Skinning Characters using Surface-Oriented Free-Form Deformations

Karan Singh
karan@paraform.com

Evangelos Kokkevis
vangelisk@home.com

Alias|wavefront
210 King St. E., Toronto, Canada M5A 1J7

Abstract

Skinning geometry effectively continues to be one of the more challenging and time consuming aspects of character setup. While anatomic and physically based approaches to skinning have been investigated, many skinned objects have no physical equivalents. Geometric approaches, which are more general and provide finer control, are thus predominantly used in the animation industry. Free-form deformations (FFD) are a powerful paradigm for the manipulation of deformable objects. Skinning objects indirectly using an FFD lattice reduces the geometric complexity that needs to be controlled by a skeleton. Many techniques have extended the original box-shaped FFD lattices to more general control lattice shapes and topologies, while preserving the notion of embedding objects within a lattice volume. This paper in contrast, proposes a surface-oriented FFD, where the space deformed by the control surface is defined by a distance function around the surface. Surface-oriented control structures bear a strong visual semblance to the geometry they deform and can be constructed from the deformable geometry automatically. They also allow localization of control lattice complexity and deformation detail, making them ideally suited to the automated skinning of characters. This approach has been successfully implemented within the *Maya2.0* animation system.

Key words: Character animation, skinning, deformers, free-form deformations.

1 Introduction

A layered approach [1] to the modeling and animation of articulated figures is a widely adopted methodology. The layers may be broadly classified into skeletal, muscle, underlying tissue and skin. These layers are largely symbolic of their contribution to the visual appearance of the animated character, since physical equivalents of bones or muscles for characters modeled from inanimate objects such as a lamp, need not exist. Hair, clothes and accessories form further layers on many characters. Layers are often omitted, collapsed together, or further classified depending on the sophistication of the application.

The *skin* is particularly important since it largely establishes the visual appearance of a character. A number of techniques for the modeling and animation of the *muscle* and *skin* layers have been investigated [1, 5, 10, 11, 13, 16, 17, 18]. The typical workflow for setting up an articulated character involves building a model representing the geometric skin of the character in some pose. An underlying skeletal structure comprising of reference frames at joints is also constructed to control the movement of the articulations. More sophisticated methodologies sometimes also model underlying bones and muscles using geometry. The skinning algorithm is responsible for deforming the geometric skin to respond to the motion of the underlying skeleton. Skinning approaches can be classified as geometric or physically based. While a number of physical models for muscle and skin [4, 11, 16, 18] exist, techniques used in the animation industry continue to be predominantly geometric [1, 5], because of their generality and control.

Figure 1: Surface-oriented FFD overlaid on a character

It is worth noting that characters are often modeled using a number of adjacent parametric surface patches, for reasons of smoothness and ease of modeling, texturing and rendering. Animators, however, would rather deal with a single contiguous skin surface since it obviates is-

sues of continuity between adjacent pieces of skin during animation. A geometric skinning approach that presents a single skin interface for setup and animation is thus desirable. Subdivision surfaces provide the smoothness and parameterization characteristics of surface patches while presenting a single control mesh as an animation interface [5]. We aim to provide the animation interface and control of a subdivision surface to an underlying skin of arbitrary geometric representation.

A good skinning algorithm needs to provide an automated attachment of points representing the skin surface of a character to an underlying skeletal and muscle structure. Subsequently, it should be easy for an animator to edit the default attachments, as it is difficult to universally predict how an arbitrarily shaped object is intended to be controlled by its skeleton. Once pieces of skin are attached to corresponding pieces of muscle and bone, the strength with which pieces of the skin are deformed by parts of the skeleton should be easy to control. The effect on skin from changes in underlying layers, such as muscle shape or bone position should be easy to overlay. We avoid an explicit bone and muscle model since the range of articulated deformable objects and skin deformation effects is as vast as the animator's imagination. Instead we emphasize a system where any localized deformation can be easily specified and controlled by underlying skeletal motion. The resulting skin deformation should be smooth and predictable. Finally, the approach should be efficient allowing real-time interactivity for at least a low resolution model of the geometric skin. Empirical geometric approaches to skinning [1, 10, 13, 17] have shown realistic results at interactive rates.

