Journal Code: C A V | Article ID 1 6 8 7 | Dispatch: 03.12.15 | CE: Jan April Same
No. of Pages: 17 | ME:

COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (2015)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1687

RESEARCH ARTICLE

Q1
Segmentation-based skinning

Q2
Jorge Eduardo Ramírez Flores* and Antonio ~~Susin~~ Sánchez

MOVING – Universitat Politecnica de Catalunya (UPC), c/ Jordi Girona, 1-3, 08034 Barcelona, Spain

ABSTRACT

Skeleton-driven animation is popular by its simplicity and intuitive control of the limbs of a character. Linear blend skinning (LBS) is up to date the most efficient and simple deformation method; however, painting influence skinning weights is not intuitive, and it suffers the candy-wrapper artifact. In this paper, we propose an approach based on mesh segmentation for skinning and skeleton-driven computer animation. We propose a novel and fast method, based in watershed segmentation to deal with characters in *T-Pose* and arbitrary poses, a simple weight assign algorithm based in the rigid skinning obtained with the segmentation algorithm for the LBS deformation method, and finally, a modified version of the LBS that avoids the loss of volume in twist rotations using the segmentation stage output values. Copyright © 2015 John Wiley & Sons, Ltd.

KEYWORDS
mesh segmentation; skinning; weight assignment algorithm; rigging; computer animation

***Correspondence**
Jorge Eduardo Ramírez Flores, MOVING – Universitat Politecnica de Catalunya (UPC), c/ Jordi Girona, 1-3, 08034 Barcelona, Spain.
E-mail: jramirez@lsi.upc.edu

1. INTRODUCTION

Skeleton-driven animation is one of the most common 3D animation techniques used nowadays with applications in video games and film industry. The common pipeline begins with an artist sculpting a 3D character, creating the rig of the character mesh, and depending on the selected deformation method, a set of weights are associated from every joints of the skeleton to a specific part of the character's body. The described process is time-consuming and usually is hand made by the artist itself. The most difficult part in this pipeline is the weight creation, a proper result depends on a precise weight assignment. Another well-known problem related with the most popular deformation methods are artifacts generated at joint rotation. In linear blend skinning (*LBS*) we have the candy-wrapper artifact (a loss of volume associated with a twist rotation), dual quaternion skinning (*DQS*) avoids the candy-wrapper artifact of LBS but introduces its own artifact: the joint-bulging artifact (an artifact produced by the spheric nature of the quaternion interpolation). The main focus in this paper is mesh segmentation applied to skinning: we propose a novel and fast segmentation algorithm, that is going to be applied over a previously rigged mesh to organize a set of vertices by its spatial distribution. Each vertex in the input rigged mesh is assigned to a specific link of the underlying skeleton (segmentation stage), creating what is known as rigid skinning. A proper rigid skinning is a good starting point for a weight distribution algorithm, in [1,2] and [3] is used as starting point. We also propose a weight assignment algorithm based in the segmentation information of each vertex to create a simple and fast algorithm. Finally, we use the segmentation in the limbs of the character to create an algorithm based on LBS, but without volume loss on twist rotations, link-oriented twist scheme, and fast enough to be used in real time animations.

2. PREVIOUS WORKS

The main classification for segmentation algorithms according to Shamir [4] are *part-type* segmentation and *surface-type* segmentation. *Part-type* segmentation is oriented in partitioning the object into semantic components; *surface type* uses geometric properties of the mesh to create surface patches.

The most common application of mesh segmentation is skeleton extraction: an input mesh is taken and partitioned in segments that will represent a region that belongs to a skeleton bone; examples are found in [5,6] and [7]; in [5] is proposed a segmentation method that takes as input a set of meshes that represent an animated mesh sequence through time. The input mesh is segmented in patches that undergo approximately the same rigid transformation over time. In [6], the segmentation is based in a mapping function that creates level lines around concave areas that define area of the surface mesh. A region merge algorithm is used to prevent over segmentation; this algorithm creates a fine abstraction of hierarchy levels to merge segmented areas; the lower the hierarchy used, the lower the number of surface patches the object will have.

In [8], a segmentation method is used over rigged meshes applying Euler distances and a normal test over the surface of the input mesh; however, their segmentation method had the same goal as our method: create a weight assignment method for the LBS algorithm. They also use their segmentation algorithm to correct and preserve volume during rotations, but they are limited to twists rotations under 180°, because they are using LBS as deformation scheme, and their preservation method is applied after a joint rotation is performed with its consequent deformation; therefore, they cannot eliminate the candy-wrapper artifact. An evolution of this method is found in [3]; they improve the original method using a more robust segmentation algorithm based in voxelizing a closed mesh; then, a segmentation algorithm is performed using geodesic distances and adding deformation effects to their previous framework, but their twist rotation constraint is still present. A similar method is proposed in [9]; they also achieve mesh segmentation using a voxelization scheme to create a weight assignment algorithm for the LBS algorithm. Their voxelization algorithm is a novel method based in slicing a 3D mesh in the canonical axis directions to create a set of images that allows their algorithm to create a solid voxelized version of the input mesh. One of their most interesting features is that their algorithm can work with multiple meshes, and they are not limited to input closed meshes, but all the LBS deformation problems still remains.

Q3

Our approach is different than the classic segmentation methods that need input parameters to create a segmentation; our method is oriented to create a segmentation based on an underlying skeleton previously created, in a similar way as in [3,8] and [9], but our method works on the vertex positions as input to a region growing algorithm instead of voxelizing the target mesh or simply using Euler distances over models with ideal poses.

As mentioned earlier, LBS is one of the most popular skinning algorithms; it has been used widely in video games and film industry since 1998 [10]. In this method, the deformation for each vertex is the product of the sum of each joint in the skeleton multiplied by a weight to compute the final position of the computed vertex in a 3D mesh. This kind of computation is used extensively in methods such as *skeleton subspace deformation* [11], *enveloping* or *vertex blending* [12].

The main advantages of the LBS algorithm are its simplicity (based in a linear combination) and, as consequence, efficiency to compute (which leads to low processing times). This algorithm is used natively in professional animation software (such as AUTODESK MAYA, where it is called *smooth skinning*), but as is known, suffers from artifacts when some rotations are made by the influence joints, leading to the *collapsing elbow* [11] and the *candy-wrapper* [12] artifacts. These well-known artifacts cannot be prevented by any user painting the weights of a target mesh (usually the weight distribution are "painted" or assigned by an artist to achieve the desired effects; the automatic software made an initial approximation) because the artifacts are inherent to the LBS deformation scheme. Large number of works has been published about LBS; based in the way these methods compute the set of weights w_{ik}, they fall in one of the next categories:

- Example based: The number of papers developed within this category make it a very populated one [7,11,13–17]; all these methods' main idea is compute the weights of a 3D mesh by using a set of examples (a set of 3D meshes in different poses). The new poses will be the result of an interpolation scheme to compute the vertices' position on the target mesh. In the method known as multi-weight enveloping [13], an extension of the LBS is made by assigning more than one weight value per rigid transformation in the skinning main equation; a weight value is assigned to each element of the rigid transformation matrix. A more recent approach is the work described in [7]; the main idea of this work is using the proxy bones to compute the weights of the target mesh. In our developed method, we use the segmentation as base to generate a weight distribution algorithm using the neighborhood information of the skeleton's joints for each vertex. Q4
- Function based: These methods have two modalities:

  (1) Compute automatically the weights of the LBS.
  (2) Replace partially or totally the blending method (substituting the rigid transformation matrix by a different rigid transformation tool) or adding a correcting method to the deformation achieved with the LBS.

  In the first category, we find [18]; its main approach uses a nonlinear model to compute the weights of the LBS. The weights are computed using a polynomial function that is based in a quantity called *influence ratio r*. In [1], a function based in heat diffusion is used to compute the weights of the LBS algorithm, where the Laplacian of a discrete surface is applied over a vector, p^i is a vector using the initialization $p^i_j = 1$ (rigid skinning) if the nearest bone to vertex j is i and $p^i_j = 0$ otherwise. Finally, an H diagonal matrix is computed, which will have in H_{jj} the closest weight contribution to the vertex j. One of the latest methods to compute weights automatically is [19] where the weights are computed using a Laplacian energy function subject to an upper and lower bound constraint minimizer. This work is interesting in particular because of its general treatment of the problem; the weights are values that depend on elements called handlers; the handlers can be the elements of a cage (in 2D or 3D) or the joints of a skeleton. Therefore, it can be applied to images or skinning in 3D cage based or skeleton driven.

In the second category, we find approaches like [10], where we find a change in the interpolation method from LBS to *spherical blend skinning* (SBS); SBS uses quaternions to blend the final position of a deformed vertex; all the matrix in the LBS are changed to its equivalent in quaternions. A more sophisticated approach introduced by the authors of SBS is DQS [20]; in this work, dual quaternions are used to solve in an efficient way the well-known problems of the LBS. Dual quaternions are quaternions whose elements are dual numbers ($\hat{q} = \hat{w} + i\hat{x} + j\hat{y} + k\hat{z}$). Because of the equivalence of operations between quaternions and dual quaternions, a version of QLERP for dual quaternions is made (called dual quaternion linear blending; lately, it will be known as DQS). Interpolation with SBS and DQS eliminates the candy-wrapper artifact and produces better results than the LBS algorithm but produces what is known as the bulging joint artifact. The bulging joint is an artifact produced by the spherical interpolation that is caused by nature of quaternions; an interesting work that corrects this particular problem is [21], using rigid skinning to compute the vertex length to its main joint and correcting it when a rotation is performed. In [22], the candy-wrapper artifact is corrected by computing an additional weight $\delta\alpha_i$ per vertex based on the angles at animation and binding time. $\delta\alpha_i$ is used to compute a rotation matrix (restricted to the X canonical axis in the paper) that is multiplied prior to $M_{\delta k}$ in the LBS equation; also, an operation over the vertices to compensate the collapsing joint is introduced. This operation consists in choosing a collapsing joint; then the vertices affected by this particular joint will be recomputed by a stretch operation that is basically a vector length compensation from the chosen joint to the affected vertex. One of the methods that adds a post-processing to the LBS is [3]; the method adds a volume correction stage after the skinning deformation of a target mesh. The volume correction is treated as minimization problem of a correction vector u that is computed using Lagrange multipliers. A similar method can be found in [23]; this method is also a post-processing to correct the deformed volume obtained with LBS. The change of volume is computed by a displacement vector field and a scale factor applied to volume V. By solving the vector field, the correction of volume is performed over the deformed mesh; this method uses a set of new weights S to control the correction in a localized level; however, this method cannot solve the candy-wrapper artifact in LBS. An optimization method based on level of details can be found in [24], using LBS skinning a model's matrix transformation are precomputed based in its joints' hierarchy. If more detail in the animation is required, the method adjusts it progressively applying the matrix operations for the desired level of detail with no noticeable errors in the final animation. An extension of the LBS algorithm is found in [25]; the method uses two meshes to achieve advanced deformations. One with low level of detail that will control the final deformation of the detailed one in the control mesh, the LBS is applied and combined with bar-net deformers to achieve some degree of realism (physically based) in some selected vertices of the control mesh; the fine mesh is deformed using the wrapping method and can be directly manipulated or thought the bar-net if the user needs and advanced deformation effect such as wrinkles. Two of the latest methods are [26] and [27]. In [26], blend bones are used to approximate nonlinear skinning with a set of weights computed specifically to work with the extra blend bones. Kavan and Sorkine [27] take an interesting approach by using two deformers depending on the kind of rotation. For twist rotations, they use a quaternion deformer, and linear blend deformation is used for any other kind of rotation. Each deformer had its own set of weights that are computed and optimized using examples for some representatives poses, using biharmonic weights as base.

