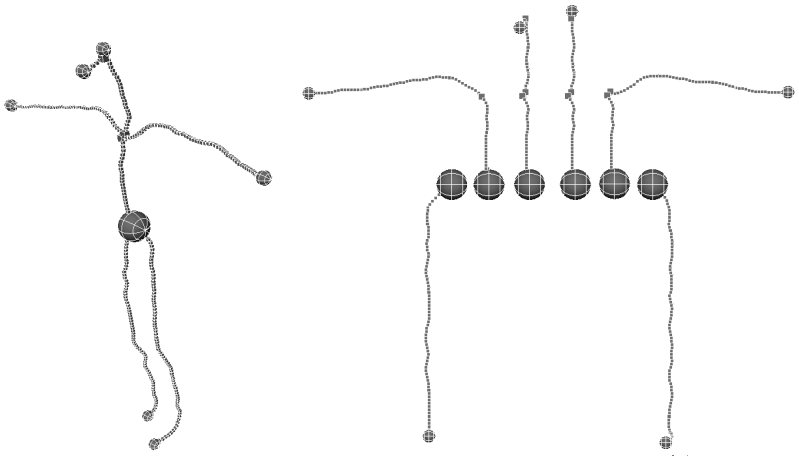
Segments are the core elements in the adjustment of a logic skeleton (rig) to a geometric skeleton. We define a segment as:
Segment: *A set of nodes traversing the skeleton from a connection node to an end node.(fig.2. b.).*
Using our definition of segment, a skeleton(geometric or logic) can be defined as:

Skeleton: *A set of segments with the same connection node as starting point (fig.2. a).*

In full body animation only five end nodes are needed (head,hands and feet) [4], furthermore the great majority of full body motion capture data is produced with five end nodes [13]. Therefore we have restricted our method to logic skeletons with five end nodes.

4 Node selection and root assignment.

The main problem of adjusting a logic skeleton to a geometric skeleton is finding the correspondence between their body segments (head,hands and feet). Logic



(a) Extracted skeleton. (b) Isolated segments.

Fig. 2: *Extracted skeletons and its segments, the third segment (root to nose) will be deleted.*

skeleton's limbs are specified by a tag, this tag can be obtained from a file, a user interface or if the model was in a specific pose it could be tagged automatically by its segments positions in the space.

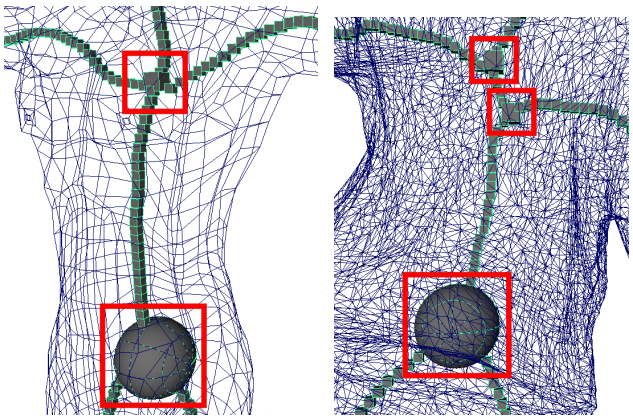
4.1 End node selection.

Geometric skeleton's limbs are not specified. The used 3D models are in an arbitrary pose, therefore there is no simple method capable of automatically tagging the limbs of a 3D model. Moreover, there are models with human like forms but with an extra limb (for instance the tail of an armadillo model). Limbs detection is a very challenging task and its out of the scope of this paper, to solve this problem we have implemented an interface that allows the user to select which are the end nodes that correspond to their appropriate limbs.
In our interface the end nodes are marked with a sphere and the flow nodes are represented by cubes. The user must decide which end node corresponds to its logic limb selecting the appropriate sphere (fig. 1 b and c).

4.2 Root assignment.

Once the limbs are assigned, we delete all the nodes that are not part of an assigned segment.(fig.2 b.).
When the segments are assigned, the number of connection nodes will decrease, and only connection nodes that represents non-articulated parts of the model(hips and chest) will be preserved.

It is customary to set the hip as the root node. In our case the hip will be one of the connection nodes but depending on the number of connection nodes the next situations can arise:



(a) Skeleton with two connection nodes.

