Shape Aware Deformation Using a Skeleton-Guided Scheme

Alvaro Cuno, Claudio Esperança and Antonio Oliveira
Federal University of Rio de Janeiro - COPPE/PESC

Abstract

This work presents a method for deforming meshes in a shape-sensible way using Moving Least Squares (MLS) optimization. It is explained how an approach for space deformation [10] can be transformed into a skeleton-guided deformation scheme, where more sensitivity to the mesh geometry is achieved. Given a mesh and its skeleton, for each joint a minimization problem is solved subject to positional constraints given by the user. The mesh geometry is then modified by blending optimal transformations computed at skeleton joints associated with each vertex of the mesh. Additionally, it is shown how a small reformulation of the MLS scheme makes it possible to affect the bending behavior of the deformation.

1. Introduction

Interactive direct modification of 3D surface meshes, specially in the context of character animation, can be defined as a process where the shape of a model is altered through constraints specified by the user. Traditionally, inverse kinematics techniques [11] are preferred for this task. These, however, require that the character being animated is attached to a very carefully designed skeleton data structure, where bone and joint movement restrictions are established manually. In addition, the movement of a limb does not usually propagate to the rest of the model, which makes the authoring of a simple animation rather time consuming.

Recently, several techniques based on optimization schemes have been proposed for this task [1, 8, 17]. Most of these focus on obtaining a new pose for a given character by means of a deformation process guided by the displacement of a few points attached to the model. This type of interaction can be effected by dragging a 2D mouse on a window canvas, thus providing a very intuitive interface for authoring animations. No restrictions are given for mouse dragging, i.e., the constraints can be separated or stretched freely and rotations need be not specified explicitly. However, it is required that the model shape should be preserved as much as possible giving the user the idea he is manipulating a real object.

Deformation techniques that avoid unnatural shearing and arbitrary scaling of the models are of special interest, since the global shape and local details are preserved after editing operations [17]. Recently, Schaefer et al. [16] used this paradigm for performing interactive image deformation. In that work, the key solved problem was how to find the rotation component for the optimal rigid-body transformation efficiently. They presented a closed-form solution using the relationship between similarity and rigid transformations obtaining attractive results. Their algorithm use as input a 2D image and a set of control points. After the user drags some of these control points, the image is altered computing a transformation restricted to be "as rigid as possible" for each element of the image (not necessarily a pixel). They used a moving least squares optimization approach for computing a varying solution for each image element.

An approach for deforming 3D models following Schaefer's ideas has been presented in [10]. Although it presents an efficient formulation for computing the rotation component in 3D, non-natural results can be obtained when the deformation is applied on surface models, as the deformation process is blind to the shape of the model. Figure 1 illustrates this problem.

The main contributions of this work are: (1) a proposal for applying "as rigid as possible transformations" sensible to the model shape, (2) a reformulation of the MLS scheme when applied to skeletons in order to provide bending control, and (3) a simplified skinning/rigging which works well even for poorly designed or automatically constructed skeletons. We propose to solve the mesh deformation problem by adapting the method presented in [10] into a skeleton-driven deformation scheme. Then, given a mesh model and its associated skeleton, we perform the deformation on the skeleton joints and project the transformations into the mesh. Sensitivity to shape is achieved by using a distance metric defined over the skeleton bones. Figure 1(d) shows more natural results when compared with results on Figure 1(b).

2. Related Work

Interactive deformation methods have attracted much attention in recent years, especially those which aim at pre-

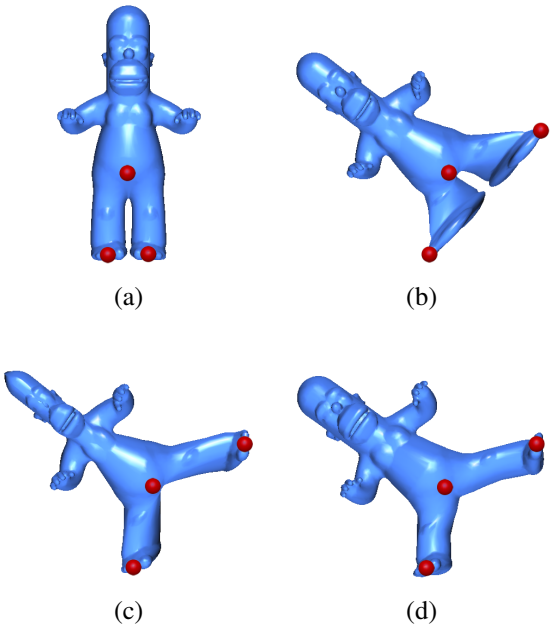


Figure 1. MLS-based deformation approaches: (a) Initial model and control points. (b) Standard MLS deformation based on optimal rigid transformations and the Euclidian metric; (c) using a geodesic path metric defined over the mesh; and (d) skeleton-guided MLS deformation.

serving surface detail and which support intuitive user interfaces [1, 8, 17]. Another frequent requirement is that results are predictable and physically plausible even when no physical properties are specified. Overall, such methods may be divided into two broad classes: those that operate on some representation of the model surface, and those which deform the space where the model is embedded.

