Simple Constrained Deformations for Geometric Modeling and Interactive Design

Paul Borrel [†] Ari Rappoport [‡]

[†] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, borrel@waston.ibm.com
[‡] Computer Science Department, The Hebrew University, Jerusalem 91904 Israel, arir@cs.huji.ac.il

Abstract. Deformations are a powerful tool for shape modeling and design. We present a new model for producing controlled spatial deformations, which we term *Simple Constrained Deformations (Scodef)*. The user defines a set of constraint points, giving a desired displacement and radius of influence for each. Each constraint point determines a local B-spline basis function centered at the constraint point, falling to zero for points beyond the radius. The deformed image of any point in space is a blend of these basis functions, using a projection matrix computed to satisfy the constraints. The deformation operates on the whole space regardless of the representation of the objects embedded inside the space. The constraints directly influence the final shape of the deformed objects, and this shape can be fine-tuned by adjusting the radius of influence of each constraint point. The computations required by the technique can be done very efficiently, and real-time interactive deformation editing on current workstations is possible.

Keywords. Geometric modeling, geometric design, deformation, spatial deformation, constraints, B-splines, interpolation.

1 Introduction

One of the most important issues in geometric modeling is providing operations for modifying an object's shape. Deformations are powerful sculpting operations since they allow high level shape modification, as opposed to manipulation of lower level geometric entities. A particularly appealing type of deformation is the spatial deformation, which operates on the whole space regardless of the representation of the deformed objects embedded inside the space.

Another approach to deformations is given in the context of physically-based modeling [Barr89]. This approach uses physical simulation to obtain realistic shapes and motions. The technique is very promising but has some drawbacks. Currently the expensive cost of the computations involved does not enable real-time interactive design. In addition, the representation of objects must be one that supports such simulations and therefore cannot be arbitrary. Finally, obtaining enough control over the final shape of the deformed objects is difficult; using physics can limit the design space, and avoiding this problem requires some sophistication [Platt88, Terzopoulos88, Celniker91]. For these reasons we chose to concentrate on purely geometric, spatial deformations.

One of the first uses of spatial deformations in the context of geometric modeling was in [Barr84], where some specific types of deformations were introduced. However, the general problems of arbitrarily shaped deformations and the way the user specifies a deformation or achieves local control are not addressed.

A popular method to define spatial deformations is by using tri-variate parametric volumes. Sederberg and Parry presented a free-form deformation (FFD) technique in which a user manipulates control points of tri-variate Bézier volumes [Sederberg86]. The same approach, using B-spline volumes, was implemented in [Griessmair89]. Coquillart extended FFD to operate on volumes with a topology different from a cube [Coquillart90]. Joy studied the applicability of tri-variate modeling, calling it parametric hyperpatch [Joy91] and providing some higher level editing operations. An integration of FFD with superquadrics for solid modeling is described in [Güdükbay90]. Chadwick *et al* used FFD for animation [Chadwick89].

These methods can express a very wide variety of deformations, but they force the user to first define control points around the region of space to deform and then manipulate these control points. This may be unnatural and tiring in case there are many of them. In addition, tri-variate volumes are based on multiplication of three basis functions, which may be too costly for interactive applications.

A general deformation model in which the deformation is defined by an arbitrary number of user specified point displacement constraints is given in [Borrel92]. The system lets the user select a solution obeying the constraints. As a result, user operations are kept to a minimum and the resulting deformation is intuitive since its definition only requires manipulating existing points, for example points on the surface of an object. The technique can be used for directly manipulating free-form deformations [Rappoport91, Hsu92].

However, the type of deformation (i.e., the nature of the mapping) is not related to the fact that it is defined by constraints; these are only used to compute parameters of a deformation function that could be computed by other means (e.g. control point manipulation). Consequently, the shape of the deformation is not strongly correlated with the constraints (except of the fact that they are satisfied).

In this paper we introduce a new type of deformation, which we term *Simple Constrained Deformation (Scodef)*. The user defines a set of constraint points, giving a desired displacement and radius of influence for each. Each constraint point determines a local B-spline basis function centered at the constraint point, falling to zero for points beyond the radius. The displacement of a point is a blend of these basis functions, obtained by a linear combination that insures that all constraints are satisfied.

