

Volume-Preserving Free-Form Solids

A thesis submitted in fulfillment
of the requirements for the degree of
Master Of Science

by

Sheffer Alla

supervised by

Dr. Ari Rappoport
and
Prof. Michel Bercovier

Institute of Computer Science
The Hebrew University of Jerusalem
Jerusalem 91904, Israel
`sheffa@cs.huji.ac.il`

Acknowledgements:

I'd like to thank my advisors in this work, Prof. Michel Bercovier and Dr. Ari Rappoport, for their help during this work. I'd like to thank Michel for sharing with me his vast mathematical knowledge and experience. I thank Ari for helping me to transform the mathematical ideas and algorithms into a user friendly, interactive modelling package and for his help in writing this paper.

I'd like to thank Dany Youlus for participating in the first steps of this work and saving me from some very nasty BUGs.

Abstract:

In this paper we present a novel method for modeling an object composed of several tensor-product solid primitives while preserving the desired volume of each primitive and ensuring high-order continuity constraints between them. We achieve those requirements with minimal necessary change from the original object. The change can be represented geometrically as the square distance between the original and resulting objects control point configurations or physically as the deformation energy required to convert the original object into the resulting one. The computation method we use utilizes the Uzawa algorithm for non-linear optimization.

We show how the algorithm can be used in an interactive environment by relaxing exactness requirements while the user interactively manipulates free-form solid primitives. On current workstations, the algorithm runs in real-time for tri-quadratic volumes and close to real-time for tri-cubic volumes.

Although in this work we limit the method description to Bézier solids, it can easily be adopted for most of the other common definitions of free-form solids, for example for NURBS. The only restriction on the mathematical definition of the solid that we have is that it should be defined as a linear combination of the control points.

Contents

1	Introduction	6
1.1	Background	6
1.2	Proposed Approach	8
2	Notations	11
3	Statement of the Problem	13
4	The Volume Function	16
4.1	Computation of the Volume	16
4.2	Computation of the Volume Derivative	19
5	The Constraints	20
5.1	Continuity Constraints	20
5.2	Position and other Constraints	21
5.3	Summary	22
6	The Energy Function	23
6.1	Computation of the Energy	23
6.2	Computation of the Energy Derivative	26
7	The Uzawa-Based Volume Preservation Algorithm	27
7.1	Lagrangian multiplier method	27
7.2	Solution method	28
8	Results	33
9	Conclusions	35
10	References	36

List of Figures

1	A quadratic Bézier in initial cube state	11
2	Example of results of the two distance functions applied on 2-D Bézier patch of 2×3 : <i>a: initial state; b: with constraints applied on right and left side points with no volume preservation; c: result of applying LSQ method on original patch with constraints; d: result of applying energy method on original patch with constraints.</i>	14
3	(left) Original elements, (right) The object created via C^1 continuity constraints.	22
4	The Uzawa-based volume preservation algorithm.	30
5	The ‘inner’ P computation by successive approximation.	32
6	The Motif interface of the system	33
7	An armchair (tri-cubic.)	39
8	An amphora (tri-cubic.)	39
9	A pitcher (3 tri-cubic and one element of $4 \times 3 \times 3$)	40
10	A phone tube (3 tri-cubic.)	40

1 Introduction

Some of the important trends in geometric modeling in the last few years have been the reliance on solid models rather than surface-based models and the enhancement of the expressive power of models, by using free-form objects in addition to the usual geometric primitives and by incorporating physical principles, all this while emphasizing interactive performance. In this paper we integrate all of these requirements in a single geometric primitive.

The primitive we use is a tri-variate tensor product volume. The use of 3-D definition of the object enables us to preserve while shaping the body it's basic properties such as volume. We present a novel algorithm for free-form volume design enabling us to model a body built from a number of tri-variate tensor product volume primitives while keeping the volume of the body as desired and keeping continuity constraints between the primitives.

1.1 Background

The common approach to representing and manipulating free-form objects is by using a boundary representation (Brep), with parametric surfaces for the boundary. Adjacencies between neighboring surface patches are stored explicitly. Using a Brep, it is inherently difficult to model physical attributes associated with the object. Such attributes are easier to consider when using parametric free-form *solids* instead of surfaces. The difference between the two is the dimension of the parameter space (two for surfaces and three for solids.)

Some previous systems have used free-form solids (e.g. [Farouki85]). However, parametric volumes are not used in the way that surfaces are used, for direct object design, but rather for design of separate deformation entities used for modification of existing objects (free-form deformations (FFD) [Sederberg86]). This can be explained by the fact that if only the boundary of the object is of interest, there is no need to use free-form solids which enable control over what happens 'inside' the object. In this paper we use free-form solids as primitives for geometric design, not as deformation entities.

FFD utilizes a tri-variate tensor-product parametric Bézier solid defined by a

lattice of control points. The defining parameter space is the unit cube. Naturally, other basis functions (such as NURBS) could be used as well [Griessmair89]. Sederberg suggested a user interface based on control point manipulation, with which is it rather difficult and tedious to obtain a desired deformation. Direct manipulation of object points instead of control point manipulation was suggested in [Borrel91, Hsu92]. The user directly moves an object point, and the system automatically computes the control point configuration yielding the desired point displacement constraints. In [Rappoport94] this method is extended to approximate (‘probabilistic’) point constraints with a non-isotropic shape parameter. [Joy91] gave methods to manipulate a group of control points in a single operation. A more general type of extension to FFD was presented in [Coquillart91], who defined an arbitrary volume and used numerical routines to compute local coordinates within this volume. Neither of the above methods attaches any physical meaning of the deformation.

Physics-based modeling is a successful research area in geometric modeling. Several papers [Terzopoulos94, Welch92, Kallay93, Moreton92, Celniker91, Greiner93] presented surface design schemes based on minimization of an energy functional subject to linear point constraints such as location and tangent vectors. We are not aware of any work using similar ideas for free-form solids. Other applications of physics-based modeling are in reconstruction and tracking [Fang92], motion control [Shapiro88], and modeling of flexible and rigid objects [Barzel88].

The only relevant reference we are aware of for volume preservation ¹ is [Aumann92], which gives an algorithm that approximates a surface of revolution by a surface which is not a surface of revolution while trying to preserve the original volume. Free-form solids are not discussed, and it seems that the algorithm is not suited for them at all. Formulae for computing the area or volume enclosed by curves and surface patches were given in [Elber94, Liu87].

¹[Sederberg86] refers to an unpublished report about volume preserving deformations, but such deformations cannot be everywhere locally satisfied with polynomial fields except for the simple case of pure shears.

1.2 Proposed Approach

As mentioned above, we use free-form solids as design primitives. In the context of solid model design in general and specifically of free-form solids, one of the most basic physical properties of a space cell is its volume size, the measure of the domain it contains. A major drawback of current user interaction techniques when applied to free-form solid design is that the user has no way of controlling the contained volume size. Currently, solid design (as opposed to using volumes for free-form deformations) is not much more than design of the surfaces bounding the volume, each of them independently.

We present a novel method for modeling an object composed of several tensor-product solids while preserving the desired volume of each primitive and ensuring high-order continuity constraints (and any linear constraints in the control points) between the primitives. The method utilizes the Uzawa algorithm for non-linear optimization, with an objective function based on deformation energy or least squares.

The algorithm is very useful for several applications. For example, hierarchical FFDs were used by [Chadwick89] for computer animation of muscles. A similar effect could be achieved by a combination of point displacement constraints and smooth increase/decrease of desired volume size. The algorithm is useful in industrial design, where basic functional requirements are automatically obeyed without imposing limitation on the creativity of the designer. When the object material is known, volume preservation means weight preservation, hence is attractive for mechanical engineering applications when the engineer designs a part or an assembly. The preservation of volume of each element of the objects enables us to keep required proportions between volumes and weights of object parts. Obviously, simple scaling of the object in order to achieve a desired volume is not possible, due to the presence of point location and continuity constraints.

