SKELETON-BASED RIGID SKINNING FOR CHARACTER ANIMATION

Andreas Vasilakis, Ioannis Fudos
Department of Computer Science, University of Ioannina, Ioannina, Greece
$\{abasilak, fudos\}$*@cs.uoi.gr*

Keywords: skeletonization, rigid skinning, character animation, mesh generation.

Abstract: Skeleton-based skinning is widely used for realistic animation of complex characters defining mesh movement as a function of the underlying skeleton. In this paper, we propose a new robust skeletal animation framework for 3D articulated models. The contribution of this work is twofold. First, we present refinement techniques for improving skeletal representation based on local characteristics which are extracted using centroids and principal axes of the character's components. Then, we use rigid skinning deformations to achieve realistic motion avoiding vertex weights. A novel method eliminates the artifacts caused by self-intersections, providing sufficiently smooth skin deformation.

1 INTRODUCTION

In video games, crowd simulations, computer generated imagery films and other applications of 3D computer graphics, achieving skin motion of deformable objects in a realistic-looking way is of utmost importance. Since, due to its versatility, *skeletal animation* is one of the most popular techniques, we focus on skin motion techniques for skeletal animation of articulated objects.

In skeletal animation, a representation model consists of at least two main layers: a highly detailed 3D surface representing the character's *skin*, and an underlying *skeleton* which is a hierarchical tree structure of *joints* connected with rigid links (*bones*) providing a kinematic model of the character.

The process of extracting a skeleton is called *skeletonization*. A skeleton acts as a special type of deformer transferring its motion to the skin by assigning each skin vertex one (*rigid skinning*) or more (*linear blending skinning-LBS*) joints as drivers. In the former case, inherent flaws arise caused by self-intersections, especially in areas around joints. In the latter case of LBS , each skin vertex is assigned multiple influences and blending weights for each joint. This scheme provides more detailed control over the results. However, the generated meshes exhibit volume loss as joints are rotated to extreme angles producing non-natural deformations (also known as collapsing joint and candy wrapper effects). Yet another problem with LBS is that tuning the LBS vertex weights tends to be a tedious and cumbersome task (Lewis et al., 2000; Mohr and Gleicher, 2003; James and Twigg, 2005). Despite these drawbacks, variations of this method are widely used in interactive computer graphics applications because they are simple and easy to implement on GPUs.

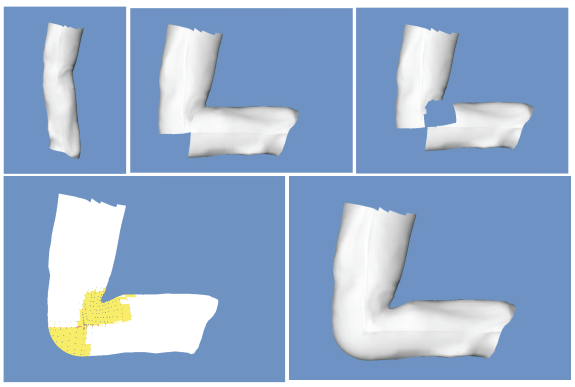
Figure 1: Robust rigid skinning. From top left to bottom right: the initial model; rotating the lower part; removing points; adding points; construct a robust blending patch

To address the above issues we present an integrated skeleton-based rigid skinning framework for animating 3D solid modular articulated models. A visually satisfactory skeleton is extracted using centroids and principal axes (Lien et al., 2006) of the components of a segmented object by performing a depth-first traversal of the skeleton hierarchy tree. We refine the produced skeleton segments with local and neighbor features to derive better skeletal representations that are more appropriate for our application. Then, we use classic rigid skinning by assigning each skin vertex to an influence bone to achieve mesh animation avoiding thus vertex weight estimation and training pose set production costs. We introduce a novel method to eliminate degeneracies and artifacts from self-intersections, especially in areas near joints, by performing alternative intuitive deformations. Figure 1 illustrates the entire process.

2 RELATED WORK

There is an abundance of research work in the literature that tackles the skeleton extraction and skinning of 3D objects from different perspectives. We focus on recent developments most closely related to animation.

