Placement of Deformable Objects

SAGI SCHEIN
and
GERSHON ELBER

With the increasing complexity of photo-realistic scenes, the question of building and placing objects in three-dimensional scenes is becoming ever more difficult.

While the question of placement of rigid objects has captured the attention of researchers in the past, this work presents an intuitive and interactive scheme to properly place deformable objects with the aid of free-form deformation tools. The presented scheme can also be used to animate the locomotion of non-rigid objects, most noticeably animals, and adapt the motion to arbitrary terrain.

The automatic construction of our free-form deformation tool is completely hidden from the end user, and hence, circumvents the difficulties typically faced in manipulating these deformation functions. Further, a precise bound on the error that is introduced by applying free-form deformations to polygonal models is presented, along with an almost-optimal adaptive refinement algorithm to achieve a certain accuracy in the mapping.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry; Object Modeling —*Curve, surface, solid and object representation*; I.3.7 [**Computer Graphics**]: Three-dimensional Graphics; realism—*Animation*; I.3.6 [**Computer Graphics**]: Methodologies and Techniques—*Interaction Techniques*

General Terms: Design

Additional Key Words and Phrases: Free-form Deformations, animation, animal locomotion

1. INTRODUCTION

The seemingly simple task of placing three-dimensional objects in a scene is fundamental to the field of geometric design and computer graphics. During the process of scene design, a lot of time and effort is spent on placing models inside the scene, switching between views and making small adjustments to the relative position of objects. The problem becomes far more difficult when soft or deformable objects are involved in this placement process.

DEFINITION 1. *A* deformable object *is made of an elastic material. The shape of a deformable object is influenced by the shape of the surrounding objects coming in contact with it, by internal properties such as elasticity, and by external factors such as gravity.*

Autor's address: G.Elber, CGGC group, Computer Science department, Technion, Haifa, Israel, 32000.
S.Schein, CGGC group, Computer Science department, Technion, Haifa, Israel, 32000.

Fig. 1. A variation on the Reptiles picture of M. C. Escher.

The proper placement of a deformable object needs to take into consideration not only the relative position of other objects inside the scene, but also the deformations that newly placed non-rigid objects undergo when in contact with other objects in the scene. Deformable objects are common in everyday life and viewer familiarity makes the task of simulating the placement of a deformable object much more challenging.

Definition 2. *The* placement of a deformable object *is a process in which the deformable object assumes a shape that is influenced by interaction with other objects in the scene, taking into consideration internal properties and external factors.*

Exact placement of deformable objects requires the simulation of the physics of the interaction and is impractical, unless powerful tools such as finite element analysis are used. Nevertheless, one can go a long way with a far simpler approach, an approach that will be quite convincing even for the sensitive human eye that is used to seeing deformable objects, such as animals and plants, cloth and upholstery, rubber and Jell-O-like materials. Specifically, the placement of animals over arbitrary terrain typically requires the contact of legs and arms and, sometimes body and tail, with the environment underneath.

In this work, we will show that Free Form Deformation (FFD) [24] can be employed toward proper and convincing placement of non-rigid objects. FFD is a

recognized powerful tool that offers a global deformation scheme to manipulate three-dimensional objects. A common metaphor of FFD is that of a cube of Jell-O. An object is embedded inside the cube, and when the Jell-O cube is deformed (skewed, bent, twisted), the embedded object undergoes a similar deformation. Unfortunately, it is difficult to prescribe and construct FFDs, a severe hindering factor in making FFD of any general use. Nevertheless, in our placement application, not only is the FFD constructed automatically but the construction process is completely hidden from the end user.

FFD is typically defined as a mapping, $F : \mathbf{v} \subset \mathbb{R}^3 \rightarrow \mathbf{V} \subset \mathbb{R}^3$. A point $p \in \mathbf{v}$ is mapped into a new location $P = F(p) \in \mathbf{V}$. In the computer graphics literature, trivariate Bézier [24] or B-spline [19] functions are the common representations for FFDs,

$$F(u,v,w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} P_{i,j,k} B_i^o(u) B_j^o(v) B_k^o(w), \qquad (1)$$
$$(u,v,w) \in [0,1] \times [0,1] \times [0,1],$$

where $B_i^o(u)$ is the i^{th} Bézier or B-spline basis function of order o, and $P_{i,j,k} \in \mathbb{R}^3$ defines the control lattice of size $l+1$,$m+1$,$n+1$. These basis functions were chosen as the deformation function for their robustness and efficiency as well as simplicity of use. Hereafter and without loss of generality, the domain $\mathbf{v}$ of the function F is assumed to be the unit cube. In the ensuing discussion, we will use both $F(p)$ and $F(u,v,w)$ to denote the same mapping of a point in $\mathbb{R}^3$.

Consider a model $\mathbf{m} \subset \mathbf{v}$. F maps $\mathbf{m}$ to $\mathbf{M} = F(\mathbf{m}) \subset \mathbf{V}$. If $\mathbf{m}$ is represented using Bézier and/or B-spline surfaces, then for each surface $\mathbf{s} \in \mathbf{m}$, $F(\mathbf{s})$ could be computed exactly by evaluating this composition over (piecewise) polynomials or rationals. In many other cases, $\mathbf{m}$ is a polygonal model. Then, a polygon or a triangle $\mathbf{t} \in \mathbf{m}$ would be mapped to a free-form triangular patch $\mathbf{T} = F(\mathbf{t})$. It is typical for $\mathbf{T}$ to be approximated by mapping only the vertices of $\mathbf{t}$ through F. The edges of the deformed polygons remain rigid, resulting in errors being introduced into the deformed model. Finer tessellation of the polygons in $\mathbf{m}$ would result in a smaller deformation error. An adaptive refinement approach was presented in [10]. Polygons are refined where the error exceeds a certain threshold. Consider edge $\overline{p_i p_j}$. The error is estimated at the middle of the edge as

$$\epsilon_{i,j} = \left\| F\left(\frac{p_i + p_j}{2}\right) - \frac{F(p_i) + F(p_j)}{2} \right\|. \qquad (2)$$

A local subdivision algorithm is employed to split edges that introduce errors larger than an allowed tolerance. This approach is simple but has two major drawbacks. First, its error estimation scheme does not bound the maximal possible error, a maximum that can occur at any point along the edge. In addition and for similar reasons, the middle point is not always the best location at which to split the edge. The location that exerts the maximal error is probably a better candidate that would result in fewer polygons for the same level of accuracy. Herein, we will propose a scheme that does exactly that.

When a designer tries to place a rigid object in a scene, the result needs to be physically plausible. Objects must never float in mid air and chairs had better

be standing on their four legs. Similar restrictions hold when dealing with the placement of deformable objects. For example, when a snake is placed on top of a staircase, its belly must follow the stair tops as would a rug when placed on the same staircase. Different deformable objects present different amounts of resistance for bending or stiffness. There are several constraints that need to be observed when placing a deformable object over an arbitrary terrain. First, the object needs to rest upon the upper surfaces of the objects on which it is placed. Then, the bending and stretching of the object should be limited and controlled. Toward this end, a mass-springs system, commonly used for cloth simulation [26], is employed to handle the complex interaction between the deformation function and the objects underneath. This interaction or contact layer of the deformation function is treated as a rug or a sheet of cloth and its behavior is simulated using a mass-springs system.

FFD is an extremely flexible and powerful tool, and it stands out in its ability to continuously manipulate and warp models of arbitrary complexity. Nevertheless and in spite of its power, it is difficult to exploit. The difficulties stem, mainly, from the fact that designing a trivariate FFD that performs a specific deformation function is complex, counter-intuitive, and necessitates the manipulation of a volumetric control lattice in $\mathbb{R}^3$. It is non trivial to achieve a desirable deformation out of an FFD volume and it is a challenging task to predict the exact deformation that the FFD function would induce on an embedded model. The use of FFD as a manipulation tool for free-form surfaces supplants the difficult problem of dealing with control meshes of surfaces with the even more intricate problem of handling control lattices of trivariate functions.

