Instant Mesh Deformation

Fausto Richetti Blanco*
Instituto de Informática
UFRGS

Manuel M. Oliveira†
Instituto de Informática
UFRGS

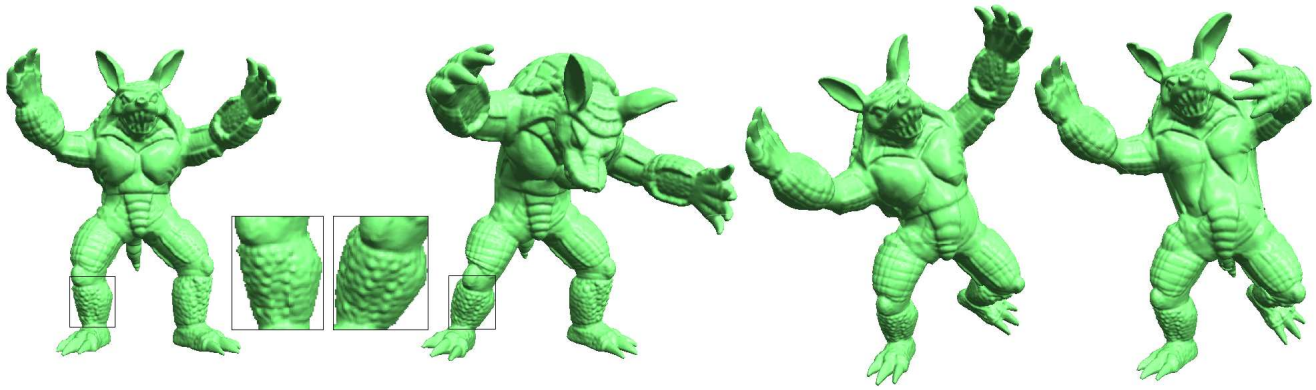

Figure 1: *Dancing Armadillo. A set of dancing poses created using our mesh deformation technique. The original pose is on the left and the rectangles depict zoomed portions of the model containing high-frequency surface details shown before and after a deformation.*

Abstract

We present an interactive mesh deformation technique based on parametric curve manipulation. A set of lines sketched over the projection of the mesh model is used to create parametric curves, which can be interactively manipulated, thus deforming the associated surfaces. Such curves can be further combined to create skeletons in a simple way, providing some extra control over the deformation process. Additionally, parametric curves can be automatically extracted from suggestive contours, allowing the deformation to be performed directly on visually-important details of the model. A major advantage of our technique is that it requires no preprocessing, allowing users to immediately produce visually-pleasing mesh deformations while using an intuitive interface. This makes it a good choice for artistic prototyping, as well as for casual users. We demonstrate that, despite its conceptual simplicity, it is quite general, producing results that are visually similar to the ones obtained with more sophisticated and computationally-intensive mesh deformation and skinning techniques.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques— [I.3.5]: Computer Graphics—Computational Geometry and Object Modeling

Keywords: interactive mesh deformation, sketch-based editing, curve-based deformation

1 Introduction

The realism of computer-generated images depends heavily on the ability to accurately represent geometric details of the objects in the scene. Modeling, however, is a labor-intensive task that can be significantly accelerated with the use of 3D scanners. Although 3D scanners provide a fast solution for the problem of sampling geometrically complex shapes, algorithms for surface reconstruction from point clouds tend to produce non-structured models consisting of a single polygonal mesh. Providing the user the means for changing pose and animating the resulting representations is key for using these models in applications such as computer movies and games. The challenge is to provide an intuitive interface and interactive feedback, while preserving surface details and accommodating user-defined constraints.

We present a new interactive technique for geometric shape deformation of 3D meshes. Our approach requires no preprocessing, allowing users to immediately create visually-pleasing results through the use of a simple and intuitive interface. As such, it provides a good choice for artistic prototyping and casual users. We use parametric curves created from 2D sketches or from automatically extracted suggestive contours [DeCarlo et al. 2003] to deform, twist and scale the associated mesh. The inherent smoothness of parametric curves is transferred to the induced global deformations. Our approach also supports some user-defined constraints, such as the specification of rigid segments for deforming articulated figures. We demonstrate that, despite its conceptual simplicity, our

*e-mail: fausto.blanco@gmail.com
†e-mail: oliveira@inf.ufrgs.br

approach is quite general, producing results that are visually similar to the ones obtained with more sophisticated mesh deformation techniques [Botsch and Kobbelt 2004; Kobbelt et al. 1998; Alexa 2003; Lipman et al. 2004; Lipman et al. 2005; Sorkine et al. 2004; Huang et al. 2006; Yu et al. 2004; Zayer et al. 2005; Zhou et al. 2005] and skinning [Mohr et al. 2003; Kry et al. 2002], which tend to require some considerable preprocessing time. One should note, however, that these techniques usually optimize some aspect of the deformation (*e.g.*, volume preservation), which our approach does not.

Our technique is considerably faster than previous approaches for achieving visually-similar results, being able to handle very large meshes at interactive rates. Moreover, it handles arbitrary meshes, including multiple connected component ones, non-orientable and non-manifold surfaces. Figure 1 shows the Armadillo model in some dancing poses obtained using our technique. The rectangles depict zoomed portions of the model containing high-frequency surface details shown before and after a deformation. These poses illustrate the use of deformations applied to the model's arms, legs, and torso, as well as the use of some bending and scaling operations.

2 Related Work

There has been a considerable amount of work on model deformation in recent years. Free-Form Deformation (FFD) and its variations perform object deformation indirectly by manipulating a set of control points (handles) that deform the space containing the object. Such handles can be defined as 3D lattices [Coquillart 1990; MacCracken and Joy 1996; Sederberg and Parry 1986], a set of curves [Chang and Rockwood 1994; Singh and Fiume 1998], or points [Hsu et al. 1992; Sumner et al. 2007].