Surface deformation methods try to meet deformation specifications by preserving differential properties of the surface and minimizing detail distortion. These techniques express the deformation requirements as a non-linear optimization problem which is then linearized and solved using standard numeric tools [15, 3, 12]. In particular, the size of the linear system depends not only on the size of the part of the model subject to deformation, but also on the size of the handles used to guide the process. In this respect, the work of Sorkine and Alexa [17] is of special interest to us since they are able to obtain convincing results using a small set of points as handles.

Space deformation schemes, on the other hand, are able to cope with a wider spectrum of model representations. The idea is to express deformations as transformations of the space in which the model is embedded. In this context, the so-called *as-rigid-as-possible* transformations are of special interest, since they tend to produce physically plausible results by avoiding unnatural shearing and non-uniform scaling of the model. By making use of a Moving Least Squares approach, Schaefer et al. [16] have recently presented a way of computing such deformations for 2D images. In order to produce as-rigid-as-possible deformations in $\Re^3$, it is necessary to obtain a rigid transformation which best approximates a mapping $f : \Re^3 \mapsto \Re^3$, $f(\mathbf{p}_i) = \mathbf{q}_i$, for given sets of points $\{\mathbf{p}_i\}, \{\mathbf{q}_i\}$. This problem is also known as point registration, or, more specifically, the Absolute Orientation Problem. Analytical solutions have been proposed, for instance, based on SVD (*Singular Value Decomposition*) [2], quaternions [13, 14], orthonormal matrices and dual quaternions [19]. Another related group of techniques includes those based on iterative schemes such as the ICP (*Iterative Closest Point*) algorithm [5] and its variants. In [10] a solution is proposed where the axis and angular parameters of the rotation are obtained. That method requires a smaller number of computations than matricial and even quaternion-based approaches, but is still quite computationally expensive, as it requires solving a depressed quartic equation for every vertex of the mesh. Other types of transformations can also be used for space deformation – Botsch et al. [6], for instance, use Radial Basis Functions (RBFs) for this purpose. The chief disadvantage of space deformation schemes is that their blindness to the model's shape may lead to inadequate results. Figure 1 illustrates this effect when using a MLS-based scheme [16, 10]. The main goal of this work is to cope with such problems by applying MLS tools on a space spanned by a skeleton for the model rather than its 3D embedding space.

3. Moving Least Squares Deformation

Let $\{\mathbf{p}_i\}$ be a set of control points and $\mathbf{x}$ be a point in $\Re^3$. After new positions $\{\mathbf{q}_i\}$ are defined for the control points, the problem consists in finding the rigid-body transformation T that minimizes $\sum_i w_i |T_{\mathbf{x}}(\mathbf{p}_i) - \mathbf{q}_i|^2$, where w_i is a weight function defined as $w_i(\mathbf{x}) = |\mathbf{p}_i - \mathbf{x}|^{-2}$ and $|\cdot|$ denotes Euclidian distance metric.

It was shown (see [16]) that the optimal MLS-based solution is defined as

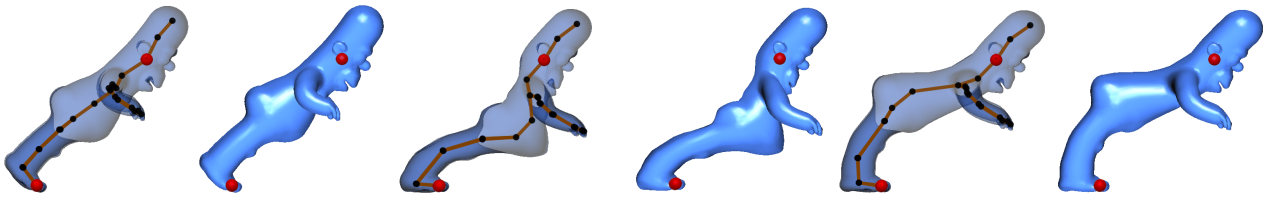
$$T(\mathbf{x}) = R(\mathbf{x} - \mathbf{p}^*) + \mathbf{q}^*, \tag{1}$$

where R is a rotation matrix and

$$\mathbf{p}^* = \frac{\sum_i w_i \mathbf{p}_i}{\sum_i w_i}, \quad \mathbf{q}^* = \frac{\sum_i w_i \mathbf{q}_i}{\sum_i w_i}. \tag{2}$$