A number of techniques have advanced the box-shaped lattices of the original FFD formulation [12], to allow for greater generality in the shapes and topologies of the control lattice. This is in accordance with the general trend for the control structures of higher level deformation techniques to bear a visual correlation to the geometry they deform [5, 14]. Most FFD approaches, however, preserve the notion of a volume enclosed by the control lattice, within which objects are embedded.

In contrast, we propose here a deformation technique that is surface-oriented. The region of space deformed by the control point structure is not the volume enclosed by the control points but is based on a distance metric from a surface defined by the control point structure. Surface-oriented deformation control structures provide a better visual representation of the geometry they deform and can typically be constructed from the deformable geometry itself (See Figure 1). They allow better localization of control lattice complexity and deformation detail as illustrated by the results in our implementation.

Our surface-oriented FFD thus aims to represent and control any underlying object by a single control polymesh. Unlike subdivision surfaces the control polymesh does not have a limit surface that represents the object. Our goal instead is to allow the advantages of a single control mesh to represent and control an object that has an alternate surface model (such as a set of surface patches, a different control mesh or even a subdivision surface). The control mesh can be used to both visually represent the surface model and to drive the deformation of the object. The control mesh can be user created or automatically synthesized from data, for example by tesselating and stitching various surface patches that define the deformed skin. In addition to their application in skinning characters, surface-oriented FFDs are a useful tool for the multi-resolution modeling and animation of objects.

The remainder of this paper is organized as follows: Section 2 describes characteristics of existing free-form deformation techniques and motivates the surface-oriented free-form deformation approach. Section 3 presents the surface-oriented deformation algorithm. Section 4 provides an analysis of the properties of the algorithm and describes an extension of it. Section 5 describes the implementation of an automated skinning workflow based on surface-oriented free-form deformations, within the modeling and animation system *Maya2.0*. Section 6 concludes with a discussion of the results obtained.

2 Free-form Deformation Techniques

In this section we present an overview of a number of existing free-form deformation techniques and contrast their properties with the characteristics of our surface-oriented deformation algorithm described in Section 3.

Free-form deformations (FFD) were originally introduced by Sederberg and Parry [12] as a general technique where objects are deformed by warping a volume of space within which the objects are embedded. The volume of space is typically defined using a structure connecting a set of control points. Spatial deformations are then accomplished by the manipulation of these control points. A one-to-one correspondence is established between points within the original and deformed volumes of space. Objects embedded within the original volume are thus deformed by mapping the point-set representing the object to their corresponding points in the deformed volume. This process typically involves calculating a parameterization of the volume based on the topology of its control point structure. The actual mapping of an undeformed point to a point in the deformed volume is then a function of the deformed positions of the control points for the given parameterization. Continuity of the corre-

spondence function is crucial to the smoothness properties of the deformed object.

Sederberg and Parry [12] used a parallelopiped lattice of control points to define a trivariate Bezier volume. The mapping of points within the parallelopiped volume to a trivariate basis is straightforward. Evaluating the deformed point is simply a matter of evaluating the Bezier equation for the deformed set of control points. Griessmair and Purgathofer [6] extended this technique to employ a trivariate B-spline basis. While these methods are simple, efficient and in popular use they suffer from the drawback of a restrictive original volume shape. Parallelopiped volumes rarely bear any visual correlation to the objects they deform and typically have a globally uniform lattice point structure that is larger than is required for the deformations to which they are applied.

Coquillart [3] extended the box-shaped lattices to allow for a richer set of shapes (EFFD), constructed by join operations applied to paralleloped lattices. The parameterization of a point within the original trivariate volume is calculated numerically, making the technique less stable than the original FFD [12] in the general case.

Chang and Rockwood [2] present an approach where a deCasteljau approach of repeated affine transformations defines the deformable space around a Bezier curve. The approach is intuitive and fast but restricted in the range and local control of the deformations it can capture.