Our algorithm uses Bézier solids of arbitrary orders as the underlying mathematical definition of a free-form solid primitive. A Bézier solid of known orders is completely specified by its control points. The input to our algorithm consists of the desired object form (set of primitives defined by their control points configurations), desired primitives volume sizes and a set of linear constraints on the control points implied by continuity requirements between the primitives or imposed di-

rectly by user. The control points configurations can either be given directly by the user through control point manipulation, or computed from point displacement constraints specified by direct solid manipulation as in [Borrel91, Hsu92]. The algorithm computes a control point configuration closest to the given one (in a deformation energy minimization or least square sense) such that the deformed primitives contain volume of the given sizes and the linear constraints are fulfilled. The algorithm does not automatically guarantee that the boundaries do not self-intersect. the constraints rising from topological correctness demand can be found in [Stewart93] and can be if desired incorporated into the set of constraints handled by our system.

Note that it is the global volume of a given free-form ‘cell’ primitive that is being preserved, not the volume of an object embedded within the cell or of local sub-cells. This approach was introduced in the finite element method for rubber type materials, but here we avoid the complexity of the penalty approach [Bercovier81] and use a duality argument to deal with the constraint, based on the Uzawa algorithm for non-linear programming [Arrow58, Ciarlet88].

Special measures were taken in order to endow the algorithm with real-time performance on current workstations. We utilize the fact that the volume size actually depends only on the boundary surfaces of the deformed primitive, hence volume size computation can be done with a subset of the control points. The inside points are of no interest to the user as well for the object representation, but are required for any physical computations on the object such as tear strength or deformation energy. The inside control points are computed once from the outside points using a 3-D variant of the Coons surface formula when energy computation is required. and then serve as input to the minimization algorithm as well. Preservation of both linear and volume constraints together requires quite heavy computations and there for can not be achieved during interactive modeling. During interaction only one set of constraints is preserved (either volume or linear). In addition, while interactively preserving the volume the algorithm relaxes its accuracy requirements during object manipulation, computing an accurate solution only when real-time performance is no longer essential while still giving the user a feeling that volume is preserved during interaction.

Although in this work we limit the method description to Bézier solids, it can easily be adopted for most of the other common definitions of free-form solids, for

example for NURBS. The only restriction on the mathematical definition of the solid that we have is that it should be defined as a linear combination of the control points.

The paper is organized as follows. Section 2 gives necessary mathematical notations. Section 3 formalizes the mathematical problem involved. Section 4 explains in detail how to compute the size of the volume enclosed by a tensor product Bézier solid and the partial derivatives of the volume size function. Section 5 explains how to ensure continuity constraints. Section 6 explains how to compute the energy required for a change of a tensor product Bézier solid from one control point configuration to another and the energy derivative. Section 7 presents the numerical algorithm used to solve the mathematical problem, and Section 8 describes our implementation and results.

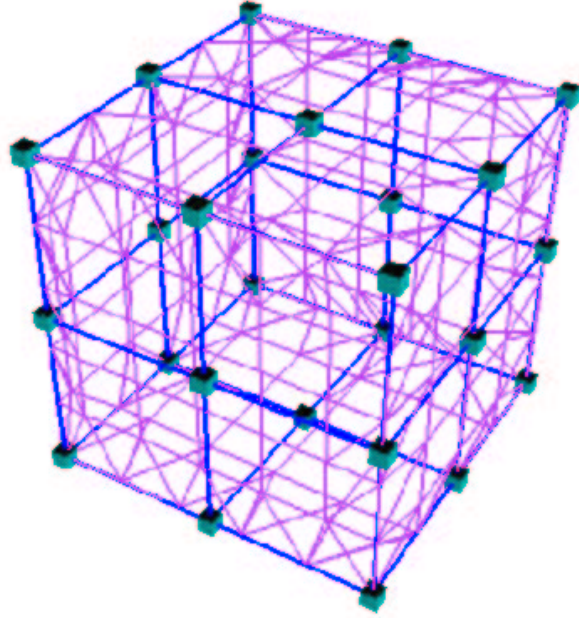


Figure 1: A quadratic Bézier in initial cube state

2 Notations

We introduce here the formal mathematical notations used during the rest of the work.

A tensor product Bézier solid is defined using a set of control points

$$P_{ijk} \in \mathbb{R}^3,$$

The image of a parametric point (u, v, w) in the unit cube is

$$F(u, v, w) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w) P_{ijk} \quad (1)$$

where $B_i^n(t)$ is the Bernstein polynomial defined by

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i.$$

Denote the x, y, z coordinates of a control point by $P_{ijk}^x, P_{ijk}^y, P_{ijk}^z$ respectively.

Denote the volume of the solid primitive defined by a set P of control points by

$$Volume(P),$$

and by

$$\frac{\partial Volume(P)}{\partial P}$$

the vector whose components are the partial derivatives

$$\frac{\partial Volume(P)}{\partial P_{abc}^x}, \frac{\partial Volume(P)}{\partial P_{abc}^y}, \frac{\partial Volume(P)}{\partial P_{abc}^z}$$

for every triplet $abc \in [1 \dots n] \times [1 \dots m] \times [1 \dots l]$.

Denote the energy of a transformation from a Bézier solid defined by a configuration Q of control points to one defined by configuration P of control points by

$$Energy(P - Q),$$

and by

$$\frac{\partial Energy(P - Q)}{\partial P}$$

the vector whose components are the partial derivatives

$$\frac{\partial Energy(P - Q)}{\partial P_{abc}^x}, \frac{\partial Energy(P - Q)}{\partial P_{abc}^y}, \frac{\partial Energy(P - Q)}{\partial P_{abc}^z}.$$

Denote the matrix representing the linear constraints by C . Denote by \bar{P} the column vector of all the control points (from all the Bézier solids in the system) $(P_{ijk}^x, P_{ijk}^y, P_{ijk}^z, 1)$, that is $\bar{P} = (P, 1)$. The constraints are achieved when $C\bar{P} = 0$.

3 Statement of the Problem

The general problem we handle is finding a control point configuration that satisfies the existing constraints (linear and volume) and which results in an object as close as possible to the given one. The change of an object can be represented in two ways, one the simpler is as the sum of squares of distances between the original control point positions and the new ones, the second more physically based is as the energy required to get from the original object to the new one.

In this section we formalize this problem as a set of mathematical requirements that the target control points configuration should satisfy.

We will denote as $Dist(P, Q)$ the distance between two objects resulting from control points locations P and Q where it can stand for:

- If using energy approach $Energy(P - Q)$.
- If using LSQ approach $\frac{1}{2}(P - Q)^T(P - Q)$.

In case we use the program to model objects directly , the modeled (“original”) objects are usually close to the desired final ones - since if a designer enforces a set of constraints on an object then during modeling he won’t violate them too much (if he wants two primitives to be connected he will not set them too much apart), in this case the the distance measure should be LSQ since we want the resulting object control points to be as close as possible to the “original” ones so the shape of the object will change as little as possible.

Another way of modeling we can use is completely physically based and in it we use as the “original” object the element in an initial state (usually the standard rectangular shape) and we deform it by implying linear constraints on some of the control points and running the energy minimization process, the resulting object then simulates the behavior of an elastic material with internal pressure (volume preservation) that was subject to the same constraints. The difference of results achieved using the two technics is shown in Figure 2.

As explained in Section 1, the “original” control point configuration could have been obtained by any method, including control point manipulation and direct manipulation of points inside/on the solid.