A Scodef deformation can be viewed as the deformation obtained by creating an arbitrary number of possibly overlapping B-spline shaped “bumps” over the space. The location and height of a bump are defined by a constraint and its width by the constraints’ radius of influence. Scodef is a constraint based deformation in two senses: first, it is defined using constraints; second, the constraints directly influence the final shape of the deformed objects, and this shape can be fine-tuned by adjusting the radius of influence of each constraint point.

The computations required by the technique can be done very efficiently, and real-time interactive deformation editing on current workstations is possible.

The paper is organized as follows. Section 2 presents the Scodef deformation and shows how to compute the deformed image of any point in space and the deformed normal at a point on a surface. In Section 3 we study some properties of Scodef which are important for interactive design, such as the inter-relationship between constraints. Section 4 details some extensions to the basic Scodef model. In Section 5 we describe our implementation of a real-time interactive deformation editor and give some examples.

2 Simple Constrained Deformation

In this section we define Scodef and show how to compute the deformed image of any point in space and the deformed normal at a point on a surface. We also state Scodef’s continuity properties.

2.1 The Deformation Model

Let n be the dimension of the space which is to be deformed. Scodef is defined by a set of an arbitrary number r of constraints. Denote by C_i the position of the point in the original space, and by D_i its displacement in the deformed space. Each constraint has a radius of influence R_i associated with it.

Let Q be a point of R^n. For conciseness, the letter Q also represents the column-matrix of the point’s coordinates. Let $d : R^n \to R^n$ be the deformation function that expresses the displacement $d(Q)$ of Q. d is given by:

$$d(Q) = \sum_{i=1}^{r} M_i f_i(Q) \tag{1}$$

where M_i is an n dimensional column vector, and f_i represents the contribution of the i-th constraint and is a scalar function of the constraint point C_i and its radius R_i:

$$f_i = B_i\left(\frac{\| Q - C_i \|}{R_i}\right)$$

B_i is a B-spline basis function centered at zero and dropping to zero at 1. f_i is a equal to 1 at C_i and scaled to drop to zero for all points whose distance from C_i is greater than the radius R_i. We denote $U_i(Q) = \frac{\|Q-C_i\|}{R_i}$, so that $f_i = B_i(U_i(Q))$. f_i is a function providing a local parameterization for Q with respect to the constraint C_i. See Figure 1.

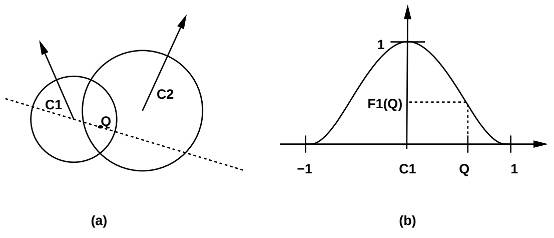

Figure 1: Computing $f_1(U(Q))$ for a constraint. f_2 would be computed in the same manner.

Equation 1 can be written concisely as $d(Q) = Mf(Q)$ where M is an $n \times r$ matrix whose columns are the M_i, and $f(Q)$ is an r dimensional column vector whose components are the f_i. Each column M_i of M scales the contribution of the i-th constraint.

2.2 Satisfying the Constraints

The deformation satisfies all the constraints by a proper choice of the matrix M. The columns of M do not have any special meaning (such as control points in some space), they are just computed to satisfy the constraints.

To satisfy each constraint C_i, M must be such that:

$$D_i = d(C_i) = Mf(C_i), \forall i = 1 \ldots n$$

Let D_{ij} be the j-th coordinate of D_i. Let M^j be the j-th row of M. M^j is an r-dimensional vector. Let the superscript T denote transposition. From the previous equation, we can write:

$$D_{ij} = M^j f(C_i) = f^T(C_i) M^{jT}.$$

Combining all constraints in a single equation, for the same j-th coordinate of space, we have:

$$D_J(C) = \begin{pmatrix} D_{1j} \\ \dots \\ D_{rj} \end{pmatrix} = \begin{pmatrix} f^T(C_1) \\ \dots \\ f^T(C_r) \end{pmatrix} M^{jT} = XM^{jT} \tag{2}$$

where X is a $r \times r$ matrix built from the images under f of the constraint points. Since X does not depend on j, we can build such a system of linear equations for each space coordinate, and all these systems will contain the same matrix X. The solution of each system is one of the rows M^j of M.