Sumner et al. [Sumner et al. 2007] use a graph to deform the space where an object is embedded. This graph is created using the object surface (which is not restricted to be a mesh) allowing the user to directly manipulate parts of the object. The authors ensure that the deformation of each graph's node is locally rigid by solving a non-linear optimization problem.

Skinning techniques [Mohr et al. 2003; Kry et al. 2002] are probably the most popular mesh deformation techniques, but they tend to require a considerable amount of time for the artist to find the correct weights for each object. Although there exist some techniques for automatically computing such weights [Kry et al. 2002] (and references therein), they are often computationally expensive and do not always produce the desired deformations.

Multiresolution techniques [Botsch and Kobbelt 2004; Guskov et al. 1999; Kobbelt et al. 1998; Zorin et al. 1997] decompose the surface into a smooth base representation (low frequency) and the surface details (high frequencies). Mesh deformation is applied directly to the base representation, after which the details are added back (as displacement vectors).

Some techniques avoid factoring the base surface representation by directly applying the deformation to the original mesh. These techniques try to preserve some differential properties of the mesh, such as discrete Laplacian coordinates [Alexa 2003; Lipman et al. 2004; Lipman et al. 2005; Sorkine et al. 2004] or gradient functions over the mesh [Yu et al. 2004; Zayer et al. 2005; Zhou et al. 2005]. All these techniques treat mesh deformation as a minimization problem and, in general, require the solution of a linear system, which tends to introduce some delay before the user can actually start deforming the model. The energy function to be minimized contains both a detail preservation term and some position constraints [Huang et al. 2006]. The detail preservation term is nonlinear as it also depends on the position constraints. For efficiency reasons, the non-linear term is often approximated by a linear one using various strategies, such as local linearization [Sorkine et al. 2004], heuristic approximations for the local rotations [Lipman et al. 2004], propagation of user-defined transformations [Yu et al. 2004], and interpolation from handles [Zayer et al. 2005; Zhou et al. 2005; Au et al. 2007].

Botsch et al. [Botsch et al. 2006] presented a physically-plausible approach for mesh deformation that uses regions of the mesh as manipulation handles. While the interaction metaphor is very intuitive, the technique is not suitable for interactive modeling sessions.

Our approach avoids the need of performing non-linear minimization by using a simple, although effective, strategy: *we represent the coordinates of the mesh vertices in the regions of interest (ROI) in terms of frame fields associated to parametric curves used to control the deformation*. Since such frame fields are instantly updated as the curves are deformed, so are the coordinates of the mesh. By using external, as opposed to local frames, our approach provides a computationally efficient and general framework for mesh deformation.

Curves have been used to guide mesh deformation [Nealen et al. 2005; Zhou et al. 2005]. Zhou et al. [Zhou et al. 2005] use WIRE curves [Singh and Fiume 1998] to specify where a few vertices of the mesh should deform to, defining constraints to a linear system. In our approach, curve deformation is directly transferred to the mesh. Nealen et al. [Nealen et al. 2005] use curves to edit details of the mesh, while we use curves to specify both global and local deformations.

3 Technique Overview

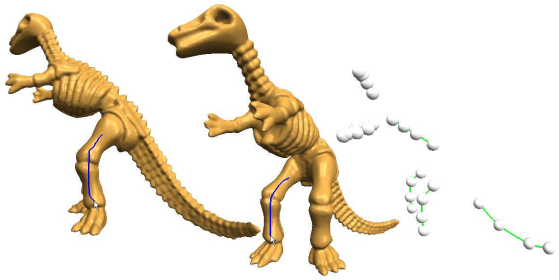

Figure 2: *Technique overview step one. The user oversketches the parts of the model he/she would like to deform (left and center). Each sketched line becomes a parametric curve in 3D that will be used as a handle for deforming the model (right). The small white balls represent the curves' control points.*

An interactive mesh-deformation session based on our approach consists of three main steps:

- *Sketching 2D curves over the 3D model* (Figure 2). The user oversketches the parts of the object he/she would like to deform. Each sketched line is turned into a parametric curve in 3D, which will be used as a handle for deforming the model. While oversketching, the user can freely change the camera's viewpoint. Figure 2 (right) shows a set of parametric curves created for different parts of the dino model. In addition, parametric curves can be automatically extracted from suggestive contours [DeCarlo et al. 2003], allowing the user to directly deform some visually-important features of the model (Section 5);
- *Connecting individual curves to form skeletons*, as shown in Figure 3. Creating skeletons with our technique is a simple

and intuitive task. However, there is no need to have a single skeleton per object. In fact, the user may even want to keep all curves separated (disconnected) from each other;

- *Deforming the model by moving the control points of the parametric curves in 3D*. Figure 4 illustrates this. On the left, the before and after states of a twist applied to the neck of the dino model. On the right, the deformation has been applied to its tail.

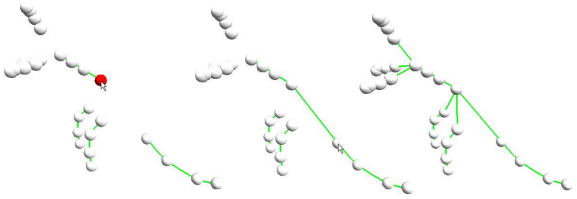

Figure 3: *Technique overview step two. The user may want to combine parametric curves in order to create complex skeletons.*

3.1 Region segmentation and handle creation

Our interface for sketching the curves is similar to the one used by [Kho and Garland 2005]. Given the sketched curves over the model, they are filtered as described in [Kho and Garland 2005] and parameterized to the [0,1] interval. Equally-spaced points in this parametric space are then projected on the mesh using the method described in [Moller and Trumbore 1997]. The projected points (on the mesh surface) are used to create an interpolating parametric curve that will be used as a deformation handle. We have chosen to use Catmull-Rom splines [Catmull and Rom 1974] because of their global smoothness, local control and interpolating characteristics. Hermite curves [Mortenson 1997] are also provided if the user wants to control the curves' tangents at the end points. The region of interest for the deformation produced by such a handle is delimited by two cutting planes positioned at the 3D curve end points. The orientation of each plane is defined by the curve's tangent at the corresponding end point.