MacCracken and Joy [7] use a volume equivalent of the Catmull-Clark subdivision scheme for surfaces to iteratively define a volume of space based on a control point structure of arbitrary topology. This is a significant step in increasing the admissible set of control lattice shapes. The technique is powerful and its only real shortcoming are the potential continuity problems of the mapping function (a combination of subdivision and interpolation) of points within the volume. The approach also suffers from the same discontinuity problems as Catmull-Clark surfaces at extraordinary vertices.

Dirichlet free-form deformations [9] is an approach based on the Voronoi structure defined within the convex hull of a set of points. While there is no restriction on the shape of the volume, the deformations are controlled solely by the parameterization defined by natural neighbour interpolants. These interpolating functions have singularities that result in unwanted deformation artifacts.

All the above approaches are strongly volume-oriented. The structure of the control points explicitly defines a volume of deformable space. The deformation function $D(P)$ for a point P can be typically represented by $D(P) = \sum_{i=1}^{n} W_i(P)P_i$, where P_i is a control point and W_i a function that maps a point P to a weight value for the control point P_i. For FFDs $W_i(P) = B_i(s,t,u)$, where B is the Bezier basis function and s,t,u the parameterization of P within the parallelopiped volume.

The property that affine transformations to the control lattice are transmitted as such to the deformed points is desirable. Suppose an affine transformation M were applied to the lattice. $D(P) = PM = \sum_{i=1}^{n} W_i(P)(P_iM)$ or $P = \sum_{i=1}^{n} W_i(P)P_i$, where P_i are the positions of the control points in the original lattice. Thus for affine transformations to be captured by the deformation, the weighted average of control points for any point in space point should be the point itself. Additionally $\sum_{i=1}^{n} W_i(P) = 1$ for the convex hull property of the deformation to hold. These properties can be verified for the approaches described thus far.

Singh and Fiume [14] provided a different direction to free-form deformations by making them surface-oriented, in that there was no explicit mapping of points between two deformable volumes. Instead points in space were associated with surface elements, parametric curves called wires. Transformation of these associated surface elements result in a deformation of space surrounding the surface element. The control structure of a surface-oriented deformer typically bears a strong visual correlation to the object it deforms. Local control over the deformation is easier and the arbitrary nature of the control point structure makes it possible to introduce detail locally without a global change to the object. At the same time it is harder to ensure continuity properties and perfectly transmitted affine transformations for surface-oriented deformations. This paper addresses these issues for a polygon based deformer.

3 Surface-oriented deformations

The surface-oriented deformation algorithm described in this paper, binds the surface S of a deformable object to a deformer object O. Manipulation of O is then tracked by S. Formally, we define the deformer O as a triple $\langle D, R, local \rangle$ where D and R are surfaces, referred to as the *driver surface* and *reference surface* and $local$ a scalar value. R is a congruent copy of the deformer surface D, that is made when surface S is bound to O. Subsequent manipulation of D causes a deviation relative to R that drives the deformation of surface S. The parameter $local$ provides control over the locality of the deformation.

For the purpose of our algorithm we need to be able to compute localized orientation and scaling information at points on the surfaces D and R. We also require a unique and intuitive correspondence between points on D and R. This section treats D and R to be polygon based surfaces of matching topology, for which these calculations are straightforward and efficient.

3.1 Overview of the Algorithm

There are three phases to the deformation process: The bind phase, the registration phase and the deformation phase. The bind phase takes place once whereas the registration and deformation phases are repeated as needed. During the bind phase, the user-specified deformer surface becomes the driver surface D. An identical copy of it becomes the reference surface R, which along with a user-specified $local$ value define the deformer object O.

Let the driver D, and reference surface R of a deformer object be represented by a collection of enumerated *control elements*. These control elements are the triangular facets of polygon based surfaces D and R (non-triangular faces of D are triangulated before the creation of R during the bind phase). There is thus a simple bijective correspondence between the control elements of D and R based on element index.

The registration phase computes how much each control element of the deformer object affects the deformation of each point P representing the surface S. This scalar value, referred to as the *influence weight* of the control element for P, is calculated using a distance metric. Control elements closer to P have a higher influence weight, and therefore affect the deformation of P more than elements further away from P. The registration phase typically takes place once, right after the bind phase and needs to be repeated only if the position of the reference surface R changes relative to the surface S.