Recognizing all these difficulties, this work offers a completely automatic construction scheme for the trivariate FFD functions. The designer is only required to sketch a 2D path that is projected on the scene and set a few parameters, such as the width and height of the mapping function, in order to fully prescribe the shape of the FFD volume. Moreover, the user does not even have to be aware of the existence of this FFD function unless he/she wishes to modify it.

The contributions of this work, in short, are

—Its proposal of a set of simple, interactive and intuitive tools to properly place deformable objects over arbitrary complex terrain, and animate their locomotion.

—Its demonstration of the usage of a mass-springs system to place a deformable object, while controlling its bending and stretching stiffness properties, in a physically plausible manner.

—Its development of precise bounds on the error introduced during the mapping of a polygonal model through an FFD function.

In Section 2, we refer to related work and research that developed some of the tools that are about to be used here. In Section 3, we define the tools and algorithms toward FFD based placement. In Section 4, we describe a few extensions to the basic algorithm. We define and present a more accurate adaptive refinement algorithm and also consider a few extensions that include giving more control over the construction stage of the trivariate and animation of deformable objects. A few more examples are presented in Section 5, and finally, we conclude in Section 6 and discuss possible future directions.

2. RELATED WORK

The problem of object placement stems mainly from the need to use 2D input and output devices when trying to manipulate objects in 3D. During the manipulation process depth perception is typically lost, and hence, even the simplest of tasks, such as placing a vase on a table, become painstaking tasks. The question of automatic placement of rigid objects inside a scene was investigated, for example, in [29]. [29] tries to minimize the amount of user intervention during the process of placing multiple rigid objects inside a scene. Their system uses a combination of physical and semantical constraints to enable the user to drop objects into the scene. The system then finds the most suitable placement, physically and semantically, for the objects. The proposed system only handled the placement of rigid objects.

Automatic placement of rigid objects was also tackled in the context of augmented reality (AR). [6] proposed a system that places virtual objects into a scene of real objects. The two main issues on which the system focused were occlusion of real objects by virtual objects and vice versa, and the placement of the virtual objects on top of real objects in the scene. The proposed system also dealt solely with rigid object placement.

Some similarity exists between the placement of deformable objects and the problem of adding fine detail to a smooth surface. There is work on cut and paste operations on free-form surfaces. For example, [4] concentrated on adding highly detailed spline surfaces to a base surface while retaining continuity constraint. Another recent example, [5] in the context of subdivision surfaces extends this notion to multi-resolution cutting and pasting operations.

The most common technique for deforming an object is Free-Form Deformation (FFD). The idea was first introduced in [24] who proposed the use of trivariate tensor product Bézier functions as the deformation functions. [19] introduced trivariate B-spline functions as the deformation function for FFD. Later work tried to elevate FFD into a design tool and remove some of its restrictions. [11] proposed the use of prismatic and cylindrical control lattices to define the deformation map. [11] also used the metaphor of sculpting with FFD by welding several lattices together. [22] used a volume subdivision scheme to generate a control lattice of arbitrary topology. Other results in this area were more interested in finding new applications for FFD. [12] used FFD to animate objects inside the deformation function. [13] proposed the usage of Deformation Displacement Maps (DDM) to generate three-dimensional textures and place them on top of curved surfaces. A good survey on the topic of deformable object modeling and FFD can be found in [18].

FFD is not the only available global deformation method. [28] defines a physically based global deformation technique. A global deformation function is used to deformed the vertices of a model. The vertices are then treated as masses and a simulation of the object's dynamics is applied to the deformed points. [2] uses the physical framework that was described in [28] to simulate non-penetrable flexible objects. The result might be more physically accurate then the approach describe herein at the expense of a higher conceptual complexity and running time. Thus, these methods seems more suitable for animation and high-quality image rendering then for interactive object design. Both [28] and [2] do not address deformable

object placement as a stand-alone problem although the proposed methods could be adapted for this problem.

FFDs are difficult to construct. Direct editing of points in the control lattice of the trivariate is seen as one simple yet counter-intuitive and tedious manipulation scheme. A different way that was proposed to circumvent the difficulties in constructing FFDs was to employ direct manipulation of FFD volumes [20]. The idea is to directly interact with the embedded model, and by solving a least squares problem, find an admissible configuration of the control lattice that would result in the desired deformation of the model. While feasible, this approach can be unstable at times and/or result in undesired side effects and global influence. In [25], the same problem is solved by means of constrained optimization that enables the derivation of an explicit solution when considering a single point constraint as well as methods to compose a complex manipulation from single point constraints.

Using FFD for the task of placement of deformable objects must be done with care so that the deformed object retains its original topology. Inept handling may result in an unnatural deformed object with self intersections. In [17], a scheme to test the injectivity of the FFD volume is proposed, by analyzing $det(J)$, the determinant of the Jacobian matrix of the FFD trivariate function F. In that paper, they also claim that an exact computation of $det(J)$ is too costly for interactive use.

The problem of simulating the complex interaction between rigid bodies and deformable surfaces, mainly in the area of cloth simulation, has been in the center of many recent results such as [8; 26; 3; 7; 9], to name just a few. These results sought to animate the creases and folds that are created when cloth interacts with a rigid object. For the most part, some form of a physically based simulation that solves the motion ODE and a collision detection system were used. The results vary in their details of how the motion ODE is solved and how collisions are handled.

3. THE PLACEMENT ALGORITHM

The placement algorithm we are about to present is heavily dependent on the constructed FFD function. In fact, the function F is the FFD that will map the geometry of a deformable object $\mathbf{m}$ residing on a planar terrain to a new object $\mathbf{M} = F(\mathbf{m})$ over arbitrary terrain. In Section 3.1, we present an automatic scheme to construct the FFD function from the terrain underneath it and represent the FFD as a trivariate B-spline function. In Section 3.2, a mass-springs simulation is employed, augmenting the quality of the contact layer or the surface that is in contact with the terrain.

3.1 The Deformation Function

Consider F, a trivariate B-spline function of order o, defined over a uniform knot sequence with open end conditions τ, in all three axes:

$$F(u,v,w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} P_{i,j,k} B^{o}_{i,\tau_{\mathbf{u}}}(u) B^{o}_{j,\tau_{\mathbf{v}}}(v) B^{o}_{k,\tau_{\mathbf{w}}}(w), \qquad (3)$$
$$(u,v,w) \in [0,1] \times [0,1] \times [0,1],$$

where $P_{i,j,k} \in \mathbb{R}^3$ are the lattice of control points of the trivariate B-spline function and $B^{o}_{i,\tau_{\mathbf{u}}}(u)$ are the i^{th} B-spline basis function. See Figure 2.

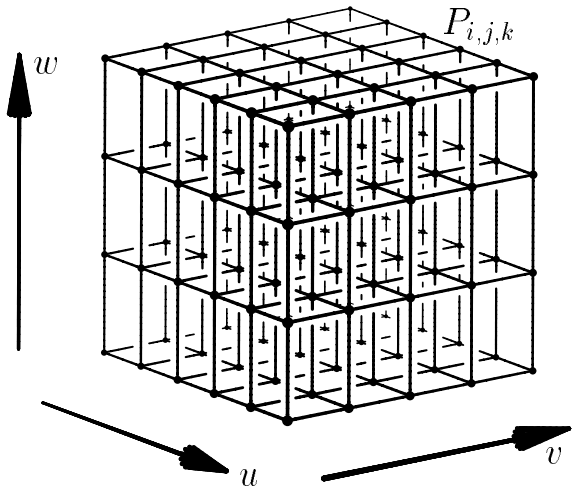

Fig. 2. FFD's control lattice.