(b) Skeleton with three connection nodes.

Fig. 3: *Root assignment cases.*

– ***Two connection nodes:***In this case the difference between the chest and the hip comes from the fact that the chest will have three segments without connection nodes(the hands and the head, fig. 3 a.) and the hip will have two (the feet). To apply this rule we are going to build two sets of segments (one per connection node), each set of segments will have its starting node in one of the connection nodes. Finally we assign the set with the least number of segments as the hip (root) of our skeleton.
– ***Three connection nodes:*** In this case we calculate the summation of the euclidian distances between flow nodes from one connection node to the other. The two nearer connection nodes will represent the chest and the other one the hip. Therefore to find the hip we create three set of segments, one per connection node. For each set we select the segment with the minimum number of flow nodes between the segment's starting node and its nearest connection node, an from these three segments we choose the one with the maximum number of flow nodes. The starting node of the selected segment will be assigned as the hip (root) of the skeleton.

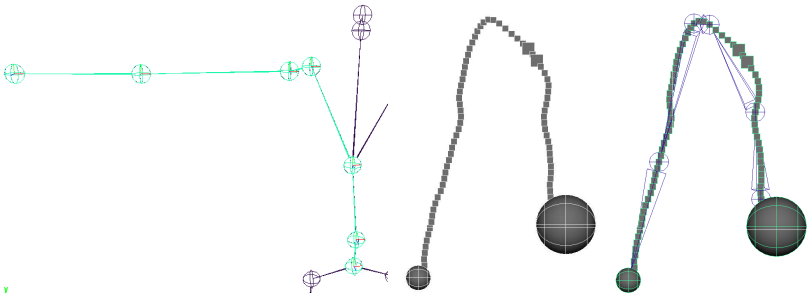
5 Skeleton adjustment.

A logic skeleton can also be viewed as set of segments, if we have followed the previous steps correctly, we must have the same number of segments in the geo-

metric and logic skeletons but in the logic skeleton we will have additional tagged nodes (elbow,neck,ankle...) that are not tagged in the geometrical one. Adjusting a logic skeleton to a geometric one is reduced to find the correspondence between logic tagged nodes and geometric untagged nodes.

5.1 Scaling segments.

As is mentioned in the section 3, our skeletons will be represented by a set of five segments. Because a segment in the logic skeleton has its equivalent in the geometric skeleton we can define a normalized distance in our skeletons segments, being *zero* the starting node position and *one* the end node position, with this distance we can find the position of the logic skeleton tagged nodes and map it to our geometric skeleton untagged nodes.
The distance of the logic skeleton segments its defined as the sum of the distance between two neighbor nodes(joints) in a segment from the root node to the end node. We have defined the distance of the geometrical skeleton segment as the sum of the distances between the center of two neighbors nodes(voxels) in a segment from the root node to the end node. Basically adjusting a logic skeleton to a geometric skeleton is finding a partition of the node graph formed by the logic skeleton segment, and map its internal nodes to its correspondent arc curve formed by the geometric skeleton segment. The union of all this mapped nodes(with its hierarchy implicit) will be the adjusted skeleton.



(a) Logic skeleton segment. (b) Geo. skel. segment. (c) Logic segment adjusted to a geo. skel. segment.

Fig. 4: *Segment adjustment.*

6 Results.

We have implemented our method as an *Autodesk Maya* plug-in with a 2.1 GHz Intel Core 2 Duo with 4 Gb RAM memory and Windows Xp 64 O.S. In the fig.5

we show the results obtained aplying our method to arbitrary models in different positions.The voxelization and skeletization time will depend on the model's pose and its number of triangles, the chosen voxel size is 0.65% of the model's height with processing times in the range of 2 and 3 seconds.The geometric skeleton creation and the logic segment adjustment processing time will be increased if more connection nodes and segments are obtained, our times are in the range of 2 to 3 seconds for models with a density of 20K and 28K triangles. (see table 1) The skin attachment of the skeleton has been done with Maya's mesh binding, the mesh deformation method used by Maya is SSD but we have modified these weights with a mesh segmentation algorithm(fig.5 last column).