The deformation phase follows the registration and is repeated every time the deformer object's driver surface D is manipulated. The influence weights calculated in the registration phase as well as the spatial difference between the control elements of the reference and driver surfaces are used to determine the deformation of each point P on the surface S.

The registration and deformation phases are now described in greater detail.

3.2 Registration

During the registration phase, the influence weights for all the control elements and points P of the deformable surface S are computed. Typically, the surface S is represented by the set of points P^S that are necessary to construct or approximate S. P^S could therefore be a set of vertices in a polygonal mesh, a set of control vertices in a free form surface or an unstructured set of points in space. The deformation is applied to points P of this set. Of the two surfaces of the deformer object O, only the reference surface R is used in this phase. In our implementation, a distance metric represented as a scalar function $f(d, local)$, is employed to compute the influence weights. The first parameter, d, is the distance of P from the control element. The second parameter, $local$, controls the rate at which the function f decays in value with an increase in distance d. We define the function $f(d, local)$ for any $d, local \geq 0$ to be:

$$f(d, local) = \frac{1}{1 + d^{local}}. \quad (1)$$

We define the distance of point P from a triangular facet to be the length of the vector $\overrightarrow{PP'}$ where P' is the point on the surface of the facet that is the closest to point P. For each point P and each control element k of the deformer object's reference surface R we define the corresponding weight $w_k^P = f(d_k^P, local)$, where d_k^P is the distance of point P from control element k. The influence weights for a point P are normalized to preserve the convex hull property described in Section 2. The normalized weight vector for point P is defined as $\mathrm{U}^P = \{u_1^P, u_2^P, \ldots, u_n^P\}$, where n is the number of control elements of the influence object's surfaces and u_k^P is the normalized weight of control element k for point P defined as $u_k^P = \frac{w_k^P}{\sum_1^n w_k^P}$. Section 5 will show that in practice u_k^P is set to zero for all but a few control elements making the approach quite efficient.

The control elements as used for this algorithm define not only a local position in space but also a local coordinate frame with axes $\overrightarrow{e_1}, \overrightarrow{e_2}, \overrightarrow{e_3}$. In other words, each control element defines its own coordinate system that can be represented compactly with a 4×4 transformation matrix Q. We denote transformation matrices of elements on the driver and reference surface as Q^D and Q^R respectively.

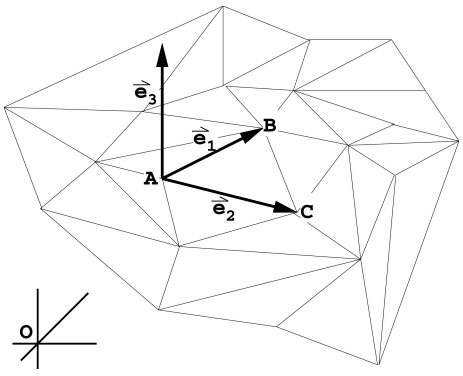

Figure 2: The coordinate system of a polymesh face.

The position vector and the attached coordinate system of a control element corresponding to a triangular facet of a polygonal surface are derived directly from the vertices and edges of the facet (See Figure 2). An arbitrary ordering may be assigned to the three vertices of the triangular facet. Referring to these three vertices as vertices A, B and C we define the coordinate system for the control

element with its origin at A and axes as :

$$\overrightarrow{e_1} = \overrightarrow{B} - \overrightarrow{A}, \overrightarrow{e_2} = \overrightarrow{C} - \overrightarrow{A}, \overrightarrow{e_3} = \frac{\overrightarrow{e}_1 \times \overrightarrow{e}_2}{||\overrightarrow{e}_1 \times \overrightarrow{e}_2||}. \quad (2)$$

In addition to the influence weights, the registration phase computes a representation of undeformed points P of surface S in the coordinate system defined by each control element k of the reference surface R. This operation can be easily carried out by inverting the transformation matrix Q_k^R of the control element. $P_k^R = P(\mathrm{Q}_k^R)^{-1}$, where P_k^R is the representation of the point P in the local coordinate system of control element k of the reference surface R.

