Discontinuous Free Form Deformations

Sagi Schein
Computer Science Department
Technion Haifa, Israel
32000
schein@cs.technion.ac.il

Gershon Elber
Computer Science Department
Technion Haifa Israel
32000
gershon@cs.technion.ac.il

Abstract

Contemporary deformation tools let designers modify the geometry of deformed models. This approach can be restrictive if the designer wants to incorporate holes or gaps into a model while deforming it into a different shape. This work presents a variant of FFD that would let the designer incorporate iso-parametric discontinuities into the deformation function. The input model is automatically split at these discontinuities, allowing the deformed model to reflect topological discontinuity changes.

We demonstrate the deformation algorithm using two different applications. The first application wraps a moving model around obstacles in a scene, splitting and then reforming it. The second application works locally, enabling the end-user to insert arbitrarily shaped cuts into the surface of the model.

1. Introduction

The deformation of geometric models has become a fundamental modeling tool in computer graphics. Following the terminology of [16], *Model Deformation* denotes the application of a non-linear transformation $F(u,v,w) : \mathbb{R}^3 \Rightarrow \mathbb{R}^3$ to a geometric model. Applying the deformation to an existing model results in a new, altered version.

There are many available deformation techniques. Some modify selected parts of a model by applying local deformations while others operate on the whole model at once and are denoted as global deformation methods. Among the global methods, one of the most widely-accepted techniques is Free Form Deformation (FFD) [19]. Here, trivariate functions, usually in the form of tensor product Bézier or B-spline functions, are used to map a box into a contorted box in $\mathbb{R}^3$. Then, an application of this FFD function to a geometric model results in a deformed version of the model. FFD is usually applied to polygonal models but it is not restricted to them. Its application to free-form models is also possible [19]. In this work however, we only consider triangular meshes.

Most deformation approaches alter the geometry of the deformed object without affecting its topology. This can become restrictive when a designer wishes to incorporate holes or tears into an existing model. A deformation tool that can potentially modify the topology of a model as part of the deformation specification will offer diverse applications as a modeling tool.

This work proposes a new FFD variant, coined *Discontinuous FFD* (DFFD). DFFD will account for the needed discontinuities and deform the model properly while automatically allowing it to split and re-form at the proper locations.

The DFFD operation is demonstrated using two different applications. The first application splits a deformable object and then wraps it around an obstacle(s) in the scene, splitting the deformable object and then re-forming it behind the obstacle(s). The second application demonstrates DFFD as a general framework for inserting cuts into the surfaces of geometric models, in real-time.

The rest of the work is organized as follows. In Section 2, we survey related work on deformation functions and geometric modeling of cuts. In Section 3, we present an overview of the proposed approach. Section 3.1 details the construction steps of a discontinuous deformation function. Section 3.2 defines two approaches to closing holes in a mapped model that result from the application of the DFFD operation. In Section 4, we examine the use of DFFDs as a generic cutting tool. Section 5 demonstrates some results that were achieved by using DFFDs and finally, in Section 6, we conclude and consider some future directions for the proposed method.

2. Related Work

Throughout the years, researchers have devised many competing deformation methodologies. Such methodologies can be classified according to the locality of their influence, the types of input that can be handled by the deformation method and whether the deformations are physically based or purely geometrical in nature. Local deformation methods

such as [12, 21, 13] are usually found to provide designers with finer control over the shape of the deformed object than global methods. Global deformation methods are more suitable for specifying large scale deformation over their source model.

Among the global deformation techniques, Free Form Deformations (FFD) [19] is probably the most widely accepted general deformation tool. FFD was first suggested by Sederberg and Parry as a mapping, $\mathbb{R}^3 \Rightarrow \mathbb{R}^3$, based on a trivariate tensor-product Bézier function. Later work by Griessmair and Purgathofer [7] suggested a B-spline representation for better localized control over the deformation process. This work was later generalized to NURBS-based FFDs by Lamousin and Waggenspack [11].

A significant shortcoming of FFD stems from the fact that it maps a box-shaped domain into a contorted-box in Euclidean space. When used to deform objects of complex geometry, the deformation function only remotely resembles the deformed object. As a result, it becomes even harder to control the final shape of the deformed object by manipulating the control points of the deformation function. To improve that shortcoming, the Extended FFD (EFFD) by Coquillart [4] combined multiple FFD volumes to construct a single deformation function that better resembles the shape of the deformed model. Later, MacCracken and Joy [14] suggested the use of arbitrary topology FFD based on subdivision volumes for free-form deformation. The method improves the localization of the deformation since the deformation mesh approximates the deformed model at the expense of higher computational complexity.

Most of the existing deformation techniques preserve the topology of the deformed object. This can be explained by the fact that common design paradigms are modeled after operations that do not change the topology of the object (bending, twisting, stretching, skewing and more). However, such deformations also prevent the introduction of holes or gaps into the models during the deformation process. Recently, Steyn and Gain [22] suggested a deformation scheme that tries to address this issue. [22] starts by extruding the three-dimensional (3D) object into a four-dimensional (4D) space followed by an application of a deformation operation on the 4D object. The extrusion is performed by making copies of the 3D object and stacking these copies as layers of a 4D object. Respective vertices of the model layers are connected, forming a 4D object comprising prismatic-shaped primitives that are later subdivided into tetrahedrons. Deformation control is achieved by using Direct Manipulation FFD [8] on the 4D object. The deformed 4D object is then intersected with a hyperplane in order to extract the deformed 3D object. The main shortcoming of this approach hinges on the fact that it requires the manipulation of objects in a 4D space. Such a manipulation may be counter-intuitive for designers who live and work in normal 3D space.