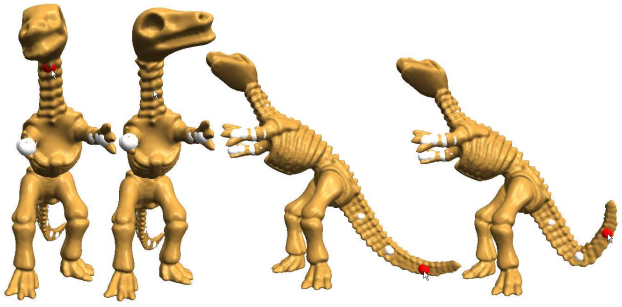

Figure 4: *Technique overview step three. By modifying a 3D curve, its deformation is transferred to the model. Twisting dino's neck (left) and lifting its tail (right).*

The control points of the parametric curves obtained as described are positioned on the mesh surface. We refer to such curves as *surface curves* and find them useful handles for deforming thick or disconnected parts of the mesh, as well as to add bumps/depressions to the surface mesh. We call *skeleton curve* a parametric 3D curve whose control points are inside the object's mesh. Skeleton curves can represent the structure of an object and be used to globally deform it. Thus, let k be the number of control points of a surface curve C. A skeleton curve S is obtained from C by: (i) creating $k-1$ planes perpendicular to C, each one halfway two consecutive control points; (ii) For each of the k subspaces delimited by the $k-1$ planes, compute the centroid of the ROI vertices falling in that subspace; (iii) use these centroids as the control points for a Catmull-Rom skeleton curve. Deciding which vertices fall in between any pair of planes is done using two dot products.

The interface allows the user to specify whether the new curve should be a surface curve or a skeleton one. Surface curves can be converted into skeleton ones and vice-versa. When converting a skeleton curve into a surface curve, the resulting control points are obtained by projecting each original control point onto the closest surface to the camera along the line connecting the control point itself and the camera's center of projection.

One can link several parametric curves by simply clicking on their control points. Curves can also be merged using the same interaction approach. This leads to a simple but effective interface for skeleton creation. Figure 3 illustrates the process. On the right, one sees a complete skeleton representing the structure of the dino model.

4 Mesh Deformation

In this section, we explain how the deformation of a curve is transferred into mesh deformations in an easy and efficient way. Briefly, we create a set of frames along the curve and represent the coordinates of the mesh vertices with respect to these local frames. As the user modifies the curve, the local frames are modified causing the associated surfaces to be deformed.

4.1 Defining Local Frames

Frenet frames [do Carmo 1976] are quite intuitive and can be computed analytically. Unfortunately, Frenet frames are not defined at inflection points or along straight segments. Moreover, at inflection points, Frenet frames can undergo some violent twists [Bloomenthal 1990]. We avoid these problems by defining coordinate frames along the curve using the following algorithm: let $\vec{u_0}$ be the curve's unit tangent vector at parameter value $t=0$ (point p_0). The second vector ($\vec{v_0}$) of the frame at $t=0$ is obtained by projecting $\vec{u_0}$ onto the world XZ plane and then normalizing and rotating the projection $\vec{u_{0p}}$ by 90 degrees (Figure 5). The third vector, $\vec{w_0}$, is obtained as the cross product $\vec{u_0} \times \vec{v_0}$. If $\vec{u_0}$ coincides with the vector $(0,1,0)$ in the world coordinate system, this procedure would fail. To avoid this problem, the components of $\vec{u_0}$ are inspected and the projection is performed on a world plane that would not lead to a null vector, by appropriately choosing one of the planes XY, YZ or XZ.

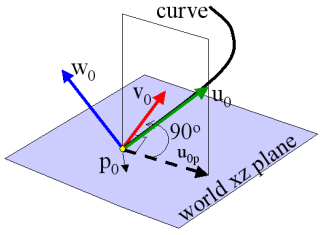

Figure 5: *The creation of a local frame at $t=0$. The vector $\vec{u_0}$ is the curve's unit tangent vector at $t=0$. $\vec{v_0}$ is obtained by normalizing and rotating by 90 degrees the projection $\vec{u_{0p}}$ of $\vec{u_0}$ onto the XZ plane. $\vec{w_0} = \vec{u_0} \times \vec{v_0}$.*

Once a frame has been created at $t=0$, we vary t in the interval [0,1] to define new local frames along the curve. We use the curve

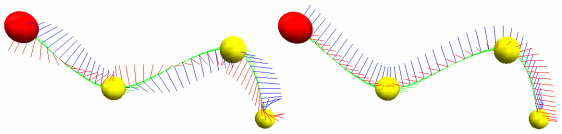

Figure 6: *Local frames along a parametric curve. Frenet frames computed analytically (left). Frames computed using our algorithm (right). Note the smooth transitions where the Frenet frames suffer sudden twists.*

position p_i and unit tangent vector $\vec{u}_i$ at $t = t_i$ to define an implicit plane at p_i. We then project the vector $\vec{v}_{i-1}$ (from the previous frame) onto this new plane, obtaining $\vec{v}_i$, after the projection has been normalized. $\vec{w}_i$ is again obtained as $\vec{w}_i = \vec{u}_i \times \vec{v}_i$. An additional operation consists in switching the sign of $\vec{w}_i$ in case the angle between $\vec{w}_i$ and $\vec{w}_{i-1}$ is bigger than 90 degrees. This simple algorithm is very efficient and produces a set of local frames along the curve that avoid the occurrence of undesirable twists. Figure 6 compares the results obtained by analytically computing a Frenet frame field along the parametric curve (left) with the result produced by our algorithm (right).