Skeletonization algorithms may be classified based on whether they work on the boundary surface (*geometric methods*) or on the inner volume (*volumetric methods*). Researchers have proposed geometric based algorithms approximating MAT to overcome some of its inefficiencies, however MAT based skeletons are still not well fitted for animation applications. Several methods (Liu et al., 2003; Ma et al., 2003) generate skeletons by constructing discrete field functions by means of the object's volume. Although accurate, such methods are usually very time consuming and they cannot be applied to animation since the volumetric information needed is not usually part of the animation model. Researchers also generate skeletons based on mesh contraction (Au et al., 2008). (Katz and Tal, 2003) extract a skeleton using a hierarchical mesh decomposition algorithm. Finally, (Lien et al., 2006) proposed an iterative approach that simultaneously generates hierarchical shape decomposition and a corresponding set of multi-resolution skeletons.

Linear blend skinning (LBS) is the most widely used technique for real-time animation due to its computational efficiency and straightforward GPU implementation (Rhee et al., 2006). Recent skinning algorithms are classified based on whether they use (*Geometric methods*) or a training set of poses (*Example based methods*) of input models. Geometric methods revert to non-linear blending of rigid transformations since deformation is inherently spherical. Numerous proposed methods have replaced the linear blending domain with simple quaternion (Hejl, 2004), spherical blending (Kavan and Žára, 2005) and dual quaternion (Kavan et al., 2008). Example-based methods remove artifacts by correcting LBS errors with storage and computation cost increase (Lewis et al., 2000; Wang and Phillips, 2002; Merry et al., 2006). Our rigid skinning approach falls under Geometric class of algorithms introducing a novel versatile, robust and efficient blending approach based on rational quadratic Bezier patches.

3 SKELETONIZATION

We have built on techniques introduced by (Lien et al., 2006) that produce refined local skeletal morphs from the components of modular models. A straightforward approach to build the skeleton of a component is to connect the opening centroids. A boundary of a component consists of the common joining points of two adjacent components generated during a mesh segmentation process. We denote the centroid of a boundary as *opening centroid*. An improvement over this method is to use the centroid or kernel centroid of the component in addition to the opening centroids. However, frequently they do not capture the component shape accurately. In (Lien et al., 2006) proposed to overcome failures of previous methods by extracting a skeleton from a component by connecting the opening centroids to the principal axis. Our approach adapts this skeleton extraction technique for use in our animation framework. Let OC be the set of the opening centroids oc_i of component C, $OC = \{oc_1, \ldots, oc_n\}$. First, we select the principal axis segment of the component's convex hull that resides within the interior of the component. For even better results we have used as major axis an axis parallel to the principal axis of the convex hull that goes through the kernel centroid of the component to ensure that skeleton segments have the least possible intersection with the component's border. We denote this major axis segment as pa. We subdivide the pa in a number of uneven segments by picking as link points the projections of the opening centroids on pa (see Figure 2). Let $P_{pa}(oc_i)$ be the projection of opening centroid oc_i on pa. Our matching algorithm uses a dynamic programming concept, enhanced with new score functions which aim at maximizing the length of the utilized pa.

After grouping has been performed, we create the set of skeleton edges. Beyond the standard con-

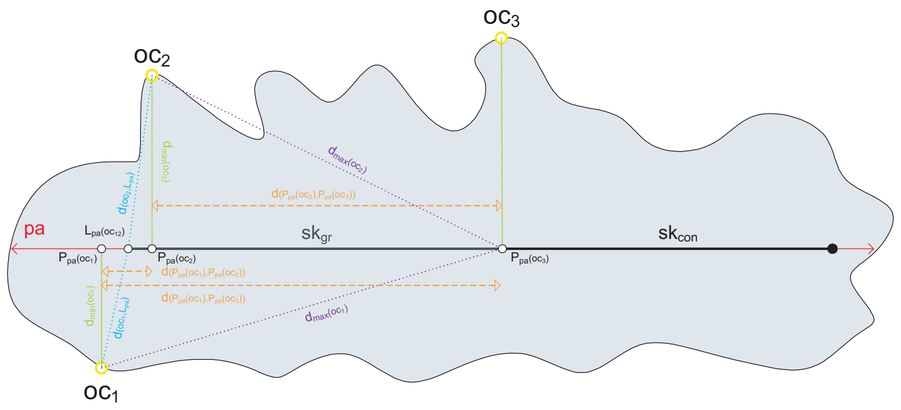

Figure 2: Skeleton segments extracted by the grouping (sk_{gr}) and the connecting (sk_{con}) algorithm.

nections between the opening centroids and the link points, we use extra skeleton edges based on the mapping result.