During the deformation phase the point P is deformed to preserve its local position, P_k^R, in the coordinate frame of the control element k of the driver surface D.

3.3 Deformation

The deformation procedure maps each point P in the set of points P^S defining the undeformed surface S to a point P_{def}. The set P_{def}^S of all points P_{def} defines the deformed object surface.

As the user manipulates the driver surface D of the deformer object, control elements on D change shape, position and orientation. Comparing the coordinate system defined by the element k on the reference surface and the element k on the driver surface makes it possible to calculate the deformation effect of element k on a point P. The effect of each control element k is weighted by the corresponding normalized weight factor u_k^P and is added to the contributions of all the other control elements.

During the registration phase, the representation of point P in the local coordinate system of the control element k on reference surface R was calculated to be P_k^R. In the deformation phase, we deform the point to have the same local representation in the coordinate system of the corresponding driver surface control element. The world space position of the point P as deformed by control element k is thus, $P_k^{def} = P_k^R \mathrm{Q}_k^D$, where Q_k^D is the transformation matrix corresponding to the local coordinate system of the control element k of the driver surface. The effect on point P of each control element is weighted by the corresponding normalized weight value stored in vector U^P. The weighted effects are added to compute the resulting point P_{def} on the deformed surface:

$$P_{def} = \sum_{k=1}^{n} u_k^P P_k^{def}. \quad (3)$$

The algorithm can be adapted to allow parametric surface patches to represent the driver and reference surface by specifying control elements as a parametric sampling of points. The sampling density is a trade-off between computational efficiency and the fidelity with which the surface is represented. Sampling at knot vector parameter values provides good local control on manipulation of the control vertices of the driver surface. The local coordinate system in Equation 2 at a sample point G, is defined as $\overrightarrow{e_1} = \overrightarrow{t_u}, \overrightarrow{e_2} = \overrightarrow{t_v}, \overrightarrow{e_3} = \overrightarrow{n}$, where $\overrightarrow{t_u}$ and $\overrightarrow{t_v}$ are the tangents at G along two parameter curves, and $\overrightarrow{n}$ is the surface normal at G.

4 Algorithm Analysis

The algorithm imposes no restriction on the topology of the deformer object or its position relative to the deformable surface S. There are, however, implicit assumptions that greatly influence the quality and control that the deformer object has over the resulting deformation.

For a control element to provide good local control of the deformation of a region of the surface S, its spatial position on the reference surface R should be closer to the surface S than other control elements of R.

Also for the deformation of the region of surface S to appear intuitive it should be proximal to the control element in absolute terms. Non-intuitive behavior may be observed for points whose projection onto the plane of the triangle does not lie within the triangle. A point P on the deformed surface can be visualised as being anchored to its projection point P_k^{proj} on the plane of the triangle k offset at a fixed distance, normal to the plane. The deformation of P thus has a clear visual correlation to the deformation of the triangle if and only if P_k^{proj} lies within this triangle.

These observations place an implicit assumption on the nature of R relative to S, in that every point P of S should be proximal to some triangle k of R and have its projection onto the plane of triangle k lie within it. This is in accordance with the motivation for our deformer to provide a lower resolution visual model of our original surface S.

4.1 Algorithm Properties

- The surface S does not deform upon being bound to a deformer object. Since the driver and the reference surfaces of the deformer object are spatially identical when bound, $P_{def} = P_k^{def} = P$ for all control elements k.

- The deformation of space defined by the algorithm is continuous and intuitive. The parameter $local$ provides good control over the localization of deformation effects.

- Rotations and translations applied to the entire driver surface are imparted precisely to the surface S, since the weight values used in equation 3 are normalized.

- Warping of space normal to the plane of the control elements is captured as a constant offset from the control element, since $\overrightarrow{e_3}$ is a normalized vector, normal to the plane of the triangle in both R and D.