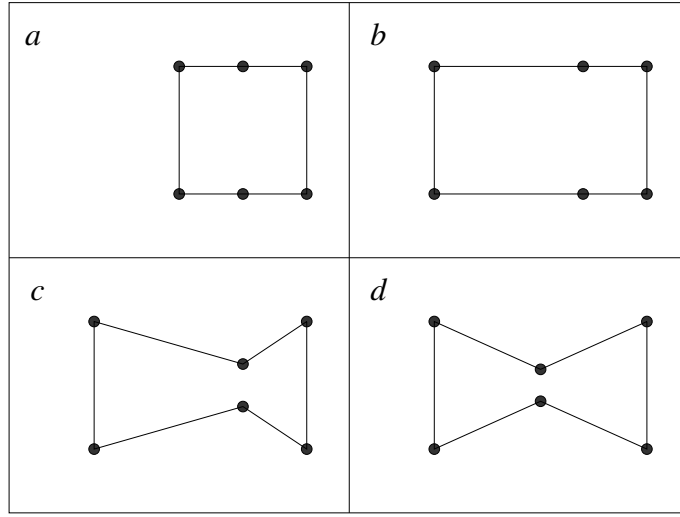


Figure 2: Example of results of the two distance functions applied on 2-D Bézier patch of 2×3 : *a*: initial state; *b*: with constraints applied on right and left side points with no volume preservation; *c*: result of applying LSQ method on original patch with constraints; *d*: result of applying energy method on original patch with constraints.

The constrained minimization problem (M) resulting from both distance computation approaches will be:

Given a control point configuration Q_1, \dots, Q_n (each Q_i representing a single tri-variate primitive), a set of corresponding volume sizes V_1, \dots, V_n and a matrix C representing linear constraints on the control points, find a new control point configuration $P = P_1, \dots, P_n$ such that the following holds:

- P minimizes the expression: $\min_{P'_i} \sum_{i=1}^n \text{Dist}(P'_i, Q_i)$,
- For each i , $\text{Volume}(P_i) = V_i$,
- $C\bar{P} = 0$ (recall that $\bar{P} = (P, 1)$).

The desired volumes V_i could be the initial volume sizes if the application is interested in volume preservation or any other number. (e.g. smoothly varying the desired volumes can be used for growing (‘dilating’) the object in animations.)

The computations of $Volume(P)$ and of $Energy(P - Q)$ are quite complicated and are explained in detail in the following sections (Section 4 and Section 6). The creation of matrix C from the continuity constraints involved is explained in Section 5.

The presented problem is non-linear since as explained in the next section the volume function and its derivative are non linear. Therefore to solve it we require a method for solving non linear constrained minimization problems. The method we use is explained in detail in Section 7.

4 The Volume Function

Volume calculation and preservation is one of the main innovations of our program. Fundamental subroutines in the Uzawa-based volume preservation algorithm to be presented in Section 7 are the computations of $Volume(P)$ and of $\frac{\partial Volume(P)}{\partial P}$. Below we show how to compute analytically the volume size of a tensor product Bézier solid. We show that the computation of the volume size can be represented as a scalar product of two vectors: one whose components are the multiplication of the coordinates of the solid's control points, and second one whose components are based on the Bézier basic functions and therefor can be computed off-line just once for each combination of orders of basis functions. It is not even essential to understand how this second vector was created; it is enough for it to be given.

Note that the way of computation we present gives us exact analytical results independent of the control point configuration, as opposed to the usual volume computation methods using volume subdivision into pyramids which requires adaptive subdivision to get a closer result and requires new subdivision computation for each control points configuration in order to get exact results.

4.1 Computation of the Volume

The size of the volume specified by a three-dimensional function $F(u, v, w)$ defined over the unit cube is

$$\int_0^1 \int_0^1 \int_0^1 J_F du dv dw$$

where J_F is the determinant of the Jacobian matrix of F :

$$J_F = \det \left(\frac{\partial F_i}{\partial x_j} \right), \quad i = x, y, z \quad x_j = u, v, w.$$

In our case F is given by Equation 1. For example, the entry in the first row and column of the Jacobian matrix is

$$\frac{\partial F_x}{\partial u} = \sum_i \sum_j \sum_k \frac{d}{du} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w} P_{ijk}^x.$$

The derivative of a Bernstein polynomial of order n can be expressed by the scaled difference of two Bernstein polynomials of order $n - 1$ [Farin92]:

$$\frac{d}{du}B_i^n(u) = n(B_{i-1}^{n-1}(u) - B_i^{n-1}(u))$$

with the convention that $B_b^a(u) = 0$ for $b < 0$, $a < 0$, or $b > a$. Denote

$$\begin{aligned} \bar{u}_{ijk} &= \frac{d}{du}B_i^n(u)B_j^{n_v}(v)B_k^{n_w}(w) = \\ &n_u(B_{i-1}^{n_u-1}(u) - B_i^{n_u-1}(u))B_j^{n_v}(v)B_k^{n_w}(w), \end{aligned} \quad (2)$$

and similarly \bar{v}_{ijk} and \bar{w}_{ijk} . The determinant J_F can be written

$$J_F = \det \begin{pmatrix} \sum_{ijk} \bar{u}_{ijk} P_{ijk}^x & \sum_{ijk} \bar{v}_{ijk} P_{ijk}^x & \sum_{ijk} \bar{w}_{ijk} P_{ijk}^x \\ \sum_{ijk} \bar{u}_{ijk} P_{ijk}^y & \sum_{ijk} \bar{v}_{ijk} P_{ijk}^y & \sum_{ijk} \bar{w}_{ijk} P_{ijk}^y \\ \sum_{ijk} \bar{u}_{ijk} P_{ijk}^z & \sum_{ijk} \bar{v}_{ijk} P_{ijk}^z & \sum_{ijk} \bar{w}_{ijk} P_{ijk}^z \end{pmatrix}, \quad (3)$$

expanding, we obtain:

$$\begin{aligned} J_F &= \sum_{ijk} \bar{u}_{ijk} P_{ijk}^x \sum_{lmn} \bar{v}_{lmn} P_{lmn}^y \sum_{opq} \bar{w}_{opq} P_{opq}^z - \sum_{ijk} \bar{u}_{ijk} P_{ijk}^x \sum_{lmn} \bar{v}_{lmn} P_{lmn}^z \sum_{opq} \bar{w}_{opq} P_{opq}^y \\ &\quad - \sum_{ijk} \bar{v}_{ijk} P_{ijk}^x \sum_{lmn} \bar{u}_{lmn} P_{lmn}^y \sum_{opq} \bar{w}_{opq} P_{opq}^z + \sum_{ijk} \bar{v}_{ijk} P_{ijk}^x \sum_{lmn} \bar{w}_{lmn} P_{lmn}^y \sum_{opq} \bar{u}_{opq} P_{opq}^z \\ &\quad + \sum_{ijk} \bar{w}_{ijk} P_{ijk}^x \sum_{lmn} \bar{u}_{lmn} P_{lmn}^y \sum_{opq} \bar{v}_{opq} P_{opq}^z - \sum_{ijk} \bar{w}_{ijk} P_{ijk}^x \sum_{lmn} \bar{v}_{lmn} P_{lmn}^y \sum_{opq} \bar{u}_{opq} P_{opq}^z. \end{aligned}$$

Since the determinant is a multilinear operator and due to the structure of the summations, we can write:

$$J_F = \sum_{ijk} \sum_{lmn} \sum_{opq} \det \begin{pmatrix} \bar{u}_{ijk} & \bar{u}_{lmn} & \bar{u}_{opq} \\ \bar{v}_{ijk} & \bar{v}_{lmn} & \bar{v}_{opq} \\ \bar{w}_{ijk} & \bar{w}_{lmn} & \bar{w}_{opq} \end{pmatrix} P_{ijk}^x P_{lmn}^y P_{opq}^z. \quad (4)$$

Let $I = ijklmnopq$ be a new index notation, in the range $1 \dots (n_u n_v n_w)^3$. Denote the determinant in Equation 4 by $\det_I(u, v, w)$, and denote

$$c_I = \int_0^1 \int_0^1 \int_0^1 \det_I(u, v, w) du dv dw.$$

Since the integral is a linear operator, the volume can be written as:

$$Volume(P) = \sum_{ijk} \sum_{lmn} \sum_{opq} c_I P_{ijk}^x P_{lmn}^y P_{opq}^z. \quad (5)$$

Let \mathbf{p} be a column vector indexed by I containing all terms of the form $P_{ijk}^x P_{lmn}^y P_{opq}^z$, and let \mathbf{c} be a column vector of the same size whose components are c_I . Then Equation 5 can be expressed as the scalar product of \mathbf{p} and \mathbf{c} :

$$Volume(P) = \mathbf{c}^T \mathbf{p}.$$

The vector \mathbf{c} depends only on the orders of the Bézier basis functions, hence can be computed once and for all for every practical order combination (the number of all practically useful order combinations is small.) Computing the elements of \mathbf{c} analytically, via symbolic integration is very complicated even for relatively small Bézier orders, therefore they can be computed using numerical integration only. We compute it using Gauss numerical integration method [Press88], which gives us an exact result since the integrated functions are polynomials. A component c_I is computed as

$$c_I = \sum_r \sum_s \sum_t w_r w_s w_t \det_I(x_r, x_s, x_t)$$

where w_r, w_s, w_t are the Gauss weights corresponding to the points x_r, x_s, x_t in the unit interval. The number of sample points on each dimension is determined according to the order of the basis function in that dimension.