To help to understand the main features of some of the main skinning algorithms, we have created Table I. **T1** Q5

Our proposed method relays on the segmentation to create a weight distribution algorithm and a deformation scheme without the candy-wrapper artifact. We use the output of the *delta value* (δ) as a normalized distribution value to compute the progressive change in the twist angle for each segmented limb (a link between the joints of the skeleton).

Our segmentation algorithm is based in part-type segmentation, using a previously rigged mesh that can be created by an artist or by an automatic method such as [28]. In our particular case, the semantic (the elements that represents a limb) parts of the part-type segmentation are already created: the underlying logic skeleton. Therefore, our algorithm is not a full automatic segmentation algorithm, and a benchmark such as the one described in [29] is not viable because of our dependency in a pre-defined skeleton and our lack of explicit control parameters. Our method is created with the purpose of detecting vertices that belong to each semantic part, in this case: the link between a joint and its child; therefore, we do not have a set of control parameters; our method is implicit, and our only control parameter is the skeleton that has been created previously. The number of segments of the segmented mesh will have a direct relationship with the number of *bones* or *links* that our skeleton rig had; the same model will have different segmentations if our method is applied to different skeletons bounded each time to the same mesh.Although our algorithm works for non-closed

Table I. Weight distribution and skinning methods features comparison.

Features	[18]	[1]	[19]	[20]	[10]	[22]	[23]	[3]	[26]	[27]
Nonlinear polynomial weight function	✓	✓	✓	×	×	×	×	×	×	×
Heat-diffusion weight function	×	✓	×	×	×	×	×	×	×	×
Rigid skinning pre-proc.	×	✓	×	×	×	×	×	×	×	×
Quaternion based	×	×	×	✓	✓	×	×	×	✓	×
Linear blend skinning matrix modification	×	×	×	×	×	✓	×	×	×	×
Post-proc. volume correction	×	×	×	×	×	×	✓	×	×	✓
Nonlinear approx., using additional bones	×	×	×	×	×	×	×	✓	×	×
Two deformers combination	×	×	×	×	×	×	×	×	✓	×

Methods: realistic skeleton driven skin deformation [18], Pinocchio [1], bounded biharmonic weights [19], dual quaternion skinning/dual quaternion iterative blending [20], spherical blend skinning [10], strech-it [22], volume-preserving mesh skinning [23], exact volume preserving skinning with shape control [3], automatic linearization of nonlinear skinning [26], and elasticity-inspired deformers for character articulation [27].

meshes and characters that are defined in multiple meshes, we will explain the method with the assumption that we have as input a single closed mesh. We have chosen a region growing method, because it is fast and depends directly on the number of vertices in a 3D mesh. Works like in [3] and [9] are similar to our method but relays in a voxelization of the input model. A voxelization process can be very time-consuming depending in the voxel size, and as is explained in [9], the main problem is to know which voxels are internal voxels to produce a solid model. One of our objectives is to propose a novel and fast method that can be used in a model with an arbitrary pose (not constrained to an ideal *T-Pose*). In our proposed weight assignment method, input mesh segmentation is the base, but we use it at a high level, because we store for each vertex their particular segmentation information to create a fast weight assignment, our approach is to create the segmentation and the weight assignment as two separated methods, because we also use the segmentation to create a new set of weights to solve the candy-wrapper artifact effectively.

Our algorithm is composed of three stages:

(1) Region growing. In this stage, we assign to each vertex a set of segments that can be the segment where it belongs.
(2) Belonging test. For each candidate segment in a vertex, defined in the previous step, we apply a set of rules to discriminate which is the most suitable segment to be assigned.
(3) Region merge. If false positives exist, we merge them with one of their surrounding neighbor regions.

In the following section, we are going to describe in detail each stage.

3. METHODS

Before we start explaining our segmentation algorithm, we will explain a key feature of our method: The mapping of the skeleton to a tree data structure (Figure 1). Skeleton mapping allows us to traverse the skeleton hierarchically, the mapping is not performed by joints but by joint pairs, that is, a joint jn_a and its child jn_b define a node in our tree data structure (a segment that we will reference as s_j) that has end joints as special cases. Therefore, a skeleton will have m number of segments for a skeleton with n joints, with $m > n$ (m is greater than n because the end nodes are counted as segments of its own) and will be related directly by their hierarchy depending their position within the tree data structure.

3.1. Region Growing

Our algorithm starts using one of the root node related segments as initial growing region. We test if a vertex v_i belongs to a segment defined by the joints jn_a and jn_b, using their coordinates to compute the orthogonal projection value δ as defined in [23]:

$$\delta = \frac{(v_i - jn_a) \cdot (jn_b - jn_a)}{\|jn_b - jn_a\|^2} \qquad (1)$$

The δ value classifies the relative position of the point and the segment: $\delta < 0$ if the point projection is before the segment, $0 < \delta < 1$ if the point projection is inside the segment, and $\delta > 1$ if the point projection is after the segment.

We apply our test for each segment in the skeleton hierarchically, using the delta function combined with region grown as tool to check if this vertex is candidate to being inside a segment for each vertex traversed.

Region growing needs a seed to begin with,; for the first segment, the seed can be a manually chosen vertex, or can be the closest vertex to the root node that belongs to the initial segment measured in Euler distance. In our case, we use a vertex that had a delta value between 0 and 1 as seed (the vertex is in the influence space of the segment). If the value of delta is greater than 1, we compute the delta value output for the child segment (next segment in hierarchy). If its value is not in the child segment influence space, we mark it as candidate for being part of the current

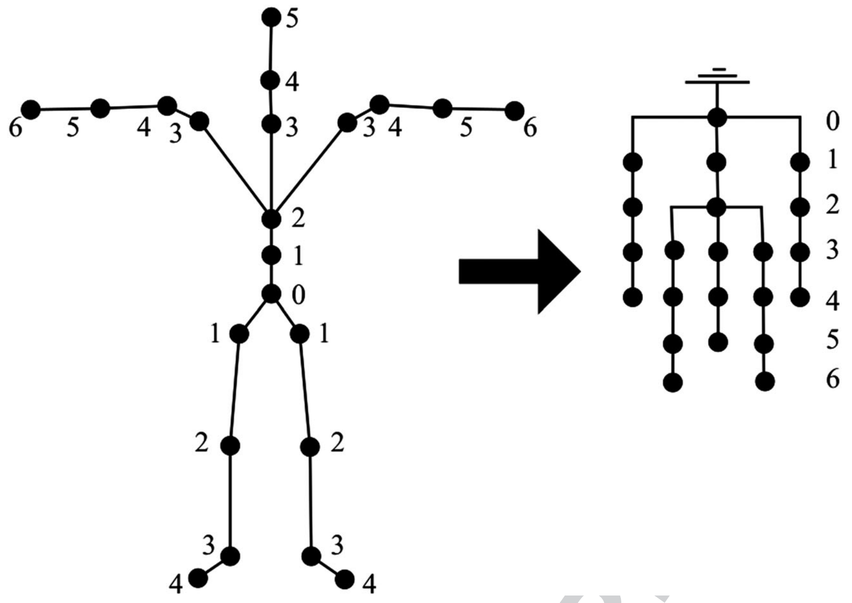


Figure 1. Equivalence between a logic skeleton used in animation and a *n*-ary hierarchy tree.

segment; therefore, only vertices with delta value greater than 0 can be candidates if all the conditions are accomplished, and as consequence, only a portion of the vertices of the mesh are checked per segment. The region-growing method marks as candidate a vertex for a specific segment; therefore, when all the segments are computed, we will have a list of segments for each vertex v_i, being v_i a candidate vertex to be part of the influence space of a segment s_j.

3.2. Belonging Test

F2 Because of the nature of the delta value, some vertices that are not part of a particular segment are marked as candidates (Figure 2). Therefore, a discrimination of segments in a candidate vertex v_i is needed. We apply the following test to select the segment s_j in the segment list denoted by Ls for each vertex v_i:

- For each segment s_j in the segments list Ls, we compute the angle θ_{ij} between the weighted normal np_i (the mean value of the sum of the normals of the vertex and its $1-$*connected* neighbors) and the vector $\vec{v_i s_j}$ (the vector that had the shortest distance d_{ij} from a segment s_j to the vertex v_i).
- The segments with an angle $\theta_{ij} > 90$ are discarded.
- End nodes cannot be discarded by the anterior rule.
- We assign the vertex v_i to the segment s_j with the lowest distance d_{ij} from the segment to the vertex.

This simple set of rules allows us to apply our algorithm in meshes that are not in the ideal *T-Pose*; as can be seen in Figure 4, it can be applied to rigged meshes with arbitrary poses.

3.3. Region Merging

F3 Figure 3(a) shows an example of vertex assignation with our *region-growing* algorithm and the *belonging test* in an arbitrary pose that results in *false positives*. This problem is caused because of the orientation of the pondered normal of some vertices with the vector $\vec{v_i s_j}$, and it depends basically on the face orientation in some vertices; we solve this problem using region merging.

Our region-merging method creates for each region s_j (corresponding to a segment) a list with subsets of vertices connected; we basically create subsets of vertices interconnected in a segment region. The largest subset in the list will be the definitive set for the computed segment region s_j; the remaining subsets will be merged, each one with its largest neighbor region (the region that had the highest number of vertices connected with the analyzed subset).

The complexity of our segmentation method is $O(Sn^2)$, being n the number of vertices in a 3D mesh, and S the number of segments created from a skeleton, the O complexity analysis of our method is included in Appendix A.

4. SEGMENTATION-BASED SKINNING

In this section, we will explain how our proposed segmentation algorithm can be used for automatically generating vertex weight information for skinning.