In [10], the rotation R in 3D is defined by an angle α and an axis $\mathbf{u}$, and it was shown that this rotation axis is given by the solution of the 3×3 system

$$(\mathrm{M} + \mathrm{M}^\mathsf{T} - \lambda \mathrm{I})\, \mathbf{u}^\mathsf{T} = \mathrm{V}^\mathsf{T},$$



(a) (b) (c)

Figure 2. (a) Model deformation with no bending control; (b) using decreasing bending weights on joints from feet to head, and (c) using increasing weights.

where

$$\mathrm{M} = \sum_i w_i \hat{\mathbf{q}}_i^{\mathsf{T}} \hat{\mathbf{p}}_i, \quad \mathrm{V} = \sum_i w_i \hat{\mathbf{p}}_i \times \hat{\mathbf{q}}_i,$$
$$\hat{\mathbf{q}}_i = \mathbf{q}_i - \mathbf{q}^*, \qquad \hat{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}^*,$$

and λ is the largest root of the depressed quartic equation

$$y^4 - 2\|\mathrm{M}\|^2 y^2 - 8\det(\mathrm{M})\, y - \mathrm{E}^4 + 2\|\mathrm{M}\|^2\mathrm{E}^2 - 8\det(\mathrm{M})\mathrm{E} + \det(\mathrm{N})(2\mathrm{E} - \mathrm{V}\mathrm{N}^{-1}\mathrm{V}^{\mathsf{T}}) = 0.$$

In the last equation, E is the trace of M and $||\cdot||$ denotes the Frobenius norm. Additionally, expressions for computing angular parameters are given by

$$\cos(\alpha) = \frac{1 - \|\mathbf{u}\|^2}{1 + \|\mathbf{u}\|^2} \quad \text{and} \quad \sin(\alpha) = \frac{-2\|\mathbf{u}\|}{1 + \|\mathbf{u}\|^2}.$$

Finally, if we substitute $\cos(\alpha)$ and $\sin(\alpha)$ in the axis-angle rotation formula, then it can be seen that the optimal rotation to be applied on vector $\mathbf{v}$ depends only on $\mathbf{u}$:

$$R(\mathbf{v}) = \mathbf{v} - \frac{2(\mathbf{u} \times (\mathbf{v} \times \mathbf{u}) + (\mathbf{v} \times \mathbf{u}))}{1 + \|\mathbf{u}\|^2}. \qquad (3)$$

The scheme described above produces a *space* deformation, i.e., it defines a $\Re^3 \rightarrow \Re^3$ mapping. The modeling of a particular deformation is influenced merely by a discrete set of $\{\mathbf{p}_i, \mathbf{q}_i\}$ pairs. As such, any model immersed in 3D-space will be subject to the same deformation, with no consideration to its shape. This is unfortunate in many important applications – character animation, in special – where the desired deformation must take into account model peculiarities such as overall shape, bone structure and the position and geometry of articulations. As an illustration, contrast the unnatural deformation shown in Figure 1(b) with the more reasonable deformation exhibited in Figure 1(d).

In an attempt to reduce such undesired artifacts we first tried to simply replace the Euclidian metric of the MLS formulation by a more "shape aware" metric. Figure 1(c) illustrates a deformation obtained with the MLS scheme using a geodesic metric, i.e., the distance between any two points on the surface of the model is measured in terms of the length of the shortest curve connecting these two points which can be drawn on the surface. Although this idea addresses some of the problems, it is incapable of coping with some undesired shrinking/stretching effects. Most of these problems are overcome with a hybrid scheme combining as-rigid-as-possible space deformations and skeleton-based animation techniques, described below. Our goal is to obtain reasonable model poses specified by suitably placing a small set of control points.

4. Bending Control

The proposed method deviates from the standard space-based scheme by applying MLS optimization on a skeleton structure, rather than directly on the model points. The idea is similar to the method based on geodesic distances discussed above, but, here, these are measured along the bones and joints of a skeleton. Changes on the skeleton are propagated to the mesh using standard linear blending skinning.

The advantages of using a skeleton are two-fold. Firstly, the skeleton is very good representation of the overall shape of the model, which means that changes to it are almost immune to small detail present on the surface of the model. Secondly, in addition to the use of control points which the method tries to interpolate, it is possible to influence the behavior of the method by altering the distances traveled along bones and joints, making some parts more "rigid" than others.