Model	***Triangles***	***Voxelizat.+Thin.(sec.)***	***Assign+adjust.(sec.)***
Woman.	28820	1.9543	1.0874
Man jumping.	20000	3.0031	3.1689
Man marching.	20000	2.5508	2.4632
Man hand standing.	20000	2.0604	1.4756

Table 1: *Processing times.*

7 Conclusions and future work.

Our method can be used as part of any animation pipeline to improve or saving time in the development of an animation rig, because it can be applied to any human-like model in any pose. Our method can be combined with any SSD technique, but due to animations files are usually in an initial pose, a transformation between the animation file's initial pose and our adjusted rig pose must be done. This transformation can be easily implemented with any matrix or quaternion library.
We have proposed an easy and practical way to adjust logic skeletons to human-like models (or at least to skeletons with five or more end points) in an arbitrary pose.

The main contribution of our method is that instead of creating a new logic skeleton [12] or taking a predefined skeleton [4] we adjust an arbitrary skeleton(and its hierarchy) with its related motion data to an extracted skeleton, which is a new approach to the existing automatic skeleton rig methods . The logic skeleton can be taken from any motion data base [13], produced by a motion simulation software or a motion capture file, the only restriction is that the logic skeleton must have five end nodes. Our method is human like models oriented, but is not restricted to them. It can be used to adjust the rig of a human to a any model with at least five end nodes in its geometric skeleton.
The main limitation of our method is that it can not deal with any pose automatically, to produce the adjusted skeleton the user must select the limbs (end nodes).

As future work we want to use a skinning procedure different than SSD, we believe that our method can be used with a cage based deformation technique such as [6]. The cage base deformation needs an effective mesh segmentation based in the links of a rig, we are currently working in an algorithm to achieve this goal.

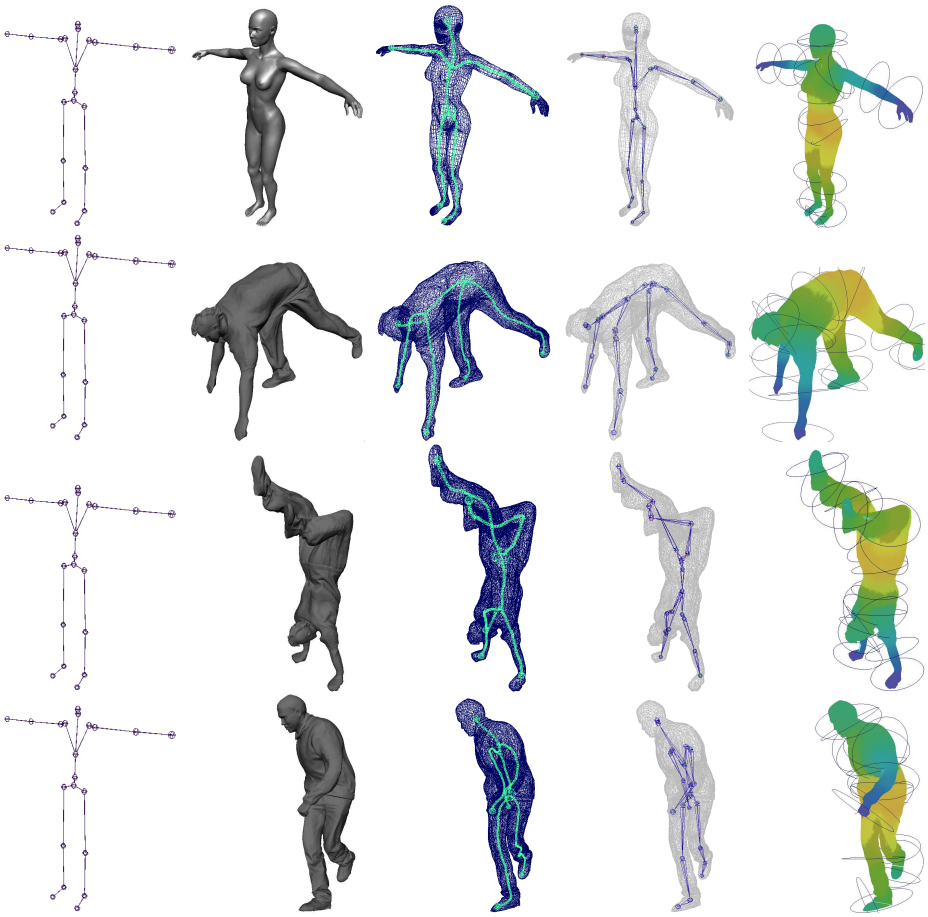


Fig. 5: *Columns:Skeleton from an animation file, arbitrary model, geometric skeleton, adjusted logic skeleton, binded model.*