The description above was simplistic for the sake of ease of explanation. Actually, volume size depends only on the boundary surfaces (Stokes' formula [Gibson44].) In Bézier curves, the end points pass through the first and last control points. In Bézier surfaces, the positions of the four boundary curves are defined only by the control points corresponding to parameter values of 0 or 1 in any of the parameters. Similarly, in Bézier volumes the boundary surfaces are not influenced at all by the 'inner' control points. The inner control points can be completely neglected during the computation of the volume size and in fact when computing the elements of the \mathbf{c} vector we will find that for any combination $ijklmnopq$ containing an inside point coordinates the values $c_{ijklmnopq}$ are equal to zero.

In practice, then, to accelerate the volume computation we let the index I run only on values of $ijklmnopq$ which define 'outer' control points. This simple observation can save an order of magnitude of floating point operations when the orders of the Bézier basis functions are high. We emphasize that in either case, run-time volume computation simply entails a scalar product between two vectors.

4.2 Computation of the Volume Derivative

The volume preservation algorithm requires the computation of the vector $\partial Volume(P)/\partial P$ whose components are of the form $\partial Volume(P)/\partial P_{abc}^r$ where r is x, y or z . For example,

$$\frac{\partial Volume(P)}{\partial P_{abc}^x} = \frac{\partial \left(\sum_{ijk} \sum_{lmn} \sum_{opq} c_{ijklmno pq} P_{ijk}^x P_{lmn}^y P_{opq}^z \right)}{\partial P_{abc}^x},$$

and since for every $ijk \neq abc$ the partial derivative vanishes we get

$$\frac{\partial Volume(P)}{\partial P_{abc}^x} = \sum_{lmn} \sum_{opq} c_{abclmno pq} P_{lmn}^y P_{opq}^z.$$

5 The Constraints

In this section we explain the different linear constraints imposed on the control points configuration required in order to achieve desired geometrical or physical effects.

5.1 Continuity Constraints

Continuity constraints between primitives in an object are essential for any object design. Continuity of order k (C^k) between two adjacent volumes $F_1(u, v, w)$ and $F_2(u, v, w)$ defined on $[0, 1]^3$ in the u direction is achieved when the following holds

$$\left(\frac{\partial^k F_1(u, v, w)}{\partial^k u} \right) (1, v', w') = \left(\frac{\partial^k F_2(u, v, w)}{\partial^k u} \right) (0, v', w')$$

for every $(v', w') \in [0, 1] \times [0, 1]$.

In our case for two adjacent primitives defined by control points configurations P and Q we will get

$$\begin{aligned} & \left(\frac{\partial^k (\sum_{ijk} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w) P_{ijk})}{\partial^k u} \right) (1, v', w') = \\ & \left(\frac{\partial^k (\sum_{ijk} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w) Q_{ijk})}{\partial^k u} \right) (0, v', w'). \end{aligned}$$

Since derivative is a linear operator we get

$$\begin{aligned} & \left(\sum_{ijk} \left(\frac{\partial^k B_i^{n_u}(u)}{\partial^k u} \right) B_j^{n_v}(v) B_k^{n_w}(w) P_{ijk} \right) (1, v', w') = \\ & \left(\sum_{ijk} \left(\frac{\partial^k B_i^{n_u}(u)}{\partial^k u} \right) B_j^{n_v}(v) B_k^{n_w}(w) Q_{ijk} \right) (0, v', w'). \end{aligned}$$

and therefore

$$\sum_{jk} B_j^{n_v}(v') B_k^{n_w}(w') \left(\sum_i \left(\frac{\partial^k B_i^{n_u}(u)}{\partial^k u} (1) P_{ijk} - \frac{\partial^k B_i^{n_u}(u)}{\partial^k u} (0) Q_{ijk} \right) \right) = 0.$$

To hold this for each $v', w' \in [0, 1] \times [0, 1]$ necessary and sufficient condition is that for $j = 1 \dots n_v, k = 1 \dots n_w$

$$\sum_i \left(\frac{\partial^k B_i^{n_u}(u)}{\partial^k u} (1) P_{ijk} - \frac{\partial^k B_i^{n_u}(u)}{\partial^k u} (0) Q_{ijk} \right) = 0,$$

thus getting a set of $n_v n_w$ linear equations in P_{ijk}, Q_{ijk} .

As mentioned in Section 4.1 the derivative of a Bernstein polynomial of order n can be expressed by the scaled difference of two Bernstein polynomials of order $n - 1$ [Farin92]:

$$\frac{d}{du} B_i^n(u) = n(B_{i-1}^{n-1}(u) - B_i^{n-1}(u))$$

with the convention that $B_b^a(u) = 0$ for $b < 0, a < 0$, or $b > a$. Since $B_l^m(0) \neq 0$ for $l = 0$ only and $B_l^m(1) \neq 0$ only for $l = m$ then the number of i for which $\frac{\partial^k B_i^{n_u}(u)}{\partial^k u}(1) \neq 0$ is $k + 1$ and same for $\frac{\partial^k B_i^{n_u}(u)}{\partial^k u}(0)$.

There for C^k continuity condition between adjacent Bézier volumes are expressed as a set of $n_v n_w$ linear equations on $k + 1$ layers of control points of each volume from the adjacent border.

For the most common cases the conditions are:

$$\begin{aligned} C^0 & P_{n_u, i, j} - Q_{1, i, j} = 0 \quad i = 1 \dots n_v, \quad j = 1 \dots n_w \\ C^1 & n_u(P_{n_u, i, j} - P_{n_u-1, i, j}) - n_u(Q_{2, i, j} - Q_{1, i, j}) = 0 \quad i = 1 \dots n_v, \quad j = 1 \dots n_w. \end{aligned}$$

Another kind of continuity constraint between elements is geometric continuity. It is more general than than parametric C^k continuity. If we can reparametrize the given volume and in the new parametrisation it will have C^k continuity than by definition the given volume has GC^k continuity. It is clear that this is a much weaker demand than C^k continuity and it is enough for visual continuity but the constraints on the control points that it imposes are much harder to express and are non-linear, therefor we do not use them in our work. For more details on geometric constraints (see [Bercovier93]).

In Figure 3 we show five elements before C^1 continuity constraints were enforced between them and after.