The main advantage of generating weights based on our segmentation algorithm is that we have identified already the main influence joint for each vertex in a mesh, which also applies for the case of meshes with arbitraryposes

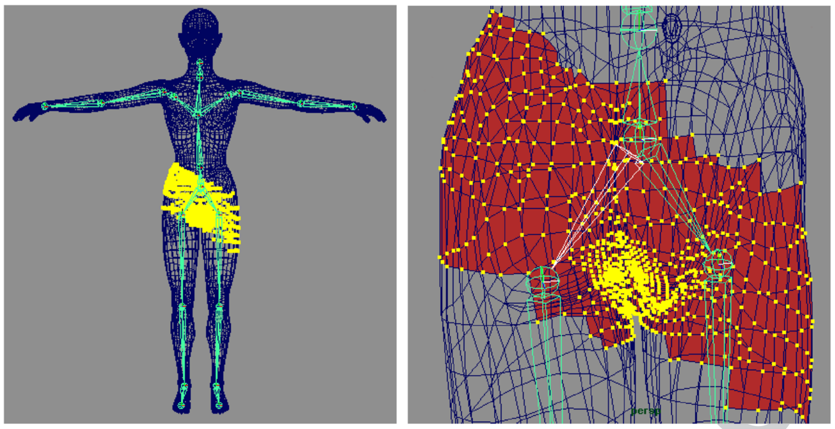

(a) Vertices of the right hip segment. (b) Right hip influence space (link in white and selected surface in red color).

Figure 2. Candidate vertices for the right-hip segment selected by our region-growing algorithm before the belonging test.

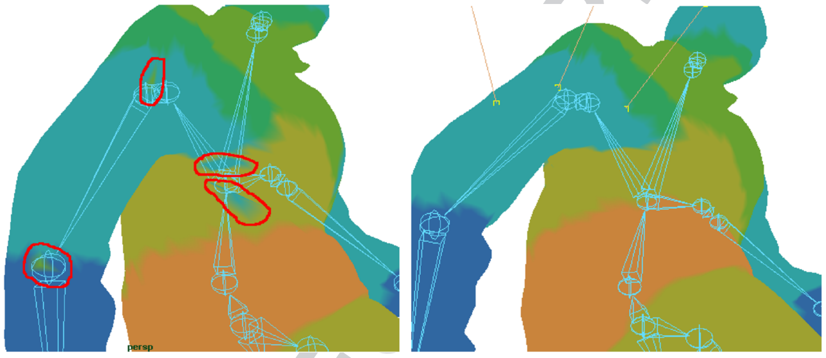

(a) (b)

Figure 3. Comparative between the same mesh before and after applying the merge region procedure; (a) false positives in a segmented mesh and (b) segmented mesh after merge region (some pondered normals in yellow).

F4 (Figure 4). Then, the weight generation is computed hierarchically involving only the joints that had a direct relation with the main joint, instead of distributing the weights by a geometrical method, where the influence joints are computed by its distance to a vertex [1]. As an example, in the automatic weight algorithm used by AUTODESK MAYA Q14 F5 artifacts are created (Figure 5); apparently the weights are calculated using a sphere with its center in the current vertex using Euclidean distances in the weight computation for each influence joint.

Using the segmentation algorithm described in the previous section for each vertex v_i, we had stored in a data structure the main influence joint jn_k. Then, as we will show next, we use a distance metric to compute the weight for the main joints an its "siblings". We express the distribution function for each weight as $w_{i_k} = F(d_{i_k})$; in our specific case, we use the normalized projection of a vertex v_i over the skeleton's links.

4.1. Selection of the Distribution Function and Distance Function.

All the algorithms that calculate automatically the weights for the LBS have explicitly or implicitly two components:

(1) *Distance function* (*DstF*). A function that calculates a number. This number can be a direct or indirect relation with a kind of distance from a vertex to the main influence joint(and consequently the main influence link). The distance function is important because its output will be used directly by the distribution function; therefore, a function that calculates the Euclidean distance will give us a different output that another one uses a geodesic distance.

(2) *Distribution function* (*DtbF*). This function takes a set of numbers and maps it to a set of values between 0 and 1. Its output will be the weights assigned to a vertex for each joint of the character. If the

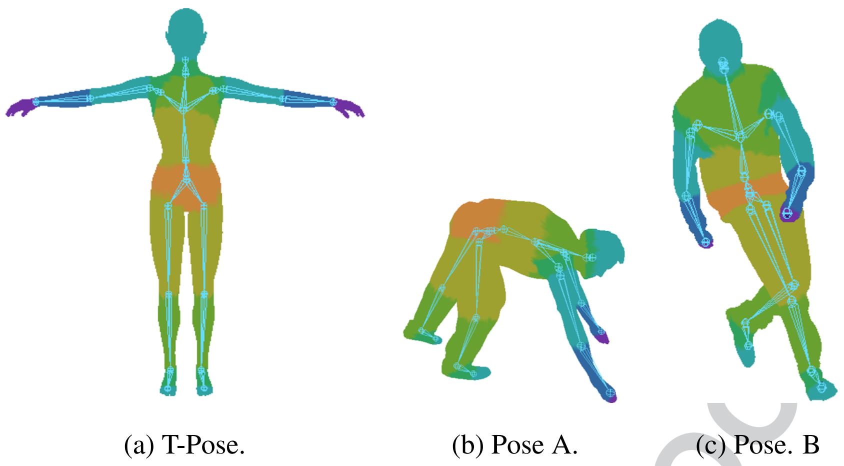


Figure 4. Segmentation in meshes with different poses, our segmentation algorithm is not restricted to a specific pose to produce an adequate mesh segmentation.

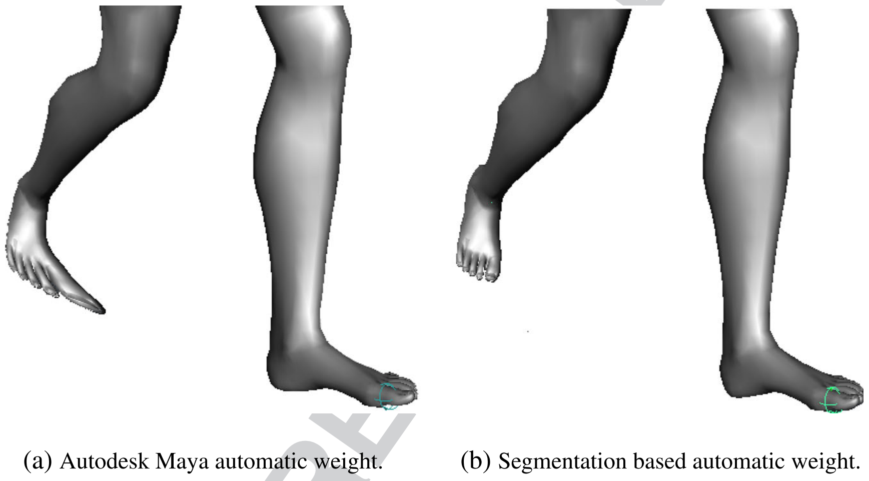

(a) Autodesk Maya automatic weight.
(b) Segmentation based automatic weight.

Figure 5. Artifact generated in an animation frame of a mesh (a) because of an improper weight assignation by the AUTODESK MAYA automatic weight assignation algorithm, corrected by our segmentation-based weight assign algorithm (b).

sum of the weight values is different from 1, the produced deformation will have artifacts depending on the influence of each joint. The distribution function is the most important part in the weight computation for the LBS algorithm; the deformation behavior of a mesh depends on the distribution function. Any function can be used as distribution function, but the quality of the output deformation can change according to the chosen function. In [1], a heat equilibrium equation is used as ~~*DstF*~~.

4.1.1. Used Distance and Distribution Functions.

In our particular case, we use the δ value described in Section 3.1 as *DstF* (a consequence of the segmentation), because its output is a normalized measure based in distance of the projection of the vertex over the link instead of the Euclidean distance from the vertex to a joint that is more dependent of the shape of the input mesh.

As distribution function (*DtbF*), we use a Gaussian function, defining the center of the function in the center of the main influence link to create a skinning behavior mostly rigid in the center of the links and smooth in the joint areas. Our *DstF* is defined as

$$f(x) = ae^{-\frac{(x-0.5)^2}{2c^2}} \qquad (2)$$

where a is the maximum value of $f(x)$ and x is the output value from the distance function. The inflection point is controlled by the value of the constant c; c is important because if we choose the incorrect value (an extremely low or a big value), we might have rotational continuity artifacts. The values computed by the function $f(x)$ are the weights for each influence joint in a particular vertex v_i, after assigning the weight values, we normalize it; if the weights are not normalized, artifacts are produced because of the sum greater than 1 in the influence joints weight values.

In our test over multiple meshes, we had used a hierarchy value of 1; for most of the vertices, this produces three influence joints for each vertex, which is the number of influence joints that is usually used for the vertices of a rigged character. The values used for the Gaussian function are $a = 1.3$ and $c = 0.25$; these parameter values generate the best results in our tests.

5. SEGMENTATION-BASED LINEAR BLEND SKINNING

In LBS, one of the main problems of this widely used deformation scheme is the "candy-wrapper" artifact. The candy wrapper in plain words is a loss of volume over the mesh caused mainly because of the abrupt change of position between two sets of vertices that is generated when a link rotates more than 60° and is at its maximum when the rotation reaches over 180° in the axis that is aligned with the link direction (Figure 6(c)). The LBS is basically a weighted sum of the set of vectors for each vertex of a polygonal mesh. A more detailed explanation can be found in [11] and [13]

F6

5.1. Our Approach.

To eliminate the loss of volume in a rotation over the link vector, we will avoid the abrupt change of angle in a twist rotation, which is the main reason of volume loss in LBS. Our approach is based on keeping the same rotation angle over all influence joints in a vertex when a twist rotation is applied; although the rotation angle will be the same for all the joints in a vertex, this angle will be assigned progressively depending on its projection on the link segment (the closest link) using δ.

In our modification over the LBS deformation scheme, we use the segmentation algorithm. A segment s_j is defined by two joints as main components: jn_a and jn_b. The vertex with "lower" hierarchy will be the main joint; therefore, twist rotations over jn_a in a segment will be applied normally. When a twist rotation is made over the joint jn_b, we compute the rotation angle progressively for each vertex on that specific segment. As can be seen in Figure 7, when we made a 180° rotation over its link axis, the LBS applies the deformation in two segments (Figure 7(a)), but our deformation scheme is applied only in one segment

F7

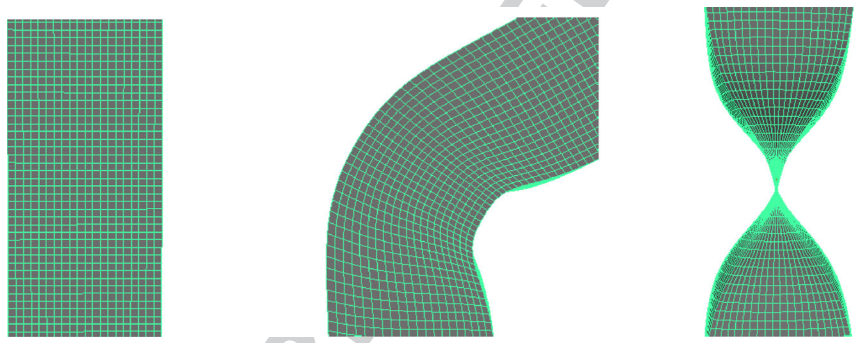

(a) Original mesh.
(b) Rotation of 60° in the x axis.
(c) Rotation of 180° in the y axis.