One of the boundary surfaces of F plays a major role in the placement process:

DEFINITION 3. *The* contact surface *of F is the surface defined by $F(u, v, 0)$.*

The contact surface as the bottom layer of F serves as the contact layer with the scene underneath. The selection of open end condition for the FFD function in Equation (3) is crucial in the context of this placement application. It coerces the FFD function to interpolate the envelope of the control lattice, and specifically, the contact surface. Therefore, points $p \in \mathbf{m}$ of the form $p = (u, v, 0) \in \mathbf{v}$ are guaranteed to be mapped to the contact surface $F(u, v, 0)$ and hence, be in contact with the terrain underneath.

In order to fully prescribe the shape of F, one needs to specify its control lattice along the u, v and w directions. Let us describe this construction process from the user's perspective. The process starts by selecting a viewing direction for the scene and sketching a planar curve, $C(u)$, on the image plane of the rendered scene along which the deformable shape is to be placed. Two planar offset curves of distances $\pm d$ are then constructed from $C(u)$, in the image plane,

$$\begin{aligned} C_o^+(u) &= C(u) + d\vec{N}(u), \\ C_o^-(u) &= C(u) - d\vec{N}(u), \end{aligned} \tag{4}$$

where $\vec{N}(u)$ is the unit normal field of $C(u)$; see Figure 3. Construct a planar ruled surface, $R(u, v)$, between $C_o^+(u)$ and $C_o^-(u)$,

$$R(u, v) = vC_o^+(u) + (1 - v)C_o^-(u), \quad 0 \le v \le 1.$$

$R(u, v)$ is in the image plane. In order to place $R(u, v)$ on top of the objects in the scene, or on the underneath terrain, rays originating from points sampled in R are fired into the scene in the viewing direction, and the first hits, if any, are recorded. Toward this end, the parametric domain of R is point-sampled in a rectangular grid, at a density that is controllable by the user. The denser this grid, the finer the terrain details that can be sensed.

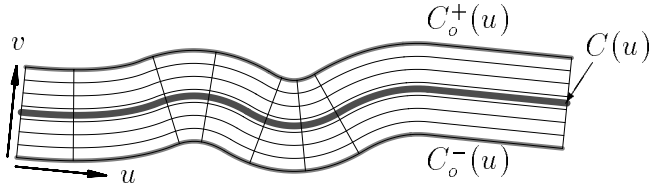

Fig. 3. Constructing two offset curves, $C_o^{\pm}(u)$, from a designer's sketch, $C(u)$, only to rule a surface $R(u,v)$, between these two offset curves.

The grid of projected points in the u and v directions is used to define the contact surface, $S(u,v) = F(u,v,0)$. A B-spline surface, $F(u,v,0)$, is re-fitted through this new grid with a parameterization that is arc-length in the v direction and chord-length in the u direction. The chord-length is estimated by the length of the middle column of control points, at $(u_i, 1/2)$. As a result, $S(u,v)$ has a uniform, constant speed in the v direction or $\left|\left|\frac{\partial S}{\partial v}\right|\right|$ is fixed for all (u,v). Further, $\left|\left|\frac{\partial S}{\partial u}\right|\right|$ is also approximately constant along $S(u,1/2)$, and varies away from $v = 1/2$ only if $C(u)$ presents curved regions.

To complete this automatic derivation of the deformation function F, we need to define the rest of the control points along the w direction, above the contact surface. We explored two alternatives. In the first, we constructed an extruded volume by extruding the contact surface along a prescribed unit direction $\mathcal{V}$. $\mathcal{V}$ is typically the viewing direction and h is the extrusion's amount:

$$F_1(u,v,w) \;=\; F(u,v,0) + wh\mathcal{V}, \qquad 0 \le w \le 1. \tag{5}$$

Figure 4 presents an extrusion volume constructed for the the contact surface of Figure 3. The resulting deformed volume F_1 may be later viewed from an arbitrary direction.

As an alternative, a ruled volume is built between the contact surface and its offset surface, by amount h,

$$F_2(u,v,w) = F(u,v,0) + wh\vec{n}(u,v), \tag{6}$$

where $\vec{n}(u,v)$ is the unit normal field of the contact surface $F(u,v,0)$. Due to the square root normalization in $\vec{n}(u,v)$ the resulting ruled volume is not rational, thus, we only approximate the ruled volume by offsetting the control mesh of the contact surface by $k\frac{h}{o_w}$ where o_w is the order of $B_k(w)$ and k is the k^{th} layer of control mesh in the w direction s.t. $P_{i,j,k} = k\frac{h}{o_w}\vec{n}(u,v) + P_{i,j,0}$.

These two approaches are exemplified in Figure 5. In (a), the extrusion scheme of F_1 is presented and compared to the offset approach in (b).

The approach creating $F_1(u,v,w)$ will never result in a self intersecting trivariate function, provided $S(u,v)$ is self intersection-free and $\mathcal{V}$ is a direction from which S is completely visible. Nevertheless, the result is skewed toward $\mathcal{V}$ and can be less natural than the second alternative. In contrast, $F_2(u,v,w)$ might generate self intersections in the volume where the contact surface has highly curved concave regions, which present a radius of (principal) curvature smaller than h. However, its intrinsic construction scheme has no biased direction $\mathcal{V}$, and therefore can be used to place objects on surfaces with an angular span of more than 180^o.

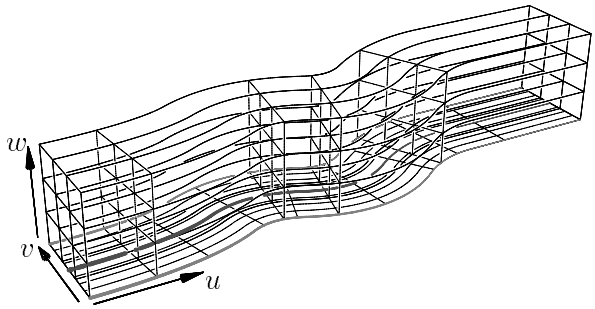

Fig. 4. Generated extrusion volume along a prescribed viewing direction for the contact surface presented in Figure 3.

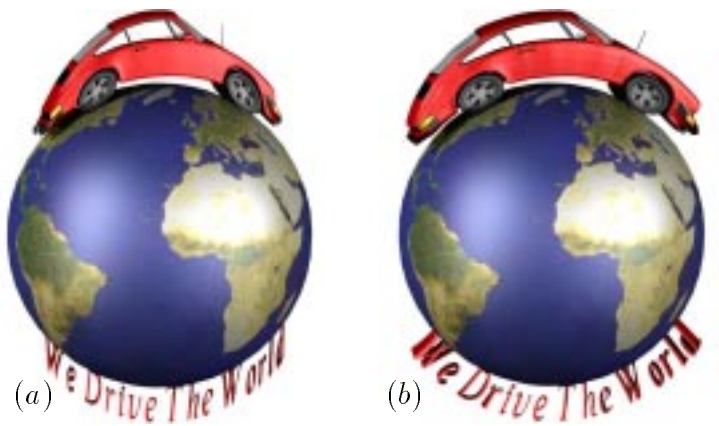

Fig. 5. An extrusion volume of the contact surface in (a), using the F_1 scheme is compared to the normal offset in (b), using the F_2 scheme in constructing the FFD.

3.2 The Mass-Springs System

Once the ray-shooting stage is complete, as described in Section 3.1, all the control points of the contact surface $S(u,v) = F(u,v,0)$ are in full contact with the scene underneath. The resulting contact surface emulates a very light fabric material with little bending and stretching resistance. One can employ a mass-springs (MS) simulation system, drawing on ideas presented in [26] or [9], to better control and mimic the behavior of more resistive material in the constructed deformation volume.

Using an MS simulation, the stretching and bending stiffness of the contact surface can be governed. Consider the control points of $S(u,v)$, $P_{i,j,0}, (i,j) \in [0,l] \times [0,m]$, where l and m are the maximal indices of the control points in the u and v directions of the control lattice, respectively.