5.2 Position and other Constraints

Besides continuity constraint we handle the following constraints as well. Fixing a point at a given location, which result in equations like

$$P_i^r - c_i^r = 0 \quad r = x, y, z$$

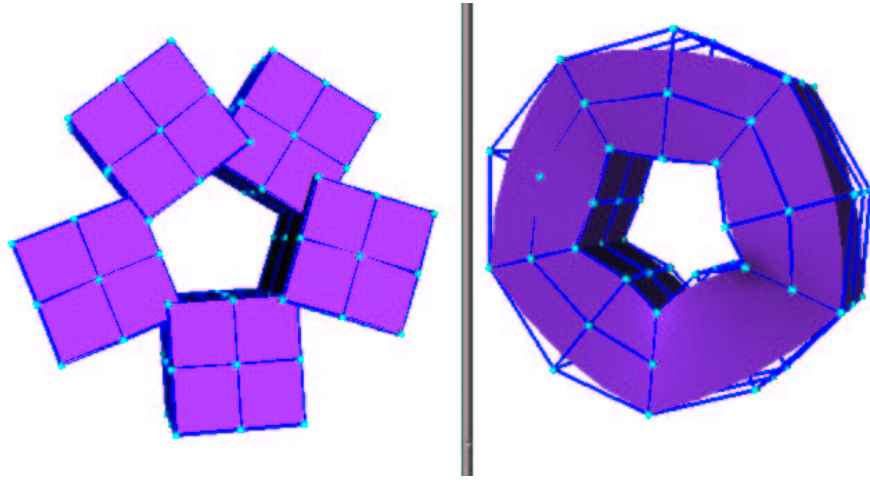


Figure 3: (left) Original elements, (right) The object created via C^1 continuity constraints.

where c_i^r is a constant .

Attaching together two points, which results in equations like

$$P_i^r - Q_i^r = 0 \quad r = x, y, z .$$

Preserving a given distance between points can easily be handled also.

5.3 Summary

A general linear equation on variables P_i is expressed as

$$\sum_i c_i P_i + c_{n+1} = 0$$

or in vector representation as

$$c^T(P, 1) = 0.$$

We denote by C the matrix whose rows are the coefficients c of the linear equations and by \bar{P} the column vector of all the control points (from all the Bézier solids in the system) $(P_{ijk}^x, P_{ijk}^y, P_{ijk}^z, 1)$, that is $\bar{P} = (P, 1)$ (Section 2).

The constraints are achieved when $C\bar{P} = 0$.

6 The Energy Function

Energy computation for deformation of a Bézier primitive from one control point configuration to another is one of the basic parts in our algorithm. Here we show that it can be computed as a product of multiplying the vector of difference between control points configurations by a precomputed matrix and by itself. The matrix elements depend only on the order of Bézier basis functions and hence it can be computed just once for each order combination.

6.1 Computation of the Energy

The energy of a deformation of a unit cube specified by a three-dimensional vector function $F(x_1, x_2, x_3)$ is usually described as

$$\int_0^1 \int_0^1 \int_0^1 \frac{1}{2} \left(\beta \sum_i \left(\frac{\partial F_i}{\partial x_i} \right)^2 + \alpha \sum_{i \neq j} \left(\left(\frac{\partial F_i}{\partial x_j} \right) + \left(\frac{\partial F_j}{\partial x_i} \right) \right)^2 \right) dx_1 dx_2 dx_3 \quad (6)$$

with α and β being material property constants [Terzopolus94].

This notation is however not correct for a general rigid body, since it doesn't give us a zero result under rotation as can be easily checked on a most basic rotation of a tri-linear unit cube. The energy as described above is preserved for transformations which include translation and rotation around a fixed given vector, and not for general rotations. Details are in [Ciarlet83, Duvaut76].

But in case of Bézier elements if using in Equation 6 instead of symbolic integration numerical under-integration (ie for Parabolic 2 point gauss, for Cubic 3 point gauss , etc) we get in fact a function which tends to preserve rotations (get zero deformation energy on general rotation), although we usually need to add a small mass matrix to the result to keep the rank of the result matrix. So for the rest of the section we continue to use Equation 6 as our energy functional keeping in mind that by integration in it we mean in fact Gaussian under-integration.

Since the integral is a linear operator we can write

$$Energy(F) = \frac{1}{2} \beta \sum_i \int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_i}{\partial x_i} \right)^2 dudvdw + \quad (7)$$

$$\frac{1}{2}\alpha \sum_{i \neq j} \left(\int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_i}{\partial x_j} \right)^2 dudvdw + \int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_j}{\partial x_i} \right)^2 dudvdw + 2 \int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_j}{\partial x_i} \right) \left(\frac{\partial F_i}{\partial x_j} \right) dudvdw \right),$$

In our case the deformations used are of modifying a body defined by one Bézier point configuration to a body defined by another one. It can easily be seen that the deformation itself is then defined as a tri-variate Bézier functions with distances between the control points of the two configurations serving as it's control point lattice. Using P as this new point lattice and using the notation of \bar{u}_{ijk} as defined in Equation 2 we can write

$$\frac{\partial F_x}{\partial x_1} = \frac{\partial F_x}{\partial u} = \sum_{ijk} \bar{u}_{ijk} P_{ijk}^x,$$

and consequently

$$\frac{\partial F_x^2}{\partial u} = \sum_{ijk} \sum_{lmn} \bar{u}_{ijk} \bar{u}_{lmn} P_{ijk}^x P_{lmn}^x.$$

So we have

$$\int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_x}{\partial u} \right)^2 dudvdw = \int_0^1 \int_0^1 \int_0^1 \left(\sum_{ijk} \sum_{lmn} \bar{u}_{ijk} \bar{u}_{lmn} P_{ijk}^x P_{lmn}^x \right) dudvdw,$$

and since integral is a linear operator we get

$$\int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_x}{\partial u} \right)^2 dudvdw = \sum_{ijk} \sum_{lmn} P_{ijk}^x P_{lmn}^x \left(\int_0^1 \int_0^1 \int_0^1 \bar{u}_{ijk} \bar{u}_{lmn} dudvdw \right).$$

Let Du be a matrix indexed by ijk and lmn , defined by

$$Du_{ijk,lmn} = \int_0^1 \int_0^1 \int_0^1 \bar{u}_{ijk} \bar{u}_{lmn} dudvdw$$

and define Dv and Dw similarly. Let Px be a column vector with components P_{ijk}^x , and similarly define Py and Pz . then we have

$$\int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_x}{\partial u} \right)^2 dudvdw = P^{xT} Du P^x.$$

The elements of Du , Dv , Dw depend only on the orders of the Bézier basis functions and can be computed using Gauss numerical integration.

Denote by Duv the matrix of the mixed derivatives given by

$$Duv_{ijk,lmn} = \int_0^1 \int_0^1 \int_0^1 \bar{u}_{ijk} \bar{v}_{lmn} dudvdw$$

and define Duw and Dvw similarly, as clearly seen using same definitions $Dvu = Duv^T$. We have

$$\int_0^1 \int_0^1 \int_0^1 \left(\frac{\partial F_y}{\partial u} \right) \left(\frac{\partial F_x}{\partial v} \right) dudvdw = P^{yT} Duv P^x.$$

The elements of the 3 matrices above can also be computed numerically.

Substituting the matrices into equation Equation 8 we have

$$\begin{aligned} Energy(F) = & \frac{1}{2} (\beta (P^{xT} Du P^x + P^{yT} Dv P^y + P^{zT} Dw P^z) + \\ & \alpha ((P^{xT} Dv P^x + P^{yT} Du P^y + P^{xT} Duv^T P^y + P^{yT} Duv P^x) + \\ & (P^{xT} Dw P^x + P^{zT} Du P^z + P^{xT} Duw^T P^z + P^{zT} Duw P^x) + \\ & (P^{yT} Dw P^y + P^{zT} Dv P^z + P^{yT} Dvw^T P^z + P^{zT} Dvw P^y))). \end{aligned}$$

And finally let D be the matrix

$$D = \frac{1}{2} \begin{pmatrix} (\beta Du + \alpha Dv + \alpha Dw) & \alpha Duv^T & \alpha Duw^T \\ \alpha Duv & (\beta Dv + \alpha Du + \alpha Dw) & \alpha Dvw^T \\ \alpha Duw & \alpha Dvw & (\beta Dw + \alpha Du + \alpha Dv) \end{pmatrix}.$$

Then we have, using P as defined above (column vector concatenating (Px, Py, Pz)).

$$Energy(P) = P^T D P.$$

All the elements of D depend only on the orders of the Bézier basis functions, hence can be computed exactly once and for all for every practical order combination, using Gauss integration, and as mentioned above for our needs we used under-integration instead of the “real” one, with a small mass matrix addition.