4.2 Extending the Algorithm

We now look at shortcomings of the approach described thus far. The first deals with non-intuitive deformations resulting from a point P being anchored to its projection P_k^{proj} on the plane of a triangle k of the reference surface, where P_k^{proj} does not lie within the triangle. It is conceivable to clamp P_k^{proj} to the closest point on the triangle boundary P_k^{close} and calculate the deformed point as two offsets from the point corresponding to P_k^{close} on the driver surface. The first offset is $P_k^{proj} - P_k^{close}$ in the plane of the triangle and the second, $P - P_k^{proj}$ normal to the plane of the triangle. While this addresses the shortcoming, the change introduces a first order discontinuity of deformation as P_k^{proj} for points P transitions across the triangle boundary.

The second shortcoming deals with the fact that the algorithm does not capture the warping of space in a direction normal to the plane of its control elements. Uniform scaling of the driver surface D, for example will scale the object precisely in the plane of the control elements of D, but maintain a constant distance from the elements in a direction perpendicular to them.

Both of these shortcomings can be attributed to the ambiguities in the perception of the behavior of space around the deformer object on manipulation of the driver surface. There are infinitely many ways by which a user can deform space such that the discrete set of points of a driver surface are manipulated to the same position. In each case, however, the behavior of the spatial neighbourhood of the points of the driver surface is different.

This ambiguity can be reduced by defining a coordinate system by introducing three additional points for every given point on the deformer surface. These points form mutually independent axes with the point on the deformer surface as the origin. The three points are subsequently subjected to the same manipulation function as the corresponding point on the deformer surface. While this coordinate system represents the space of a local linear transformation accurately, non-linear deformations are once again only approximated. This gives us some insight into the nature of spatial deformations and solutions to them by providing the user with additional control.

Every triangle k in our extended model has three local coordinate systems instead of one, centered at each vertex of the triangle and constructed during the bind phase. We register the point P by computing the local position of the point within each of the three coordinate system as described in Section 3.2. We also generate a deformed point with respect to each of the three coordinate systems of triangle k. The three deformed points are then weight averaged to a single resultant P_k^{def}. The weights in this case are provided by the barycentric coordinates of P_k^{close} (the closest point to P from triangle k). The deformed result of the various control elements are combined as in Section 3 to determine the final position of a point. It is straightforward to see that the range of deformation behavior captured above encompasses that of the the algorithm in Section 3.

5 Skinning Workflow

It is a fairly common practice in the animation industry to model articulated figures using a number of surface patches. Joint regions such as the shoulder in Figure 3 are particularly problematic to skin. This is because the range and degrees of freedom of the joint cause large variations in the motion of points. It is also the case that often a number of patches converge in the region around a joint, making the problem of skinning the geometry while maintaining smoothness and continuity across the patches a formidable task.

Figure 3: Shirt skinned using surface-oriented FFDs

A single surface-oriented deformer can abstract this underlying patch complexity so the user has to deal with the more tractable problem of skinning a single lower resolution object that bears a close visual semblance to the actual geometry. We prescribe a simple workflow that largely automates the entire skinning process. The shirt in Figure 3 was skinned with such a deformer.

The basic skinning workflow involves the construction of a single surface-oriented deformer around a character. This deformer is essentially a low-resolution representation of the character (See Figure 1). More importantly the resolution is adaptive, to allow a greater resolution of control points in the region of character joints.

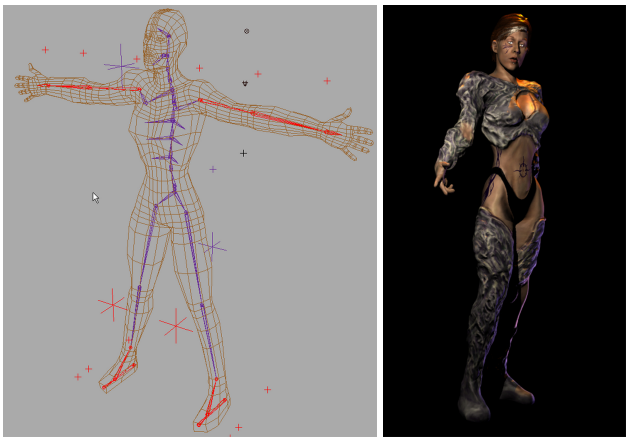

Figure 4: Skinning workflow: polyhedral deformer bound to skeleton(left), deformed surface (right)