Figure 6. Linear blend skinning deformation method applied to a cylinder (a), a proper deformation in the x-axis (b), and the candy-wrapper artifact produced by a twist rotation (c).

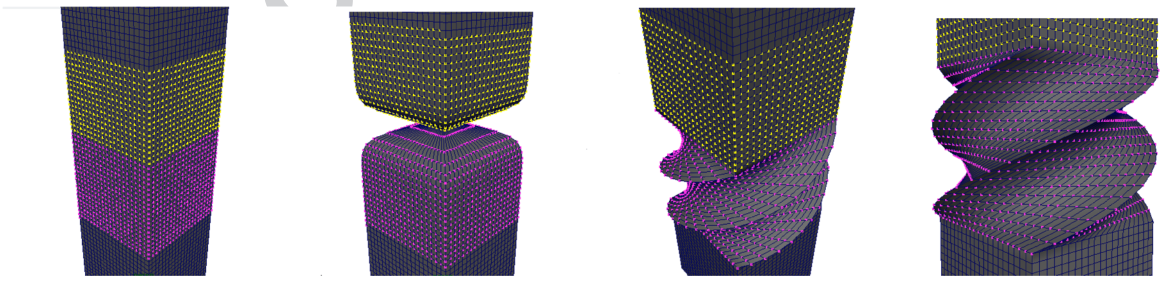


Figure 7. Deformation of a bar (a) with linear blend skinning (b), the deformation is applied in two segments with its respective loss of volume produced by a twist rotation greater than 180°. In our approach, we apply the deformation solely in one segment, preventing the loss of volume and self intersection (c) and (d).

(Figure 7(b)); we believe that this is a more natural behavior if we take the way of how a human limb deforms itself.

5.1.1. Simplest Case.

To explain our deform scheme, we start with the simplest case: a link with the same direction of the canonical axis. A twist rotation over a link axis can be classified by the hierarchy that has the rotating joint jn_k in the segment s_j of a particular vertex v_i.

(1) Our segmentation algorithm is applied to the target mesh obtaining an additional weight that will be the value $\delta(v_i)$ for each vertex and its assigned segment s_j. This value will be stored to be used in step 4.

(2) For a vertex v_i in the target mesh, all the joints that influence v_i are stored and sorted in a list by its hierarchy. Inside our list of influence joints, every time a child joint is added, its hierarchy will be increased by one of its father hierarchy value.

(3) If a rotation with angle θ_i is performed over the joint jn_b of the segment assigned to v_i, then θ_i is stored for computation. As we had mentioned, a segment made by the joints, jn_a and jn_b $(\overrightarrow{ba})$, is parallel to the canonical axis in this case.

(4) For a joint jn_k, which is also the joint jn_a in the segment s_j assigned of a vertex v_i, we will compute the rotation angle as $\theta'_i = \theta_i \delta(v_i)$. The rotation matrix $\mathrm{Mjn}_{\delta \mathrm{k}}$ is computed with θ'_i; in the chain of rotations, $\mathrm{Mjn}_{\delta \mathrm{k}}$ will be multiplied by $\mathrm{Mjn_k}$:

$$\mathrm{Mjn}'_{\delta \mathrm{k}} = \left(\prod_{i}^{k} \mathrm{Mjn_i} \right) \mathrm{Mjn}_{\delta \mathrm{k}} \tag{3}$$

(5) The joints with different hierarchy than jn_k need to be rotated with the same angle of the segmented link axis; then, the expression for any joint with hierarchy lower than jn_k will be

$$\mathrm{M}'_{\delta \mathrm{k-n}} = \left(\prod_{0}^{k-n} \mathrm{Mjn_i} \prod_{k-n+1}^{k} \mathrm{Mjn'_h} \right) \mathrm{Mjn}_{\delta \mathrm{k}} \tag{4}$$

where $\prod_{k-n+1}^{k} \mathrm{Mjn'_h}$ is an iterative product of rotation matrices that will have only rotations over the link axis of every joint with higher hierarchy between the joints jn_{k-n} and jn_k.

(6) Joints with higher hierarchy than jn_k will have the same rotation of the assigned link axis. In a similar way as in the previous point, any joint with higher hierarchy than jn_k (jn_{k+n}) needs to be multiplied by the negative angle of the link axis of each of the previous joints. The expression for a joint with higher hierarchy than jn_k will be

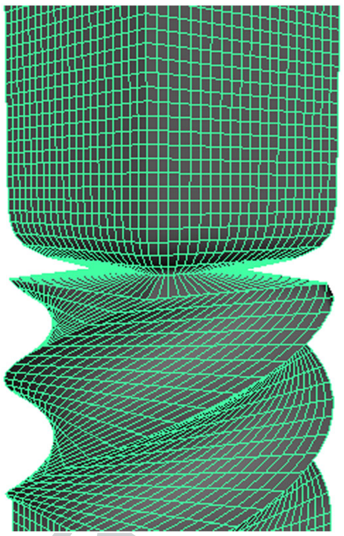


Figure 8. Candy-wrapper artifact in the upper part of a segment produced by not applying our method to joints with hierarchy greater than j_n.

$$\mathrm{M}'_{\delta \mathrm{k+n}} = \left(\prod_{0}^{k+n} \mathrm{Mjn_i} \prod_{k+1}^{k+n} \mathrm{Mjn'_h} \right) \mathrm{Mjn}_{\delta \mathrm{k}} \tag{5}$$

where $\prod_{k+1}^{k+n} \mathrm{Mjn'_h}$ is an iterative product of rotation matrices that will have negative angle rotation over the link axis of every joint with lower hierarchy between joints jn_{k+1} and jn_{k+n}.

(7) If a rotation over the parent segment of v_i assigned segment is performed, a rotation over the joint jn_a of the segment has been made. This rotation needs to be corrected; otherwise, the candy-wrapper artifact will be present again (Figure 8). To prevent this situation, the chain of rotations for influence joints with hierarchy lower than jn_a must be multiplied by the rotation chain described in Equation (4) but with the rotation matrix $\mathrm{Mjn}_{\delta \mathrm{k}}$ as identity $(\theta = 0)$. F8

As has been mentioned previously, the idea to avoid the loss of volume is based on having the same rotation of the segmented link in all the influence joints when a rotation is performed but only if that rotation is over the segmented link of a vertex. As can be seen in Equation (4), the rotation expressed by $\prod_{k}^{k-n+1} \mathrm{Mjn'_h}$ product has to be applied before any rotation from the skeleton space to the world coordinates has been performed. If the product is applied after the original set of rotation $\left(\prod_{0}^{k-n} \mathrm{Mjn_i}\right)$, volume loss artifacts are produced. Putting all the previous cases and information in one expression, we obtain

$$p'_i = \sum w_m \mathrm{M}'_{\delta \mathrm{m}} \mathrm{M_{Lm}} \mathrm{p_i} \tag{6}$$

where $\mathrm{M}'_{\delta \mathrm{m}}$ will be $\mathrm{Mjn}'_{\delta \mathrm{k}}$, $\mathrm{M}'_{\delta \mathrm{k-n}}$ or $\mathrm{M}'_{\delta \mathrm{k+n}}$ depending on the hierarchy of the joint.

5.1.2. General Case.

The general case of our method takes into account that a limb of a virtual character is not always aligned with the canonical axis; therefore, when a rotation over a limb is made, the rotation had to be over the axis made by the link of the joints jn_a and jn_b. In Equation (6), we made the assumption that this link is aligned with one of the axis; in the general case, we will take the axis made by the link and we will rotate around it the segmented vertices of the mesh. The rotation over an arbitrary link can be carried out with quaternions or with its equivalent in matrix rotation computed by Rodrigues rotation formula using the following expression:

$$v_R = v\cos\theta + (u \times v)\sin\theta + u(u \cdot v)(1 - \cos\theta) \quad (7)$$

if we express the cross product $(u \times v)$ as the rotation matrix $\mathrm{M}_{\mathrm{u}\times}\mathrm{v}$, the Rodrigues formula can be expressed as

$$R_u = I + \mathrm{M}^2_{\mathrm{u}\times}(1 - \cos\theta) + \mathrm{M}_{\mathrm{u}\times}\sin\theta$$

therefore, $v_R = R_u v$.

When a rotation is applied in a link l_k with an arbitrary position, we use the next procedure to compute the rotation over a vertex.

(1) We identify the nearest axis ax_k in orientation with l_k.
(2) The angle θ is extracted if a rotation over the ax_k exists.
(3) If the angle between ax_k and l_k is greater than 90°, we take the negative value of θ as the rotation angle using the negative of $\mathrm{M}_{\mathrm{u}\times}$.
(4) We apply Equations (3), (4), or (5) depending on the case, but $\mathrm{M}'_{\delta\mathrm{m}}$ is replaced with R_u with $\theta'_i = \theta_i\delta(v_i)$ as rotation angle and $\|l_k\|$ as u, where l_k will be the link of the segment where the influence joint jn_k is assigned.

Our approach is similar to [22] but with substantial differences: we use the δ value to obtain an additional weight that had the purpose of being the rotation amount of the twist rotation; our rotation scheme applies twist rotations over segments instead of the classic way that is over the influenced joint vertices. We apply our deformation correction in more than one influence joint down and up in hierarchy, and finally, we are not restricted to canonical axis only.

6. RESULTS

T2 As can be seen in Table II, the times of our segmentation algorithm depends on the number of vertices of the input model mesh. The weight assignment algorithm uses the segmented vertices; therefore, the segmentation method used on the input mesh does not have an impact in the weight assignment processing times, but their output values depend on the segmentation output. All the programming and test of our algorithms where performed in an *Intel Core i3* at 2.1 GHz, 6 GB in RAM, Windows 7 with Visual Studio 2010 with a 64 bits *C++* compiler and tested over AUTODESK MAYA 2014.

Table II. Segmentation processing times.

Model	Num. Vert.	Seg. (sec.)	Weight Assg. (sec.)
Low res.	6488	0.39	16.53
Mid res.	15 576	0.88	21.45
High res.	172 974	32.3	293.51

Table III. Comparison between deformation methods (processing times).

Rotation #	DualQuat (ms)	LBS (ms)	SLBS (ms)
1	0.062	0.052	0.055
2	0.071	0.041	0.054
3	0.056	0.053	0.053
4	0.054	0.044	0.054
5	0.063	0.055	0.055
6	0.061	0.041	0.056

LBS, linear blend skinning; SLBS, segmentation-based linear blend skinning.

In the skinning world, it is customary to compare the performance of a new proposed skinning algorithm with the most popular algorithm because of its simplicity and its linear nature: LBS, probably LBS has the best performance of all the skinning algorithms used up to date. We also compare the performance of our algorithm with another popular skinning solution: DQS. DQS has a lower performance than LBS, but it solves one of its main problems: the well-known candy-wrapper artifact. Therefore, these are the two main algorithms to compare a new proposed method in the field.