6.2 Computation of the Energy Derivative

To minimize the deformation energy the algorithm requires the computation of the vector $\partial Energy(P)/\partial P$ whose components are of the form $\partial Energy(P)/\partial P_{abc}^r$ where r is x, y or z . It is easy to see that, for example,

$$\frac{\partial Energy(P)}{\partial P_{abc}^x} = 2 \sum_r \sum_{lmn} D_{xabc,rlmn} P_{lmn}^r,$$

since for every $rijk \neq xabc$ the partial derivative of $Energy(P)$ according to P_{abc}^x vanishes.

7 The Uzawa-Based Volume Preservation Algorithm

In this section we explain in detail the algorithm we use for solution of the problem as defined in Section 3.

7.1 Lagrangian multiplier method

To convert the constrained minimization problem (M):

minimize $\sum_i^n Dist(P_i, Q_i)$ subject to the constraints $C\bar{P} = 0$ and $Volume(P_i) - V_i = 0, \quad i = 1 \dots n,$

into an unconstrained min-max problem we define a new functional L called the Lagrangian associated with the problem (M), by

$$L(P, \lambda, \gamma) = \sum_i Dist(P_i, Q_i) + \sum_i \lambda_i (Volume(P_i) - V_i) + \gamma C\bar{P}$$

where γ is vector with size the number of linear constraints. The vector $(\lambda_1 \dots \lambda_n, \gamma_1 \dots \gamma_m)$ is called the Lagrange multipliers vector, λ_i is called the Lagrange multiplier for the constraint $Volume(P_i) - V_i = 0$ and γ_j is called the Lagrange multiplier for the constraint $C_j\bar{P} = 0$ (C_j stands for row j of C).

As explained in [Ciarlet88], the constrained minimization problem (M) can be reformulated as finding a solution to the unconstrained min-max problem (S) defined by

$$(S) \quad \max_{\lambda, \gamma} \min_P L(P, \lambda, \gamma).$$

A necessary condition for a triplet (P, λ, γ) to be a solution of (S) is the vanishing of the partial derivatives:

$$\frac{\partial L}{\partial P} = \frac{\partial L}{\partial \lambda} = \frac{\partial L}{\partial \gamma} = 0,$$

which means for each $i = 1 \dots n$, and for each point coordinate $j = 1 \dots n_u n_v n_w, \quad r = x, y, z$:

$$\frac{\partial L}{\partial P_{i,j}^r} = \frac{\partial Dist(P_i, Q_i)}{\partial P_{i,j}^r} + \lambda_i \frac{\partial Volume(P_i)}{\partial P_{i,j}^r} + \gamma C_{i,j}^r = 0, \quad (8)$$

where $C|_{i,j}^r$ means the column of C that multiplies the point $P_{i,j}^r$ in \bar{P} ,
and

$$\frac{\partial L}{\partial \lambda_i} = Volume(P_i) - V_i = 0,$$

and for γ

$$\frac{\partial L}{\partial \gamma_j} = C_j \bar{P} = 0, \quad j = 1 \dots m.$$

7.2 Solution method

As mentioned above the constraints of $Volume(P_i) = V_i$ are non-linear and therefore the usual direct methods (such as LDL^T , Gauss elimination) can't be used to solve (S). Therefore we use a version of the Uzawa method tailored to our problem. Uzawa's method is an iterative method allowing one to solve an inequality constrained minimization problem by replacing it with a sequence of unconstrained minimization problems (see Ciarlet[88]). Since we don't have inequality constraints we can use a simpler version of the Uzawa method as explained below.

Given a problem (M) the iteration starts with an arbitrary values for $\lambda^0 \in R_+^n, \gamma^0 \in R_+^m$ (we start with 0 for both), and with an initial value for P^0 for which we use Q . These initial guesses are especially suitable in an interactive setting, where it is expected that Q will not change much after the constraints are satisfied. A sequence of triplets

$$(P^k, \lambda^k, \gamma^k) \in V \times R_+^n \times R_+^m, \quad k \geq 0$$

is defined by means of the following iterations:

$$P^k : \sum_i Dist(P_i) + \sum_i \lambda_i (Volume(P_i) - V_i) + \gamma C \bar{P} = \inf_{P' \in V} \left(\sum_i Dist(P'_i) + \sum_i \lambda_i (Volume(P'_i) - V_i) + \gamma C \bar{P}' \right) \quad (9)$$

. and

$$\lambda_i^{k+1} : \lambda_i^{k+1} = \max(\lambda_i^k + \rho_1 (Volume(P_i) - V_i), 0) \quad 1 \leq i \leq n, \quad (10)$$

$$\gamma_j^{k+1} \quad : \quad \gamma_j^{k+1} = \max(\gamma_j^k + \rho_2(C_j \bar{P}), 0) \quad 1 \leq j \leq m, \quad (11)$$

The problem in Equation 9 is solved in each iteration k as explained bellow. The algorithm runs until the constraints are satisfied or the number of iterations exceeds a given limit.

Pseudo-code for the algorithm is shown on next page (Figure 4). The initial values for P , λ and γ are set in lines 1 and 2. Line 3 computes the current volumes v_i and line 4 initializes the loop counter k .

The main (“outer”) loop of the algorithm is performed in lines 5-10. The loop iterates while the constraints are not satisfied, stopping after the limit on the number of iterations has been reached. In each iteration the system in Equation 8 resulting from Equation 9 is solved (line 6) and the current value of λ and γ is updated using the tuning parameters ρ_1 and ρ_2 respectively (lines 7-8) Line 11 returns P as the answer.

The choice of tuning parameters ρ_1 and ρ_2 . as used in Equation 10 and Equation 11 is practically the most difficult issue when using Uzawa’s method. Each type of problem has its own best range of values for ρ_i . In our case we found it best to use $\rho_1 = \rho_2 = 0.15$ if the *Dist* function used is *Energy*, and $\rho_1 = \rho_2 = 0.5$ if the *Dist* function used is the squares sum. In general the larger the ρ the faster the convergence goes but the risk is of overstepping the convergence point with a ρ too big and then not converging.

Input:

A set of control point configurations Q_1, \dots, Q_n

A set of desired volumes V_1, \dots, V_n

A matrix C of linear constraints

Output:

A set of new control point configurations Q_1, \dots, Q_n

that satisfies the Goal conditions.

Parameters:

Convergence tolerances $\delta_{vol}, \delta_{cons}, \delta_{dist}$

Iteration limits L, L_{dist}

Tuning parameters ρ_1, ρ_2

Goal:

Minimize $\sum_i Dist(P_i, Q_i)$ subject to $Volume(P_i) = V_i, C\bar{P} = 0$

Algorithm:

1 $P = Q$

2 $\lambda = 0, \gamma = 0$

3 $v_i = Volume(Q_i) \quad i = 1 \dots n$

4 $k = 0$

5 **while** $\sum_i |v_i - V_i| > \delta_{vol}$ **and** $\|C\bar{P}\| > \delta_{cons}$ **and** $k < L$

6 **find** $P_1 \dots P_n$ that solve $\frac{\partial Dist(P_i)}{\partial P_i} + \lambda_i \frac{\partial Volume(P_i)}{\partial P_i} + \gamma C|_i = 0$

7 $\lambda = \lambda + \rho_1(v - V)$

8 $\gamma = \gamma + \rho_2 C\bar{P}$

9 $v_i = Volume(P_i) \quad i = 1 \dots n$

10 $k = k + 1$

end

11 **return** P

Figure 4: The Uzawa-based volume preservation algorithm.

The “Inner” problem is of solving Equation 8 for P with the given γ and λ . It is a non linear problem and we solve it by a successive approximation method on P , that is we iteratively compute new values for P based on Equation 8 until the distance between two successive iterations is small enough (that is $\| P - P' \|^2 < \delta_{dist}$). This is just one of the possible technics for solving a set of non linear equations and other methods can be used .

