Automatic Adjustment of Rigs to Extracted Skeletons

Jorge E. Ramirez, Antonio Susin, and Xavier Lligadas

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

Abstract. In the animation, the process of rigging a character is an elaborated and time consuming task. The rig is developed for a specific character, and it can not be reused in other meshes. In this paper we present a method to automatically adjust a human-like character rig to an arbitrary human-like 3D mesh, using a extracted skeleton obtained from the input mesh. Our method is based on the selection and extraction of feature points, to find an equivalence between an extracted skeleton and the animation rig.

Key words: Animation, rig adjustment, skeletonization, thinning, voxelization

1 Introduction.

One of the most time consuming tasks in animation is character modeling, commonly when a character is animated a skeleton (rig) is created, this rig usually is adjusted to a specific 3D mesh. If the same rig and animation data wants to be used on a different mesh, an artist have to spend effort and time to adjust it to a new mesh.While the skeleton extraction is a well known problem ([4],[3])however the equivalence between the voxels or points, and their adjustment to an animation rig has not been explored as deeply as the skeletonization problem.
Our work take as a base the skeleton extraction, many approaches have emerged about how to solve this particular problem. In 2D this problem was solved using hexagonal sampling [14] as an alternative to the classic square sampling.
In 3D one of the most difunded are thinning algorithms ([4],[3],[11],[10]) which works by using a voxelizated version of a model and removing voxels from the surface until an skeleton remains. In [13] it is used a Euclidean distance and Discrete medial surface to extract a 3D skeleton. A penalized algorithm [12] based in a modified dijkstra method is used for skeleton extraction, and a hybrid method [6] use a modified version of the thinning algorithm mixed with force fields to refine the process. Our work can be compared with the results obtained in [7] but this work is based on the embedding of a rig, instead of its adjustment to an extracted skeleton.
The purpose of this paper is to reduce this adjustment task. In order to improve the adjustment process, we propose a method that automates the rigging process of a human like 3D mesh. Our method is composed by the next stages:

1. Model Voxelization: A 3D mesh is transformed into a set of voxels. This transformation is a discretization of the surfaces of the triangles of the mesh.
2. Skeleton extraction:Taking as a base the voxelizated model, we apply a skeleton extraction based on a secuencial thinning algorithm.
3. Points selection and their refinement: From the extracted skeleton we obtain and refine information about the input model; we also make an equivalence of points between the rig joints and the feature points of the extracted skeleton.
4. Scale and adjustment of joints: once we have made an equivalence of points we scale and adjust the joints of the rig to make them totally fit into the skeleton extracted from the mesh.

The skeleton adjust is a complex problem that depends on the input 3D mesh; to successfully apply this method we need to restrict the problem to the following conditions:

- The input 3D mesh has to be closed.
- The input must have an approximated fixed pose. In our particular the mesh needs to be stand up with the arms extended (T-pose).
- The 3D mesh has to have human like proportions.

Our method is based on a correct skeleton extraction to establish equivalences between feature points of the extracted skeleton and joints of the animation rig.

2 Model Voxelization

Voxelization is the process used to transform a closed 3d triangle mesh into to a voxel subset in a defined space [9], in general space voxelization is equivalent to a coordinate transformation and a map function to determine the vertex and faces position of our original model in voxel's space . Basically our voxelization process has been performed following the next steps:

1. Space Voxelization: We create a bounding box containing the model and then we subdivide it into voxels.
2. Mesh Voxelization: The vertex and faces of the triangular mesh are transformed into voxels.
3. Filling the voxelizated mesh: The empty voxelizated mesh is filled with voxels and transformed into a closed solid model.

Basically voxelizing a model is the transformation of the triangles to voxels. This task has been faced by different approaches like [3]. Our approach is based on the midpoint-line algorithm [1] extended to the 3D space. We have choose the midpoint-line algorithm because of its low processing cost, achieved by only using integers and additions to paint a line. For each triangle we transform their edges into voxels using the 3D midpoint-line algorithm, then we keep painting lines between edges until the entire triangle surface is filled. The voxels space struct is a simple list of bytes where a byte represent a voxel in the space, to move between voxels we rely on pointer arithmetic.
Once we have voxelizated the mesh, we get a set of surface voxels and a set of empty voxels within it. For the skeletization stage we need to fill the empty voxel set, the filling algorithm is an extension to 3D of the flood algorithm [1] used

in image processing. When the model is voxelizated, a resolution space problem appears: if we are using low resolution (big voxel size) the map function will map some vertex to the same voxel. The problem emerges when the space between limbs or between a limb and the body of the mesh is lower than one voxel, in this particular case the voxelization process will merge two sections of the model. To solve this problem, we store three different voxels values: 0 for empty voxels, 1 for surface voxels and $n > 1$ for filled voxels. Only filled voxels will be used in the thinning process.