A completely different approach, which can be viewed as topological alteration of deformed models, was presented by Chen et al. [2]. [2] suggest the use of transfer functions as tools that prescribe spatial modifications on a volumetric dataset. Transfer functions are usually used to define a mapping between two different domains, such as heat and color. [2] extend this notion by suggesting transfer functions that map between locations in space, $\mathbb{R}^3 \Rightarrow \mathbb{R}^3$. Such transfer functions are denoted Spatial Transfer Functions (STF). STF are applied to volumetric models, resulting in non-structured volumetric models that may later be sampled on a structured grid of voxels and rendered using standard volume visualization techniques. By properly designing the STF, the user can introduce holes into an original solid volumetric model.

While the introduction of feature lines as part of the surface modeling process is of major importance in shape design, not many works are concentrated on the use of C^{-1} for design. Ellens and Cohen [5] suggested *teared surfaces* as a scheme for the incorporation of non-iso-parametric lines of C^{-1} discontinuity inside B-spline surfaces. The crux of their work was the localization of the influence of B-spline control points with the addition of a local overlapping layers of control points to accommodate a C^{-1} discontinuity. Points in the parametric domain are tagged as belonging to either side of the discontinuity curve. The evaluation of a teared surface is done by selecting the proper layer of control points.

Cutting through meshes is a common operation in the field of surgical simulation. Most systems employ a complex collision detection scheme and, at times, a haptic feedback mechanism. In the rest of this overview, we focus mostly on the geometric part of these efforts, namely the parts that deal with cutting through meshes.

Bielser et al. [1] described a procedure that cuts tetrahedral meshes. One problem with their approach, as well as with every other subdivision-based approach, is that the complexity of the mesh increases as the cutting operation progresses. To prevent this problem, Neinhuys et al. [17] proposed locally aligning the mesh to the curve that the virtual scalpel traces. This approach alleviates the increase in the size of the mesh but it also suffers from some drawbacks. First, the face alignment procedure can produce zero-volume tetrahedras that get in the way of the Finite Element Analysis (FEA) deformation engine. Second, if textures are mapped over the model, this process may cause undesirable stretches. Finally, if the scalpel is cutting through fine model details, those details tend to vanish since the mesh near them is altered. To combat the creation of degeneracies in the model, Neinhuys and Van der Stappen [18] combined mesh cutting with a Delauny-based triangulation approach. Local edge-flip operations are used on the faces that are affected by the cutting operation. The result of employing edge-flips is the elimination of triangles with large circumference. [18] consider surface meshes only.

3. The Discontinuous Deformation Algorithm

The DFFD algorithm operates in two phases. The first phase constructs a discontinuous deformation function. The selection of B-spline functions as the FFD tool lets us exploit

a rich set of modeling techniques that are readily available for surface modeling. For the task at hand, knot insertion [3] will be used to introduce a potential iso-parametric C^{-1} discontinuity into a tensor product B-spline FFD. The proposed algorithm automatically manipulates the control points of the deformation function to allow the deformation of the geometric model around the potential discontinuity and the creation of gaps or tears in the model. The two sides of the iso-parametric C^{-1} discontinuity retain their former spatial continuity in every place but the regions of influence of the modified control points. This easily ensures the continuity of the deformed model everywhere but for the openings.

A deformation operation is a mapping $\mathbb{R}^3 \Rightarrow \mathbb{R}^3$. In most cases the deformation is applied to polygonal models that discretely approximate a free-form object. The vertices of the model are deformed while the edges linearly connect these deformed vertices. Thus, edges and faces that cross the discontinuity are oblivious to the potential presence of that discontinuity and, consequently, do not reflect the needed topological change. To overcome this problem, the proposed algorithm incorporates an automatic model-split operation that splits the edges and faces that cross all discontinuities in the DFFD. In Section 3.1, we discuss the construction of the discontinuous deformation function and the automatic split operation.

Definition 1 *A* hole *is introduced into a valid two-manifold geometric model resulting in an increase of the genus of the model while retaining its two-manifold validity. An* opening *in a geometric model introduces a boundary-cut that transforms it from a valid two-manifold geometric model into a non-manifold geometric model.*

One undesirable result of our model-split operation is the introduction of openings into the model. Openings can be undesirable for applications that require closed, two-manifold, valid models. To seam such openings, two automatic approaches are presented. The first approach applies Boolean operations to the model before the deformation is applied in order to identify and fill the opening. Since Boolean operations are not always tractable, a second method that stitches openings is presented. This second method requires an additional manipulation of the deformation function and may yield a noticeable stretching effect in the deformed model. Section 3.2 elaborates on these two options for seaming an opening in a model.

3.1. Construction of the Discontinuous Deformation Function

As described above, a trivariate tensor product B-spline function serves here as the deformation function:

$$\begin{aligned} F(u,v,w) &= \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} P_{ijk}B_i^o(u)B_j^o(v)B_k^o(w), \qquad (1)\\ (u,v,w) &\in [U_{min},U_{max})\times[V_{min},V_{max})\times[W_{min},W_{max}), \end{aligned}$$

where P_{ijk} are the control points and $B_i^o(u)$ are the univariate basis functions of order o, in all three directions. In this application, we initially use uniform knot sequences in all directions, which complete the initial definition of a B-spline function. This function is continuous with a continuity that is governed by o, the order of the basis functions.

The insertion of o knots into a knot sequence of curve $C(t)$ of order o at t_0 would result in the interpolation of two different control points at $C(t_0)$, which, for clarity, are referred to as $C(t_0^-)$ and $C(t_0^+)$, accommodating a potential C^{-1} discontinuity. The same knot insertion algorithm is directly applicable to any of the three parametric directions of the trivariate tensor product B-spline function F in Equation (1). The result of inserting o knots at a parameter $u_{min} < u_0 < u_{max}$ introduces two identical and adjacent control mesh planes along a potential C^{-1} discontinuity. In the ensuing discussion, we assume that the knots are inserted in the u parametric direction of $F(u,v,w)$.