Bloomenthal [Bloomenthal 1990] describes a similar algorithm for obtaining reference frames along a space curve C. In his approach, the first frame F_0 is computed using the curve's tangent (T_0) and principal normal (N_0) vectors at point p_0. If C has zero curvature at p_0, N_0 is not defined and any vector perpendicular to T_0 can be used instead. The third vector of the frame is obtained as $B_0 = T_0 \times N_0$. The frame F_1 at point p_1, with tangent T_1, is then obtained by rotating F_0 around the vector $T_0 \times T_1$ in such a way that T_0 and T_1 coincide. Note, however, that the occurrence of straight segments along the parametric curve C is relatively common in our application. In such situations, $T_0 \times T_1$ is undefined, and even though $F_1 = F_0$, it is necessary to check if T_0 and T_1 are collinear and handle this as a special case. Although Bloomenthal's approach can be used to generate frame fields along our parametric curves, we prefer to use the technique described in this section because it naturally handles straight segments without the need to treat special cases.

4.2 Local Coordinates for the vertices in the ROI

Given the set of frames along a reference curve C, one can represent the coordinates of each vertex in the region of interest of C in terms of its frames. Thus, let π_{k-1} and π_k be the planes spanned by the pairs of frame vectors ($\vec{v}_{k-1}$, $\vec{w}_{k-1}$) and ($\vec{v}_k$, $\vec{w}_k$), respectively (Figure 7). All vertices v_j delimited by π_{k-1} and π_k will have their coordinates expressed with respect to the frame $F_{k-1} = (\vec{u}_{k-1}, \vec{v}_{k-1}, \vec{w}_{k-1})$. If an end point of C is not linked to other curves, C's defomation will affect all vertices of the mesh beyond that point. More specifically, if the first control point of C is not connected to another curve, all vertices before π_0 (first plane) will be rigidly transformed by F_0. Likewise, if the last control point of C is not connected to another curve, all vertices after π_n (last plane) will be rigidly transformed by F_n. The case involving blending between linked curves will be discussed later in this section.

Let $(\alpha_m, \beta_m, \gamma_m)$ be the coordinates of vertex v_m expressed in terms of frame F_{k-1} (*i.e.*, $v_m = p_{k-1} + \alpha_m \vec{u}_{k-1} + \beta_m \vec{v}_{k-1} + \gamma_m \vec{w}_{k-1}$), where p_{k-1} are the coordinates of the origin of frame F_{k-1}. Also let v_{mp} and v_{mn} be the projections of v_m onto the planes π_{k-1} and π_k, respectively. Such projections are obtained as the intersections of the line defined by v_m and the vector $\vec{u}_{k-1}$ with the planes π_{k-1} and π_k, respectively. We then associate to v_m the ratio $r_m = dist(v_m, v_{mp})/dist(v_{mn}, v_{mp})$ (Figure 7), computed based on the undeformed curve. *dist* is the Euclidean distance between

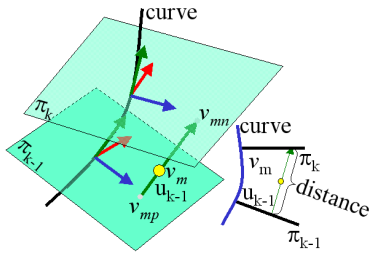

Figure 7: *During deformations, the position of vertex v_m is maintained in the same relative position along $\vec{u_{k-1}}$ w.r.t. its projections v_{mp} and v_{mn} on π_{k-1} and π_k, respectively.*

two points in 3D. This ratio will be preserved during deformations.

When the user modifies a parametric curve C, its set of frames are recomputed for the same values of the parameter t. Let C' be such a deformed curve. The first frame of C' cannot be created using the same approach used for C, because C''s tangent vector at $t = 0$ could require a different world plane for projection, causing the sets of frames from C and C' to completely diverge. Thus, let $(\vec{u}_0, \vec{v}_0, \vec{w}_0)$ be the coordinate frame of C at $t = 0$ and let $\vec{u'}_0$ be the unit tangent vector of C' at $t = 0$. Also, let ω be the angle between the vectors $\vec{u}_0$ and $\vec{u'}_0$ and let $\vec{r} = \vec{u}_0 \times \vec{u'}_0$. Thus, the vectors $\vec{v'}_0$ and $\vec{w'}_0$ are obtained from $\vec{v}_0$ and $\vec{w}_0$, respectively, by rotating them around $\vec{r}$ by ω degrees. This is the same kind of operation that Bloomenthal [Bloomenthal 1990] uses to propagate a frame field along a curve, but here it is used only to adjust the first frame of the deformed curve. Once the first frame of C' has been defined, its remaining frames are obtained using the same procedure defined for the subsequent frames of the original curve.