3 Skeleton extraction.

A skeleton is an object shape descriptor, in this paper the skeleton extraction is achieved by applying a thinning algorithm to an input 3D mesh. Basically a thinning algorithm is a process where the voxels that belongs to the surface (voxels where one or more of their faces are in contact with an empty voxel [8]) are deleted if the deletion does not affect the topology of the empty and filled voxels sets.
A voxel is a discretization of a point in a finite space[8], a simple point is a voxel whose removal does not change the topology of a voxel set. In this paper we are using $(26, 6)$ adjacency[8], where a black point is a voxel with a 0 as stored info and a white point is one voxel with value $n \geq 1$. As it was defined in section 2 we will work with voxels whose value is greater than 1. Most of the thinning algorithms are parallel as e.g. [10],[11], this kind of algorithms are useful if we have a large amount of data, in that case it is necessary to process the data using multiple processors. In our particular case the dimension of the discretized models are limited (see section 6),and altought we can implement this kind of algorithms with multi thread programming, we do prefer a more optimal algorithm (parallel algorithms generates latency due to the synchronization of subdivided tasks) to process the voxelizated model. The used algorithm is based on a secuencial version of the thinning algorithm[4] but with some added modifications to fit our needs and to optimize it.
In the thinning algorithm described in [4], the process is applied in 6 directions (*up,down,left,right,front* and *rear*) this stages of the algorithm are called *sub iterations*. Within each *sub iteration* a set of tests are applied to know if a voxel it is a deletion candidate. All deletion candidates are stored in a list.After the deletion candidate selection, the same tests done in the previous step are applied to the deletion candidates. If a voxel is still a deletion candidate after some of the voxels of the list were deleted, the voxel is deleted and his stored value will be set to 0.
We have modified the deletion candidates selection algorithm by adding a sorting process. In the sorting process the first positions of the deletion candidate list will be filled with highest delete direction value voxels, making the thinning a more robust process, obtaining a better approximation of the skeleton to the medial line of the model.

4 Points selection and their refinement.

From the voxelized skeleton extracted in section 3 we can find the correspondence between white points (voxels) and joints of an animation rig by using the white points neighborhood information.

4.1 Point classification.

A white point can be classified depending on the number of neighbors surrounding it:

– *End points*: Points which only have one neighbor. Normally this points represents the end of a model's limb..
– *Flow points*: Referees to point which have two neighbors. This points are part of a flow or a tubular segment in the skeleton.
– *Internal points*: Points with more than two neighbors. Points that give us information about the skeleton union segments.

Once we have classified the points, we can start to look for a correspondence between joints of an animation rig and the points within our extracted skeleton.

4.2 Animation Rig.

Animation rig data is represented using a hierarchic model. The hierarchy is dependent on the data file format. Developing our approach, we decided to create a set of data structures to represent an animation rig, and left the hierarchy problem of a specific animation format to a separated module. In our structures we start with a joint as the root element,this root node is the model's gravity center. All the other joints will be represented as nodes, a node can have multiple child nodes(internal node), or it can have none(leaf node). A data structure that easily adapts to this kind of hierarchy is a tree with multiple child per nodes.

4.3 Internal point refinement.

To classify the points of the extracted skeleton we have two options: check all the voxels in the space and classify all the points we find, or we can traverse only our skeleton points and classify them. We have developed a traversal algorithm that depends on the number of neighbors of a point:

Traversal algorithm

```
function traversal (p){
   inter<stack>;
   PUSH_STACK(inter,p);
   while IS_NOT_EMPTY inter{
      neignum=GET_NEIGHBORHOOD(p);
      if(neignum >2){
        REG_INTERNAL_NODE(p);
        PUSH_STACK(inter,p);
      }
      else if(neignum = 1){
         REG_END_NODE(p);
```

```
          p=POP_STACK(inter);
        }
        else{
          REG_FLOW_NODE(p);
        }
        p=GET_FIRST_NEIGHBOR_LEFT(p);
      }
    }
```

Where the functions

- `GET_NEIGHBORHOOD(p)`: Returns the numbers of neighbors of the point p.
- `GET_FIRST_NEIGHBOR_LEFT(p)`:Returns the first unregistered neighbor of the point p.