In Figure 5 we show how to solve the “inner” problem on P by successive approximation for the two types of *Dist* functions used.

Usually when solving physically based problems by Lagrange multipliers method the additional variables added as multipliers have physical meaning. In our case one can interpret λ as an inner hydrostatic pressure to keep the volume at a given value. We are looking for the value of that pressure: the Uzawa outer step can be seen as augmenting or diminishing the hydrostatic pressure until convergence. This tuning is done with the parameter ρ_1 . This observation relates our method to so-called mixed finite element methods for the Stokes problem [Hughes87]. In our case we have constant pressure for each small volume element.

```

Solution of  $\frac{\partial Dist(P_i)}{\partial P_i} + \lambda_i \frac{\partial Volume(P_i)}{\partial P_i} + \gamma C|_i = 0$ 
for  $i = 1 \dots n$  :

1  if dist function is energy
2       $P_{ij}^{new} = Q_{ij} - \frac{\sum_{k \neq j} 2D_{jk}(P_{ik} - Q_{ik}) + \lambda_i \frac{\partial Volume(P_i)}{\partial P_{ij}} + \gamma C|_{ij}}{2D_{jj}}$ 
        for each control point coordinate  $i, j$ 
3  else
4       $P_{ij}^{new} = Q_{ij} - \lambda_i \frac{\partial Volume(P_i)}{\partial P_{ij}} - \gamma C|_{ij}$ 
        for each control point coordinate  $i, j$ 
5  endif
6   $l = 0$ 
7  while  $\|P - P^{new}\|^2 > \delta_{dist}$  and  $i < L_{dist}$ 
8       $P = P^{new}$ 
9      if dist function is energy
10          $P_{ij}^{new} = Q_{ij} - \frac{\sum_{k \neq j} 2D_{jk}(P_{ik} - Q_{ik}) + \lambda_i \frac{\partial Volume(P_i)}{\partial P_{ij}} + \gamma C|_{ij}}{2D_{jj}}$ 
           for each control point coordinate  $i, j$ 
11        else
12          $P_{ij}^{new} = Q_{ij} - \lambda_i \frac{\partial Volume(P_i)}{\partial P_{ij}} - \gamma C|_{ij}$ 
           for each control point coordinate  $i, j$ 
13        endif
14         $l = l + 1$ 
15 end
15 return  $P^{new}$ 

```

Figure 5: The ‘inner’ P computation by successive approximation.

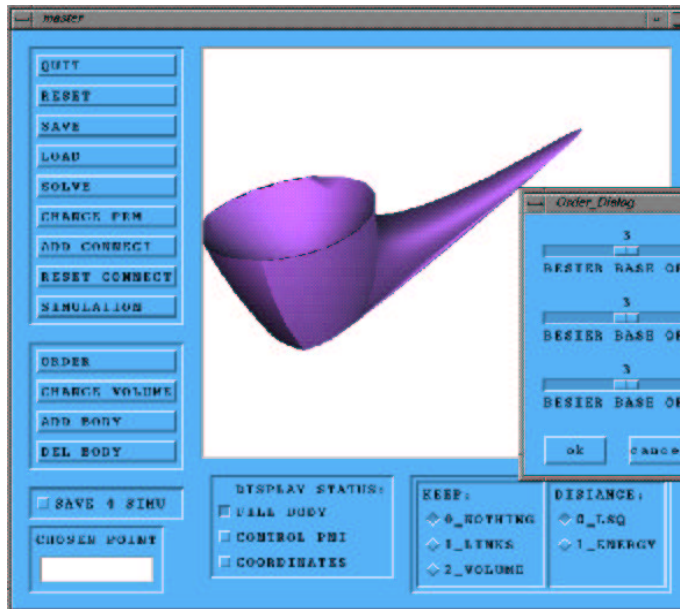


Figure 6: The Motif interface of the system

8 Results

The algorithm was implemented in C under Unix using SGI/GL for graphics and Motif for the user interface. The interface lets the user work with a number of Bézier primitives, the order of each selectable by the user. In the initial state the primitives are displayed as unit cubes (cubes whose volume is 1.) Control points on each primitive can be selected and manipulated in 3-D. We did not implement direct manipulation of boundary surface points since it is immaterial to the problem being tackled. The primitives as whole can be selected as well and manipulated. The interface is shown in Figure 6.

Constraints are inserted via a Motif-based user interface where the type of the constraint is set and then through direct point manipulation the points or surfaces involved are chosen.

There are two methods for object design. In the first method, volume preservation can be turned off during interaction and performed only when arriving at a desired configuration. In the second method, it can be turned on during the

whole interaction process. The first option is necessary since for high orders the performance is not fully interactive.

Due to complexity of computations in the interactive stage we can't satisfy volume and linear constraints simultaneously, so the user has to choose which is preferred.

There are three sets of parameters to the algorithm: parameters that influence volume preservation during interaction while the user drags the mouse, parameters that are for solving volume constraint on leaving the mouse and parameters for global computation when solving all the constraints. Typically, for the interaction mode the iteration limits are lower and the convergence tolerances are larger than for the final mode, for the global computation the tolerances usually do not increase but the iteration limits are larger and the ρ_i used are smaller.

Different sets of parameters do not cause divergence of the algorithm, since during interaction the current configuration is very close to a solution satisfying the volume constraint, and the algorithm needs fewer iterations to reach a solution. The parameter sets can be tuned using a Motif dialog box.

There is dialog box containing a scale widget that defines the desired volume for a chosen primitive. The volume preservation algorithm is performed repeatedly while the scale is dragged.

Tri-quadratic free-form volume design is fully interactive. For a typical movement of a single control point, to reach a final volume tolerance of 10^{-4} and a final distance tolerance of 10^{-3} requires about 15 outer iterations, each of them with 1-2 inner iterations. This takes about 3 seconds on Silicon Graphics workstations with a MIPS R-4000 processor. During interaction it is enough to set both tolerances to 10^{-2} , in which case the solution is completed in real-time.

For a tri-cubic free-form volume, to reach the same tolerances requires about 25-30 outer iterations, each of them with one inner iteration. This takes about 15 seconds. When both tolerances are set to 10^{-2} during interaction the solver takes about 3 seconds. This means that tri-cubic interaction could be done in real-time in the very near future (perhaps even now on an R-4400 processor, which we do not have.)

The running times above are of course dependent on the number of linear constraints and on how far the current configuration is from their solution.

Figures 7-10 show examples of objects designed using the system. The first two are modeled from one primitive with volume 1. The pitcher was modeled from three tri-cubic primitives with C^1 continuity between them (the pitcher body) and one primitive of $4 \times 3 \times 3$ as the handle attached with C^0 continuity to the body. The phone was modeled from three tri-cubic primitives with C^1 continuity conditions between them, its parts were scaled by volume modifications to create the right proportions between them while keeping the desired shape and continuity.

9 Conclusions

We presented an approach for modeling with free-form solid primitives while preserving the volume contained within each primitive and satisfying continuity constraints between the primitives. The algorithm is based on the Uzawa method for non-linear optimization. Careful tuning allowed it to be interactive for tri-quadratic volume elements and almost interactive for tri-cubic elements. The algorithm has several possible applications in computer animation, industrial design and mechanical engineering. It broadens the scope of physics-based geometric modeling.