The idea of influencing the bending behavior of the deformation arises from the analysis of the translation component of the scheme. Observing Equations 1 and 2, one notices that merely changing the values for w_i does not alter the shape of the set of $\mathbf{q}^*$ values. In other words, let A, B and C be three collinear joints on a given skeleton; then altering the bone lengths between them may cause their relative distance to vary, but they will still be collinear, if we ignore rotation components. This suggests that the compu-

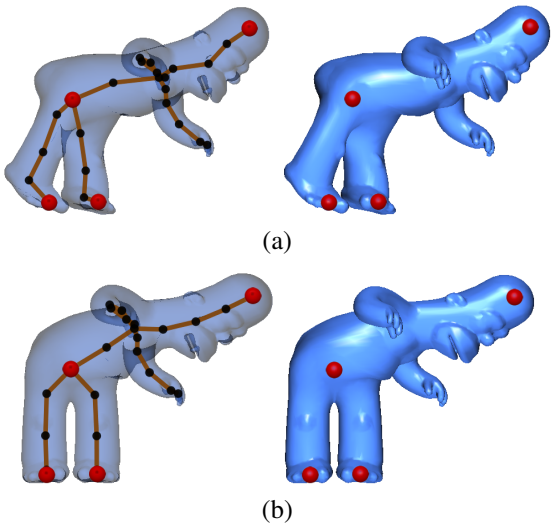


Figure 3. With no bending control, the movement of the head influences the feet (a). The use of constant bending weights isolates the legs from movement of the torso (b).

tation of **q*** values is based on different w_i values than those used for **p***. Let us then rewrite the equation for **q*** as follows

$$\mathbf{q}^* = \mathbf{p}^* + \frac{\sum_i w'_i (\mathbf{q}_i - \mathbf{p}_i)}{\sum_i w'_i}. \quad (4)$$

Notice that when $w'_i = w_i$, this is identical to Eq. 2. All it remains is to establish convenient values for w'_i in order to obtain the desired bending behavior. By using increasing / decreasing values along a sequence of joints between two control points, it is possible to obtain parabolas of different concavities. Figure 2 illustrates this effect on the "Homer" model. On the other hand, the use of constant bending weights in a sequence of joints does not alter the overall shape of the skeleton curve, but diminishes its deformation influence on the remaining joints. This is shown in Figure 3.

5. Skeleton-Driven Mesh Deformation

The scheme is divided into two phases: the setup (rigging) phase, performed only once, and the deformation phase, performed once for each desired pose.

In a nutshell, the rigging phase first extracts from the model an approximate skeleton, i.e., a tree-like structure composed of joints and bones (Figure 4(a)). Next, each mesh vertex is associated with a skeleton joint (Figure 4(b)). Lastly, a weight distribution process establishes the relative influence of each joint in the deformation of any given mesh vertex (Figure 4(c)).

In the deformation phase, the user first defines a set $\{\mathbf{p}_i\}$ of control points on the skeleton (red dots in Figure 4(d)). These, in turn, are interactively moved to target positions $\{\mathbf{q}_i\}$ which will cause the joints to be modified following the MLS scheme introduced in Section 3, but with the distances along the skeleton measured in terms of path lengths rather than using the Euclidian norm (Figure 4(e)). For each joint j, centroids $\{\mathbf{p}^*_j\}$ and $\{\mathbf{q}^*_j\}$ as well as rotation vectors $\{\mathbf{u}_j\}$ are cached. Observe that, unlike skeleton-based character animation, our scheme does not guarantee that each bone will be transformed rigidly. Finally, a linear blending of transformations is computed for each mesh vertex using the cached values (Figure 4 (e)).

5.1. Skeleton Extraction

Almost any automated method for skeleton extraction – see [9] for a survey – may be used in our scheme since, unlike skeletons used in traditional animation schemes, it requires no particular topology or strict correspondence between skeleton and mesh. In fact, the skeleton may even be quickly sketched manually, since – the examples shown in this paper, for instance, were obtained in this fashion.

5.2. Mesh Segmentation and Rigging

Once the skeleton is known, all mesh vertices are partitioned into skin sets S_j, where each set S_j contains all vertices primarily influenced by the transformation of a given joint j. A skin set S_j is defined as the union of two sets of mesh vertices (see Figure 5): a *primary skin* P_j and a *secondary skin* Q_j.

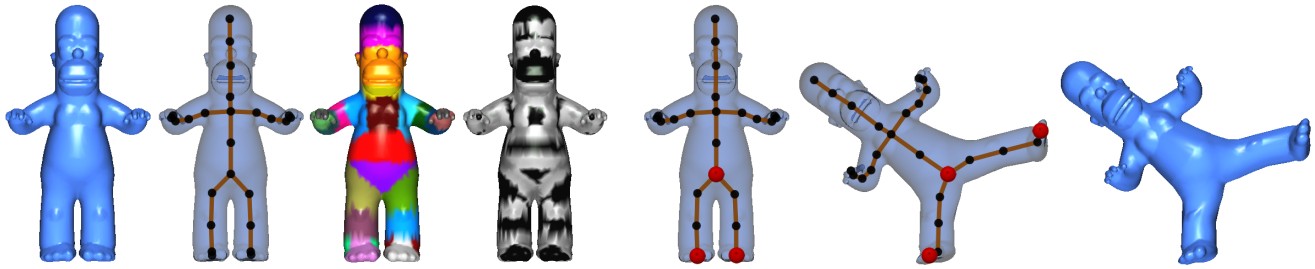
The *primary skin* P_j is the set of mesh vertices **v** such that, among all joints of the skeleton which are *visible* to **v**, j is the closest. By visible, it is meant that a line segment from **v** to joint j does not intersect the mesh. The *secondary skin* Q_j of a joint j is the set of vertices for which joint j is the solution of the problem