Definition 2 *A potential C^{-1} discontinuity due to the insertion of o knots into the parametric domain of a trivariate B-spline function F of order o at $u = u_0$ defines two potential* discontinuity planes. *These planes are denoted as $F^- = F(u_0^-,v,w)$ and $F^+ = F(u_0^+,v,w)$, respectively.*

The newly introduced discontinuity planes of control points, which are inserted into the deformation function, and the control points in their immediate vicinity, will provide the extra degree of flexibility that is needed in order to wrap the space around an obstacle.

In order to define the DFFD function, the algorithm starts by constructing a trivariate B-spline function that spans somewhat beyond all the obstacles in the scene. The obstacles are static objects around which we wish to split and wrap a moving deformable object. The algorithm uses the axis-aligned bounding box, $B = [x_{min},x_{max}],[y_{min},y_{max}],[z_{min},z_{max}]$, of these obstacles to deduce the dimensions of the scene. The parametric domain of the DFFD function is then prescribed to follow B. In order to fully determine the deformation function, F, the user supplies the order, o, of the basis functions and the number of control points in each direction, l, m and n. Uniformly-spaced knot sequences in each parametric direction complete the specification of F. Then, the control points of the deformation function, P_{ijk}, are placed uniformly inside B.

The process of building up the DFFD function continues by inserting potential C^{-1} discontinuities into the initial trivariate function, F. Let the center of mass of the i^{th} obstacle, O_i, in the u parametric direction be u_i^0 (see Figure 1). o knots are inserted at u_i^o, resulting in a potential C^{-1} discontinuity at that parameter. Control points that are associated with the $u = u_i^o$ parameter, or are in close proximity, are relocated in order to avoid obstacle O_i. In other words, we seek to modify $F(u,v,w)$ so that its range will have an empty intersection with the obstacles, $F(u,v,w) \cap O_i = \emptyset, \forall i$.

In order to achieve this object avoidance, O_i is intersected with a series of w-parallel planes $W_k : w = w_k, 0 \leq k < n$. For each W_k, the set of intersection points, $W_k \cap O_i$, is then used to construct two boundary curves, $C^+(v)$ and $C^-(v)$, which wrap O_i at plane W_k (see Figure 1). Each pair of control points, P_{ijk}, from F^- and F^+ along the C^{-1} discontinuity, are projected onto $C^-(v)$ and $C^+(v)$, respectively, resulting in a gap in the range of F that approximates the shape of O_i.

The projection procedure, though simple to implement and comprehend, also possesses some limitations. A control point, P_{ijk}, on the discontinuity plane, could be projected over adjacent control points, resulting in self-intersections in the deformation function. Self-intersections in the deformation function are undesirable and may lead to self-intersections in the deformed model (see [6]). Denote by P^u_{ijk} the component of the control point P_{ijk} and by $V^{u\pm}_{ijk}$ the computed translation vector for control point $P^{\pm}_{ijk} \in F^{\pm}$ along the u axis. The influence of $V^{u\pm}_{ijk}$ is propagated to neighboring vertices such that $P^{\pm}_{qjk} = P^{\pm}_{ijk} + \kappa(P_{qjk}, P_{ijk})V^{u\pm}_{ijk}$, where $\kappa(P_{qjk}, P_{ijk})$ is a finite kernel filter that decays from one to zero as P_{qjk} moves away from P_{ijk}. In this work, we used a truncated Gaussian function and a quadratic B-spline function for $\kappa(\cdot,\cdot)$. Proper specification of $\kappa(\cdot,\cdot)$ is needed to prevent the introduction of self-intersections into the deformation function, on the one hand, and excessively wide deformations, on the other. In this application the kernel parameters, such as the variance and cut-off level in the case of the Gaussian kernel and the shape of the B-spline filter, are controlled by the user.

As described in [6], the presence of zeros in the Jacobian of F, $J(F) = < \frac{\partial F}{\partial u} \times \frac{\partial F}{\partial y}, \frac{\partial F}{\partial w} >$, signals when the application of a kernel function, $\kappa(\cdot,\cdot)$, introduces self intersections into F. Let F be a trivariate tensor product function with control points P_{ijk} that are uniformly spread over a 3D box. Then,

Theorem 1 *Given F, a trivariate tensor product B-spline function with uniformly distributed control points along a 3D box P_{ijk}, such that $P^u_{i+1,j,k} > P^u_{i,j,k}$ and $\|J(F)\| > 0$. Assume we move control points along the u axis such that $P^u_{i+1,j,k} > P^u_{i,j,k}$ is preserved $\forall i, j, k$ after the application of $\kappa(\cdot,\cdot)$ to F. Then $\|J(F)\| > 0$ still holds.*

Proof: $\frac{\partial F}{\partial u}$ continues to point to the $+u$ direction and $\frac{\partial F}{\partial v}$ retains its $+v$ component. Hence, the projection of $\frac{\partial F}{\partial u} \times \frac{\partial F}{\partial v}$ on the w axis never vanishes. Since all control points move orthogonally to the w direction, $\frac{\partial F}{\partial w}$ retains its $+w$ coefficient and the inner product $< \frac{\partial F}{\partial u} \times \frac{\partial F}{\partial v}, \frac{\partial F}{\partial w} >$ remains positive. □

Therefore, it is sufficient to test for the occurrence of foldovers of control points in the u direction to identify the introduction of zeros into the Jacobian of F.