Each control point, $P_{i,j,0}$, assumes a mass of $\mathcal{M}$, and is connected to four of its neighbors, $P_{i\pm1,j\pm1,0}$, with four springs, of equal spring coefficient, k. Unlike traditional MS simulations, $P_{i,j,0}$ here has a single, directional degree of freedom along which to move, $\vec{V}_{P_{i,j}}$, a directional constraint that is in either the $\mathcal{V}$ direction (for F_1) or the direction of $\vec{n}(u_i, v_j)$ (for F_2). $\vec{V}_{P_{i,j}}$ is assumed to be a unit vector.

The simulation is applied until the system reaches an equilibrium state. Collision detection is used to prevent the masses from penetrating the scene underneath.

For every mass, $\mathcal{M}$, positioned at $P_{i,j,0}$, Hooke's law is used to calculate the mass' acceleration at time $t + \Delta t$. Denote by $\vec{D}_{i,j}^{k,l}$ the Euclidean distance vector from $P_{i,j,0}$ to $P_{k,l,0}$ and let ρ be a velocity damping factor. Then, the acceleration $\vec{a}_{i,j}$ of mass $P_{i,j,0}$ equals

$$\vec{a}_{i,j}^{*} = \frac{k}{\mathcal{M}} \sum_{\substack{k\,=\,i\,\pm\,1 \\ l\,=\,j\,\pm\,1}} \vec{D}_{i,j}^{k,l} + \rho \vec{v}_{i,j}\,,$$

with $\vec{v}_{i,j}$ denoting the velocity of $P_{i,j,0}$. Point $P_{i,j,0}$ is restricted to move in the positive or negative direction of $\vec{V}_{P_{i,j}}$. Hence, only $\vec{a}_{i,j} =< \vec{a}_{i,j}^{*}, \vec{V}_{P_{i,j}} > \vec{V}_{P_{i,j}}$ is considered. From $\vec{a}_{i,j}$, the new positions and velocities are calculated using an explicit Euler integration scheme, following [9],

$$\begin{aligned} \vec{v}_{i,j}(t + \Delta t) &= \vec{v}_{i,j}(t) + \vec{a}_{i,j}(t)\Delta t, \\ \vec{p}_{i,j}(t + \Delta t) &= \vec{p}_{i,j}(t) + \vec{v}_{i,j}(t)\Delta t. \end{aligned}$$

The velocity damping term, $\rho \vec{v}_{i,j}(t)$, is needed to stabilize the simulation process and can be thought of as the external friction of the system. ρ is typically negative. We also need to define the termination conditions for the simulation. Toward this end, we identify masses that need further simulation steps. The candidates for further simulations are selected using the following method. For each mass, $\mathcal{M}$, at position $P_{i,j,0}$, we consider the four neighboring masses, and calculate the four cross products of vectors toward every consecutive clockwise neighboring pair. If the angles between these four vectors exceed a user defined threshold, we consider that mass as a candidate for further optimization. Once no masses are found to be candidates for further optimization, we declare convergence.

Assume point $P_{i,j,0}$ needs to undergo an MS simulation. We select a neighborhood of points around it, and consider only these points in the process of the springs simulation. The neighborhood size enables the designer to influence the final shape of the contact surface, with larger neighborhoods resulting in a less tight contact (see Figure 6).

When the contact surface assumes its final shape, the rest of the FFD volume is calculated as before, either as an extrusion volume along a prescribed direction or a ruled volume along the offset, as was described in Section 3.1.

4. SOME EXTENSIONS

In Section 3, the FFD was constructed automatically from a sketched curve and a few parameters such as sampling density and the deformation volume width and height. Yet, we also provide several additional degrees of freedom that are described in Section 4.1. The question of proper refinement of polygonal models embedded in the FFD is discussed in Section 4.2. In Sections 4.3, we explore the animation possibilities into which the proposed approach can be extended.

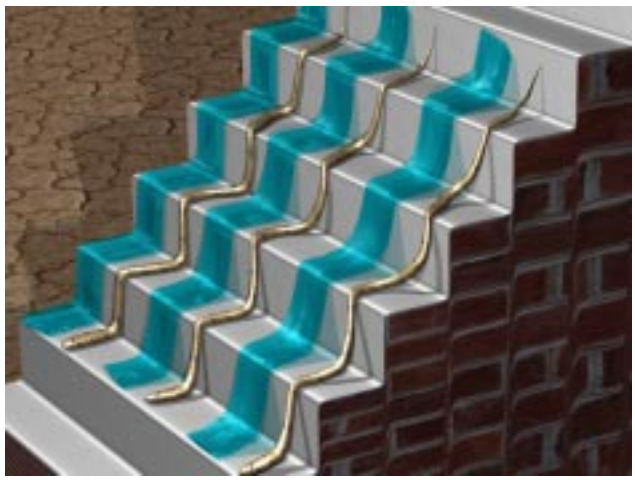

Fig. 6. MS optimization of FFD volumes. The left volume was generated without any springs optimization, the middle volume employed an optimization using a small neighborhood and the right one used a large neighborhood. The resulting positioned snakes are presented before the trivariate volumes.

4.1 Further Manipulating the Trivariate

So far, we have described a basic automatic construction scheme of the control lattice of the trivariate B-spline function. This basic approach could be extended in several ways. For example, the designer might wish to stretch the object in some parts of the deformation volume and squeeze it in others. In order to achieve this fine and local tuning ability, we generalize the construction scheme of the offset curves (see Equation (4)),

$$C_o^{\pm}(u) \;=\; C(u) \pm d(u)\vec{N}(u), \tag{7}$$

where $d(u)$ is a newly introduced, second scalar curve with the same parametric domain as $C(u)$. $d(u)$ serves as a scaling function, leading to a variable offset curve that enables us to locally control the dimensions of the constructed volume in the v direction.

Similarly, we enable the designer to manipulate the *height* of a subset of control points of $F(u, v, w)$. A scalar height function $h(u)$ or even $h(u, v)$ could be directly prescribed, replacing h in Equations (5) and (6). Alternatively, interactive intuitive editing tools are provided for minor yet of arbitrary resolution, updates of the lattice points of the FFD. The local changes of control point $P_{i,j,k}$ are constrained to the direction $\vec{V}_{P_{i,j}}$. The designer navigates a semi-transparent editing sphere of varying radius and press control points above the contact surface that are found inside the sphere. Three types of influence decay were tested: a Gaussian tool where the influence of the tool decreases exponentially from the center, a hat type tool, where the influence of the tool decreases linearly, and a box-shaped tool that applies the same pressure on all points in the radius of influence. The radius of the tool is a user defined parameter which enables multi-resolution control over the affected area. Figure 7 shows the effect of tools of different resolution when applied to the FFD volume. Figures 8 shows the effect of a pressing tool in a real scene. In

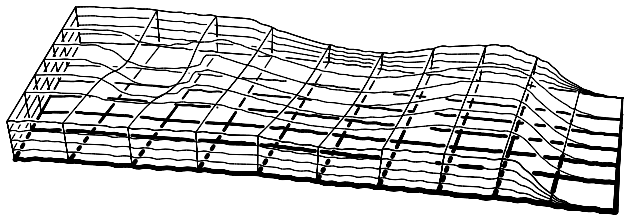

Fig. 7. The result of applying different pressing tools to the FFD volume, from small influence (left) to large (right).