Once the traversal has been done, we will have all the internal, end and flow nodes registered in a data structure. As the reader can see in figure 1 we will obtain multiple internal points, sometimes the distance between points will be smaller than the minimum limb length(the minimum length for the forearm is 0.146 of the height of the model [2]). This situation will difficult the correspondence process between the rig joints and the extracted skeleton points, that's happening because it is ambiguous which one of them has to be taken as the relevant point of the area. To overcome this problem we simplify a set of closer internal points into one, by identifying the point with higher number of neighbors and attracting the surrounding closer internal points to it (figure 1).

Fig. 1. *Left: internal points of a extracted skeleton(diamonds). Right: single internal point after the refinement.*

4.4 Point correspondence.

Once the skeleton is extracted from the voxelizated model, we have to deal with the problem of finding a correspondence between the skeleton points and the rig joints. To solve this problem we make some assumptions: the rig must have five joints as end nodes (if the rig is highly detailed it will have end nodes in the hands fingers and eyes of the model) this five end nodes will represent, feet, hands and head. Using our skeleton end points we will set a correspondence between the skeleton end points and the rig end joints. As a first step we have to find the orientation of our skeleton, to deal with this problem we will use an axis aligned bounding box to divide the space in eight parts (figure 2). If a skeleton is correctly orientated it will have more end points in the superior parts ($Q_{1-1} - Q_{1-4}$) of the bounding box and the feet end nodes will be placed in

Q_{2-3} and Q_{2-4}.We made eight stacks to represent the bounding box parts, the end and internal points are stored in each stack depending of its position within the box. To assign the end points we follow the next steps:

– Head: Is assigned to the end point with the Y axis max value in the upper part of the bounding box stacks (Q_1 set).
– Right hand: Is assigned to the end point with the X axis min value in Q_{1-2} and Q_{1-3} stacks of the bounding box.
– Left hand: Is assigned to the end point with the X axis max value in Q_{1-1} and Q_{1-4} stacks of the bounding box.
– Left foot: Is assigned to the end point with Y axis min value in the Q_{2-4} stack of the bounding box.
– Right foot: Is assigned to the end point with Y axis min value in the Q_{2-3} stack of the bounding box.

5 Scale and adjustment of joints.

To adjust a rig to an extracted skeleton it is necessary to choose an effective scale method. In our approach, instead of using the naive solution and only calculate a single scale factor for the rig, we have chosen scaling in parts to force the end and root points to fit into the extracted skeleton. Once the scaling has been done we apply an adjustment stage to the left internal joints of the rig, moving the scaled position to the closest voxel in the extracted skeleton.

5.1 Scaling in parts.

The scaling of the end and root points is a simple task when the correspondence of points has been done correctly. For each of the assigned points in section 4.4 three scale factor are calculated, one for each coordinate. To make the calculation of the scale factors, we divide both the rig and the extracted model into five bounding boxes. The distributions of the bounding boxes are: four for the hands and feet (fig 3) and one for all the points that are outside of these boxes.
The first four bounding boxes have one of its vertex placed over the root point (or joint depending of which set of boxes we are calculating) and its opposite vertex in one of the four limbs end points. The fifth bounding box is the bonding box that encapsulates the model, but its scale factors are mean factors calculated using the previous four bounding boxes scale factors.

The scale factors are calculated following the next steps:

– For each model (the rig and the extracted skeleton) we will have five bounding boxes ($[bbA_1, bbA_2..., bbA_5]$ and $[bbB_1, bbB_2, ..., bbB_5]$).
– For each bounding box we find the distance between the first and the opposite bounding box vertex for each axis: $bbA_1 = (dbbA_{1a}, dbbA_{1b}, dbbA_{1c})$.
– We divide the distances of each axis of the extracted skeleton bounding boxes between the distances of the rig bounding boxes:
$sfbb_1 = (dbbB_{1a}/(dbbA_{1a} + \epsilon), dbbB_{1b}/(dbbA_{1b} + \epsilon), dbbB_{1c}/(dbbA_{1c} + \epsilon))$.