This projection procedure limits the types of obstacles that can be properly embedded. Let P_{ijk} be a control point along the inserted C^{-1} discontinuity. Then, the projection operation of that point on the curve $C^{\pm}(v)$ should be unique, which allows only obstacles that are monotonic relative to the v direction. In addition, one should recall that $C^{\pm}(v)$ only approximates the shape of the obstacle. For objects with complex geometry such an approximation might be too crude, revealing gaps between the obstacle and the deformed object, or causing the deformed object to penetrate the obstacle. This problem may be alleviated by inserting more degrees of freedom into the deformation function.

Thus far, we have described the insertion of a single discontinuity into the DFFD function. Conceptually, the insertion of multiple discontinuities is similar; see Figures 10 and 11. However, since each control point in the control volume has a finite region of influence, discontinuities that are too closely packed in the U direction will interfere with each other. To eliminate such interference, there should be enough degrees of freedom between each pair of obstacles, O_i and O_j. The minimal number of required control points in the direction of the manipulation depends on the order of the deformation function, o, and on the support of the kernel function, $\kappa(\cdot,\cdot)$.

3.2. Splitting and Stitching the Model

The constructed deformation function, F, as presented in Section 3.1, when used to naively deform a polygonal model, M, does not achieve its goal. While the vertices of M are properly mapped, the edges and faces of M are not evaluated through the deformation function. Edges that connect vertices on opposite sides of the discontinuity would simply cross it; faces that share these edges would present the same problem. To prevent the phenomena from occurring, M must be split at each discontinuity, $u = u_i, \forall i$, before the de-

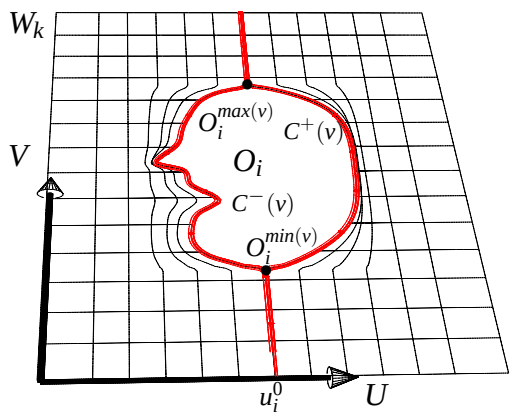

Figure 1. The cross-section of obstacle O_i in plane W_k, in the shape of a silhouette of a human face, guides the construction of the DFFD function. The two boundary curves, $C^+(u)$ and $C^-(u)$, on the cutting plane W_k are shown in red.

formation takes place. The splitting operation is performed in the parametric domain of F, which is also the object space of M. Since the object resides in the parametric domain of F, the split operation is done relative to a flat iso-parametric surface. Denote by $E_{j,k} = \overline{V_j, V_k}$ the edge connecting vertices V_j and V_k, and let v_j^u and v_k^u be the u coordinates of V_j and V_k, respectively. Without loss of generality, assume $v_j^u < v_k^u$ and recall that u_i is the splitting parameter. Then, if $v_j^u < u_i < v_k^u$, $E_{j,k}$ is a crossing edge that must be split. The result of the polygon splitting phase is a modified model, $\bar{M}$, which contains no edges that cross the discontinuity As a result of the potential C^{-1} discontinuity in F, as described in Section 3.1, points with $u \geq u_i$ would be mapped to one side of the discontinuity, while points with $u < u_i$ would be mapped to the opposite side. Because the above edge-splitting procedure introduces new points with $u = u_i$, the algorithm moves these points to the proper side by adding (subtracting) a small amount, setting $u_i^{\pm} = u_i \pm \varepsilon$.

Mapping $\bar{M}$ through F will result in openings in the mapped model, $F(\bar{M})$, along the discontinuities of F (see Figure 2(a)). In other words, as a result of the DFFD operation, a valid two-manifold model, M, will become a non-manifold one, with openings with boundaries. This behavior may not be desirable in many cases and ways to close these openings must be sought. We propose two approaches to close these openings. The first one employs Boolean operations [15, 10]. Let $M_F = F(M)$ be the mapped model and let $\mathcal{P}^+$ be an infinite cutting plane defined by the split parameter u_i and capturing the half space $u \geq u_i$. Then, $M^+ = F(M * \mathcal{P}^+)$ and $M^- = F(M - \mathcal{P}^+)$ define the two split parts of M_F, where $*$ and $-$ represent the intersection and subtraction Boolean operations, respectively. If M is a valid two-manifold, these operations are well defined. M^+ and M^- are two closed objects that are fully contained in domains $[u_{min}, u_i)$ and $[u_i, u_{max})$, respectively. Hence, $F(M^+)$ and $F(M^-)$ are two closed objects at the opposite sides of the discontinuity, with added caps at the intersection. See Figure 2(b) for an example of an application of this method to openings in a model.

A potential difficulty in the general use of Boolean operations is that many polygonal models are not proper two-manifolds, for which Boolean operations are not applicable. Moreover, the caps that Boolean operations create for seaming the openings are only C^0-continuous. For a C^1 smooth object, M, such low continuity might be insufficient. Thus, a second option that offers better continuity, while assuming nothing about the mapped object M but demands an additional manipulation of the trivariate deformation function, is suggested.