Depending of the relative location of the constraint points C_i, solving these n systems may be trivial (e.g. when no C_i is within the influence zone of any other constraint), or may require special techniques (e.g. when two or more constraint points C_i are coincident). This issue is discussed in detail in Section 3. For now, we simply assume that a an inverse of X is available.

2.3 Continuity

The continuity nature of Scodef is easy to determine. Scodef maps R^n to itself using a linear combination (defined by the matrix M) of blending functions f_i. Every coordinate of f is of the form $f_i = B_i(U_i(Q))$, a composition of two functions. U_i is infinitely differentiable everywhere except at the constraint point where it is only C^1 (continuous first derivative). B_i is a B-spline basis function, hence infinitely differentiable everywhere except at the knots where the continuity is at most the degree of the B-spline minus one[1]. Naturally, by duplicating knots the continuity can be reduced.

If the fact that Scodef is not C^2 at the constraint point is disturbing, other functions U_i with better continuity behaviour can be used. The requirements from U_i are that it should be zero at the constraint C_i, monotonely increasing, and larger than unity for points at distance larger than R_i from C_i.

2.4 Deformed Normal at a Surface Point

In many applications the deformation will be applied to the surface of a 3D object. In order to perform various calculations on the deformed surface we need a way to compute the normal at a point $d(Q)$. Assume that the surface is parameterized by two parameters u, v so that $Q = Q(u, v)$ and that we are able to compute the tangents $T_u(Q), T_v(Q)$ according to each parameter in the original space.

To compute the normal $N(d(Q))$ at a deformed point $d(Q)$ we have to first compute the tangents $T_u(d(Q)), T_v(d(Q))$ at this point and then take their normalized cross product. We cannot perform the deformation on the original normal since local differential properties of the surface are not maintained under the deformation.

[1]We do not address the issue of geometric continuity here.

Computing the tangent at a surface point requires computing the Jacobian $J_d(Q)$ of the deformation function at that point. This is a simple consequence of the chain rule for taking the derivative of a function:

$$\frac{\partial}{\partial u} d(Q(u,v)) = \frac{\partial}{\partial Q} d(Q) \frac{\partial}{\partial u} Q(u,v) = J_d(Q) T_u(Q)$$

The matrix M is constant over the whole space. Recall that we denote $U_i(Q) = \frac{\|Q-C_i\|}{R_i}$. If we denote $U(Q) = (U_1(Q), \cdots, U_r(Q))$, Then we have

$$J_d(Q) = \frac{\partial}{\partial Q} d(Q) = M \frac{\partial f}{\partial Q} = M \frac{\partial f}{\partial U} \frac{\partial U}{\partial Q}$$

where $\frac{\partial f}{\partial U}$ is the $r \times r$ Jacobian matrix of $f(U)$ and $\frac{\partial U}{\partial Q}$ is the $r \times n$ Jacobian matrix of $U(Q)$. It is easy to see that

$$\frac{\partial f_i}{\partial U_j} = \begin{cases} 0 & i \neq j \\ \frac{\partial B_i(U)}{\partial U} \,|_{U=U_i} & i = j \end{cases}$$

and

$$\frac{\partial U_i}{\partial Q_j} = \frac{\frac{\partial}{\partial Q_j} \| Q - C_i \|}{R_i} = \frac{Q_j - C_{ij}}{R_i \| Q - C_i \|}$$

where C_{ij} is the j-th coordinate of the constraint point C_i. Given the derivative of a B-spline basis function at a point, the above equations provide an easy way of computing the Jacobian of the deformation function.

3 Scodef Properties

In this section we study some properties of Scodef which are important from the point of view of interactive design.