$$\min_{j \in \text{joints}} g(\mathbf{v}, P_j),$$

where $g(a, X)$ is the *geodesic* distance between a point a and a set of points X. Recall that the geodesic distance between two points of a surface is the length of the shortest path on the surface which connects them.

It should be mentioned that exact algorithms to compute these sets are fairly costly. Thus, rather than computing exact geodesic path lengths, we only consider paths composed of mesh edges. This makes it possible to employ Dijkstra's path length algorithm starting from vertices on the border of primary skin sets to find secondary skin vertices.

Similarly, visibility determination using, say, a ray-casting algorithm is unduly expensive for the task at hand.



(a) (b) (c) (d) (e)

Figure 4. Skeleton-guided deformation process. In the setup phase, (a) a skeleton is embedded into the mesh. (b) Mesh is segmented in clusters. (c) Influence of joints defined for each vertex. Vertices on dark regions will be influenced by one only joint and vertices on other regions will be influenced by more than one joint. In the deformation phase, (d) control points (red) are defined over some skeleton joints (black). (e) After moving one control point the skeleton is deformed and positions of mesh vertices are modified projecting transformations computed on associated joints.

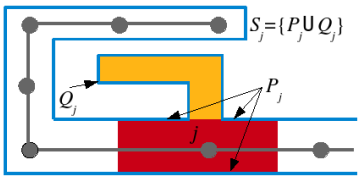


Figure 5. The primary skin P_j of joint j is the border of the red region, and the secondary skin Q_j is the border of the yellow region.

Rather, we use a visibility propagation algorithm which employs a "local" visibility property. A joint j is locally visible to a vertex $\mathbf{v}$ if the angle between the (inwards-facing) normal at $\mathbf{v}$ and the vector from $\mathbf{v}$ to joint j is smaller than 90 degrees (we assume that skeleton is always placed inside the mesh). Let $L(j)$ be the set of mesh vertices locally visible to j and which are closer to j than to any other joint k to which it is also locally visible. Then, the algorithm for finding the primary skin of joint j is the following:

1. Initialize $P(j)$ with $\{$**seed**$\}$, where **seed** is the mesh vertex in $L(j)$ which is closest to j.
2. Let $V = L(j) - P(j)$. Search V for a vertex $\mathbf{v}'$ such that $\mathbf{v}'$ is an edge neighbor of some vertex $\mathbf{v} \in P(j)$.
3. If such a vertex is found, place it in $P(j)$ and repeat step (2).
4. If $V \neq \emptyset$, select another **seed** $\in V$ by making sure that **seed** is *globally* visible using ray-casting. If none can be found, stop. Otherwise, include **seed** in $P(j)$ and repeat step (2).

This algorithm is very quick, as finding edge neighbors and testing for local visibility are constant-time operations. If other seeds have to be found by ray-casting (step 4 above), the process is constrained to a neighborhood of j. In our experience, even if a vertex that should belong to a primary skin is not found, most of the time the algorithm to compute secondary skins assigns that vertex to the correct joint. Moreover, it should be stressed that the skinning process used for deforming the model is fairly robust to vertex assignment errors. Figure 6 shows the segmentation results for a few selected models.

5.3. Skin Weight Determination

Despite the fact that each mesh vertex is assigned to a single skin set S_j, joints other than j may also influence its deformed position. One must, therefore, determine a weighting[1] scheme capable of ascertaining the smoothness of the deformed mesh. Recent approaches to this problem (see, for instance, [4]) emulate a heat diffusion process from the skeleton to the mesh, which occurs only within its interior as if the mesh surface were a perfect thermal insulator. To avoid the complexity of solving a 3D heat diffusion problem constrained to a polyhedron, it is considered that the heat emitted by joint j is totally directed to set S_j. From S_j that heat is diffused to the whole mesh. Now, assume that joint j has temperature 1 whereas all other joints have temperature 0. Let $w_{\mathbf{v}}$ represent the current temperature of a mesh vertex $\mathbf{v}$. Then, since it is considered that energy is radially spread from the joints to the mesh (and vice-versa), the heat flow between a vertex $\mathbf{v} \in S_k$ and joint k is given by $\alpha(1 - w_{\mathbf{v}})/||\mathbf{v} - k||^2$ if $k = j$, and