8 Acknowledgments.

Authors are supported by grants from the "Consejo Nacional del Ciencia y Tecnologia " (CONACYT) MEX, TIN2007-67982-C02-01 ESP, "El comissionat per

a universitats i recerca del departament d'innovació d'universitats i empreses de la Generalitat de Catalunya" and the European Social Fund, and we thank to our fellows of the LABSID for their valuable discussions.

References

1. D. Brunner and G. Brunnett: An extended concept of voxel neighborhoods for correct thinning in mesh segmentation. In: SCCG '05: Proceedings of the 21st spring conference on Computer graphics, pp. 119–125. ACM Press, New York (2005)
2. K. Palágyi and E. Sorantin and E. Balogh and A. Kuba and C. Halmai and B. Erdohelyi and K. Hausegger: A Sequential 3D Thinning Algorithm and Its Medical Applications. In IPMI '01: Proceedings of the 17th International Conference on Information Processing in Medical Imaging, pp.409–415. Springer-Verlag (2001)
3. Pin-Chou Liu and Fu-Che Wu and Wan-Chun Ma and Rung-Huei Liang and Ming Ouhyoung: Automatic Animation Skeleton Construction Using Repulsive Force Field. In PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications, pp.409 IEEE Computer Society (2003)
4. Ilya Baran and Jovan Popović: Automatic rigging and animation of 3D characters. In SIGGRAPH '07: ACM SIGGRAPH 2007 papers, ACM San Diego, California (2007)
5. Jorge E. Ramirez, Xavier Lligadas and Antonio Susin: Automatic Adjustment of Rigs to Extracted Skeletons . In AMDO '08: Proceedings of the 5th international conference on Articulated Motion and Deformable Objects. Port d'Andratx, Mallorca, Spain,pp.409–418. Springer-Verlag (2008)
6. Ju Tao, Zhou Qian-Yi, van de Panne Michiel, Cohen-Or Daniel and Neumann Ulrich: Reusable skinning templates using cage-based deformations. In SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers, Singapore, ACM NY, USA (2008)
7. Christophe Lohou and Gilles Bertrand:A 3D 12-subiteration thinning algorithm based on P-simple points. In Discrete Appl. Math., Vol. 139, Num. 1-3, pp.171–195. Elsevier Science Publishers B. V. Amsterdam, The Netherlands, The Netherlands (2004)
8. Kálmán Palágyi and Attila Kuba:A 3D 6-subiteration thinning algorithm for extracting medial lines. Pattern Recogn. Lett., Vol. 19, Num. 7, pp.613–627. Elsevier Science Inc. New York, NY, USA(1998)
9. Ingmar Bitter and Arie E. Kaufman and Mie Sato:Penalized-Distance Volumetric Skeleton Algorithm. IEEE Transactions on Visualization and Computer Graphics., Vol. 7, Num. 3, pp.195–206. IEEE Educational Activities Department. Piscataway, NJ, USA(2001)
10. Lawson Wade and Richard E. Parent: Automated generation of control skeletons for use in animation. The Visual Computer., Vol. 18, Num. 2, pp.97–110. Springer Berlin / Heidelberg(2002)
11. Richard C. Staunton: An analysis of hexagonal thinning algorithms and skeletal shape representation. Pattern Recognition., Vol. 29, Num. 7, pp.1131–1146. Elsevier Science B.V. U.K.(1996)
12. Pan, JunJun and Yang, Xiaosong and Xie, Xin and Willis, Philip and Zhang, Jian J.: Automatic rigging for animation characters with 3D silhouette. Comput. Animat. Virtual Worlds., Vol. 20,Issue 2 – 3,pp.121–131. John Wiley and Sons Ltd.(2009)
13. ACCAD - Motion Capture Lab: `http://accad.osu.edu/research/mocap/mocap_data.htm`