3.1 Disjoint Constraints

We say that a constraint C_i *influences* a point Q if the distance between them is smaller than the radius associated with the constraint: $\| C_i - Q \| < R_i$. Two constraints are said to be *disjoint* if neither one influences the other. A set of constraints is disjoint if they are pairwise disjoint. A set of constraints is *completely disjoint* if their radii do not overlap. See Figure 2.

Suppose that the constraint set C of a Scodef is disjoint. Then for every constraint C_i we have

$$U_j(C_i) = \frac{\| C_i - C_j \|}{R_j} = \begin{cases} > 1 & i \neq j \\ 0 & i = j \end{cases}$$

therefore $f_j(C_i) = \delta_{ij}$, and $f(C_i) = (0, \cdots, 0, 1, 0, \cdots, 0)^T$, zero at all coordinates except of 1 at coordinate i. Recall that M_i denotes the i-th column of the projection matrix M. We have

$$D_i = d(C_i) = (M_1, \cdots, M_r) f(C_i) = M_i.$$

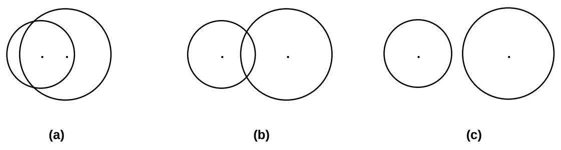

Figure 2: Non disjoint (a), disjoint (b), and completely disjoint (c) constraint pairs.

The i-th column of the matrix M is equal to the displacement $d(C_i)$ of the constraint point C_i.

Let us now consider any point Q. We have

$$d(Q) = \sum_{i=1}^{r} B_i(\frac{\| C_i - Q \|}{R_i})D_i$$

which can be phrased as follows: when the constraints are disjoint the displacement of a point Q is a weighted average of the displacements of the constraints, the weight being inversely proportional to the distance to a constraint.

3.2 Influence of One Constraint

Suppose there is only one constraint C_1 defining the Scodef. The matrix M will have one column, the displacement vector of the constraint, since this situation is a special case of the analysis in the previous sub-section. For every point lying outside of the radius of influence of the constraint, the f vector will be zero, and the point will not be deformed, i.e. its displacement vector will be zero.

A point Q inside the radius of influence will have a non-zero f vector, which in this case is simply a scalar s since $r = 1$. The displacement vector of Q is a vector in the direction of $d(C_1)$ scaled by the scalar s. All points influenced by C_1 will move in the same direction, and the distance they move depends upon their distance to the constraint. Note that no point will move a distance larger than the displacement of a constraint, since s is monotonely decreasing and has its maximum on the constraint.

The above result is valid also when there are a number of constraints which are completely disjoint. In this case the columns of M are again the displacement vectors of the constraint, and every point in space is either influenced by one constraint or by none of them. A Scodef defined by a set of r completely disjoint constraints can therefore be regarded as separable into r independent Scodefs. See Figure 3 for an example. The 2D space deformation is visualized in the figure by presenting its effect on a regular 2D grid (this figure is a 2D figure even though it gives the impression of a 3D figure). Observe that the deformation created by the constraint on the right (marked with an arrow) is not a one-to-one mapping.

Figure 3: A 2D Scodef separable into a set of three independent Scodefs, visualized by its effect on a regular grid.

3.3 Control of the Mutual Influence of Constraints

In this sub-section we analyze the mutual influence of constraints on the deformation and describe methods to control this influence.

3.3.1 Influence of Non Disjoint Constraints

When two constraints C_1 and C_2 are non disjoint, the deformation of points influenced by both constraints is the weighted average of their contribution:

$$D(Q) = M_1 f_1(Q) + M_2 f_2(Q)$$

Using D_1 and D_2, this system is solved for M_1 and M_2 as explained earlier. Changing one constraint (e.g. changing D_1) modifies the displacements of points that are not within the radius of influence of that constraint because the solution of the above system is modified, thus modifying the weight of the other constraint (i.e. M_2). This behavior may lead to non-intuitive results in the interactive editing of constraints since changing one constraint may affect points that are far away from the constraints' zone of influence. See below for a solution to this problem.