As we have carried out in Section 6.0.3, a sequence of six deformations is applied to a test model (a bar) and are reported in Table III. The implementation of our segmentation-based LBS for general purpose can be almost six times slower than LBS; in our test, we use an optimized version of segmentation-based LBS for three segments (the main one, their father, and child) applying it to a total of four joints. All the remaining influence joints with or without twist rotation will be solved with LBS; four joints are the usual number of influence joints for almost all vertices in a rigged mesh; even if that is not the case, the influence weight is commonly pretty low for joints related in second degree to the main segment's joints that can be solved with LBS without affecting the output (as can be seen in Figure 9). With the optimization, our algorithm is very fast having a difference of 0.01 milliseconds with LBS, and it is faster than DQS; in terms of quality, our algorithm shows the proper results without the candy-wrapper artifact of the LBS or the artifacts caused by the weight distribution showed in the DQS algorithm that are solved properly with dual quaternion iterative blending (DIB). T3 F9

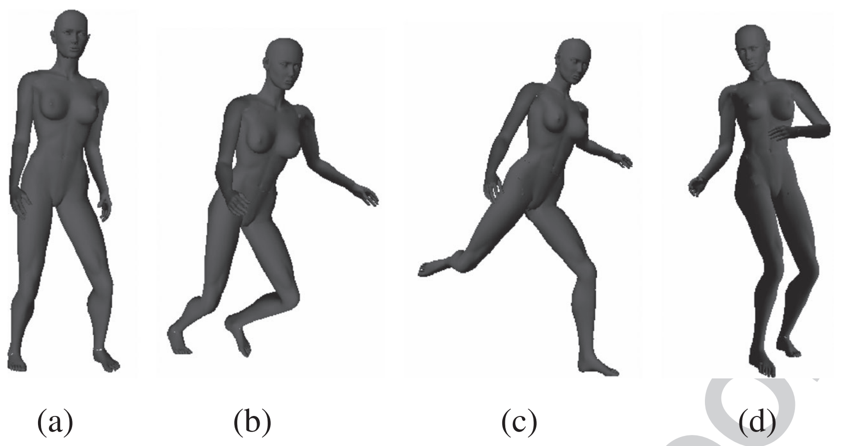


Figure 9. Segmentation-based linear blend skinning output applied over an animated character in different frames over time (character hitting a soccer ball).

6.0.3. Volume Preservation.

To know how much volume is lost when a rotation is made with our method, we have made a set of rotations over a mesh. The volume is computed using a tetrahedron representation with negative volumes (a negative volume will be computed for each face in the input mesh with negative direction in its surface normal); the result is in the next table. In all cases, the set of weights for the deformation methods are the same. We apply six rotations over the joints jn_1 to jn_3 of the five joints in the bar mesh with an initial volume of 32 units, leaving left of the test the end joints (jn_0 and jn_4). The set of rotations are planned to show the behavior of every deformation scheme; the results that are shown in Table IV are error percentages, where $e_i = \frac{V_0 - V_i}{V_0}$. The rotations in sequence are T4

(1) 180° in the y axis, joint jn_1.
(2) 200° in the y axis, joint jn_2.
(3) 120° in the y axis, joint jn_3.
(4) 90° in the x axis, joint jn_1.
(5) 60° in the z axis, joint jn_2.
(6) 80° in the z axis, joint jn_3.

Table IV. Comparison between output volumes from deformation methods (error percentage).

Rotation #	DualQuat (%)	LBS (%)	SLBS (%)
1	0.026	6.516	0.108
2	0.657	12.236	0.242
3	1.022	16.572	0.303
4	0.941	14.985	1.974
5	0.676	13.938	2.719
6	0.870	13.404	3.934

LBS, linear blend skinning; SLBS, segmentation-based linear blend skinning.

In Table IV, as expected, the method that had lost volume the most is LBS, followed by our method with DQS with the best performance of the three methods. Figures 10 and 11 show the surface areas were DQS and LBS operates, one with the test performed and a new test of a 180 twist rotation over the arm of two characters; the surface areas are lower in each one than the area were our method operates because the main weight distribution is the same for all the methods. In our method, we have two set of weights: F10 F11

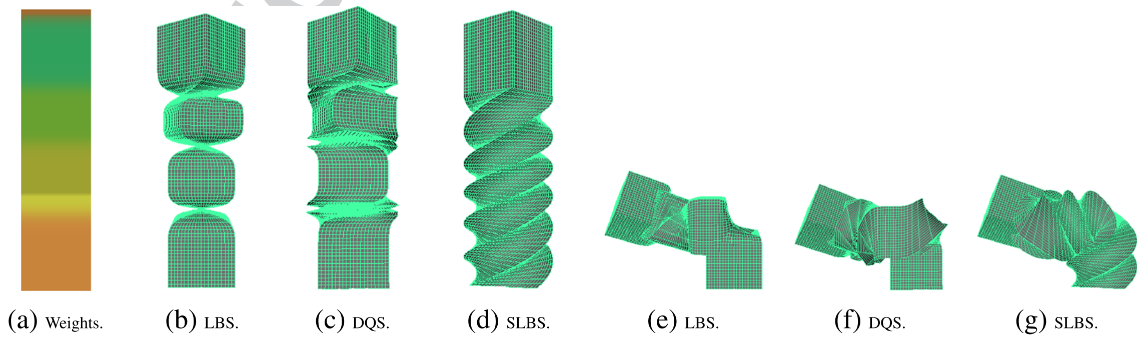


Figure 10. Output volumes for different deformation methods. From (b) to (d), rotation 3 and from (e) to (g), rotation 6. LBS, linear blend skinning; DQS, dual quaternion skinning; SLBS, segmentation-based linear blend skinning.

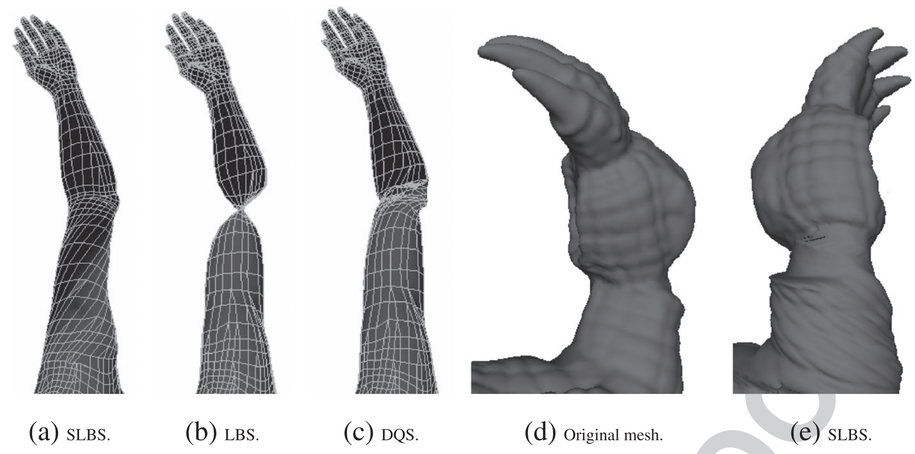


Figure 11. Deformation over an arm for two meshes performed by different deformation methods, with a 180° rotation. SLBS, segmentation-based linear blend skinning; LBS, linear blend skinning; DQS, dual quaternion skinning.

Table V. Comparison between output errors from two models with different volume magnitude.

Rotation #	20.8 units model (%)	72 units model (%)
1	0.092	0.108
2	0.266	0.242
3	0.343	0.303
4	1.967	1.956
5	2.673	2.714
6	3.898	3.943

the main (taken directly from LBS) will have effect over all the rotations that are not aligned with the segmented link axis (twist rotations); the second set is obtained from the δ value. If a behavior different than the lineal one obtained through the δ value is desired, an output function must be applied over the results δ. Our method can manage degenerated cases such as rotations equal and greater than 180° because of its progressive nature. As an example of this feature, in LBS, if a rotation angle θ is greater than 180°, θ will be equivalent to the difference between θ and 360°; in general, the rotation angle θ in LBS will behave by the relation $\theta' = \theta - 360\left(\lfloor\frac{\theta}{180}\rfloor - \lfloor\frac{\theta}{360}\rfloor\right)$; in DQS, the rotation about 360° produces serious artifacts as is showed in [26]; only DIB produces a correct output. With our method, this degenerated case is properly solved, because θ is changing smoothly between vertices by the δ value, instead of changing θ depending on the weights values.

To test how stable is our deform method, we have modified the bar model; we have made two modifications: one varying tuning down the total volume of our bar and other increasing the volume. The same set of rotations had been applied to these modified models; the results are shown in Table V.

As seen in Table V, the variation between the two models are indicative of a stable method. When the results of the set of rotations of the original model (Table IV) and the result of the output errors on the modified volume models are compared, the output errors are similar.

6.1. Discussion

The segmentation in the automatic rigging algorithms [2] and [30] shares one main feature in their segmentation: the segmentation is part of the skeleton extraction process; therefore, their segmentation will fit perfectly with the segments of their output skeleton. Our case is different; we are not working directly with the vertices of the target mesh to produce a skeleton; we instead take a rigged mesh and produce a segmentation depending of the bound skeleton to the target mesh. Our automatic weight assign algorithm was developed with the same objective as the one showed in [1]: create a set of weights having only as input a 3D rigged mesh. Other algorithms such as [11,13,15–17,27,31,32] had a set of examples to compute (or extend in some cases) the weights of each vertex in a character 3D mesh. Works like [3,8] and [23] preserve the volume of a mesh after its deformation; however, they compensate the loss of volume as a post-process; their results are notable, but it adds computation time to the animation pipeline, and they can only solve properly twist rotations with an angle $0° < \theta < 180°$, because if $\theta \geq 180°$, a self intersection artifact is produced, and it cannot be corrected by any volume preservation post-process; our proposed method solves correctly twist rotation angles greater than 180° with low loss of volume. A main point of comparison in the case of the skinning algorithm is the work developed by Kavan in [20]; DQS uses the same weight influence base of LBS and also corrects the *candy wrapper*, but it also introduces a bulging artifact in rotation over limbs such as elbows or knees by the nonlinear behavior (similar to a sphere) of DQS; our method does not have that kind of problem because it uses fundamentally LBS with the exception of twist rotations. Another problem addressed in [26] are the artifacts that DQS

Table VI. Skinning algorithms main advantages comparison.

Features	DQS [20]	Segmentation-based skinning	LBS	DIB [20]	ALNS [26]
Uses LBS weights	√	×	√	√	×
Solves candy-wrapper artifact	√	√	×	√	√
Solves or not produces bulging artifact	×	√	√	×	√
Produces correct results with rotations over 360°	×	√	×	√	?
Short processing times	√	√	√	×	√

DQS, dual quaternion skinning; LBS, linear blend skinning; DIB, dual quaternion iterative blending; ALNS, automatic linearization of nonlinear skinning.