In order to close the openings that are formed in a model as a result of the DFFD operation, the deformation function can also be further manipulated. Modify F such that $F(u_i, v, w) = F(u_i, v, w_s), \forall w$, where w_s serves as the w-stitching parametric level. As a result, all control points of F interpolating the discontinuity plane at $u = u_i$ would be stitched together at a single parametric level $w = w_s$. w_s could be derived in several manners, and currently, the algorithm selects w_s to be the average w-value of all the vertices, V_i, with $v_i^u = u_i$, in $\bar{M}$. To complete the stitching operation, the control points of F, $P_{ijk} = (u_i, v_j, w_k)$ on the two discontinuity planes are moved to $P_{ijk} = (u_i, v_j, w_s), \forall j, k$. Figure 3 presents the result of stitching shut a rounded hole in a DFFD trivariate function. A properly stitched DFFD function, F, has the property that every application of F to $\bar{M}$ will preserve the two-manifold property in $F(\bar{M})$, if $\bar{M}$ is a two-manifold geometric object. No new openings are introduced into $F(\bar{M})$ and the openings along the discontinuity $u = u_i$ are stitched together into a closed seam.

$C^{\pm}(v)$ are C^0 continuous in two locations, at the minimal and maximal v values of O_i, $O_i^{min(v)}$ and $O_i^{max(v)}$ (see Figure 1). Projecting control points P_{ijk} onto $C^{\pm}(v)$ is insufficient to capture this low continuity in $C^{\pm}_{(v)}$. Hence, we insert $o-1$ knots in the v parametric direction at parameters v^i_{max} and v^i_{min} corresponding to the two C^0 locations $O_i^{min(v)}$ and $O_i^{max(v)}$, respectively. Figure 4 shows the difference with and without an interpolation constraint in the v direction of the DFFD function at $O_i^{min(v)}$ and $O_i^{max(v)}$.

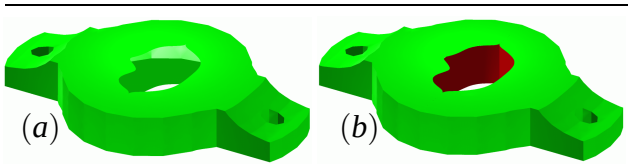

Figure 2. A non-trivial shape, the silhouette of a human face, (see also Figure 1) is molded into the center of a simple mechanical part using a DFFD operation. In (a), the edges of the opening are left open, while in (b), Boolean operations are used to cap the introduced opening.

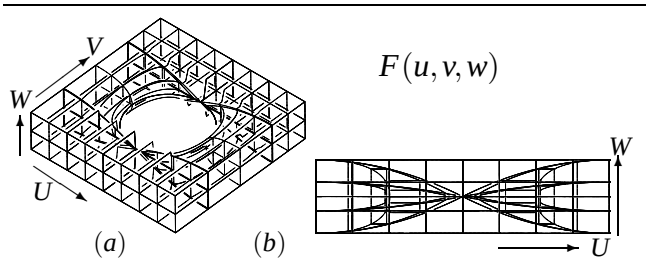

Figure 3. In (a), a modified trivariate function, F, which has a circular hole in it, is stitched together, closing the opening. The stitching operation preserves a closed mapped object, $F(\bar{M})$. In (b), a side-view of F is shown.

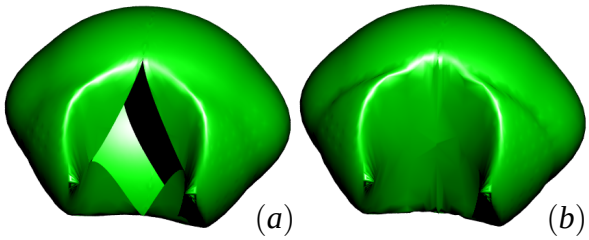

Figure 4. A model of a sphere is squashed in the vertical direction and wrapped around a cylindrical obstacle in front. In (a) no interpolation constraint is imposed on the control points in the v direction, which results in an opening in the middle of the model. (b) shows the effect of inserting $o-1$ knots in the v direction at $O_i^{max(v)}$, thereby closing the gap.

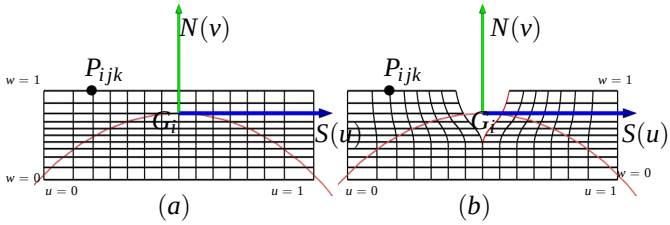

Figure 5. A cross-section of a local DFFD function placed over a cylindrical surface. The direction of the scalpel $G(v)$ is oriented into the page. (a) shows the DFFD function prior to the insertion of discontinuities, while (b) shows the same function after the introduction of the discontinuity cut; see also Figure 6.

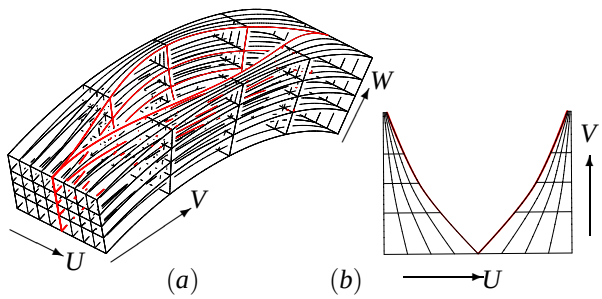

Figure 6. In (a), a general view of a local DFFD function is shown. The two red iso-parametric surfaces depict the cut. (b) presents the iso-surface $F(u,0.5,w)$.

4. Local DFFD

Section 3 described the construction and manipulation of the DFFD algorithm from the point of view of a global deformation algorithm. Alternatively, the DFFD algorithm can also be treated as a local mesh-splitting algorithm that cuts the surface of polygonal meshes while properly handling the shape and deepness of the cut and preserving its two-manifold property. The algorithm models the shape and deepness of the cut produced by the shape of the cutting tool. Moreover, by properly designing the the DFFD function, the algorithm can simulate the time-response of the deformable object in the vicinity of the cut.