(a)

(b)

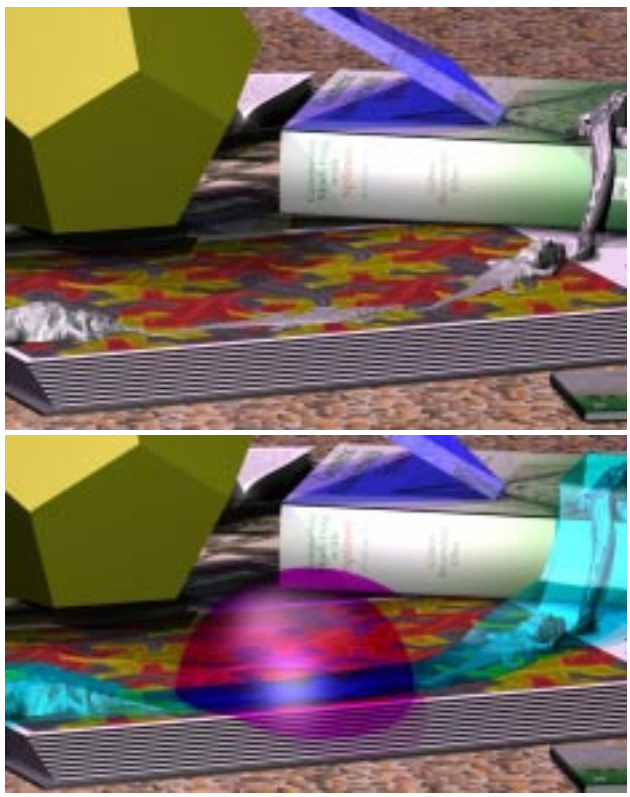

Fig. 8. (a) shows the result of the pressing spherical tool. In (b), the sphere of influence of the tool is shown in magenta whereas the trivariate itself in presented in cyan. Both are drawn transparently.

addition to these editing capabilities of the lattice above the contact surface, the control points of the contact surface could be independently pulled and/or pushed along $\vec{V}_{P_{i,j}}$, moving the entire column of control points above it the same amount. That is, the translation of $P_{i,j,0}$ coerces all control points $P_{i,j,k}$, $k > 0$ to follow along.

4.2 Refinement Algorithm

One undesired result of mapping the vertices of a polygonal model through a deformation function, instead of the exact composition evaluation, is the fact that

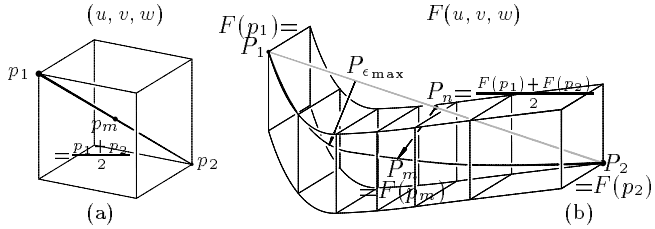

Fig. 9. A deformation of a polygonal edge between p_1 and p_2 in (a) is approximated by mapping these two points to the gray line shown in (b). The maximal error of this approximation, at $P_{\epsilon_{\max}}$, is clearly not at the middle of the domain. $P_{\epsilon_{\max}}$ is indeed a better candidate for further refinement.

neither the edges nor the faces of the model deform. A simple remedy that reduces these deformation artifacts subdivides the edges and introduces new, finer faces into the mesh. Consider two vertices, p_1 and p_2, of edge $\overline{p_1p_2}$ in some polygonal model. If the error in the mapping of $\overline{p_1p_2}$ (see Equation (2)) $\epsilon_{1,2}$, exceeds a certain distance threshold, the edge is subdivided at the middle point, $p_m = \frac{p_1+p_2}{2}$, and the polygons on both sides of these edges are divided as well. This algorithm is described in [10].

As already explained in Section 1, the problem with this simple algorithm is twofold. In general, the middle point is neither the location with the maximal error, nor is it the optimal location at which to subdivide. Consider, for example, a situation where most of the deformation is concentrated in one end of the edge (see Figure 9). Subdividing at P_m makes little sense since that part of the deformed edge is almost straight. A superior location for subdivision would be $P_{\epsilon_{\max}}$, where the error, $\epsilon_{1,2}$, assumes its maximal value. Subdividing at $P_{\epsilon_{\max}}$ offers a more rapid convergence and minimizes the number of subdivisions and the size of the refined model, while achieving the same accuracy.

Parameterize edge $\overline{p_1p_2} \subset \mathbf{v}$ as

$$\begin{aligned} e_p(t) &= tp_2 + (1-t)p_1, \quad 0 \le t \le 1 \\ &= (u_p(t), v_p(t), w_p(t)), \end{aligned}$$

for some linear functions $(u_p(t), v_p(t), w_p(t))$, and let

$$\begin{aligned} & E_p(t) \\ & = F(tp_2 + (1-t)p_1) \\ & = F(u_p(t), v_p(t), w_p(t)) \\ & = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} P_{i,j,k} B^o_{i,\tau_{\mathbf{u}}}(u_p(t)) B^o_{j,\tau_{\mathbf{v}}}(v_p(t)) B^o_{k,\tau_{\mathbf{w}}}(w_p(t)), \end{aligned}$$

be the deformed edge in $\mathbf{V}$ (see also Equation (3)).

The deformation error function along the mapped edge $E_p(t)$ equals

$$\epsilon(t) = dist(E_p(t), \overline{P_1P_2})$$

where $dist(\cdot)$ measures the distance from $E_p(t)$ to line $\overline{P_1P_2}$. The same parameter t that maximizes $\epsilon(t)$, at $P_{\epsilon_{max}}$ (see Figure 9), should also be selected as the subdivision location of edge $e_p(t)$. $E_p(t)$ is a degree $3(o-1)$ piecewise polynomial B-spline

curve where the distance square function $\epsilon^2(t)$ is a scalar piecewise polynomial B-spline curve of degree $6(o-1)$. Finding the extremum of $\epsilon^2(t)$ requires the solution of a degree $6(o-1)-1$ constraint, $\frac{\epsilon^2(t)}{dt}=0$. This computation is exact yet it is a difficult and time consuming task. Instead, one can examine the control polygon of the scalar polynomial curve of the squared error, $\epsilon^2(t)$, and select the nodal point of its maximal coefficient. The nodal value is a parameter that is considered a close parameter value for the free-form shape to the control point with which it is associated. The nodal points are also known as the Greville abssica [15]. See [15] for ways to compute these parameters.

Looking for the control point of $E_p(t)$ with the maximal distance from line $\overline{P_1P_2}$ and using its nodal value as the refinement location is significantly simpler and more efficient and would achieve a similar result. Again, $E_p(t)$ is a piecewise polynomial of degree $3(o-1)$ and hence could be represented as a degree $3(o-1)$ B-spline curve. Compute the distance of the control points of this B-spline curve to line $\overline{P_1P_2}$ and select the nodal value of the control point with the maximal distance. The rest of the refinement algorithm follows the same lines as the refinement algorithm of [10].

4.3 Animation

The idea of using FFD for non-linear animations first appeared in [12]. Combining the ideas of non-rigid object placement, presented here, with the ability to animate the deformed object further reveals the capabilities of the proposed method. First, a FFD volume is constructed along a sketched path following Section 3.1. Once the FFD volume is in place, the model is translated in the parametric domain of the FFD along its u axis, the axis that corresponds to the direction of the curve that the designer sketched. Recall the chord-length parameterization of $F(u,v,w)$ (Equation (3)) and assume that model $\mathbf{m}$ spans a fraction q of the u domain of $[0,1]$. Then, $\mathbf{m}$ is animated by placing it from $[0,q]$ and up-to $[1-q,1]$. Thus, the model is animated along a user defined route in the scene. Typically, the length of the embedded object, spanning $[0,q]$, would be much smaller than the length of the u parametric domain of the FFD. Further, periodic sequences prescribing the locomotion of objects or animals over planar domain could be used, allowing for a two level animation - first, of the locomotion of the object or animal itself, such as walking or sliding, that is combined with a second warping FFD function that adapts that locomotion to the local terrain. As a result, a locomotion sequence of an object, such as a moving animal, which was designed on a planar surface along a straight trajectory could be used to generate an animation of the same object on an arbitrary terrain and along an arbitrary path.

5. EXAMPLES

Figure 10 presents one snapshot from a short movie that portrays the locomotion of several ants in a row. The planar locomotion of the ants along a linear path was adapted to this complex terrain using an automatically created FFD from a single sketched curve. See figure 11 shows a sequence of frames from the same ant animation movie.