3.3.2 Singularities

Assume a Scodef is defined by two constraints having the same radius of influence, which is larger than the distance between them, and the same knot vector of their B-splines. In this case we have $f_2(C_1) = f_1(C_2) = a$, so

$$(d(C_1)d(C_2)) = M \begin{pmatrix} 1 & a \\ a & 1 \end{pmatrix}$$

Solving this system for M yields

$$M_1 = \frac{1}{1-a^2} d(C_1) - \frac{a}{1-a^2} d(C_2)$$

$$M_2 = \frac{-a}{1-a^2} d(C_1) + \frac{1}{1-a^2} d(C_2)$$

Now assume that $d(C_1) = -d(C_2)$, then $M_1 = -M_2 = \frac{1}{1-a} d(C_1)$, and

$$d(Q) = \frac{d(C_1)}{1-a} (B_1(Q) - B_2(Q))$$

We can draw some conclusions from the above equations.

- As the two constraints get closer to each other, a approaches 1 and the elements of M get larger.
- If $d(C_1) \neq d(C_2)$ and the two constraints coincide ($a = 1$) there is no M that can satisfy the constraints (this is natural since it means that a single point should have two different images under the deformation).
- Since the magnitude of $d(C_1)$ is independent of $f_1(Q) - f_2(Q)$, a point Q in the vicinity of the constraints can move very far away. Visually, space 'tearing' phenomena can occur when the distance between two constraints is much smaller than the magnitude of their radii (Figure 4).

Figure 4: Space 'tearing' as a result of Scodef singularities. The deformation is visualized here by its effect on a set of points lying on an horizontal line.

The space deformation in the neighbourhood of constraints which are close to each other (relative to their radii) is not well behaved.

3.3.3 Improving the Control with Constraint Duplication

The two problems mentioned previously arise from the fact that the deformation is unique given a set of constraints: the system of equations (2) is not redundant, and the deformation is imposed by its unique solution. It would instead be more powerful if the system had an infinity of solutions, and if the user had the ability to select one according to his needs.

A very simple and efficient way of creating redundancy in the system, while conserving Scodef's simplicity, is constraint duplication: the user has an option to declare that a chosen constraint is *duplicated*. To the user, this means that this constraint now possesses two radii instead of one. The user can manipulate both to gain finer control over the deformation in the vicinity of that constraint.

The matrix X in equation (2) now becomes singular. In order to find a solution, we use the pseudo-inverse of X [Bouillon71]. The pseudo-inverse is not expensive to compute [Greville60] and gives by default the least squares solution of the system. If needed, an optimization term could be added, as in [Borrel91], to find other solutions. However, in our case we simply want to find other solutions to the initial set of constraints, not to the system that contains the duplicated constraints. This can be achieved by varying the radii of the duplicated constraints: every time new radii values are provided, a new system of equations is created and solved using the standard least square solution.

A duplicated constraint has two radii. The larger one determines the range of influence of the constraint. The smaller one intuitively determines to what extent the points are influenced. Specifically, decreasing the smaller radius decreases the influence that the constraint has on neighbouring constraints. As a result, better control is provided on how non disjoint constraints influence each other, and the undesired oscillations of space in the vicinity of a singularity are reduced. Figure 5 shows how the duplication technique reduces the distortions of Figure 4.

Figure 5: The effect of a duplicated constraint.

4 Extensions

In this section we discuss two extensions to Scodef, whose goals are to enhance the blending functions used in the deformation.

4.1 Choosing Other Interpolation Functions

The definition of Scodef given so far is based on the use of B-spline functions to interpolate between constraint points and undeformed regions. Although the B-spline enjoys properties that facilitate design (e.g. continuity, local control), other interpolation functions could be chosen as well. A useful modification of the basic B-spline function we experimented with allows translating a whole (undeformed) chunk of space which remains connected to the rest of the space by B-spline interpolation: the B-spline is split into two parts which are connected around the origin by an horizontal segment.