Finally we assign each of the joints of the rig to a bounding box. If a joint it is inside a bounding box we apply its associated scale factor over it, therefore the scale factors applied to a joint will be dependent of its position. The head is treated as a special case and is adjusted separately. All the remaining points will be assigned to the fifth box.

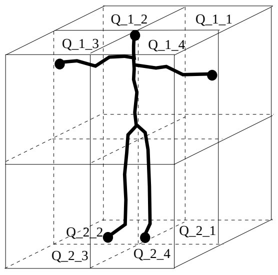

Fig. 2. *Extracted skeleton encapsulated in a bounding box.*

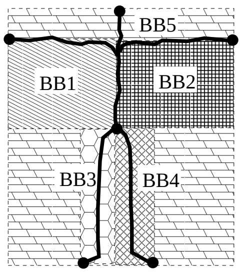

Fig. 3. *The bounding box used in the scale stage.*

5.2 Joint adjust.

After scaling the joints to the root and end points, we are sure that they are adjusted to the extracted skeleton(the scaling adjust is implicitly), the remaining joints will probably be outside but near of our skeleton. To find which point of the skeleton correspond to every one of the joints without correspondence, spheres are used to find the closest skeleton point of a joint. The process is simple, for each joint we follow the next steps:

1. If the joint is inside the volume of a voxel in the skeleton, we modify its position to the center of the voxel in the skeleton.
2. If the joint is outside of the skeleton we start to search in the surrounding voxels for the closest one.
   - We start with a sphere with one voxel length.
   - If a neighbor point is not within the range, we rise the sphere radius by one voxel length.
   - If we find a skeleton voxel within the sphere, we set the joint position to the voxel position (fig. 4).
   - We iterate until we find a voxel to assign the joint.

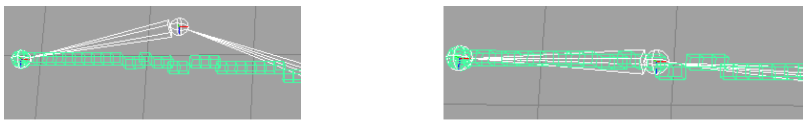

Fig. 4. *Left: an arm segment before adjustment. Right: arm segment after the adjustment.*

6 Results.

In this paper we have presented a method to adjust a rig to an arbitrary closed mesh. The results presented in this work (figure 5, table 1) depends directly on the resolution in voxels of the box encapsulating the model; For instance, if the resolution is lower than 100 voxels (in Y axis) the processing time will be reduced but the results may be inaccurate; on the other hand, if the resolution is higher than 200 voxels, the time will be considerably increased. The extracted skeleton will be a better shape descriptor, but the extra information increases the computation time, and in the best case scenario the extra information is ignored or may induce errors during the algorithm execution. Based on our experience, we believe that optimal resolution comprises from 121 to 149 voxels in the Y axis (the two remaining coordinates will be calculated based on model proportions) gives us the best results and computations times (about 6 seconds running as a maya plugin). Finally we have tested our method in four models(fig. 5): a human male and female in low resolution, a human female in medium resolution, and a human like character(alien) in low resolution. In the three human models we have obtained good results, but in the character like model, our method has failed because it has more than one end point in the feet. In the upper part of the body, the human like model, shows good results. Our tests were made on a laptop HP Turion 64 X2 with 2 gigas of RAM and Windows Xp.

Model	***Triangles***	***Resolution(voxels)***	***P. Time(sec.)***
Woman Mid. Res.	28820	130×143×23	5.28006
Woman Low Res.	12944	129×127×25	3.29999
Man Low Res.	12944	129×127×25	3.389999
Alien Low Res.	11408	121×115×30	2.90999

Table 1. *Results from the models in the figure 5*

7 Future work.

To add robustness and improve our method many things could be done. In the stage of voxelization we depend on the format of the meshes. At this moment we only can use single closed meshes, meshes composed of multiple sub meshes can not be solved at this moment but we believe that in the future this could be achieved. The correspondence stage is fundamental to obtain a satisfactory result. This stage can be improved so that it could deals with models that does not have the correct human proportions but that resemble the human body (toon characters). We have many ideas to expand this work, but we believe that adding a skinning process such as SSD or Radial Basis [5] to the mesh will be a valuable contribution. Also we would like to improve the joint adjustment; because, as it is shown in fig.5 the joint can be placed out of the articulation, causing odd effects during the mesh animation.