The local DFFD cutting operation comprises three steps. First, the end-user defines the region that should be influenced by the cutting operation and constructs a DFFD function. Then, the algorithm identifies the geometry that is embedded in the spatial support of the DFFD function. Assuming a polygonal geometry, we need to locally parameterize the vertices of these polygons relative to the range of the DFFD function. Finally, the DFFD should be evaluated by computing the deformation of the target surface polygons, splitting polygons that cross the discontinuities of the DFFD function and adding the polygons that define the deepness of the cut.

To define the DFFD function, we locate the *generator curve* on the surfaces of the geometry to be cut. A series of mouse-recorded hit points on the surface is used to construct a B-spline curve that approximates the generator curve, $G(v)$, in a least-squares sense. From $G(v)$ we also derive the *tangent curve*, $T(v) = \frac{dG(v)}{dv}$. Moreover, from the surface normals at the hit points we construct a *normal curve*, $N(v)$ (see Figure 5). Denote by G_i a sampled position along $G(v)$, and by T_i and N_i the normalized unit vectors sampled along $T(v)$ and $N(v)$, respectively; let $S_i = T_i \times N_i$. Then, the control points P_{ijk} of the DFFD trivariate function F are defined as

$$P_{ijk} = G_j + \left(\left\lfloor \frac{l}{2} \right\rfloor - i\right)\alpha S_i + \left(\left\lfloor \frac{n}{2} \right\rfloor - k\right)\beta N_i, \quad (2)$$
$$0 \le i < l\ ,\ 0 \le j < m\ ,\ 0 \le k < n,$$

where l, m and n follow the notation of Equation (1), and α and β are two user-defined offset-amount scaling factors. Equation (2) positions the trivariate deformation function such that the generator curve, $G(v)$, is approximately equidistant from the outer envelope of the DFFD function, F, at $F(\frac{1}{2}, v, \frac{1}{2})$, assuming $[0,1]$ domain in u and w. In a similar way to the construction of F in Section 3.1, o knots are inserted in the middle of the u parametric axis of F, resulting in a potential C^{-1} discontinuity. Since the DFFD construction scheme generates curved-shaped deformation functions, a different algorithm is used to model the shape of the discontinuity. To form the cut, the control points that reside near and on the discontinuity, $P^{\pm}_{ijk}$ for $i = \lfloor \frac{l}{2} \rfloor$, are moved along the u parametric axis. The amount of movement is controlled by the kernel function, $\kappa(P^{\pm}_{ijk}, P^{\pm}_{\lfloor \frac{l}{2} \rfloor jk})$; see, for example, Figure 6.

The next step of the cutting algorithm is the detection of polygons in the cut geometry that are contained in the range of the deformation function. Here, we select all the polygons that were traversed by the generator curve as well as polygons with close proximity whose vertices are contained in the spatial domain of F. Vertices that are contained in the domain of the deformation function are selected and parameterized by their local coordinates inside the volume of F. Let V_i be a vertex on the discontinuity. We parameterize V_i such that $u = \frac{1}{2}$, $v = \arg\min_v \|G(v) - V_i\|$, and $w = \frac{1}{2}$. Edges and faces that cross the potential discontinuity are split and the deformation function is evaluated (as in Section 3.1).

As was described in Section 3.2, we should maintain the two-manifold validity of the model in the presence of cutting operations. Here, we can use the assumption that our cutting operation is local, which means that the cutting tool always cuts a single patch of the input geometry. This assumption lets us construct a new geometry for the deepness of the cut, in the shape of the cutting tool. An evaluation of the original and new geometries through the DFFD function completes the construction of the deepness of the cut, and also guarantees the validity of the surface. The introduction of new geometry to close the opening can be further viewed as another option to seaming the opening that is formed during the DFFD operation. We presented this method in another paper [20] that focused on the applications of DFFD as a cutting framework for surgical simulations.

5. Results

We present some results of applying DFFD operations to various models. Figure 7(a) shows the application of the embedding algorithm to the Stanford bunny model with a cylindrical shaped tool entering from above. The opening in the model that results is clearly visible. In Figure 7(b), the stitching algorithm was applied to the DFFD function and then the deformation was applied to the model, resulting in a valid two-manifold object.

Figure 8 demonstrates the use of local DFFD operations for real-time cutting operations. The shape of the DFFD is designed to produce an incision in a model of a face. Each incision on the face corresponds to an application of a local DFFD operation. The deepness of the cut is modeled and textured to replicate the influence of a cut to the model.

Cases exist where the stretching introduced by the stitching method are significant. If the mapped model, M, is a two-manifold, Boolean operations can be used to generate caps over the holes that are introduced into the model by the DFFD operation. Figure 9 shows several snapshots of an animation sequence of a moving bulldozer. A redwood tree splits open to admit the moving tractor; the split is shown as a gradually widening gap. Boolean operations were used to define the caps in the interior of the tree, which are textured with a wood-like texture. In Figure 10, an animation sequence of a walking robot is shown. Multiple discontinuities

(a) (b)

Figure 7. The Stanford bunny model with a cylinder-tool slicing through from above. In (a), the model was left open so that its inside is exposed. In (b), the DFFD's stitching method is used to close the opening.