Given the set of frames of curve C', the new 3D coordinates of v_m after deformation could be computed simply as $v'_m = p'_{k-1} + \alpha_m \vec{u'}_{k-1} + \beta_m \vec{v'}_{k-1} + \gamma_m \vec{w}_{k-1}$, where p'_{k-1} is the point on C' for which $t = t_{k-1}$. This, however, would not take into account possible stretching applied to the curve when control points are moved apart. In order to transfer the stretching to the mesh, we scale v'_m's component along the u'_0 direction according the current distance between π'_{k-1} and π'_k times the ratio r_m. Thus, let v'_{mp} and v'_{mn} be the projections of the deformed vertex onto the planes π'_{k-1} and π'_k, respectively, where $v'_{mp} = p'_{k-1} + \beta_m \vec{v'}_{k-1} + \gamma_m \vec{w}_{k-1}$. v'_{mn} is found by calculating the intersection of the line defined by v'_{mp} and $\vec{u'}_{k-1}$ with π'_k (Figure 7). The new 3D coordinates of v_m are then recomputed as $v'_m = v'_{mp} + \alpha_m(r_m) dist(v'_{mp}, v'_{mn}) \vec{u'}_{k-1}$.

The number of frames used along the curve is important to guarantee the smoothness of the deformation. If very few planes are used, the curve behavior may not transfer well to the deformed mesh. In all examples shown in the paper and accompanying video, we used 200 frames for each curve.

Twisting, bending and scaling: Once the coordinates of the vertices in a ROI have been represented w.r.t. the frames in a handle curve, one can deform the mesh. Free-form deformations are performed by interactively moving the curve's control points around. Some other interesting effects, such as twisting and scaling, are obtained by operating directly over the frame field of the curve. For instance, by interpolating, along a segment of C, a local rotation of the frames vectors $\vec{v}_i$ and $\vec{w}_i$ around $\vec{u}_i$ produces some twisting effect. Non-uniform and localized scaling effects can be achieved by independently scaling the vectors that form the frames. All these operations can be combined to create more complex deformations.

Figure 8 illustrates the results produced by twisting and bending operations. Figure 8 (a) shows a block as the original mesh. A twisted version of the block is shown in (b) and was obtained simply by interpolating a rotation along the frame field of the handle curve, which is shown to its right. Figure 8 (c) shows the result of bending the block, obtained by moving the curve's control points (shown over). In (d) one sees the combined result of twisting and bending the block. Note the smooth results and how easy these non-trivial deformations are obtained.

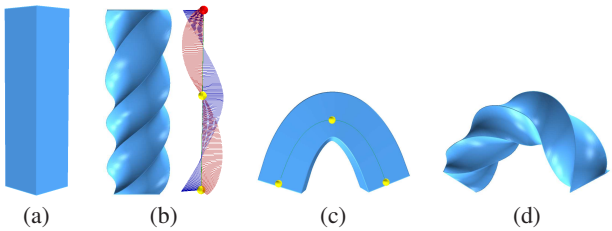

Figure 8: *Twisting and bending operations performed with our technique. A reference block (a). Twisted block obtained by interpolating a rotation along the curve's frame field (b). (c) Bent block obtained by moving the curve's control points (shown over). (d) Twisted and bent block combining the transformations (b) and (c).*

Blending between curves: To avoid possible artifacts in parts of the mesh where one handle is linked to another (*e.g.*, in the skeleton shown in Figure 3 right, several curves are connected at their endpoints), a blending function is used to transition among the deformations. This blending is performed automatically, being completely transparent to the user. A zero-mean Gaussian blending function ($b(x) = e^{-x^2/(2\sigma^2)}$) is associated to each control point of a curve C linked to another curve. Thus, let P be the set of all control points linked to a given end control point p_j of C and let p_a be the centroid of P. Empirically, we found that a standard deviation $\sigma = \sqrt{dist(p_j, p_a)}$ works fine. Let v_m be a vertex outside the ROI of curve C. In order to guarantee a smooth transition between the deformed and non-deformed regions of the mesh, we compute the influence of p_j over v_m as the weight $w_m = b(dist(v_m, p_j))$. Thus, during the deformation induced by C, the new coordinates of v_m are given by $v_m = (1 - w_m)v_m + w_m v_{rm}$, where v_{rm} are the coordinates that v_m would have if it had been rigidly transformed according to the transformation applied to plane π_j (defined by the frame vectors $\vec{v}_j$ and $\vec{w}_j$ at p_j). Since the function $b(dist(v_m, p_j))$ quickly approaches zero as the distance between v_m and p_j increases, we achieve a smooth transition between the two regions.

Local self-intersection avoidance: Self-intersection is a common problem in mesh deformation and several approaches for trying to avoid it have been proposed, especially in the context of character skinning [Mohr et al. 2003; Kry et al. 2002]. We take advantage of the curve's frame field to devise a simple but effective way of automatically avoiding local self-intersections. Let V_k be the set of mesh vertices in the ROI of curve C that are represented in C's frame ($\vec{u}_k$,$\vec{v}_k$, $\vec{w}_k$). The projection of V_k onto π_k defines a circle of influence for V_k (Figure 9). By guaranteeing that the circles of influence of all such planes do not intersect, self-intersections on the deformed mesh are avoided.

Let π_{k-1} and π_k be two such planes with origins at p_{k-1} and p_k created along the curve at parametric values t_{k-1} and t_k, respectively. Let l be the intersection line between π_{k-1} and π_k. Also, let r_{k-1} and r_k be the radii of the circles of influence at π_{k-1} and π_k, respectively. If $dist(l, p_{k-1}) < r_{k-1}$ and $dist(l, p_k) < r_k$ then an intersection is assumed. To avoid self-intersections, we rotate