1 Not to be confused with the weights used in the MLS formulation.

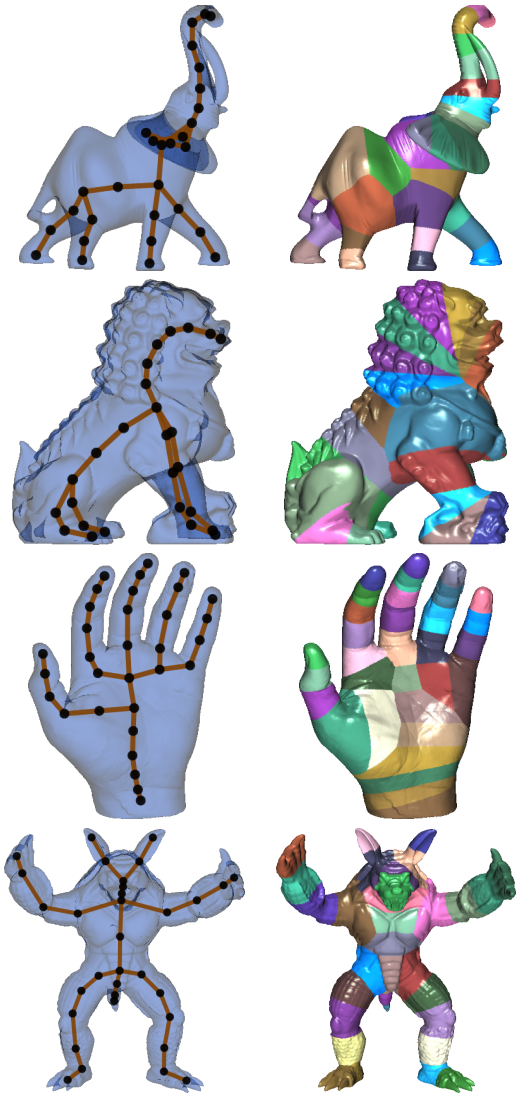


Figure 6. Segmentation results.

$-\alpha w_{\mathbf{v}}/||\mathbf{v}-k||^2$ otherwise, where α is a global constant. When an equilibrium state is reached, that flow must be equal to the flow between $\mathbf{v}$ and its neighbors, which can be estimated by $\sum_{\mathbf{u}\in N(\mathbf{v})}(w_{\mathbf{u}}-w_{\mathbf{v}})\Delta_{\mathbf{uv}}$, where $N(\mathbf{v})$ is the set of edge neighbors of $\mathbf{v}$ and $\Delta_{\mathbf{uv}}$ is the weight assigned to edge $(\mathbf{u},\mathbf{v})$, which in our prototype is given by the cotangent form of the Laplace-Beltrami operator [8]. Steady state temperatures determined by the heat emitted by j, are, then, obtained by solving a system of n equations (one for each vertex $\mathbf{v}$) of the form

$$\alpha(R_j(\mathbf{v})-\frac{w_{\mathbf{v}}}{||\mathbf{v}-k||^2})+\sum_{\mathbf{u}\in N(\mathbf{v})}(w_{\mathbf{u}}-w_{\mathbf{v}})\Delta_{\mathbf{uv}}=0$$

where $R_j(\mathbf{v})$ is the characteristic function of S_j, i.e., equal to 1 if $\mathbf{v}\in S_j$ and 0 otherwise. Each vertex temperature $w_{\mathbf{v}}$ thus obtained, is taken as the weight of joint j in the linear blending of transformations applied to $\mathbf{v}$. In order to determine all weights, a similar system must be solved for all joints.

It should be mentioned that this formulation has several interesting properties. First of all, the weights obtained for each joint j minimize a discretization of a functional of the form $\int_{\mathbf{v}\in M}(\nabla(w_{\mathbf{v}})^2+\alpha(R_j(\mathbf{v})-w_{\mathbf{v}})^2 d\mathbf{v}$. In other words, it not only yields smooth values of $w_{\mathbf{v}}$, but also makes it follow the information given by the rigging. Besides, it ensures that the sum of joint weights is 1 at every vertex and makes the "amount of blending" at a vertex be regulated by its distance to the skeleton.

6. Results

Figure 7 shows four poses of armadillo model obtained with the tecnhique. Since the mesh is deformed in a nearly rigid way, local detail tends to be preserved (notice, for instance, the wrinkles and bumps on the hands, legs and hump). More deformation results can be seen in Figure 8. These examples were obtained setting four or five control points only. The time for computing these deformations is directly dominated by the setup phase, that is, the construction of the rigging structure. In contrast, the deformation phase must compute the MLS optimal transformations for joints positions only, while the mesh deformation proper is achieved using a cheap linear blending skinning. Deformations in Figure 8 were obtained interactively at around 15 FPS in a PC equipped with a Pentium Core 2 Duo processor running at 2×2.13 GHz and a NVidia GeForce 8300 GTS graphics card. The size of models, number of joints and control points are presented in Table 1.