Our approach improves the principal axis orientation by introducing two modifications, the first is based on local features and the second uses knowledge inherited from the component hierarchy. Finally, we propose to add four more joints which represent the two ends of the other axes segments making the skeletal representation more topologically expressive.

3.1 Grouping Algorithm

Our approach manages to minimize the total mapping length and the cardinality of the set of link points (score function F_1), while at the same time maximizes the used pa length (score function F_2).

Let $d(\vec{p}_1, \vec{p}_2)$ be the euclidean distance between point $\vec{p}_1$ and point $\vec{p}_2$. For a set of opening centroids, $oc_{ij} = \langle oc_i, \dots, oc_j \rangle$, we compute their link points on pa as the average of their projections by

$$L_{pa}(oc_{ij}) = \frac{\sum_{k=i}^{j} P_{pa}(oc_k)}{|oc_{ij}|} \tag{1}$$

$\forall\ oc_k \in oc_{ij}$, we evaluate its normalized variation as

$$V(oc_k) = \frac{d(oc_k, L_{pa}(oc_{ij})) - d_{min}(oc_k)}{d_{max}(oc_k) - d_{min}(oc_k)} \tag{2}$$

where $d_{min}(oc_k) = d(oc_k, P_{pa}(oc_k))$ and $d_{max}(oc_k) = \max_{1 \le l \le |OC|}\{d(oc_k, oc_l)\}$.

Moreover, we define the ratio of the pa length which is vanished after grouping as the pa length which is generated among the projections of these opening centroids divided by the maximum used pa length.

Definition 1. The *normalized merging score function* F_1 for oc_{ij} group set is defined as the average of the total distance cost and the ratio of the not used pa length,

$$F_1(oc_{ij}) = \frac{\frac{\sum_{k=i}^{j} V(oc_k)}{|oc_{ij}|} + \frac{d(P_{pa}(oc_i), P_{pa}(oc_j))}{d(P_{pa}(oc_1), P_{pa}(|OC|))}}{2} \tag{3}$$

Definition 2. The *normalized separating score function* F_2 for groups $G_x = \langle oc_i^x, \dots, oc_l^x \rangle$ and $G_y = \langle oc_{l+1}^y, \dots, oc_j^y \rangle$ is defined as the average of the sums of their total distance cost and the complement of the ratio of pa length which is generated between these groups,

$$F_2(G_x, G_y) = \frac{\frac{\sum V(oc_k^x)}{|G_x|} + \frac{\sum V(oc_k^y)}{|G_y|} + (1 - gR_{pa})}{3} \tag{4}$$

where $gR_{pa} = \frac{d(P_{pa}(oc_l^x), P_{pa}(oc_{l+1}^y))}{d(P_{pa}(oc_1), P_{pa}(|OC|))}$.

3.2 Connecting Algorithm

The final skeleton of the component contains line segments that connect opening centroids to link points and line segments that interconnect link points. Opening centroid grouping often generates skeletons that do not capture important topological information. Therefore, the connection algorithm works with the following rules:

1. If all opening centroids are grouped to one link point and this point is close to:
   - pa's centroid, we connect it with the pa's end points on both sides of its centroid.
   - one of pa's end points, we connect it with the pa's end point on the other side of pa's centroid.
2. If opening centroids are connecting to more than one link point and these points are close to:

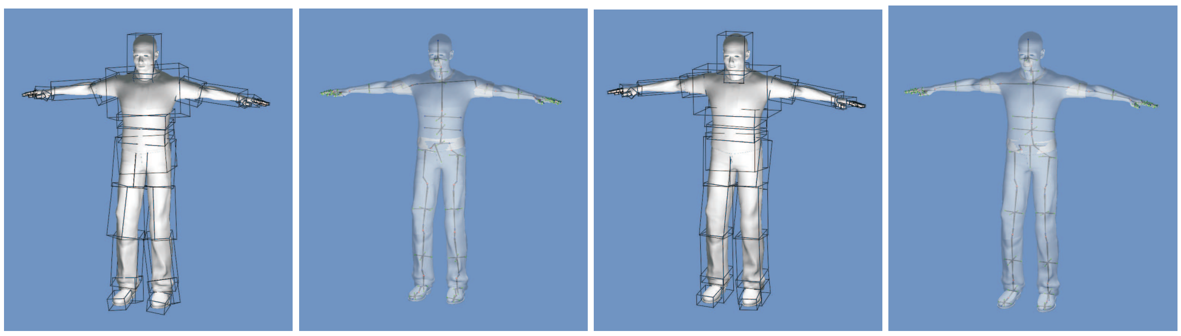

Figure 3: Oriented bounding boxes and skeletons of “Human” components: (left) Original configuration (right) After alignment.

- *pa*’s centroid, we connect the first and the last link points with the *pa*’s end points on both sides of *pa*’s centroid, respectively.
- one of *pa*’s end points, we connect the first or the last link point with the *pa*’s end point on the other side of *pa*’s centroid.

3.3 Local Refinement

To achieve optimal or satisfactory orientation results, we approximately align principal directions along some qualitative features through slight modification. The proposed algorithm performs *weighted* vector alignment for each opening centroid. If the component has more than one opening centroids, we align with regards to the closest principal direction by $angle = \alpha * weight$, where α is the angle between the given vector and its closest axis and weight $w = \frac{1}{|OC|+1}$ except from the opening centroid which lies at the root. This is assigned an increased weight : $2 * w$, such that $\sum_{i=1}^{|OC|} w_i = 1$.

3.4 Hierarchical Refinement

We extend the orientation improvements performed individually on each component by performing optimal fitting of the local principal directions of neighbor components. The algorithm starts from the children of the root node of the skeleton tree performing the same process to their children until it reaches the tree leaves. For each component, this process first aligns with regards to the parent’s principal direction which is closer to the child’s principal axis. Subsequently, from the rest of the child’s and parent’s principal directions we detect those that are closer and aligns them. Note that fitting two principal directions from different components, results in rotating the other principal directions too.

4 ADVANCED RIGID SKINNING

We have developed a novel technique to address the major flaws from self-intersections in areas around the joints providing sufficiently smooth skin deformation. Briefly, our approach first removes the skin vertices of the overlapping component parts and then adds new vertices to fill in the gap. Finally, for each frame it constructs a blending mesh that produces a smooth surface using a robust triangulation method.

4.1 Removing Vertices

The first step of our technique is to detect which vertices from each of the collided components are located inside each other. A fast approximate test is to use the oriented bounding box (OBB) since accurate methods require a lot of computational resources. We start from the boundary points to perform the fewer tests since they have the highest probability to be inside the other component’s OBB and move through their 1-ring of neighbors until we reach the points which are in the outer side of the OBB. Two points are 1-ring neighbors if they belong to the same facet.

4.2 Adding Vertices

Definition 3. There are three point sets that will use to construct the blending mesh:

- *Boundary* is the set of boundary points that have not been removed.
- *Replaced* is the point set that will replace the points that have been removed.
- *In-between* is the point set which consists of the average of the initial and final positions of the opening points which have been removed ($O_{i,initial}, O_i$ points).

We present two blending techniques; the first one called *front blending patch construction* uses only the *Boundary* set and the second called *rear blending patch construction* uses both the *Replaced* and *In-between* sets.

4.2.1 Front Blending Patch Construction: FBPC

The observation behind this process is that when we rotate a component about an arbitrary axis, movement of every point $B \in Boundary$ will describe a circular arc. The task is to find new points by interpolating from the point's position $(B_0 = q_0 B q_0^{-1})$ before the rotation q_0 to the point's position after rotation $(B'_0 = q_1 B q_1^{-1})$ q_1 (see Figure 4). We use fast linear interpolation of quaternions (QLERP) since the interpolation along the shortest segment hardly causes any observable skin defect (Kavan and Žára, 2005).