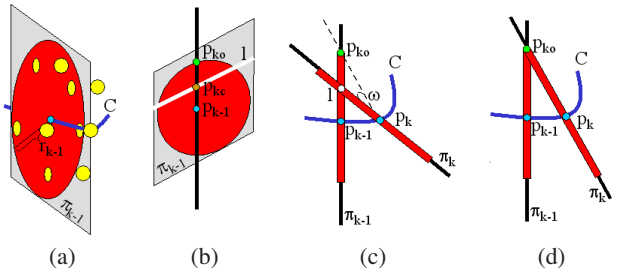

Figure 9: *automatically avoiding local self intersections. (a) The projection of the vertices associated to a plane π_{k-1} (onto π_{k-1}) defines a circle of influence with radii r_{k-1}. (b) l is the line of intersection of π_{k-1} and π_k. p_{kc} is the point on l closest to p_{k-1}, the point on C evaluated at $t = t_{k-1}$. p_{ko} is a point outside the circle of influence of π_{k-1}. (c) The plane π_k is rotated by ω degrees around the line parallel to l passing through p_k. (d) After rotation, self-intersections no longer occur.*

the plane π_k so that the circles of influence in π_{k-1} and in π_k do not self-intersect anymore. Thus, let p_{kc} be the point on l that is closest to the p_{k-1}. Also, let p_{ko} be another point along the segment connecting p_{k-1} and l passing through p_{kc}, such that p_{ko} is the closest point to p_{k-1} outside the circle of influence of π_{k-1} (Figure 9 b). Let ω be the angle between the vectors $\vec{a}_c$ and $\vec{a}_o$, defined as $\vec{a}_c = (p_{kc} - p_k)$ and $\vec{a}_o = (p_{ko} - p_k)$. Self-intersection is avoided by rotating the frame F_k by ω degrees around the direction of line l (Figures 9 c and d). Note that in order to avoid self-intersection, we have forced the vector $\vec{u}_k$ of the local frame not to coincide with the curve's tangent direction at $t = t_k$.

5 Suggestive Contours

Suggestive contours [DeCarlo et al. 2003] are points on the surface that are not contours yet but would become with a slightly change in the cameras viewpoint. Together, contours and suggestive contours convey the shape of an object quite well and that explains why these lines are often used in non-photorealistic rendering. As these lines are simpler than the mesh itself but still represent the major object details, using them as handles seems to be a promising approach for deforming complex objects, as the user can abstract some parts of the mesh and concentrate on the features that convey more information.

Creating parametric curves from automatically-extracted contours and suggestive contour lines and transferring their deformations to the actual meshes is similar to what has been described for sketched lines in the previous sections. However, a blending function based on approximated geodesic distances (as opposed to Euclidean distances) should be used for better results. Such an approximation can be calculated using a region-growing algorithm starting at the triangles where a (suggestive) contour lies and stopping when another (suggestive) contour is found.

Figure 10 illustrates the steps associated with the use of (suggestive) contours to perform mesh deformation. The two images on the left show a view of the Homer model (a) and its corresponding (suggestive) contours (b). The lines shown in (b) are automatically extracted and converted into parametric lines. Figure 10(c) shows the resulting parametric lines with their corresponding control points seen as little white spheres. Figure 10(d) shows the result obtained by moving some control points of the parametric curves associated to the model's mouth and eyebrows.

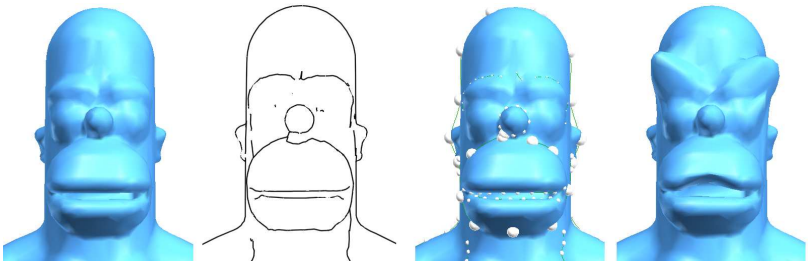

Figure 10: *Suggestive contours can be automatically computed for 3D models and distorted to perform mesh deformation.*

6 Results

We have implemented the algorithms presented in this paper using C++ and have used OpenGL for visualization. We have applied our technique to deform several models. The performance measurements were carried on an AMD Athlon 64 3700+ processor running a 32-bit Microsoft Windows XP with 2GB of RAM memory.

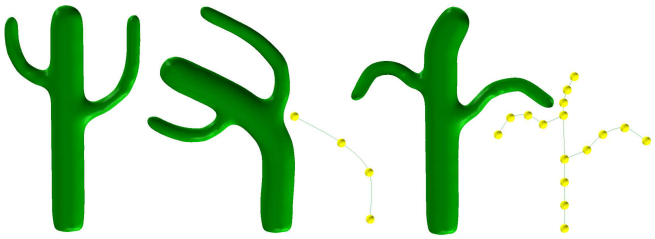

Figure 11: *The cactus model (left) was deformed using a single handle curve sketched over its trunk (center). Independent deformations can be applied to the individual branches by building a skeleton and deforming the curves associated to the branches (right).*

The examples shown in the paper stress how easy deformations can be performed with our approach. Figure 1 shows the Armadillo model in several poses and demonstrate our approach's ability to perform large deformations using a simple interface. The legs and arms were modified by translating some control points. In the second example of Figure 1 the Armadillo's body was scaled and bent. Figure 8 shows a block being twisted and bent using a handle curve with 3 control points. Note the smoothness of the twisting and the bending.

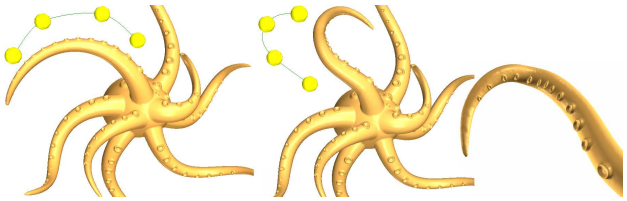

Figure 12: *Deformation including large rotations. One of the tentacles of the original octopus model (left) was deformed (center). A close-up view of the deformed tentacle seen from the back (right).*

Figure 11 shows a cactus model (left) being deformed by a single curve sketched over its trunk (center). Note how the branches nicely follow the deformation. Figure 11 (right) illustrates the versatility of our technique, which also allows the branches to be deformed independently, by using a skeleton that mimics the structure of the cactus. This capability gives animators freedom to express themselves artistically. Figure 12 (center) shows a large and smooth deformation of one tentacle of the octopus model (left). Figure 12 (right) shows a close-up view of the deformed tentacle seen from the back.