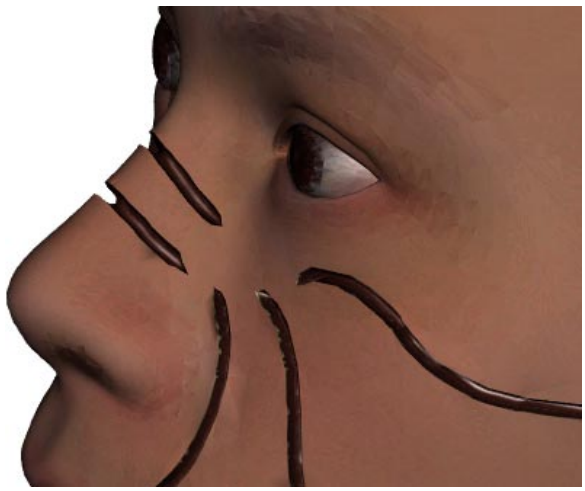

Figure 8. The local DFFD method was used to introduce several cuts into the cheek and nose. The inner part of the cut is shown, with flesh-like texture.

are introduced into the DFFD function, one for each bar. The robot simultaneously passes through several vertical parallel bars. When the robot reaches the bars, it splits apart, only to re-form beyond the bars. In this case, neither the DFFD stitching technique nor the Boolean operation method was used. As a result, the robot's interior is exposed during the split.

Figure 11 demonstrates that DFFD is independent of the orientation of the model. The robot model is placed in three different orientations relative to three semi-transparent parallel bars.

The DFFD operation is fairly cheap in terms of execution

Figure 9. Snapshots from an animation sequence of a bulldozer moving through a forming hole in a tree. Boolean operations were used to compute the interior of the hole, which is also wood-textured.

Figure 10. Snapshots from an animation sequence of a robot walking through vertical bars. The model was left open to reveal its inner parts. In this example, the bars that were originally used to generate the opening were replaced by slimmer bars to better expose the splitting effect.

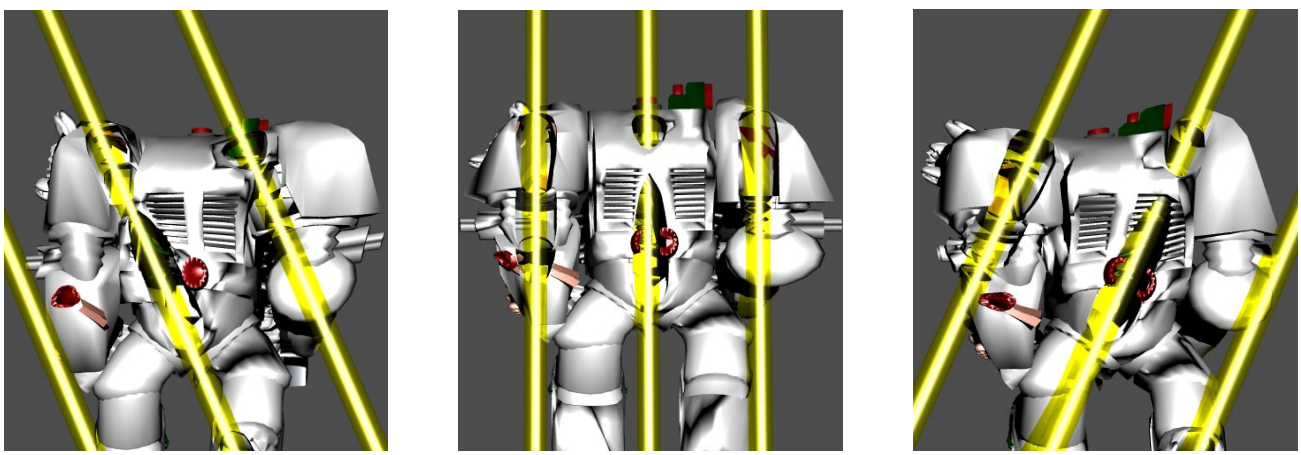

Figure 11. The robot model is cut with bars with different orientations. The DFFD construction is not limited to deformable objects with the same orientation as the obstacles.

time. A DFFD function with several thousand control points can be constructed at interactive rates, regardless of the complexity of any deformed model. The evaluation time of any model is linear in the number of vertices. For example, a single deformation of the model of the robot in Figure 10, which contains about 32000 vertices, took about one second on a *Pentium*^{tm}4 PC with 512 MB of RAM. For local DFFD (Section 4), both the construction rates of the DFFD function and the deformation time pose no limit on the real-time characteristics of the system.

6. Discussion and Conclusions

We have presented an algorithm for modeling cuts and openings in general meshes. The method builds upon a variant of FFD that supports the introduction of C^{-1} discontinuities into a deformation function. The algorithm employs standard knot insertions, followed by a manipulation of the control volume of the deformation function. To achieve a deformation that adheres to the discontinuities in the deformation function, a model-split algorithm is also defined. The deformed model is first split in the parametric domain of F. Since the inputs to the algorithm are surface-based meshes, we also present the means to close or stitch the openings that are introduced into the model as a result of applying the DFFD operation.

The DFFD operation has been demonstrated using two applications. The first automatically wraps deformable objects around obstacles that are placed in a scene. Several animation sequences that demonstrate the applicability of the proposed method are shown. The second defines a local DFFD function that cuts the surface in real-time. The embedded parts are parameterized relative to the DFFD volume and then split and re-formed, resulting in an arbitrarily shaped cut. We foresee that surgical simulators could benefit from such cutting operations. We expect that a careful tissue-based analysis of the cutting models, such as FEA, could be encoded into the cutting DFFD to a user-defined precision. Then, a purely geometric deformation phase would cut into the material in real-time, modifying the shape of the cut to resemble the expected shape. Applying DFFD to local areas of a model has two direct benefits. First, since only part of the model undergoes a deformation, the algorithm can achieve interactive manipulation rates. Second, the local version of the algorithm lets the designer specify arbitrarily-shaped incisions, alleviating the limitation of iso-parametric cuts.