Model	Vertices	Joints	Control Points
Armadillo	165K	33	5
Elephant	185K	29	5
Hand	195K	32	6
Dragon	247K	26	5

Table 1. Data size of models in Figure 8.

6.1. Limitations.

The use of positional constraints may be cumbersome or insufficient for expressing certain poses. For instance, the rotation of a hand around the axis of the forearm can not be expressed if the hand bones are not transversal to the forearm bone.

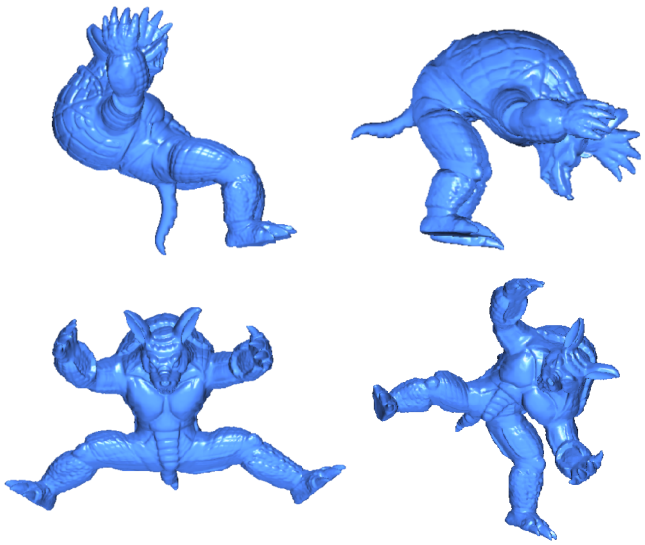


Figure 7. Several poses of the Armadillo model obtained with the proposed scheme.

Although the proposed scheme is very cheap computationally, the results are somewhat inferior to other more sophisticated surface deformation approaches such as Laplacian-based techniques (e.g., [8]) or non-linear methods (e.g., [7]). In order to address this, we are currently investigating the introduction of other types of restrictions such as those described in [18] in the MLS minimization process.

Another important issue is the fact that transformations are computed only at joint positions and propagated along the bones. This may lead to unnatural bending such as that shown in Figure 9, which may be alleviated with a careful authoring of the blending weights. Alternatively, the scheme could be modified so that the transformations are computed for bones and blended at joints. It should be stressed, however, that this limitation is inherited from the classic skeleton-based linear blending scheme.

7. Conclusions

In this work, we propose a skeleton-driven "as-rigid-as-possible" deformation scheme, transforming the original space-deforming algorithm into one which is more sensible to the geometry of meshes. Deformed poses are obtained by dragging control points and joint weights can be used to affect the bending behavior. The method employs a rigging procedure which requires a very simple set of operations – essentially, the computation of distances from vertices to joints and a progressive expansion process in the mesh graph.

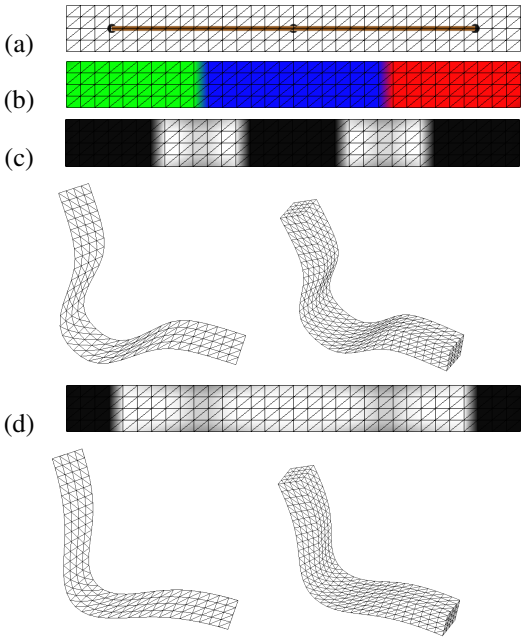


Figure 9. Deformations obtained using different weights distribution. (a) A bar model and its skeleton. (b) Segmentation of mesh vertices. Each vertex is associated with the closest joint. (c) and (d) illustrate two weight distributions and the corresponding deformations.

Results show a more plausible deformation of models when compared with the original formulation. The proposed method is fairly simple making it an attractive alternative to standard inverse kinematics in the sense that it depends less drastically on skeleton constraints, making it possible to employ automatically constructed skeletons.

The approach is also more efficient than other more involved mesh deformation schemes, since only skeleton joints must be transformed directly by the MLS formulation, making it possible to interactively deform models composed of up to a few hundred thousand vertices.