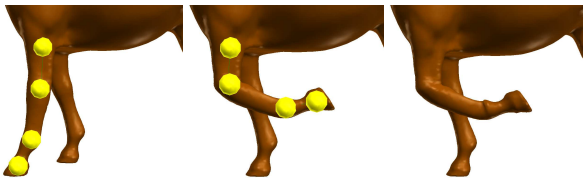

Figure 13: *Deforming a horse leg (left) using skeleton constraints. Each segment between two adjacent control points in the handle is a Hermite curve with small tangent vectors. The resulting deformation (right) preserves the rigidity of the limb segments.*

Figure 13 illustrates the use of deformation with skeleton constraints (*i.e.*, preserving the rigidity of limb segments in articulated figures). While Huang et al. [Huang et al. 2006] enforce such constraints via non-linear least-squares optimization and require that each articulated segment be a closed mesh, our approach imposes no restriction on the mesh topology. On the other hand, Huang et al.'s approach can preserve volume, which ours cannot. We implement skeleton constraints by treating each segment between two adjacent control points as a Hermite curve [Mortenson 1997] with small tangent vectors. This produces straight articulated segments with slightly bent endings, as desired. Figure 13 (left) shows a horse leg in its original position with a handle curve superimposed. The resulting deformed leg is shown on the right. Using our approach, the user can deform each leg at a time in real time.

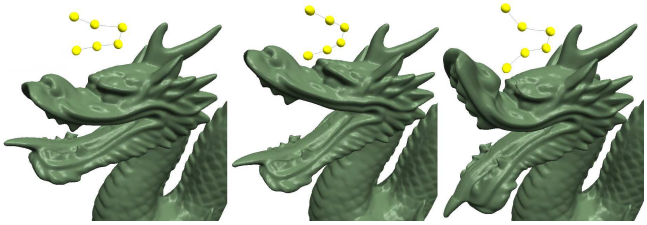

Figure 14: *Deformations of the dragon model. Left: original model. Center and right: dragon mouth deformed in different ways. Handle curves shown on top.*

Figure 14 shows the dragon model with its mouth opened in two different ways. The original model is shown on the left for comparison and the handle curves are shown on top. Figure 15 shows another example of deformation produced with our technique. The

result shown on the right was obtained with a one-step translation of a single control point of a handle curve associated to the model on the left.

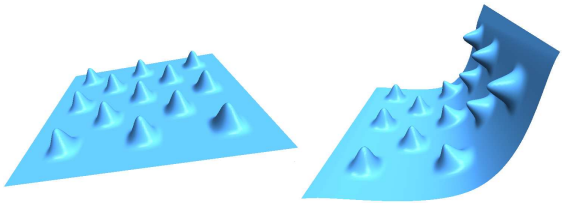

Figure 15: *Original model (left). A deformation performed using our technique by translating a single control point (right). Note how the local structures are plausibly re-oriented on the resulting surface.*

Multiple-component meshes, non-manifold and non-orientable surfaces pose some challenge to differential-based approaches, which require computing and/or propagating a discrete frame field based on the mesh local properties. As the frames used in our approach live on the handle curve, all these configurations are treated in a natural and uniform way. Figure 16 (left) shows a horse model after a ring of polygons has been removed to create two disconnected components. A handle curve with four control points has been attached to the back of the horse and used to create the deformation shown on the right. Note that, although disconnected, the deformed parts align nicely.

Figure 17 demonstrates the power of our metaphor by performing a deformation on non-orientable and non-manifold surfaces. On the left, one sees a Moebius strip (top) and a non-manifold surface. The images on the center and on the right show the deformed models after moving some control points in each of their corresponding handles.

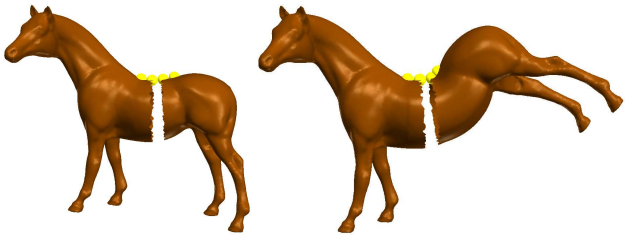

Figure 16: *Deformation of a mesh with multiple components. Horse model with a ring of polygons removed (left). Deformed model obtained by moving some control points. Note how the parts still align nicely after the deformation.*

Model	Vertices	Triangles	Setup (s)	Deform. (fps)
Horse	19,851	39,698	0.0545	59.52
Dino	56,194	112,384	0.1550	34.60
Armadillo	172,974	345,944	0.2914	14.94
Dragon	437,645	871,414	1.0697	7.12
Buddha	543,652	1,087,716	3.4130	4.50

Table 1: *Times obtained using our technique for deforming several models. The right column shows the performance of the deformation (in fps) when deforming all vertices of the model.*

Table 1 provides some performance statistics for our technique. The setup time corresponds to the time required for region segmentation (Section 3.1), which is performed only once per curve, when it is created. For these measurements, we have transformed all vertices of the mesh every time the user modified the curve. Note that we only need to deform the vertices falling in the ROI of the modified curve. By transforming all vertices of the model, we define a lower bound for the performance of our technique when applied to that particular model. These numbers (last column of Table 1) indicate that our technique can operate at interactive rates, even with meshes composed by hundreds of thousand vertices. Depending on the size of the ROI, our approach can handle meshes containing millions of polygons in real time. These numbers compare favorably to all previously known mesh deformation techniques.