Nothing prevents us from using rigid objects in this placement process. While one could clearly construct a rigid, trilinear FFD that preserves the rigidity of the

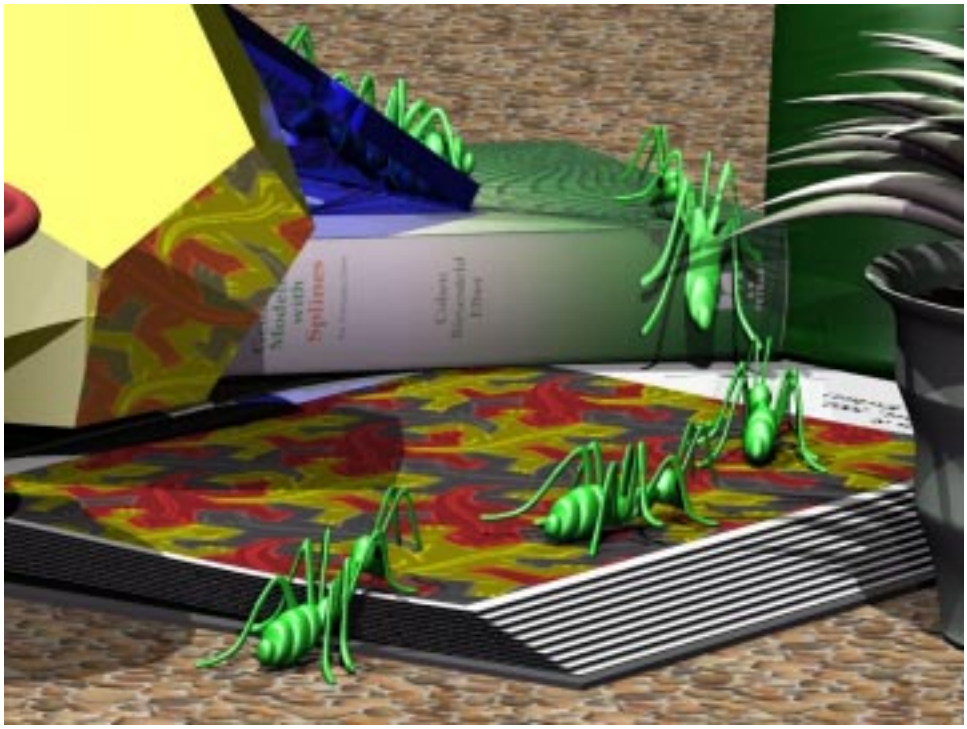

Fig. 10. A snapshot from a moving line of ants over this complex terrain. The locomotion of the ants over planar terrain is adapted to an animation over this complex terrain with the aid of FFD based placement.

object, aiding in the placement, animation, and locomotion of rigid objects, we could also allow the bending of rigid objects as if they were soft. Figure 12 contains snapshots from an animated Porsche warped to overcome the obstacle in this scene, again using an automatically created FFD. Figure 13 shows a series of frames from a moving tractor animation. The highly stiff and massive tractor is deformed to create a cartoon-like effect.

The use of MS simulation greatly increases the level of realism that can be expected in the placement process. In Figure 14, a snake is placed over a complex terrain. Only with the aid of MS simulation, we were able to bridge the deep gap between the different objects in the scene underneath. A careful examination of Figure 12 will also reveal a snake in the background, placed with the help of a MS simulation. An animation sequence in which the snake is climbing in the same scene is showed in 15. A spring simulation is used to elevate the body of the snake from the floor of the scene when it moves from the dodecahedron to the orange box in the bucket on the right hand side of the image.

Fonts and text are typically formed out of Bézier curves. Herein, we show how three-dimensional models of strings of characters could be placed on top of curved objects and in arbitrary directions. The width and height of the text can easily be controlled by the same methods that were described in Section 3. The resulting string of deformed characters can also be animated as proposed in Section 4.3.

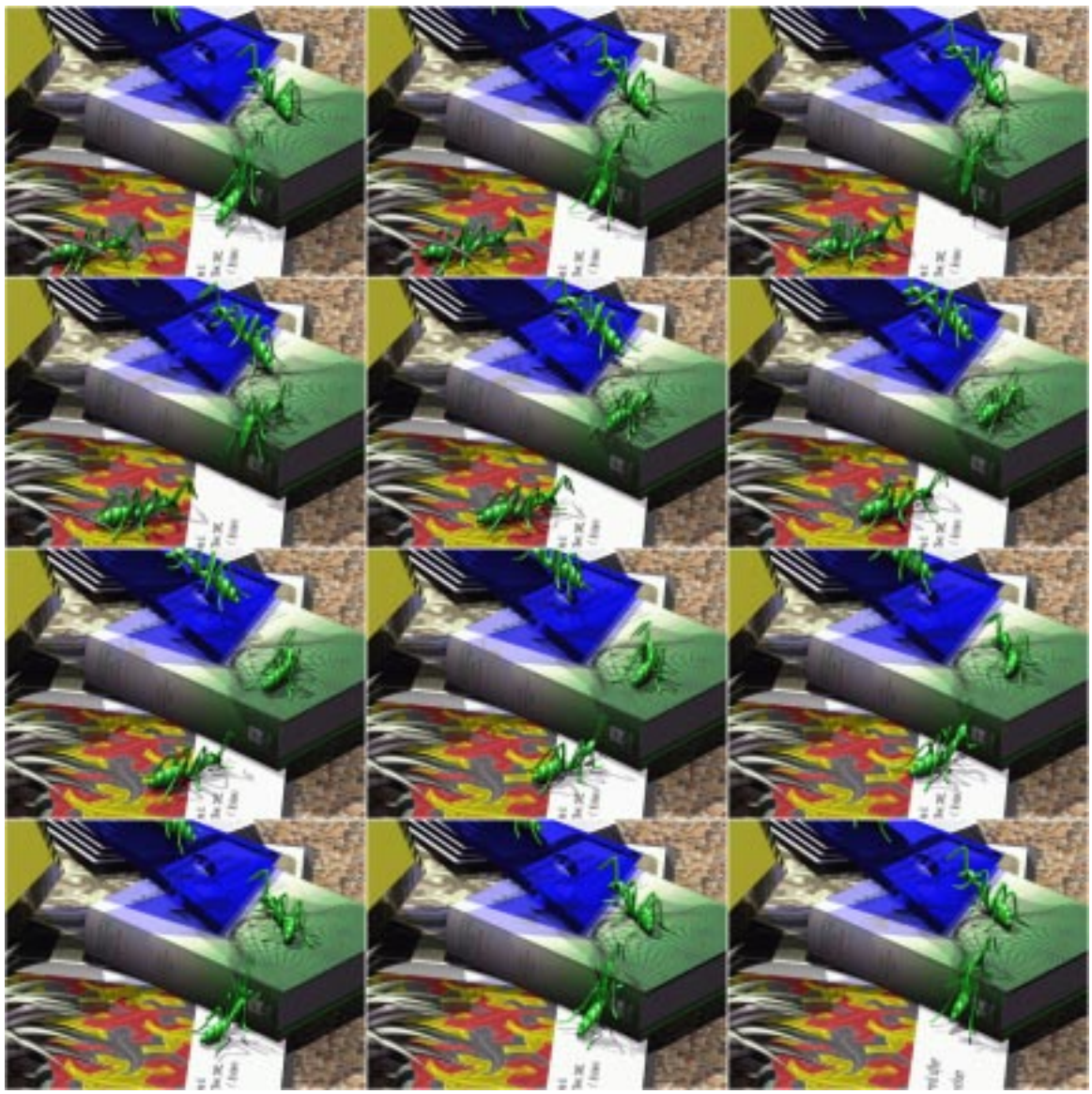

Fig. 11. A line of ants climbing in a reconstruction of Escher's reptiles scene. The image is composed of frames from an animation sequence.

Fig. 12. A sequence of deformed Porsche cars. The cars' movement is animated over a complex terrain with the help of the FFD placement tool. Note the placed snake in the background.

Several examples of this application are shown in Figure 16. This specific example also demonstrates the ability to employ variable offset curves (see Equation (7)).