4.2.2 Rear Blending Patch Construction: RBPC

We have to fill in the gap in the area where we have removed points from both components. The major issue is that we only have the *Replaced points* from each component without any correspondence between them. So, we propose to use as extra point set, called *In-between* points.

A grouping function is developed to create *triplets* from each $InB_i \in$ *In-between* and one point from child's and parent's *Replaced* sets (R_j^C, R_k^P points, respectively). The feature of each triplet is that the plane defined by its points, has the minimum dihedral angle with the plane defined by the joint position and the kernel centroids of the participating components (plane PL). Then, we construct blending arcs given the triplet as control points using a Rational Bezier representation (Fudos and Hoffmann, 1996). The tangent vectors at R_j^C, R_k^P defined as the projections on PL of the vectors going from child's and parent's kernel centroids to the joint position, respectively. To prevent the generated meshes from exhibiting volume loss as joints rotate to extreme angles we replace InB_i point with V_m. We define V_m as the average of the InB_i point and the intersection of their tangents (I_p) and then evaluate the tangents (T^C, T^P) to the vectors going from R_j^C, R_k^P to InB_i (see Figure 4).

4.3 Blending Mesh

The points derived through the two blending processes constitute the scattered 3D data which will be triangulated. In this work, we use the *Tight Cocone* (Dey and Goswami, 2003) algorithm to construct water-tight surfaces. Since we need only two parts of the generated spheroid to fill the holes, we must eliminate a number of unnecessary facets to derive optimal lighting results. The evaluation of the patch point normal vectors is achieved by averaging the normals of the output surface facets and the normals of the pre-existant component facets that share the point.

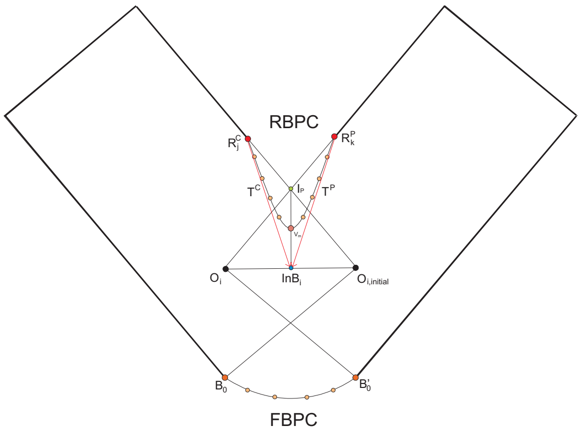

Figure 4: Introducing additional points and blending curves.

5 COMPLEXITY AND PERFORMANCE EVALUATION

The input to our framework is a segmented polyhedron model and its associated skeleton hierarchy information. At the following experiments we decompose characters into components using the *Blender* (*http://www.blender.org/*) software. A summary of the characteristing studied models (courtesy of (e-frontier, 2006; AIM@SHAPE, 2008)) is given in Table 1.

We evaluate our proposed algorithms with respect to two criteria: quality and performance. All tests were performed on an 2.4 GHz Intel Core Quad with 2GB of RAM.

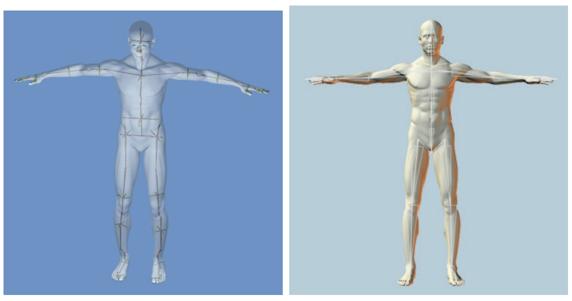
Figure 5: Comparison on a human model between our refined skeleton and a default hand-made skeleton.