Geometric representations such as parametric surface patches and implicit functions have well established tesselation algorithms. Polymesh decimation algorithms have also been well studied [8, 15]. For the common case where the underlying geometric skin comprises of a number of surface patches, the patches are tesselated independently and then stitched together to form a single deformer object. The stitched mesh represents all or a large section of the skin of an articulated character. The underlying geometry is bound to and controlled by the deformer object using the algorithm described in Section 3. The deformer object is bound to the underlying skeleton using any number of techniques [1, 14, 18]. We find that in practice it is often worthwhile to define the motion of individual points by keyposing them against various joint angle positions. The reduced point complexity of the deformer object makes this a reasonable task that allows complete customizability to be layered over the basic motion of the points of the deformer object as dictated by the basic binding technique used (See Figure 4). Finer local control may also be achieved at any point of time by subdividing triangles in a problematic region to generate a larger number of control elements. Non-triangular deformer polymeshes are internally triangulated, so as not to subject a user to unnecessary visual clutter.

A common problem with techniques that use Euclidean distance to determine correspondence between the deformed and deformer object, is that quite often regions of the deformer object will strongly influence regions of the deformable surface which happen to be spatially proximal but are quite distinct in the eyes of the user. A clear example of this can be seen in Figure 5 where the deformer region of the right thigh pulls on part of the left thigh geometry even though it should not affect it at all. Our implementation, therefore constructs, for each point P, a subset of contributing control elements C_P, from the set of control elements of the deformer object for a given point. As can be seen from Equation 1 the function f rapidly decays in value with distance such that the normalized influence weights are likely to be significantly larger than zero for only a small number of control elements. By default the control elements with a significant non-zero influence define C_P. The set, however, is under user control and may be edited if necessary. Thus by removing the control elements of the right thigh from the contributing control element sets for points of the left leg we can get the desired behavior.

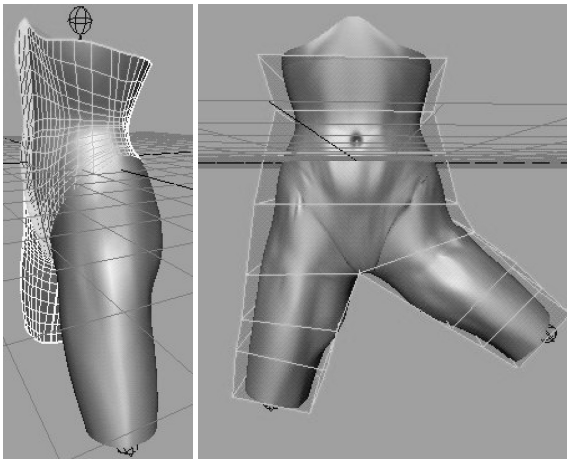

Figure 5: Deformation using a surface-oriented FFD

The algorithm described in this paper has been implemented as a general deformation technique within our modeling and animation system *Maya2.0*. The skinning of geometry has been automated as summarized below.

1. Polygonize surface patches or other geometric representation of objects to be skinned. Decimate and weld polygon objects as required to generate a few low resolution deformer objects.

2. Bind the deformer driver surface points to the skeleton. Points are rigidly attached to the Euclidean closest limb by default.

3. The control points of the deformer can be then keyposed against various skeletal and muscle attributes to generate custom skinning behavior.

4. The various parameters of the surface-oriented freeform deformation and the contributing element sets for various points may also be edited.

6 Results and Conclusion

We find in practice that surface-oriented free-form deformations address many requirements of geometric skinning. The deformer object itself provides a reasonably accurate low resolution representation of the skinned geometry, making it perfectly suitable as a stand-in for highly interactive animation tasks. The process is largely automated and may be all an animator needs for a quick setup. More importantly, however, the animator still has control at the finest level, through increasing degrees of detail. An analysis of the algorithms in Section 3 and 4, show them to be robust, efficient and of predictable deformation behavior and continuity. This is corroborated by the practical results shown in this paper and on a few of the characters of the animation short Bingo.