The length of the horizontal segment is an additional parameter r_i associated to each constraint C_i. Observe that setting this parameter to zero achieves the normal Scodef deformation. The functions f_i are now replaced by:

$$f_i(Q) = \begin{cases} 1 & \| Q - C_i \| < r_i \\ B_i(\frac{\|Q-C_i\|-r_i}{R_i}) & \| Q - C_i \| > r_i \end{cases}$$

4.2 Extended Radius of Influence

The definition of Scodef can be modified to accommodate even finer control of the range of influence of a constraint. The new range treats each space coordinate differently so as to provide a-symmetric, non-isotropic deformation of space around constraints.

The extended radius of influence is a n dimensional column-matrix. In 3D, denote $R_i = [R_{ix} R_{iy} R_{iz}]^T$ the extended radius of influence of the i-th constraint. The new f_i function is now defined as follows:

$$f_i = \sqrt{\frac{(Q_x - C_{ix})^2}{R_{ix}^2} + \frac{(Q_y - C_{iy})^2}{R_{iy}^2} + \frac{(Q_z - C_{iz})^2}{R_{iz}^2}}$$

Changing the three components of R_i allows tuning the deformation independently along the three space coordinates.

4.3 Local Parameterization of the Deformation

The extended radius of influence controls the shape of the deformation with respect to the axes of a reference coordinate system. Because an object to deform may be in arbitrary position and orientation in this coordinate system, it is more powerful to define the shape of the deformation with respect to some coordinate system associated to the constraint. This is called local parametrization of the deformation.

Associate to each constraint a local coordinate system, defined by n direction vectors (they need not be orthogonal, although this would be the most natural case). Further, associate a B-spline with each of these direction vectors. Together, these define a box centered at the constraint, oriented according to the direction vectors, and whose dimensions are the range of non-zero support of the B-splines.

The function $U_i(Q, C)$ now computes the position of a point Q in a local coordinate system of constraint C_i, i.e., the local coordinates of Q within the box. The function $f_i(Q)$ is now the tensor product of the direction B-splines of C_i evaluated at the local coordinate values.

This technique allows finer control over the deformation, but the number of B-spline evaluations (the critical performance factor) is exactly n times as before since f_i now evaluates n B-splines instead of one. In addition, we also have the cost of multiplying the B-spline values to obtain each coordinate of $f(Q)$. These are $n - 1$ multiplications per coordinate where previously there were none.

5 Implementation

We implemented an experimental system to visualize the types of deformations achieved by Scodef and to deform objects. The system is written in C and runs on IBM RS/6000 workstations under the AIX operating system. The user interface is implemented with the SigVig user interface toolkit server [Emmerik91].

The system is meant to experiment with the usability of Scodef for computer-aided design. The program is capable of importing existing object models (including models created with the Catia[2] solid modeler), interactively defining and editing constraints, and visualizing the deformed object. Programming the Scodef required only pseudo-inverse and B-spline evaluation routines; the rest was standard data structure manipulation.

Recall that Scodef is a spatial deformation; the whole space is deformed, not only an object embedded inside it. During interaction, an object is deformed by transforming a set of points. The points are vertices of triangular meshes that approximate the object's surface. The user can control the accuracy of the approximation by modifying the number of sample points on the object.

[2]Catia is a trademark of Dassault Systèmes.

It is possible to interactively modify the position and the radius of a constraint and visualize the result in real time. This efficient performance is due to the small number of computations that Scodef requires relative to other deformation techniques.

6 Discussion

We presented a new spatial deformation technique, termed *simple constrained deformation (Scodef)*. The deformation is defined by point displacement constraints. A B-spline basis function is positioned at each constraint and scaled according to a radius of influence associated with the constraint. The image of any point in space under the deformation is a linear blend of these blending functions, computed such that the constraints are satisfied. The final shape of the deformation can be fine-tuned by adjusting the radius of each constraint and the knot vectors of the B-splines. The computational effort required by the technique is small enough to enable interactive deformation editing on current workstations.

The process of defining and obtaining the deformation is basically a process of scattered data interpolation [Lancaster86]. Some techniques in interpolation theory, especially the kriging method for reconstructing height fields from random samples, resemble the Scodef model. However, they mostly emphasize mathematical properties of the resulting interpolant which are not important for interactive design. For example, the kriging method uses complex minimization of a functional via a covariance matrix, and no mechanism is given for simple local control. We believe that the present paper is the first application of such techniques for general shape design.