6. CONCLUSIONS AND FUTURE WORK

We have shown that FFD can serve as an intuitive and powerful design tool toward the placement of deformable objects. Combined with mass springs simulation, a simple, intuitive and yet effective tool is offered to handle the complex and painstaking process of placing non-rigid objects over arbitrary terrains. The presented placement tool was interactively used to position static deformable objects in a scene, as well as animate soft objects in highly complex scenes. In addition, one can clearly employ the presented scheme to place rigid objects with arbitrary degrees of flexibility, bending and stretching, employing a trilinear FFD that is either skewed, conformal, or isometric to its parametric domain.

The MS simulation presents simple controls over the rigidity of FFD, and hence over the embedded non-rigid object. These controls directly reflect on the visual resistance to stretching, and more so, resistance to bending of the non-rigid object. Further, not only can the MS simulation ease the tension along sharp corners in the terrain underneath the object, but this same simulation can aid in bridging deep gaps and hanging non-rigid geometry.

In the presented construction FFD scheme, the v and w directions are always linear. Clearly a more complex, higher degree, construction schemes could be explored for the v and/or w directions. Such higher order representations could allow, for example, the representation of wiggles along u and/or w, or represent bending or tilting toward some prescribed directions. These new degrees of freedom could, for example, serve as a first order approximation of wind forces that are applied to deformed objects.

Fig. 13. A series of deformed tractor models taken from an animation sequence that was generated by the deformation and placement tool.

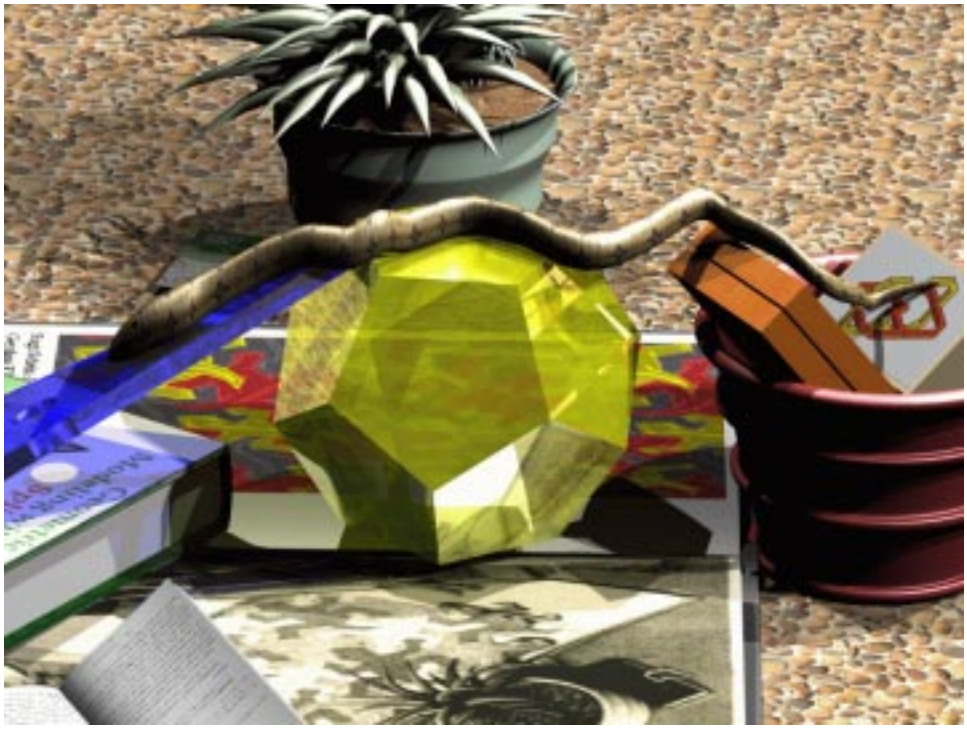

Fig. 14. A snake is placed in the scene. An MS simulation is used to bridge the gap between the red jar and the yellow dodecahedron.

The control over the rigidity of the placed object is currently only at the global level of the constructed FFD with or without the MS simulation. Nonetheless, it might be useful to control this rigidity at the local level of different parts of the placed object. For example, the head of a real snake is far less emendable to bending than the rest of its body. As the extreme example, a human skeleton should yield to bending only at the joints. Such an ability would allow the use of FFD to place skeletal objects over arbitrary terrain as well.

Since the scene underneath is discretely sampled, sharp edges could be missed, resulting in aliasing and rounding of corners with the possible consequence of the penetration of the placed object into these corners. Adaptive sampling of the sharp and/or discontinuous features during the reconstruction of the contact surface could alleviate this difficulty, a difficulty that is, in essence, shared by any cloth simulation scheme.

The current positioning tool can be viewed as a first order approximation of placement of deformable objects. It can currently handle only the deformation of a soft object when it is in contact with rigid objects. We would like to successfully handle cases where two deformable objects are in contact, and moreover when such contact leaves marks in the objects. Such an extension will enable us, for example, to animate the movement of a snake on a grassy surface or sand, and/or simulate the sinking of a person into a sofa. Such extensions are likely to necessitate the support of physical simulation of interactions between different soft materials.

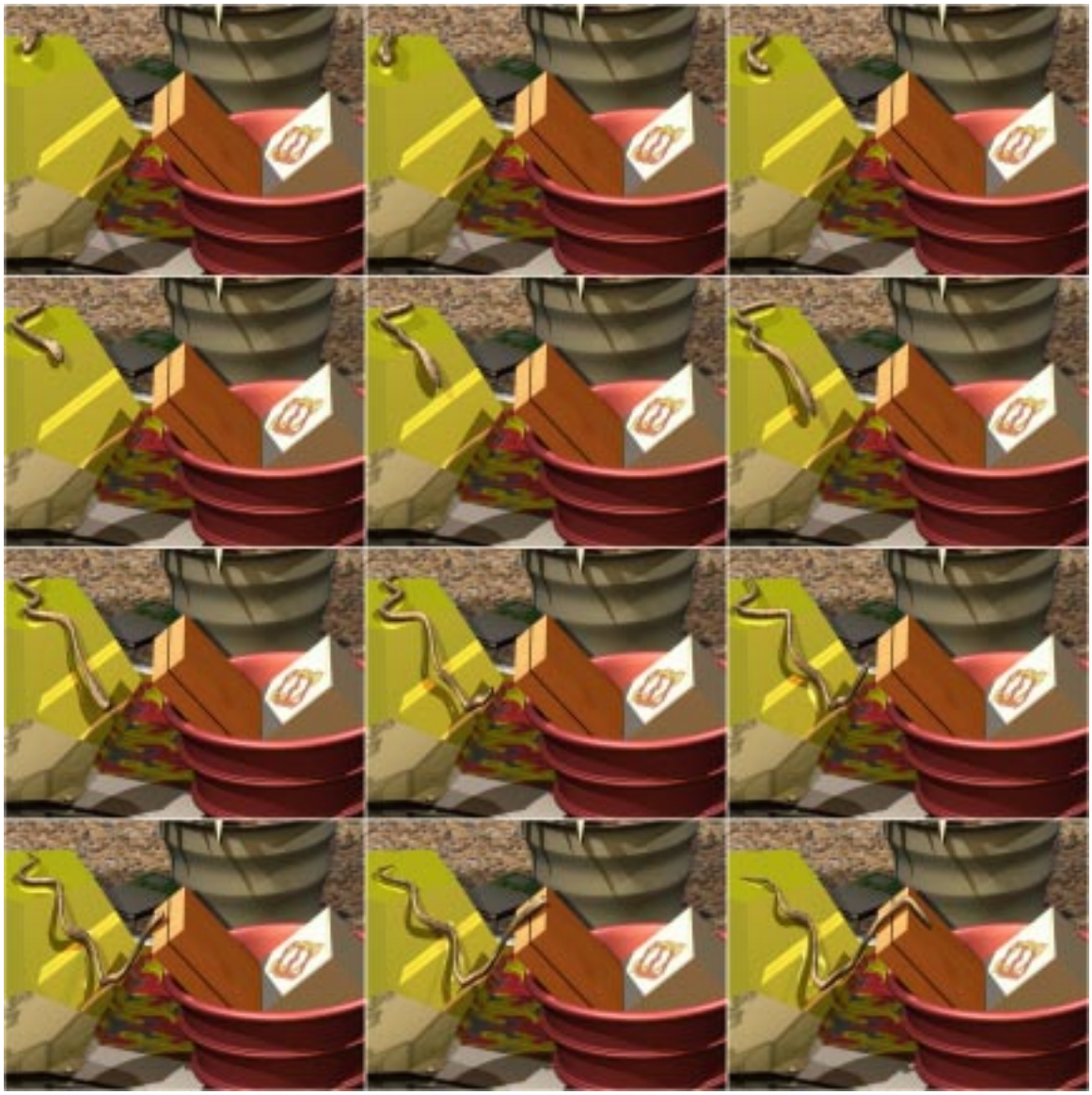

Fig. 15. A moving snake moving inside a scene that reconstructs Escher's Reptiles picture.