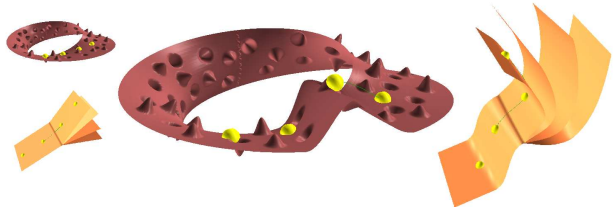

Figure 17: *Deforming non-orientable (center) and non-manifold (right) surfaces.*

7 Conclusion

We have presented an interactive technique for immediate high-quality mesh deformation. It is intended as an attractive alternative for artists and users looking for a simple and quick way of producing plausible deformations with interactive feedback. Our approach is based on the metaphor of transferring smooth deformations from parametric curves to complex 3D models, producing visually-pleasing deformations. The use of parametric curves allows our technique to be implemented using a very intuitive interface, and giving the user fine control over the deformation. Skeleton constraints and local self-intersection avoidance are efficiently enforced, and the use of suggestive contours to guide model deformation provides a way of directly editing visually-important object details. We have demonstrated the effectiveness and versatility of our approach by using it to deform, bend and twist several models and showing that our results are visually pleasing.

By using a set of frames to deform the space along the curve, our approach is quite general, handling non-orientable and non-manifold surfaces, and meshes comprised of multiple components. Unlike other mesh deformation techniques, our approach does not optimize any aspect of the deformation and, therefore, it cannot guarantee that the model's original properties (*e.g.*, volume) will be preserved after the deformation. However, given its relatively low computational cost, our technique is considerably faster than previous approaches for achieving visually-similar results, being able to handle very large meshes at interactive rates. Thus, it provides a good alternative for artistic prototyping and casual users.

Using other types of parametric curves seems a promising area of future exploration, as the properties of the curve are directly transferred to the deformed mesh. One possibility would be to perform physically plausible deformations using the technique described in [Lenoir et al. 2005]. Our technique is also suitable for GPU acceleration, as the deformation imposed by the curve can be applied to all vertices in parallel. This should make our approach even more appealing for prototyping applications.

Acknowledgements

The Armadillo and Dragon models were provided by the Stanford 3D scanning repository. The Dino model is a courtesy of Cyberware. The plane with bumps in Figure 15 was kindly provided by Mario Botsch and the Octopus model by Mark Pauly. The Homer model was provided by the AIM@SHAPE project. Vitor Pamplona modeled the Moebius strip and the non-manifold surface. The cactus model is from the web page of A. Ben Hamza. The authors would like thank Francisco Pinto and the anonymous reviewers for their comments and suggestions. Fausto Richetti Blanco was supported by CNPq (Process 130817/2005-8). Additional support was provided by Microsoft Brazil.

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2, 105–114.

AU, O. K.-C., FU, H., TAI, C.-L., AND COHEN-OR, D. 2007. Handle-aware isolines for scalable shape editing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, ACM Press, New York, NY, USA, 83.

BLOOMENTHAL, J. 1990. Calculation of reference frames along a space curve. *Graphics Gems I*, 567–571.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3, 630–634.

BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. Primo: Coupled prisms for intuitive surface modeling. *Eurographics Symposium on Geometry Processing 2006*, 11–20.

CATMULL, E. E., AND ROM, R. J. 1974. A class of local interpolating splines. *Computer Aided Geometric Design*, 317–326.

CHANG, Y.-K., AND ROCKWOOD, A. P. 1994. A generalized de casteljau approach to 3d free-form deformation. In *SIGGRAPH '94*, 257–260.

COQUILLART, S. 1990. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *SIGGRAPH*, 187–196.

DECARLO, D., ET AL. 2003. Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3, 848–855.

DO CARMO, M. P. 1976. *Differential Geometry of Curves and Surfaces*. Prentice Hall.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. In *SIGGRAPH*, 325–334.

HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. In *SIGGRAPH '92*, 177–184.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. In *SIGGRAPH '06*, 1126–1134.

KHO, Y., AND GARLAND, M. 2005. Sketching mesh deformations. In *SI3D '05*, 147–154.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98*, 105–114.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *SCA '02*, ACM Press, New York, NY, USA, 153–159.

LENOIR, J., GRISONI, L., CHAILLOU, C., AND MESEURE, P. 2005. Adaptive resolution of 1d mechanical b-spline. In *GRAPHITE '05*, 395–403.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., ROSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proc. Shape Modeling International*, 181–190.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH '96*, 181–188.

MOHR, A., TOKHEIM, L., AND GLEICHER, M. 2003. Direct manipulation of interactive character skins. In *Proc. SI3D*, 27–30.

MOLLER, T., AND TRUMBORE, B. 1997. Fast, minimum storage ray-triangle intersection. *J. Graph. Tools 2*, 1, 21–28.

MORTENSON, M. 1997. *Gometric Modeling, 2nd edition*. John Wiley and Sons, Inc.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3, 1142–1147.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *SIGGRAPH '86*, 151–160.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *SIGGRAPH '98*, 405–414.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *SGP '04*, 175–184.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3, 80.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3, 644–651.

ZAYER, R., RÖSSL, C., KARNI, Z., AND SEIDEL, H.-P. 2005. Harmonic guidance for surface deformation. In *Eurographics 2005*, vol. 24, 601–609.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph. 24*, 3, 496–503.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *SIGGRAPH '97*, 259–268.