Table 1 provides computation times for extracting skeletons from the above models. Most of the time

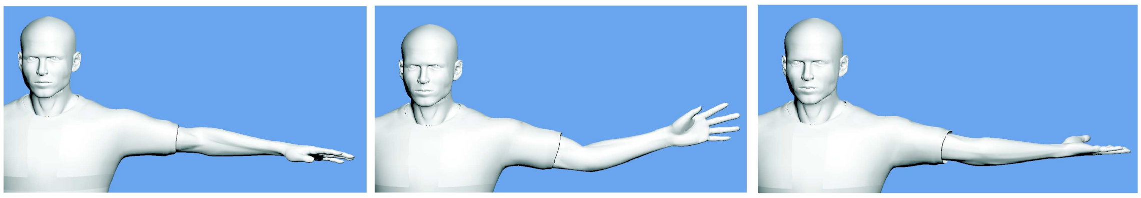

Figure 6: Avoidance of the typical LBS "candy-wrapper" artifact. (Left) is a reference pose, (middle) upper arm is rotated 90^o about its axis, and on (right) upper arm is rotated 180^o about its axis.

is due to the principal directions and kernels centroid computation which is performed in $\sum_{i=0}^{k} O(n_i \log n_i)$ and $\sum_{i=0}^{k} O(n_i \log n_i) + O(r_i \log r_i)$ time respectively, for a character with k components where the ith component has n_i vertices and r_i kernel points. The local and hierarchical refinement methods times are negligible compared to the overall performance. However, we observe from measures of Table 1 that skeleton extraction complexity appear to be nearly linear on the number of triangles.

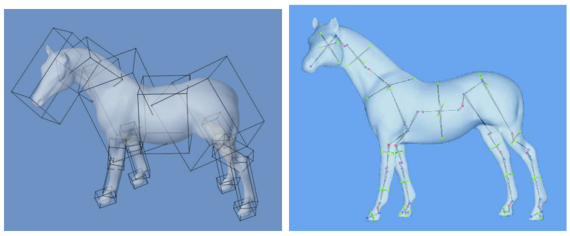

Figure 7: Refined oriented bounding boxes and skeletons of the components of a horse model.

It's difficult to compare our method with other recent methods, since the input model should be segmented. Figure 3 illustrates the qualitative superiority of our refined skeletons and aligned OBBs over the original principal axis algorithm. In general, our method generates refined skeletons in less than half a minute for dense models (Figures 3, 5, 7). Thus, if our method is used in conjunction with a fast decomposition method, it will be a very efficient overall process. Figure 5 illustrates the similarity between the default skeleton generated by *Poser* software and a skeleton extracted by our method.

Table 1: Experimental models and performance of their skeletonization steps (time measured in seconds).

Model	Vertices	Kernel Centroid	Principal Axis	Total
Cow	3825	1.982	0.868	2.995
Horse	8964	2.086	0.924	3.178
Dilo	26214	4.701	2.022	7.150
Human	74014	10.244	4.649	14.893

We create animations by re-targeting hand-made motion data to the skeletons extracted from the original meshes. Figures 6 and 8 show animated poses using our rigid skinning technique. We also provide snapshots depicting a closer view of a human knee mesh to demonstrate the robustness of the skinning process (Figure 1). These visual results confirm that our method is indeed free of artifacts exhibited by previous LBS methods working only on a single mesh.

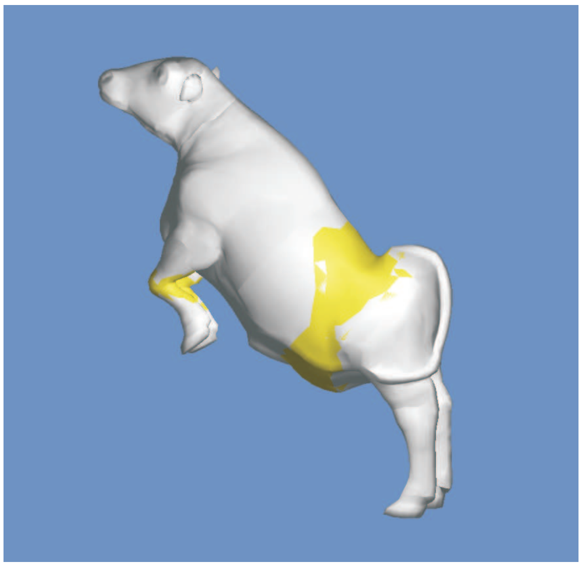

Figure 8: Cow multiple part animation. Patched parts are highlighted in yellow.