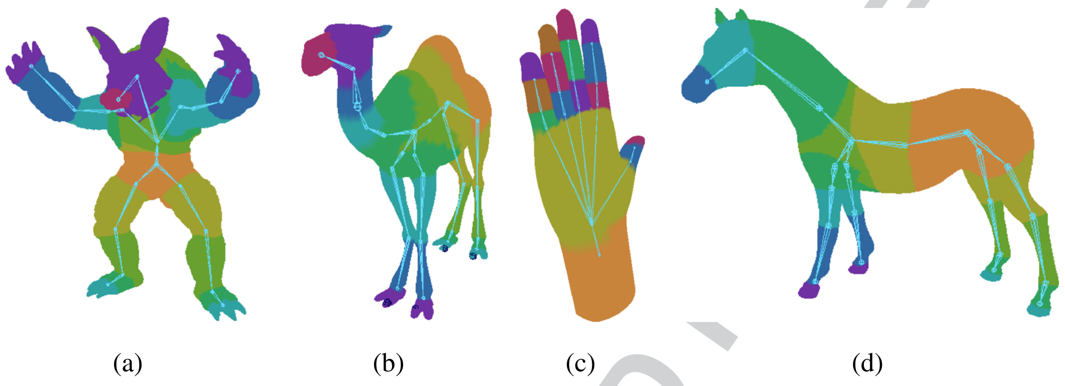


Figure 12. Segmentation algorithm applied over meshes with different shapes and number of joints in their skeletons.

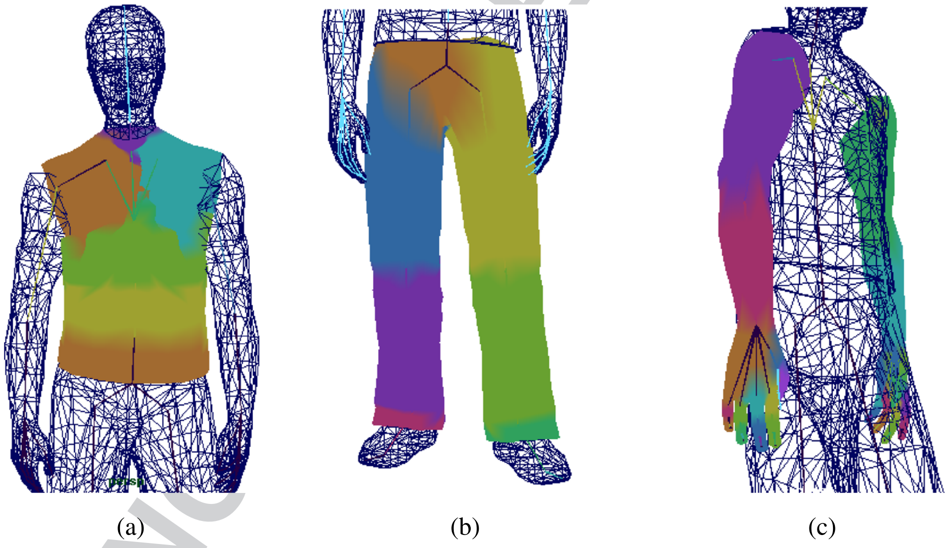


Figure 13. Segmentation applied over a multi-mesh character.

creates when the rotation over the link axis is close to 360°, artifacts that our solution does not have with the trade off of having longer computing times than LBS and DQS in its unoptimized version but similar in quality to DIB, which is more than five times slower than DQS [26]; in its optimized implementation, our method is approximately as fast as LBS. The method used in [27] is similar to our approach in the sense that they use as base LBS; for rotations in local coordinates *XY* plane (swing) and for rotations over local *z* axis (twist), they change to a nonlinear interpolation method (an approximation to DQS). They also apply the twist rotation in the middle of the link segment and not over the target joint in a similar way we apply it in our extension of LBS. When a rotation is made over a segment link axis, their performance is not clear because of its lack of processing times on the skinning stage, but their main overhead is present when the two deformers are evaluated; because of its close relation to DQS, we believe that deformations closer or higher than 360° lead to artifacts present also [26] that are properly solved by DIB or our method; we had created Table VI that shows the features that our method share with other geometry-based skinning methods.

7. CONCLUSIONS AND FUTURE WORK

We have developed a novel segmentation algorithm that works over the vertices of an input rigged mesh only; their main features are that it is not restricted to a specific pose (can deal with rigged meshes in arbitrary poses), arbitrary number of joints in the rig, or to a specific kind or shape (Figure 12 shows the segmentation algorithm applied to some non-anthropomorphic meshes), and it is low in computation times. Because it works solely with the vertices of a rigged mesh, its adaptation to rigs with multiple meshes is easy (Figure 13). Segmentation is important because it is a concept that simplifies and makes easier the weight assign process, while other algorithms depend on minimization algorithms or uses the weights assigned by other methods as a starting point. A good segmentation can be useful even for digital 3D artist as base to *paint* influence weights on a desired model. The widely used LBS algorithm had the well-known *candy-wrapper* artifact; to overcome this problem, we had developed a skinning algorithm based in LBS. Our algorithm can handle advance deformations (twists over a link greater than 180°), without volume loss or unrealistic artifacts, is not dependent on examples, uses weights generated for an LBS deformation scheme, and generates automatically the extra weights needed. Our method was developed entirely in AUTODESK MAYA as a plug-in in ANSI C++; this had the objective of making easier the diffusion around the animation community and being independent of the hardware used. Although the project was made using the Visual C++ compiler of Microsoft, with some changes, a port to Mac, Linux, or another operating system that supports Maya and a C++ compiler will be possible.

Our segmentation algorithm can be improved in the region-growing stage; we are using Euclidean distances and projections to discriminate the candidates, but in input meshes with arbitrary poses, the task can be difficult, leading to false positives that must be refined by hand. To solve this problem, we want to explore an algorithm that uses geodesic distances (such as [33]) to unfold the arbitrary pose to something closest to T-Pose; therefore, we can use geodesic distances directly in the region-growing stage and use a simpler rule set to discriminate candidates.

To improve our skinning weight assign algorithm, we want to explore an algorithm based on examples, such as [27], that allow us to apply the information obtained by the examples in models with similar shape. We are using a Gaussian function as weight distribution function, but we want to test which are the results with different kind of functions such as Bezier curves or a function of high order to produce smoother transitions between two connected segments to avoid weight-based artifacts. The algorithms depicted in this paper are sequential because of the nature of our implementation as a Maya plug-in; an interesting alternative will be a parallelized version in CUDA to improve its performance. The distribution function of the extra weight in our skinning algorithm is linear; a different distribution function may lead to different twist behavior. A topic to explore in our skinning method as future work will be volume preservation; to achieve this goal, we can plug an algorithm to the output of our skinning method to achieve a volume loss of 0% or closer; although the volume loss of our deformation algorithm is low, we want to test their improvement with a volume-correction algorithm such as [23].

REFERENCES

1. Baran I, Popović J. Automatic rigging and animation of 3D characters. In *SIGGRAPH '07: ACM SIGGRAPH 2007 Papers*, New York, NY, USA, 2007; 72, ACM.
2. Bharaj G, Thormählen T, Seidel H-P, Theobalt C. Automatically rigging multi-component characters. *Computer Graphics Forum* 2012; **31**: 755–764. Wiley Online Library.
3. Rohmer D, Hahmann S, Cani M-P. Exact volume preserving skinning with shape control. In *Proceedings of the 2009 ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation,* SCA '09, New York, NY, USA, 2009; 83–92, ACM.
4. Shamir A. A survey on mesh segmentation techniques. *Computer Graphics Forum* 2008; **27**(6): 1539–1556.
5. de Aguiar E, Theobalt C, Thrun S, Seidel H-P. Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum* 2008-09-18; **27**(2): 389–397.
6. Tierny J, Vandeborre J-P, Daoudi M. Topology driven 3D mesh hierarchical segmentation. In *Shape Modeling International*, IEEE Computer Society, 2007; 215–220.
7. James DL, Twigg CD. Skinning mesh animations. In *ACM SIGGRAPH 2005 Papers,* SIGGRAPH '05, New York, NY, USA, 2005; 399–407, ACM.
8. Rohmer D, Hahmann S, Cani M-P. Local volume preservation for skinned characters. *Computer Graphics Forum* 2008.
9. Dionne O, de Lasa M. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation,* SCA '13, New York, NY, USA, 2013; 173–180, ACM.
10. Kavan L, Žára J. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games,* I3D '05, New York, NY, USA, 2005; 9–16, ACM.

11. Lewis JP, Cordner M, Fong N. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques,* SIGGRAPH '00, New York, NY, USA, 2000; 165–172, ACM Press/Addison-Wesley Publishing Co.
12. Jacka D, Reid A, Merry B. A comparison of linear skinning techniques for character animation. In *Afrigraph*, 2007; 177–186, ACM.
13. Wang XC, Phillips C. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation,* SCA '02, New York, NY, USA, 2002; 129–138, ACM.
14. Sloan P-PJ, Rose CF, III, Cohen MF. Shape by example. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics,* I3D '01, New York, NY, USA, 2001; 135–143, ACM.
15. Mohr A, Gleicher M. Building efficient, accurate character skins from examples. In *ACM SIGGRAPH 2003 Papers,* SIGGRAPH '03, New York, NY, USA, 2003; 562–568, ACM.
16. Merry B, Marais P, Gain J. Animation space: a truly linear framework for character animation. *ACM Transactions on Graphics* 2006; **25**(4): 1400–1423.
17. Q19 Mohr A, Gleicher M. Deformation sensitive decimation. *Technical Report,* 2003.
18. Yang XS, Zhang JJ. Realistic skeleton driven skin deformation. In *Proceedings of the 2005 International Conference on Computational Science and Its Applications—Volume Part III,* ICCSA'05, Berlin, Heidelberg, 2005; 1109–1118, Springer-Verlag.
19. Jacobson A, Baran I, Popovic J, Sorkine O. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics* 2011; **30**(4): 78.
20. Kavan L, Collins S, Žára J, O'Sullivan C. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics* 2008; **27**(4): 105:1–105:23.
21. Kim YB, Han JH. Bulging-free dual quaternion skinning. *Computer Animation and Virtual Worlds* 2014; **25**(3-4): 321–329.
22. Yang X, Zhang JJ. Stretch it—realistic smooth skinning. In *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation,* CGIV '06, Washington, DC, USA, 2006; 323–328, IEEE Computer Society.
23. von Funck W, Theisel H, Seidel HP. Volume-preserving mesh skinning. *Proceedings of Vision, Modeling, and Visualization 2008* 2008: 409.
24. Pilgrim S, Steed A, Aguado A. Progressive skinning for character animation. *Computer Animation and Virtual Worlds* 2007; **18**(4-5): 473–481.
25. Zhang JJ, Yang X, Zhao Y. Bar-net driven skinning for character animation. *Computer Animation and Virtual Worlds* 2007; **18**(4-5): 437–446.
26. Kavan L, Collins S, O'Sullivan C. Automatic linearization of nonlinear skinning. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, 2009; 49–56, ACM.
27. Kavan L, Sorkine O. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics (TOG)* 2012; **31**(6): 196.
28. Ramirez JE, Lligadas X, Susin A. Adjusting animation rigs to human-like 3D models. In *AMDO '10: Proceedings of the 6th International Conference on Articulated Motion and Deformable Objects*, Berlin, Heidelberg, 2010; 300–310, Springer-Verlag.
29. Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 2009; **28**(3). Q20
30. Pan JJ, Yang X, Xie X, Willis P, Zhang JJ. Automatic rigging for animation characters with 3D silhouette. *Computer Animation and Virtual Worlds* 2009; **20**(2/3): 121–131.
31. Weber O, Sorkine O, Lipman Y, Gotsman C. Context-aware skeletal shape deformation. *Computer Graphics Forum* 2007; **26**: 265–274. Wiley Online Library.
32. Jacobson A, Baran I, Kavan L, Popović J, Sorkine O. Fast automatic skinning transformations. *ACM Transactions on Graphics (TOG)* 2012; **31**(4): 77.
33. Katz S, Leifman G, Tal A. Mesh segmentation using feature point and core extraction. *The Visual Computer* 2005; **21**(8-10): 649–658.