Our implementation combines multiple deformers on an object as described by Singh and Fiume [14]. Surface-oriented FFDs have also been used as a compelling modeling and animation tool with much of the appeal and control of a subdivision surface. Figure 6 shows the physical simulation of a polymesh deformer draping over a table top controlling a NURBS surface rendered with a checker texture.

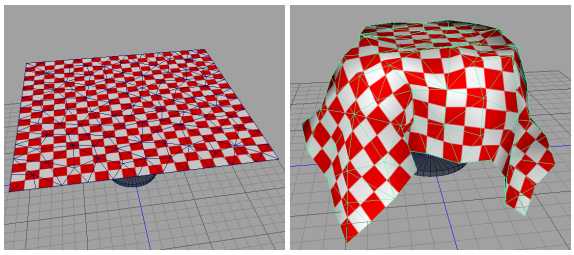

Figure 6: Physical simulation of a polyhedral mesh controlling a superposed NURBS tablecloth

While the extended algorithm in Section 4 gives us greater control and the property that all affine transformations to the deformer object are imparted perfectly to the underlying geometry, we find the algorithm of Section 3 to be simpler for an animator to understand and sufficient for the skinning application discussed in this paper.

Acknowledgements

We thank Barbara Balents and the *Maya* team for their help in the design and implementation of this technique and Paul Thuriot for providing us with invaluable case studies of character setup using surface-oriented FFDs.

References

[1] J. Chadwick, D. Haumann and R. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3):234–243, 1989.

[2] Y.K. Chang and A.P. Rockwood. A generalized de Casteljau approach to 3D free-form deformation. *Computer Graphics*, 28(4):257–260, 1994.

[3] S. Coquillart. Extended free-form deformations: A sculpting tool for 3D geometric modeling. *Computer Graphics*, 24(4):187–196, 1990.

[4] D. Chen and D. Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *Computer Graphics*, 26:89–98, 1992.

[5] T. DeRose, M. Kass and T. Truong. Subdivision surface for character animation. *Computer Graphics*, 85–94, 1998.

[6] J. Griessmair and W. Purgathofer. Deformation of solids with trivariate B-splines. *Eurographics 89*, 137–148.

[7] R. MacCracken and K. Joy. Free-form deformations with lattices of arbitrary topology. *Computer Graphics*, 181–189, 1996.

[8] A. Lee, W. Sweldens, P. Schroder, L. Cowsar and D. Dobkin. MAPS: Multiresolution adaptive parametrization of surfaces. *Computer Graphics*, 95–105, 1998.

[9] L. Moccozet and N. Magnenat Thalmann. Dirichlet free-form deformations and their application to hand simulation. *Computer Animation*, 93–102, 1997.

[10] N. Magnetat-Thalmann, D. Thalmann. Human body deformations using Joint Dependent Local Operators and Finite Element Theory. *Making Them Move*, Morgan Kaufmann, 243–262.

[11] F. Scheepers and R. Parent and W. Carlson and S. May Anatomy-Based Modeling of the Human Musculature. *Computer Graphics*, 163–172, 1997.

[12] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20:151–160, 1986.

[13] K. Singh, J. Ohya and R. Parent. Human figure synthesis and animation for virtual space teleconferencing. *IEEE VRAIS*, 118–126, 1995.

[14] K. Singh and E. Fiume Wires: A geometric deformation technique. *Computer Graphics*, 405–414, 1998.

[15] G. Taubin, A. Gueziec, W. Horn and F. Lazarus Progressive forest split compression. *Computer Graphics*, 123–133, 1998.

[16] Y. Lee, D. Terzopoulos and K. Waters. Realistic modeling for facial animation. *Computer Graphics*, 55–62, 1995.

[17] M. Walter and A. Fournier. Growing and animating polygonal models of animals. *Eurographics*, 151–158, 1997.

[18] J. Wilhelms and A. Van Gelder. Anatomically Based Modeling. *Computer Graphics*, 173–180, 1997.