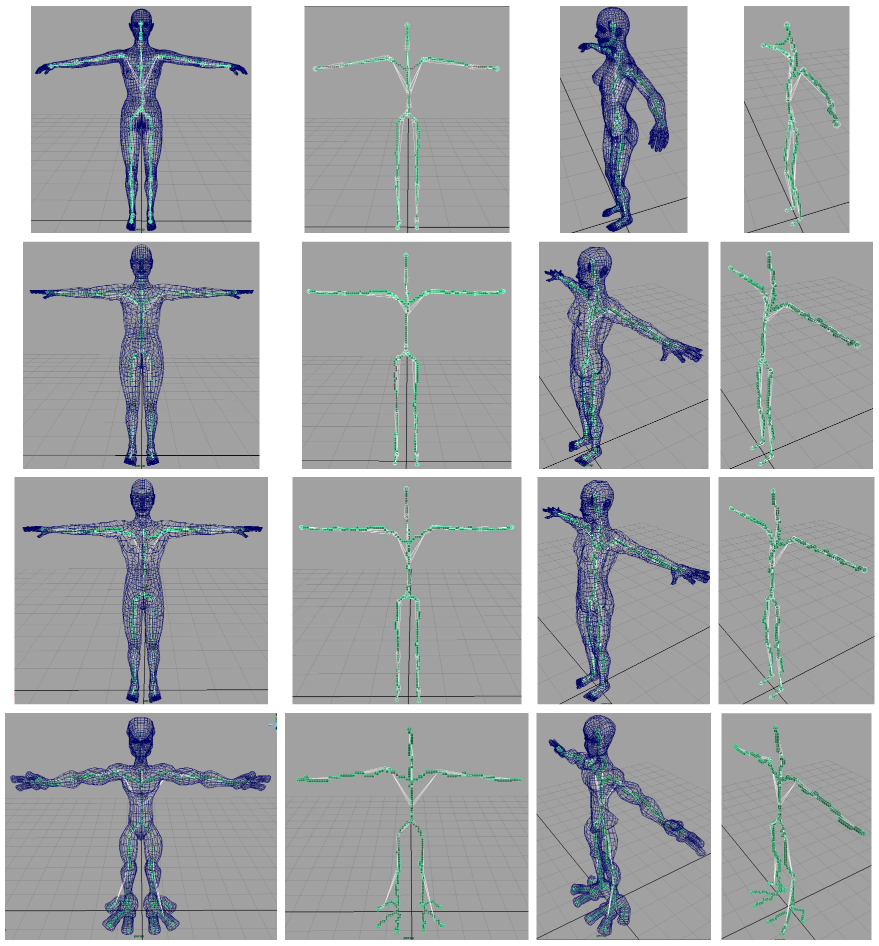

Fig. 5. *Results of the skeleton adjust to a rig in different models. Rows: Woman Mid. Res.,Woman Low Res., Man Low Res.,Alien Low Res.*

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’innovaci d’universitats i empreses de la Generalitat de Catalunya” and the European Social Fund, and our fellows of the LABSID for their valuable discussions.

References

1. Foley, van Dam, Feiner, Hughes.: Computer Graphics: Principles and Practice. ADDISON WESLEY, (1996)

2. Plagenhoef S.: Patterns of Human Motion. Prenctice-Hall, Inc., Englewood N.J., (1971)
3. 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)
4. 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)
5. J. P. Lewis and Matt Cordner and Nickson Fong: Pose Space Deformations: A Unified Approach to Shape Interpolation a nd Skeleton-Driven Deformation. In Siggraph 2000, Computer Graphics Proceedings, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, Washington, DC, USA (2000)
6. 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)
7. Ilya Baran and Jovan Popović: Automatic rigging and animation of 3D characters. In SIGGRAPH '07: ACM SIGGRAPH 2007 papers, ACM San Diego, California (2007)
8. G. Bertrand:A Boolean characterization of three-dimensional simple points. In Pattern Recogn. Lett, Vol. 17, Num. 2, pp.115–124. Elsevier Science Inc. New York, NY, USA(1994)
9. D. Cohen-Or and A. Kaufman: Fundamentals of surface voxelization. In Graph. Models Image Process., Vol 57, Num. 6, pp.453–461. Academic Press, Inc.Orlando, FL, USA(1995)
10. 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)
11. 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)
12. 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)
13. 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)
14. 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)