APPENDIX A: COMPLEXITY ANALYSIS OF THE SEGMENTATION ALGORITHM

The calculation of the complexity O of the segmentation algorithm based in our code implementation is

$$\begin{array}{ll}
\sum_{i=1}^{S} a_{1i}n_v \cdot a_{2i}n_v + & \text{Region grow.} \\
\sum_{j=1}^{n}(a_{3j}v_js + a_{4j}v_js) + n_vS + \\ \sum_{k=1}^{m}(v_ka_{5k}n_v) + a_6n_vS + & \text{Vertex belong test.} \\
n_v + \sum_{l=1}^{s}(n_va_{7l} \cdot a_{8l}n_v + a_{9l}n_v) & \text{Region merge.}
\end{array}$$

where $0 \leqq a_1 \ldots a_9 \leqq 1$ are constants, n_v is the total number of vertex in a 3D rigged mesh, and S is the total number of segments produced by the skeleton bounded to the mesh.

The simplification of this formula taking some of the constants $a1 \ldots a9$ as 1 is

$$\begin{aligned}
&n\,2_vS+ && \text{Region grow.}\\
&3n_vS+mn_v^2+a_6n_vS+ && \text{Vertex belong test.}\\
&n_v+n_v^2S+n_vS= && \text{Region merge.}\\
&2n_v^2S+n_v^2+Sn_v(4+a_6)+n_v= &&\\
&2(S+1)n_v^2+n_v(S(4+a_6)+1)= &&\\
&n_v^2S+n_vS\left(b+\tfrac{1}{s}\right)= &&\\
&n_v^2S &&
\end{aligned}$$

therefore, the complexity of our segmentation algorithm is $O\left(Sn^2\right)$.

AUTHORS' BIOGRAPHIES

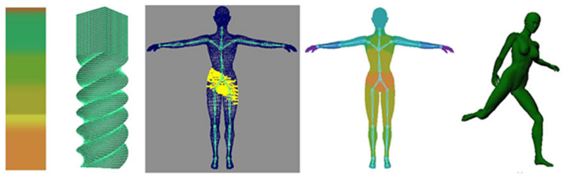
Jorge Eduardo Ramírez Flores ■■

Antonio Susin Sánchez ■■ Q21

Research Article

Segmentation-based skinning

Jorge Eduardo Ramírez Flores and Antonio Susin Sánchez



In this paper, we propose an approach based on mesh segmentation for skinning and skeleton-driven animation. Our method is based in watershed segmentation to deal with characters in T-Pose and arbitrary poses; the segmentation is the core algorithm of our method; we use it to develop a simple weight that assigns the LBS deformation method and a modified version of the LBS that avoids the loss of volume (candy-wrapper artifact) in twist rotations.

Author Query Form

Journal: Computer Animation and Virtual Worlds

Article: cav_1687

Dear Author,

During the copyediting of your paper, the following queries arose. Please respond to these by annotating your proof with the necessary changes/additions.

- If you intend to annotate your proof electronically, please refer to the E-annotation guidelines.
- If you intend to annotate your proof by means of hard-copy mark-up, please use the standard proofreading marks in annotating corrections. If manually writing corrections on your proof and returning it by fax, do not write too close to the edge of the paper. Please remember that illegible mark-ups may delay publication.

Whether you opt for hard-copy or electronic annotation of your proof, we recommend that you provide additional clarification of answers to queries by entering your answers on the query sheet, in addition to the text mark-up.

Query No.	Query	Remark
Q1	AUTHOR: Please check the changes made in the short title and article title if appropriate.	√
Q2	AUTHOR: Please check that authors and their affiliations are correct.	X
Q3	AUTHOR: 'silicing' has been changed to 'slicing' based on context. Please check if correct.	√
Q4	AUTHOR: multi-weight enveloping. Is this the correct definition for MWE? Please change if this is incorrect.	√
Q5	AUTHOR: Tables 5, 1, 2, 3, and 4 have been renumbered to Tables 1, 2, 3, 4, and 5, respectively, according to citation order. Please check.	√
Q6	AUTHOR: Please check that all tables are presented correctly.	√
Q7	AUTHOR: Figures 5, 1, 2, 3, 4, 11, 9, and 10 have been renumbered to Figures 1, 2, 3, 4, 5, 9, 10, and 11, respectively, according to citation order. Please check.	√
Q8	AUTHOR: realistic skeleton driven skin deformation. Is this the correct definition for RSDSD? Please change if this is incorrect.	√
Q9	AUTHOR: bounded biharmonic weights. Is this the correct definition for BBW? Please change if this is incorrect.	√
Q10	AUTHOR: volume-preserving mesh skinning. Is this the correct definition for VPMS? Please change if this is incorrect.	√

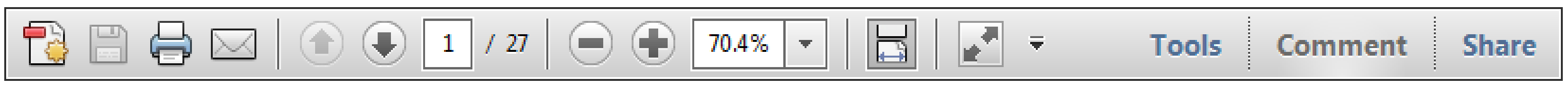
Query No.	Query	Remark
Q11	AUTHOR: exact volume preserving skinning with shape control. Is this the correct definition for EVPSSC? Please change if this is incorrect.	√
Q12	AUTHOR: automatic linearization of nonlinear skinning. Is this the correct definition for ALNS? Please change if this is incorrect.	√
Q13	AUTHOR: elasticity-inspired deformers for character articulation. Is this the correct definition for EIDCA? Please change if this is incorrect.	√
Q14	AUTHOR: Figure 5 was not cited in the text. An attempt has been made to insert the figure into a relevant point in the text—please check that this is OK. If not, please provide clear guidance on where it should be cited in the text.	√
Q15	AUTHOR: 'complier' has been changed to 'compiler' based on context. Please check if correct.	√
Q16	AUTHOR: If References 1, 19, 23, 27, and 32 are not one-page articles, please supply the first and last pages for these articles.	💬
Q17	AUTHOR: Please provide the location where the proceedings/conference was held for References 6, 12, 23, and 26.	💬
Q18	AUTHOR: Please provide volume number and page range for Reference 8.	💬
Q19	AUTHOR: Please provide name of organization and city location for Reference 17.	💬
Q20	AUTHOR: Please provide page range for Reference 29.	💬
Q21	AUTHOR: Please provide authors' biographies with photos.	

WILEY

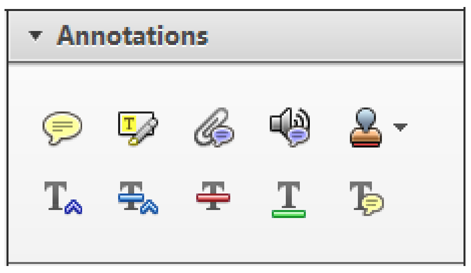
USING e-ANNOTATION TOOLS FOR ELECTRONIC PROOF CORRECTION

Required software to e-Annotate PDFs: Adobe Acrobat Professional or Adobe Reader (version 7.0 or above). (Note that this document uses screenshots from Adobe Reader X)
The latest version of Acrobat Reader can be downloaded for free at: http://get.adobe.com/uk/reader/

Once you have Acrobat Reader open on your computer, click on the Comment tab at the right of the toolbar:



This will open up a panel down the right side of the document. The majority of tools you will use for annotating your proof will be in the Annotations section, pictured opposite. We've picked out some of these tools below:

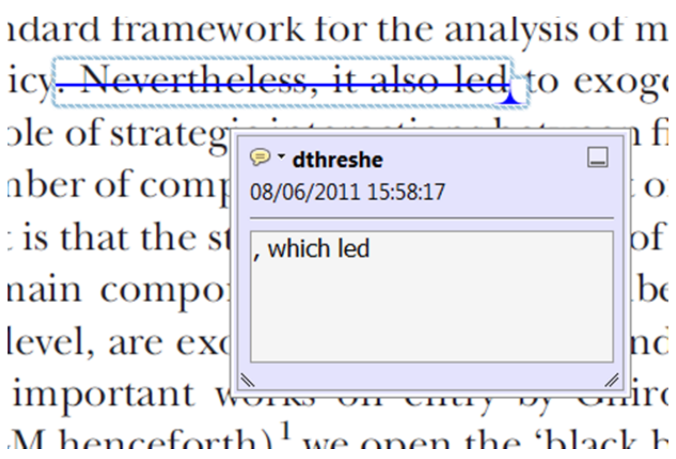


1. Replace (Ins) Tool – for replacing text.

Strikes a line through text and opens up a text box where replacement text can be entered.

How to use it

- Highlight a word or sentence.
- Click on the Replace (Ins) icon in the Annotations section.
- Type the replacement text into the blue box that appears.



ıdard framework for the analysis of m
icy~~. Nevertheless, it also led~~ to exog
ɔle of strateg
ıber of comp
: is that the st
nain compo
level, are exc
important
M henceforth)¹ we open the 'black b

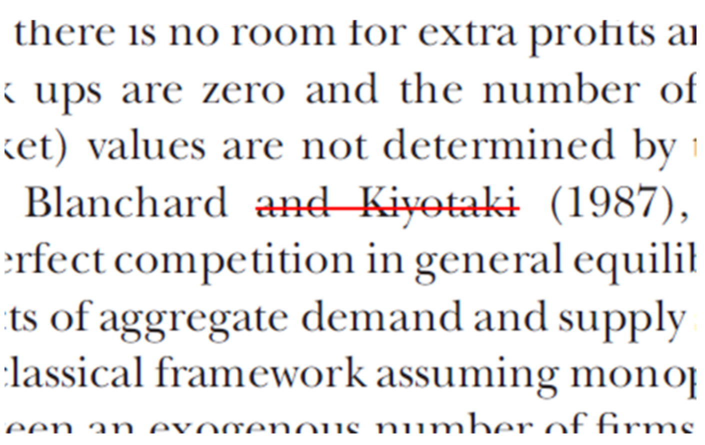
dthreshe
08/06/2011 15:58:17
, which led

2. Strikethrough (Del) Tool – for deleting text.

Strikes a red line through text that is to be deleted.