There are a number of issues which still have to be investigated. Currently in order to compose two deformations of an object we apply the deformations in sequence to a set of points lying on the object's boundary. This amounts to a crude linear approximation of the deformations; to get closer to the real composition more accurate methods are needed. A related issue is the visualization of the deformed object. A method such as direct ray casting can be used to obtain visualization results which are better than polygonal rendering.

Regarding deformations in general, a deformation model that enables curve and area constraints (as opposed to only point constraints) should be developed. Our current model cannot be generalized to deal with such constraints. In addition, different deformation techniques should be compared to see which technique is applicable in which setting. Clearly, there is a place for each of the techniques discussed in Section 1, but their relative strengths and weaknesses for specific applications are not well understood.

Acknowledgements

The work presented here has benefited from an earlier collaboration with Dominique Bechmann and from insights by Jarek Rossignac. The comments of the referees have

greatly improved the quality of the paper.

References

[Barr84] Barr, A.H., Global and local deformations of solid primitives, *Computer Graphics* 17(3):21-30, 1984, (Siggraph 84).

[Barr89] Barr, A., Witkin, A. (editors), Topics in Physically Based Modeling, ACM Siggraph 89 course notes 30, 1989.

[Bechmann92] Bechmann, D., Dubreuil, N., Animation through space and time based on a space deformation model, *Eurographics Workshop on Animation and Simulation*, 1992.

[Borrel91] Borrel, P., Bechmann, D., Deformation of n-dimensional objects, *Intl. Journal of Computational Geometry and Applications*, 1(4), 1991. Also in *ACM Symposium on Solid Modeling*, Austin, June 5-7, 1991, pp 351-370.

[Boullion71] Boullion, T.L., Odell, P.L., Generalized Inverse Matrices, Wiley, New-York, 1971.

[Celniker91] Celniker, G., Gossard, D., Deformable curve and surface finite elements for free form shape design, *Computer Graphics* 25(4):257-266, 1991 (Siggraph 91).

[Chadwick89] Chadwick, J.E., Haumann, D.R., Parent, R.E., Layered construction for deformable animated characters, *Computer Graphics* 23(3):243-252, 1989 (Siggraph 89).

[Coquillart90] Coquillart, S., Extended free form deformation: a sculpturing tool for 3D geometric modeling, *Computer Graphics* 24(4):187-193, 1990 (Siggraph 90).

[Emmerik91] Emmerik, M.J.G.M. van, Rappoport, A., SigVig: a concept for separation of application and user interface toolkit, IBM Research Report, submitted for publication, 1991.

[Griessmair89] Griessmair, J., Purgathofer, W., Deformation of solids with trivariate B-splines, *Eurographics '89*, 137-148, 1989.

[Güdükbay90] Güdükbay, U., Özgüc, B., Free form solid modeling using deformations, *Computers and Graphics*, 14(3/4):491-500, 1990.

[Hsu92] Hsu, W.M., Hughes, J.F., Kaufman, H., Direct manipulation of free-form deformations, *Computer Graphics* 26(2):177-184, 1992 (Siggraph 92).

[Joy91] Joy, K., Utilizing parametric hyperpatch methods for modeling and display of free form solids, *ACM Symposium on Solid Modeling*, Austin, June 5-7, 1991, pp. 245-254.

[Lancaster86] Lancaster, P., Salkauskas, K., Curve and Surface Fitting: an Introduction, Academic Press, 1986.

[Platt88] Platt, J., Barr., A.H., Constraints methods for flexible models, *Computer Graphics* 22(4), 1988 (Siggraph 88).

[Rappoport91] Rappoport, A., About deformations, internal IBM manuscript, March 1991.

[Sederberg86] Sederberg, T.W., Parry, S.R., Free-form deformation of solid geometric models, *Computer Graphics* 20(4):151-160, 1986 (SIGGRAPH ’86).

[Terzopoulos88] Terzopoulos, D., Witkin, A., Physically based methods with rigid and deformable components, *IEEE Computer Graphics and Applications* 8:41-51, 1988.

Figure 6: caption ...