However, the overall processing time of our skinning method is very high for a real-time animation due to the computation of new points which takes $O(n_1 \cdot n_2 \cdot n_3)$ time to complete (n_1 and n_2 are the numbers of *Replaced* points of moving and its parent components and n_3 is the cardinality of *In-between* set) and the triangulation algorithm $O(n^2)$ complexity, where n is the number of the constructed points. For instance, Table 2 shows the time performance of our rigid skinning animating figure's 1 human's knee from initial pose to extreme angles where moving and its parent components consist of 891 and 1000 vertices, respectively.

Table 2: Rigid skinning performance of human's knee animation (time measured in seconds).

Frame	n_1	n_2	n_3	n	**Time**
1	26	28	24	64	0.390
2	27	28	23	93	0.421
3	26	31	22	116	0.456
4	27	32	23	150	0.510
5	28	34	24	188	0.572
6	35	37	24	214	0.654
Mean	28	32	23	137	0.501

6 CONCLUSIONS AND FUTURE WORK

We have proposed a robust skeleton-based animation framework for 3D characters. Approximate refinement algorithms have been presented to improve extracted skeleton's orientation. We have developed a novel rigid skinning method that eliminates the potential shortcomings from self-intersections, providing plausible mesh deformations.

Considering completeness there is need for further investigating in a more quantitative manner the grouping functions. Further, one could try replacing the angle-weighted algorithm with an optimization method which will compute the principal axis orientation more accurately.

Finally, to achieve real-time animation an alternative triangulation method should be employed and its GPU realization should be investigated.

REFERENCES

AIM@SHAPE (2008). *http://shapes.aim-at-shape.net/.*

Au, O. K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D., and Lee, T.-Y. (2008). Skeleton extraction by mesh contraction. *ACM Trans. Graph.*, 27(3):1–10.

Dey, T. K. and Goswami, S. (2003). Tight cocone: a water-tight surface reconstructor. In *ACM SM '03*, pages 127–134.

e-frontier (2006). Poser 7. *http://my.smithmicro.com/win/poser/index.html.*

Fudos, I. and Hoffmann, C. M. (1996). Constraint-based parametric conics for CAD. *Computer-aided Design*, 28(2):91–100.

Hejl, J. (2004). Hardware skinning with quaternions. *Game Programming Gems*, 4:487–495.

James, D. L. and Twigg, C. D. (2005). Skinning mesh animations. *ACM Transactions on Graphics*, 24:399–407.

Katz, S. and Tal, A. (2003). Hierarchical mesh decomposition using fuzzy clustering and cuts. In *ACM SIGGRAPH '03*, pages 954–961.

Kavan, L., Collins, S., Zara, J., and O'Sullivan, C. (2008). Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4).

Kavan, L. and Žára, J. (2005). Spherical blend skinning: a real-time deformation of articulated models. In *Proc. of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16. ACM.

Lewis, J. P., Cordner, M., and Fong, N. (2000). Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *ACM SIGGRAPH '00*, pages 165–172.

Lien, J.-M., Keyser, J., and Amato, N. M. (2006). Simultaneous shape decomposition and skeletonization. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, pages 219–228.

Liu, P.-C., Wu, F.-C., Ma, W.-C., Liang, R.-H., and Ouhyoung, M. (2003). Automatic animation skeleton using repulsive force field. *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 409–413.

Ma, W.-C., Wu, F.-C., and Ouhyoung, M. (2003). Skeleton extraction of 3d objects with radial basis functions. In *Proceedings of the Shape Modeling International 2003*, page 207.

Merry, B., Marais, P., and Gain, J. (2006). Animation space: A truly linear framework for character animation. *ACM Trans. Graph.*, 25(4):1400–1423.

Mohr, A. and Gleicher, M. (2003). Building efficient, accurate character skins from examples. *ACM Trans. Graph.*, 22(3):562–568.

Rhee, T., Lewis, J., and Neumann, U. (2006). Real-time weighted pose-space deformation on the GPU. *Computer Graphics Forum*, 25(3):439–448.

Wang, X. C. and Phillips, C. (2002). Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM symposium on Computer animation*, pages 129–138.