Potentially disturbing artifacts that might be created during the placement process are self intersections in the deformed object. A necessary condition for a locally self intersection-free FFD, and hence, a locally self intersection-free deformed object, is that the Jacobian of the deformation function, $J(F) \neq 0$, never vanishes, as was proposed in [17]. The computation of $J(F)$ can be performed symbolically using the tools that were developed in [14].

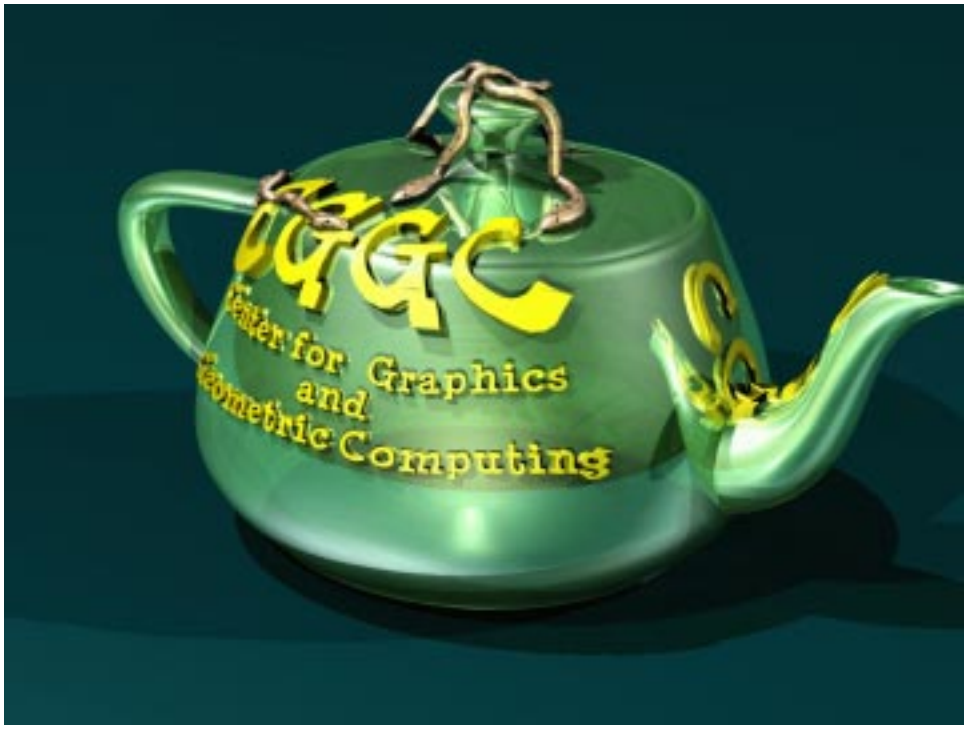

Fig. 16. A deformed "CGGC" group logo and three snakes placed over different surfaces of the Utah teapot. The snakes exemplify the ability of placement objects on top of other previously deformed objects (snakes and text).

Visualization and possibly correction of these singular locations would enable the designer to further fine tune the construction of the FFD function at these locations. Yet, the question of detection and elimination of global self intersections in trivariate functions remains completely open.

7. ACKNOWLEDGMENTS

The implementation of the FFD algorithm was conducted with the aid of the IRIT [21] modeling environment, as were all the scenes shown in this work.

The visualization task were accomplished with the aid of VTK [27], and the user interface for this application was built with the help of FLTK (Fast Light Toolkit) [16]. The Porsche and alligator models were downloaded from the 3dCafe web site [1].

All the presented scenes were rendered using the POVRAY ray tracer [23].

REFERENCES

3dCafe. 3dcafe, www.3dcafe.com, 2002.

David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 303–308. ACM Press, 1992.

David Baraff and Andrew Witkin. Large steps in cloth simulation. *Computer Graphics*, 32(Annual Conference Series):43–54, 1998.

Cristin Barghiel, Richard Bartels, and David Forsey. Pasting spline surfaces. In *Mathematical Methods For Curves and Surfaces*, pages 31–40. Vanderbilt University Press, 1994.

Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 312–321. ACM Press, 2002.

David E. Breen, Ross T. Whitaker, Eric Rose, and Mihran Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22, 1996.

Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM SIGGRAPH 2002*, volume 21, pages 594–603, 2002.

Michel Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. *Computer Graphics*, 26(2):99–104, July 1992.

Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 604–611. ACM Press, 2002.

Clint Chua and Ulrich Neumann. A layered approach to deformable modeling and animation. In *IEEE Computer Animation*, pages 184–191, November 2001.

Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, volume 24, pages 187–196. ACM Press, August 1990.

Sabine Coquillart and Pierre Jancne. Animated free-form deformation: an interactive animation technique. In Forest Baskett, editor, *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, volume 25, pages 23–26. ACM Press, July 1991.

Gershon Elber. Geometric deformation-displacement maps. In Sabine Coquillart, Heung-Yeung Shum, and Shi-Min Hu, editors, *The Tenth Pacific Conference on Computer Graphics and Application*, pages 156–165, October 2002.

Gershon Elber and Myung-Soo Kim. Geometric constraint solver using multivariate rational spline functions. In *The 6th Acm Symposium On Solid Modeling And Applications*, pages 1–10, 2001.

Gerald Farin. *Curves and Surfaces for CAGD*. Academic Press, 4 edition, 1996.

FLTK. Fast light toolkit, www.fltk.org, 2002.

James E. Gain and Neil A. Dodgson. Preventing self-intersection under free-form deformation. In *IEEE Transactions on Visualization and Computer Graphics*, volume 7(4), pages 289–298. IEEE Computer Society, 2001.

Sarah F. F. Gibson and Brian Mirtich. A survey of deformable modeling in computer graphics. Technical report, MERL - A Mitsubshi Electric Research Laboratory, November 1997. Internal document id TR-97-19.

Josef Griessmair and Werner Purgathofer. Deformation of solids with trivariate b-splines. In *Eurograpghic 89*, pages 137–148, 1989.

William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992.

Irit. *Irit Solid Modeling Environment. www.cs.technion.ac.il/~irit*, 2002.

Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 181–188. ACM Press, 1996.

POVRAY. Persistence of vision - www.povray.org, 2002.

T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. volume 20, pages 151–160, August 1986.

Hu Shi-Min, Zhang Hui, Tai Chiew-Lan, and Sun Jia-Guang. Direct manipulation of ffd: Efficient explicit solutions and decomposible multiple point constraints. *The Visual Computers*, 17(6):370–379, 2001.

Jeffrey A. Thingvold and Elaine Cohen. Physical modeling with B-spline surfaces for interactive design and animation. In Rich Riesenfeld and Carlo Sequin, editors, *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24, pages 129–137, March 1990.

VTK. The visualization toolkit, www.vtk.org, 2002.

Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 243–252. ACM Press, 1990.

Ken Xu, James stewart, and Eugene Fiume. Constraint based automatic placement for scene composition. Master's thesis, University of Toronto, 2001.