How to use it

- Highlight a word or sentence.
- Click on the Strikethrough (Del) icon in the Annotations section.



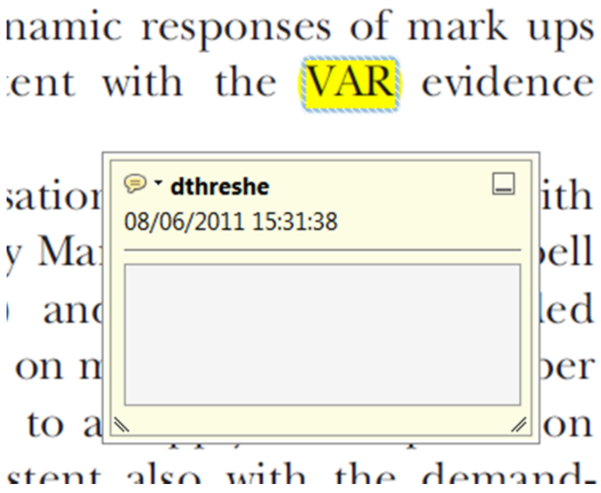
there is no room for extra profits a
ups are zero and the number of
et) values are not determined by
Blanchard ~~and Kiyotaki~~ (1987),
rfect competition in general equili
ts of aggregate demand and supply
lassical framework assuming mono
een an exogenous number of firms

3. Add note to text Tool – for highlighting a section to be changed to bold or italic.

Highlights text in yellow and opens up a text box where comments can be entered.

How to use it

- Highlight the relevant section of text.
- Click on the Add note to text icon in the Annotations section.
- Type instruction on what should be changed regarding the text into the yellow box that appears.



namic responses of mark ups
ent with the VAR evidence
sation ith
y Ma ell
an ed
on m ber
to a on
stent also with the demand-

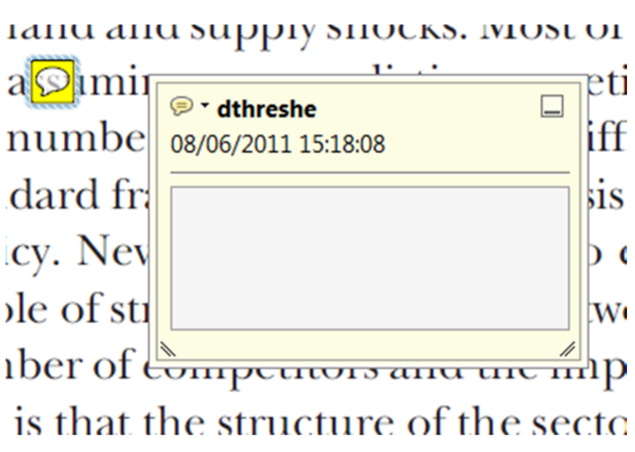
dthreshe
08/06/2011 15:31:38

4. Add sticky note Tool – for making notes at specific points in the text.

Marks a point in the proof where a comment needs to be highlighted.

How to use it

- Click on the Add sticky note icon in the Annotations section.
- Click at the point in the proof where the comment should be inserted.
- Type the comment into the yellow box that appears.



land and supply shocks. Most of
a mi eti
numbe iff
dard fr sis
cy. Nev
ɔle of st
ıber of competitors and the imp
is that the structure of the secto

dthreshe
08/06/2011 15:18:08

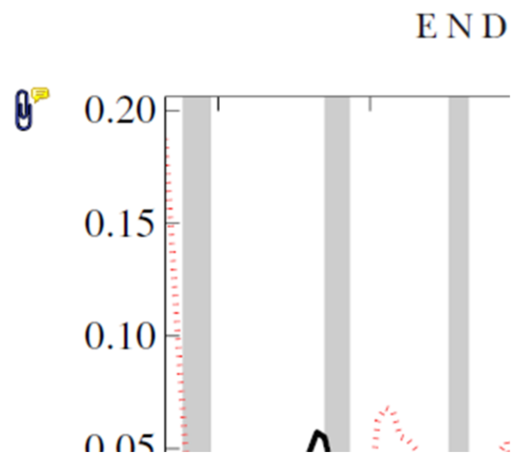
USING e-ANNOTATION TOOLS FOR ELECTRONIC PROOF CORRECTION

5. Attach File Tool – for inserting large amounts of text or replacement figures.

Inserts an icon linking to the attached file in the appropriate pace in the text.

How to use it

- Click on the Attach File icon in the Annotations section.
- Click on the proof to where you'd like the attached file to be linked.
- Select the file to be attached from your computer or network.
- Select the colour and type of icon that will appear in the proof. Click OK.

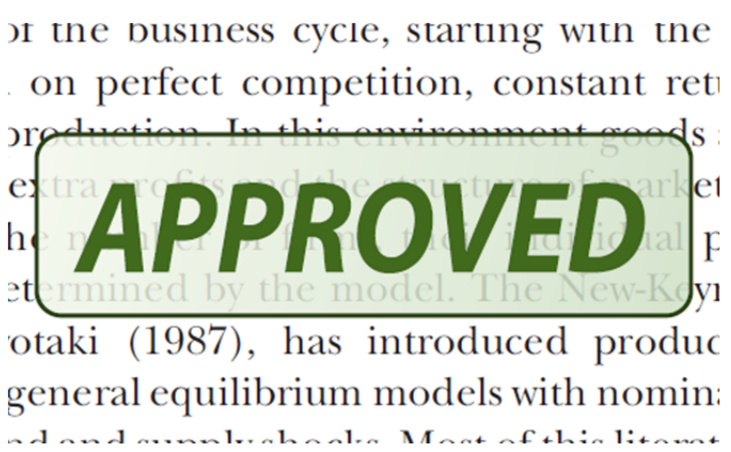


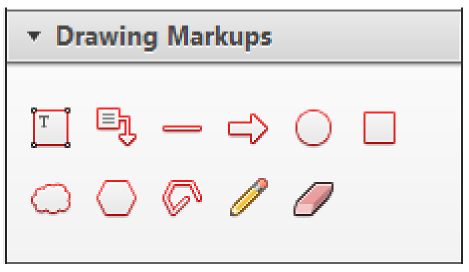
6. Add stamp Tool – for approving a proof if no corrections are required.

Inserts a selected stamp onto an appropriate place in the proof.

How to use it

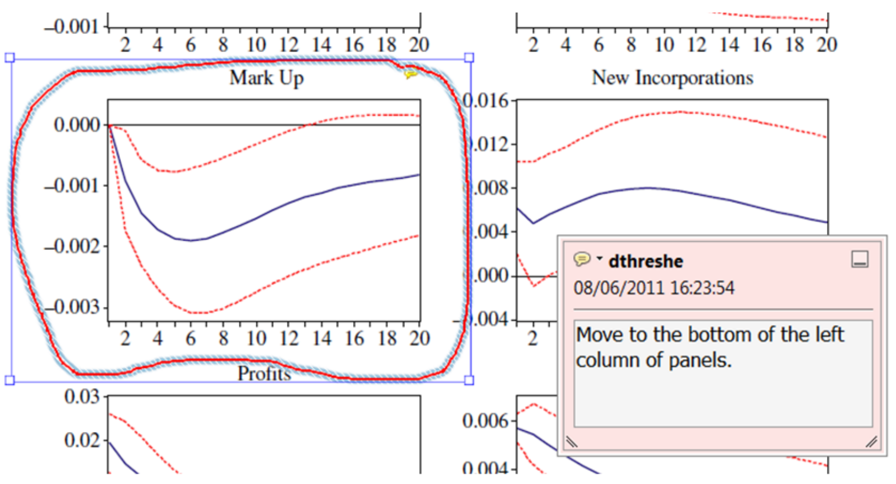
- Click on the Add stamp icon in the Annotations section.
- Select the stamp you want to use. (The Approved stamp is usually available directly in the menu that appears).
- Click on the proof where you'd like the stamp to appear. (Where a proof is to be approved as it is, this would normally be on the first page).





7. Drawing Markups Tools – for drawing shapes, lines and freeform annotations on proofs and commenting on these marks.

Allows shapes, lines and freeform annotations to be drawn on proofs and for comment to be made on these marks..

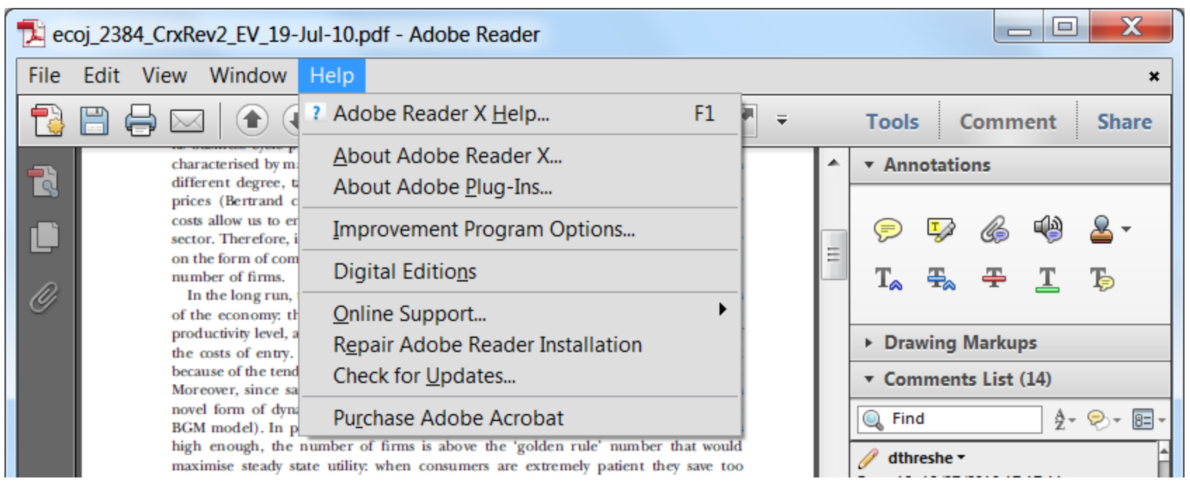


Move to the bottom of the left column of panels.

How to use it

- Click on one of the shapes in the Drawing Markups section.
- Click on the proof at the relevant point and draw the selected shape with the cursor.
- To add a comment to the drawn shape, move the cursor over the shape until an arrowhead appears.
- Double click on the shape and type any text in the red box that appears.

For further information on how to annotate proofs, click on the Help menu to reveal a list of further options:



ecoj_2384_CrxRev2_EV_19-Jul-10.pdf - Adobe Reader