Acknowledgement. The first author is supported by a CAPES/CNPq fellowship - IEL Nacional - Brasil. All models in this paper were obtained from AIM@SHAPE repository.

References

[1] M. Alexa. Interactive shape editing. ACM SIGGRAPH Courses, 2006.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987.

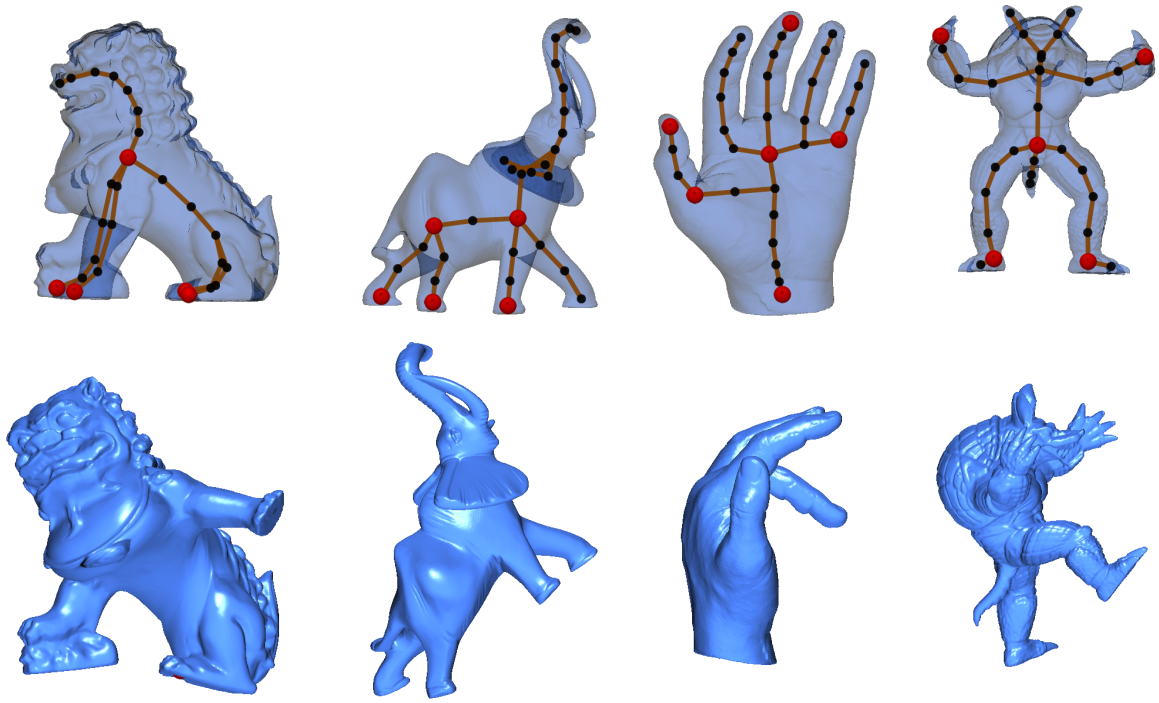


Figure 8. Deformation examples.

[3] O. K.-C. Au, H. Fu, C.-L. Tai, and D. Cohen-Or. Handle-aware isolines for scalable shape editing. *ACM Trans. Graph.*, 26(3):83, 2007.

[4] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72, 2007.

[5] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2):239–258, Feb. 1992.

[6] M. Botsch and L. Kobbelt. Real-time shape editing using radial basis functions. *Computer Graphics Forum*, 24(3):611–621, 2005.

[7] M. Botsch, M. Pauly, M. Wicke, and M. Gross. Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, 26(3):339–347, Sept. 2007.

[8] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.

[9] N. D. Cornea and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007.

[10] A. Cuno, C. Esperança, A. Oliveira, and P. R. Cavalcanti. 3D as-rigid-as-possible deformations using MLS. In *Proceedings of the 27th Computer Graphics International Conference*, pages 115–122, Petropolis, RJ, Brazil, May 2007.

[11] M. Fêdor. Application of inverse kinematics for skeleton manipulation in real-time. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, pages 203–212, New York, NY, USA, 2003. ACM.

[12] H. Fu, O. K.-C. Au, and C.-L. Tai. Effective derivation of similarity transformations for implicit laplacian mesh editing. *Computer Graphics Forum*, 26(1):34–45, Mar. 2007.

[13] B. K. P. Horn. Closed form solutions of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, Apr. 1987.

[14] K. Kanatani. Analysis of 3-d rotation fitting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):543–549, 1994.

[15] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):1142–1147, 2005.

[16] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25(3):533–540, July 2006.

[17] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *SGP' 07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[18] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Transactions on Graphics*, 26(3):80:1–80:7, July 2007.

[19] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-D location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54(3):358–367, Nov. 1991.