10 References

- [Arrow58] Arrow, K., Hurwicz, L., Uzawa H., *Studies in Linear and Nonlinear Programming*, Stanford University Press, Stanford, CA, 1958.
- [Aumann92] Aumann, G., Two algorithms for volume-preserving approximations of surfaces of revolution, *Computer-Aided Design*, 24(12):651-657, 1992.
- [Barzel88] Barzel, R., Barr, A.H., A modeling system based on dynamic constraints, *Computer Graphics* 22(4):179-188, 1988 (SIGGRAPH 88).
- [Bercovier81] Bercovier, M., Hasbani, Y., Gilon, Y., Bathe, K.J., A finite element procedure for non-linear incompressible elasticity, invited paper, Symp. on Hybrid and Mixed Methods, S. Atluri, (ed.), Wiley, 1981.
- [Bercovier93] Bercovier, M., Yacoby A Minimization, constraints and composite Bézier curves. Leibniz center for Research in Computer Science, March 1993.
- [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.
- [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).
- [Ciarlet88] Ciarlet, P.G., *Introduction to Numerical Linear Algebra and Optimization*, Cambridge University Press, 1988.
- [Ciarlet83] Ciarlet, P.G., *Lectures on Three-Dimensional Elasticity*, Tata Institute, 1983
- [Coquillart90] Coquillart, S., Extended free form deformation: a sculpturing tool for 3D geometric modeling, *Computer Graphics* 24(4):187-193, 1990 (SIGGRAPH 90).
- [Duvaut76] Duvaut, G., Lions, J.L., *Inequalities in Mechanics and Physics*, Springer-Verlag, 1976

- [Elber94] Elber, G., Symbolic and numeric computation in curve interrogation. Technical Report, Computer Science Department, Technion, Israel, 1994.
- [Fang92] Fang, L., Gossard, D.C., Reconstruction for smooth parametric surfaces from unorganized data points, SPIE Vol. 1830, Curves and Surfaces in Computer Vision and Graphics III, 1992, pp. 226-236.
- [Farin92] Farin, G., Curves and Surfaces for Computer Aided Geometric Design, 3rd. edition, Academic Press, 1992.
- [Farouki85] Farouki, R., Hinds, J., A hierarchy of geometric forms, *IEEE Computer Graphics and Applications*, 5(5):51-78, 1985.
- [Gibson44] Gibson, G.A., Advanced Calculus, Macmillan, London, 1944.
- [Greiner93] Greiner, G., Seidel, H.-P., Curvature continuous blend surfaces. *IFIP Conference on Geometric Modeling in Computer Graphics*, Genova, Italy, June 1993. Published in: Falcidieno, B., Kunii T.L. (eds), *Geometric Modeling in Computer Graphics*, pp. 309-317, Springer, 1993
- [Griessmair89] Griessmair, J., Purgathofer, W., Deformation of solids with tri-variate B-splines, *Eurographics '89*, 137-148, 1989.
- [Hsu92] Hsu, W.M., Hughes, J.F., Kaufman, H., Direct manipulation of free-form deformations, *Computer Graphics* 26(2):177-184, 1992 (SIGGRAPH 92).
- [Hughes87] Hughes87, T.J.R., The Finite Element Method, Prentice-Hall, Engelwood Cliffs, 1987.
- [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.
- [Kallay93] Kallay, M., Constrained optimization in surface design, in: Falcidieno, B. and Kunii, T.L. (eds), *Modeling in Computer Graphics*, pp. 85-94, Springer-Verlag, 1993.
- [Liu87] Liu, D., Algorithms for computing area and volume bounded by Bézier curves and surfaces, *Mathematica Numerica Sinica*, 9:327-336, 1987.
- [Moreton92] Moreton, H., Séquin, C., Functional optimization for fair surface design, *Computer Graphics* 26(2):167-176, 1992 (SIGGRAPH '92).

- [Press88] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., Numerical Recipes in C, Cambridge University Press, 1988.
- [Rappoport94] Rappoport, A., Hel-Or, Y., Werman, M., Interactive design of smooth objects using probabilistic point constraints. *ACM Transactions on Graphics*, 13(2):156-176, 1994.
- [Sederberg86] Sederberg, T.W., Parry, S.R., Free-form deformation of solid geometric models, *Computer Graphics* 20(4):151-160, 1986 (SIGGRAPH '86).
- [Shapiro88] Shapiro Brotman, L., Netravali, A.N., Motion interpolation by optimal control, *Computer Graphics*, 22(4):309-316, 1988 (SIGGRAPH 88).
- [Stewart93] Stewart, N. F., Sufficient conditions for correct topological form in tolerance specification, *Computer-Aided Design*, 25(1):39-48, 1993.
- [Terzopoulos94] Terzopoulos, D., Qin, H., Dynamic NURBS with geometric constraints for interactive sculpting, *ACM Transactions On Graphics*, 13(2):103-136, 1994.
- [Welch92] Welch, W., Witkin, A., Variational surface modeling, *Computer Graphics* 26(2):157-166, 1992 (SIGGRAPH '92).

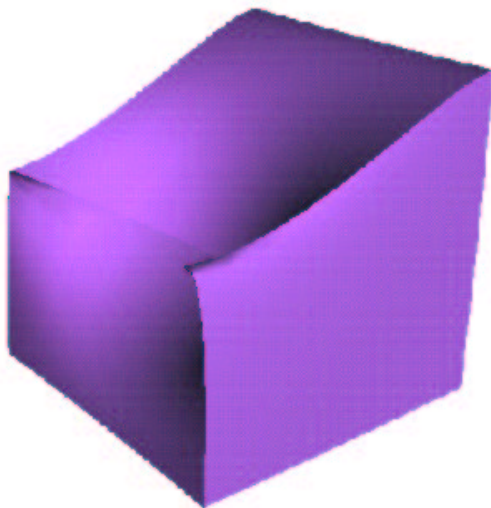


Figure 7: An armchair (tri-cubic.)



Figure 8: An amphora (tri-cubic.)

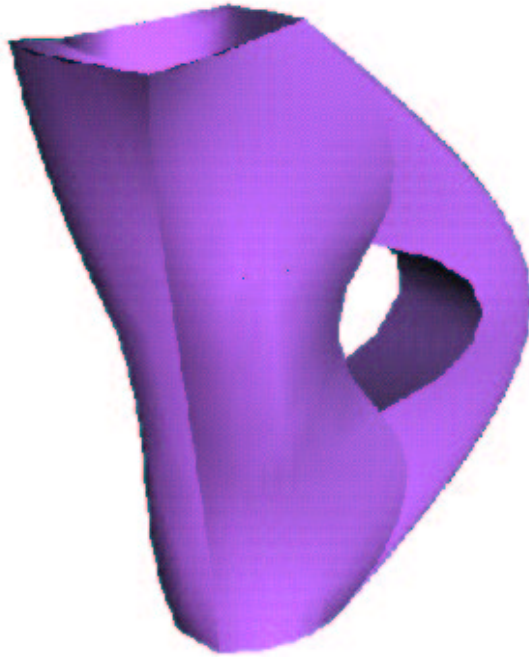


Figure 9: A pitcher (3 tri-cubic and one element of $4 \times 3 \times 3$)

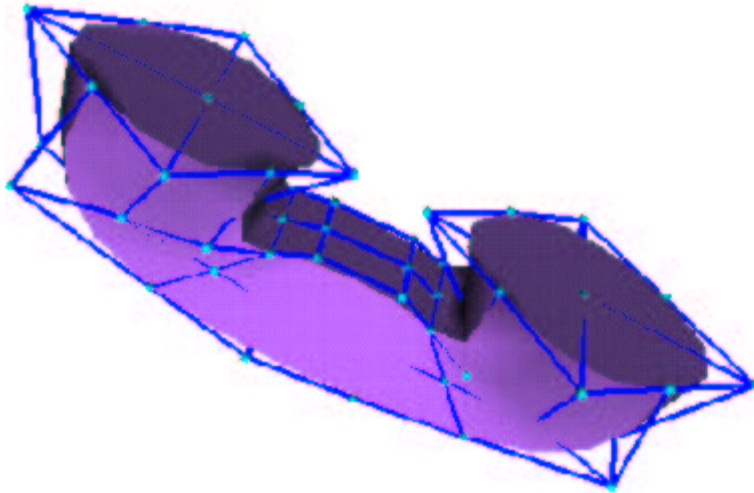


Figure 10: A phone tube (3 tri-cubic.)