DFFD operates by inserting iso-parametric discontinuities into the deformation function. Nothing prevents us from composing multiple DFFD operations on a single model, each in a different direction. An anticipated problem with such a composition will be in places where discontinuity planes intersect. One simple solution would be to compute the effect of a DFFD with a single iso-parametric discontinuity over an input geometry and then feed the result into a second, rotated in space, DFFD function.

The presented algorithm also has some limitations. First, currently it can only handle iso-parametric cuts. This shortcoming is a direct result of using knot insertion to incorporate a potential C^{-1} discontinuity. Enabling arbitrarily shaped discontinuity contours possibly as trimmed volumetric DFFD functions, would broaden the class of available cuts. Toward this end, intuitive ways should be developed to model the complex shape of the trimming surfaces. One approach that might present a feasible research direction is an extension to direct manipulation tools [9] to the problem of trimming surfaces design.

A second limitation of the algorithm results in the use of a projection procedure to construct the DFFD function. Such a projection procedure can only handle obstacles that are monotone relative to the axis of movement, v, which implies that each projected point might intersect $C^{\pm}(v)$ exactly once. In order to broaden the class of objects that can be used for design of the DFFD function, other construction methods should be explored.

As presented, the algorithm cuts through surface-based meshes or models. In order to improve the usability of the algorithm for surgical simulations, we need to support cuts through volumetric meshes. Since our deformation model is volumetric by nature, we foresee no conceptual problem here. However, surgical simulations require the deformation engines to work in real-time.

Similarly, in order to make DFFDs into a more general design tool, the support for free-form rational geometries should be considered. Supporting free-form surface-based models could be added by supporting trimmed surface in the deformable models.

7. Acknowledgment

The bunny model was downloaded from the Stanford 3D scanning repository - graphics.stanford.edu/data/3Dscanrep/. The walking robot model was downloaded from the 3D cafe web site - www.3dcafe.com/asp/scifi.asp. All the scenes that are shown in this work were modeled using the IRIT [10] modeling system.

This work was supported in part by the Bar-Nir Bergreen Software Technology Center of Excellence and through a donation from IBM's Shared University Research (SUR - www-3.ibm.com/software/info/university/sur/) program to the Technion in 2003 and by the EU through the Network of Excellence AIM@SHAPE Contract IST 506766. We would wish to express special thanks for Guy Sela for developing the software that was used for demonstrating the local DFFD operation.

References

[1] D. Bielser, V. A. Maiwald, and M. H. Gross. Interactive cuts through 3-dimensional soft tissue. In *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 31–38. The Eurographics Association and Blackwell Publishers, 1999.

[2] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions - a unified approach to specifying deformation in volume modeling and animation. In *Proceedings of 3rd International Workshop on Volume Graphics*, Tokyo, Japan, July 2003.

[3] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete bsplines and subdivision techniques in computer aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14:87–111, 1980.

[4] S. 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, Aug. 1990.

[5] M. S. Ellens and E. Cohen. An approach to c^{-1} and c^0 feature lines. *Mathematical Methods for Curves and Surfaces*, pages 121–132, 1995.

[6] J. E. Gain and N. 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.

[7] J. Griessmair and W. Purgathofer. Deformation of solids with trivariate b-splines. In *Eurograpghic 89*, pages 137–148, 1989.

[8] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992.

[9] S.-M. Hu, H. Zhang, C.-L. Tai, and J.-G. Sun. Direct manipulation of ffd: Efficient explicit solutions and decomposible multiple point constraints. *The Visual Computer*, 17(6):370–379, 2001.

[10] Irit. *Irit 9.0 Solid Modeling Environment. www.cs.technion.ac.il/∼irit*, 2002.

[11] H. Lamousin and W. Waggernspack. Nurbs-based free-form deformations. *IEEE Computer Graphics and Applications*, 14(6):59–65, Nov. 1994.

[12] F. Lazarus, S. Coquillart, and P. Jancene. Axial deformations: an interactive deformation technique. *Computer-Aided Design*, 26(8):607–613, Aug 1994.

[13] I. Llamas, B. Kim, J. Gargus, J. Rossignac, and C. D. Show. Twister: A space-warp operator for the two-handed editing of 3d shapes. In *ACM Siggraph Proceedings*, volume 22, pages 663–668, July 2003.

[14] R. MacCracken and K. 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.

[15] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, Maryland, U.S.A., 1988.

[16] T. Milliron, R. J. Jensen, R. Barzel, and A. Finkelstein. A framework for geometric warps and deformations. *ACM Transactions on Graphics*, 21(1):20–51, 2002.

[17] H.-W. Nienhuys and A. F. van der Stappen. A surgery simulation supporting cuts and finite element deformation. In *Medical Image Computing and Computer-Assisted Intervention*, volume 2208 of *Lecture Notes in Computer Science*, pages 153–160, Utrecht, The Netherlands, October 2001. Springer-Verlag.

[18] H.-W. Nienhuys and A. F. van der Stappen. A delaunay approach to interactive cutting in triangulated surfaces. In *Fifth International Workshop on Algorithmic Foundations of Robotics*, volume 7 of *Springer Tracts in Advanced Robotics*, pages 113–129, Nice, France, December 2004. Springer-Verlag Berlin Heidelberg.

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

[20] G. Sela, S. Schein, and G. Elber. Real-time incision simulation using discontinuous free form deformation. In S. Cotin and D. Metaxes, editors, *International Symposium on Medical Simulation*, volume 3078 of *LNCS*, pages 114–123, Cambridge, MA, USA, june 2004. Springer.

[21] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 405–414. ACM Press, 1998.

[22] B. Steyn and J. Gain. Topology alteration for virtual sculpting using spatial deformation. In *AFRIGRAPH 2003: Proceedings of the 2nd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 63–68, Cape Town, South Africa, Feb 2003. ACM press.