Detail Preserving Deformation of B-spline Surfaces with Volume Constraint

Basile Sauvage[1]*, Stefanie Hahmann[2], Georges-Pierre Bonneau[2], Gershon Elber[3]

July 11, 2007

[1] LSIIT, Louis Pasteur University, Strasbourg, France
[2] Laboratoire Jean Kuntzmann, University of Grenoble, France
[3] CGGC, Technion, Haifa, Israel

Abstract

Geometric constraints have proved to be helpful for shape modeling. Moreover, they are efficient aids in controlling deformations and enhancing animation realism.
The present paper addresses the deformation of B-spline surfaces while constraining the volume enclosed by the surface. Both uniform and non-uniform frameworks are considered. The use of level-of-detail (LoD) editing allows the preservation of fine details during coarse deformations of the shape. The key contribution of this paper is the computation of the volume with respect to the appropriate basis for LoD editing: the volume is expressed through all levels of resolution as a trilinear form and recursive formulas are developed to make the computation efficient. The volume constrained is maintained through a minimization process for which we develop closed solutions. Real-time deformations are reached thanks to sparse data structures and efficient algorithms.

keywords: B-spline surfaces, constrained deformation, volume preserving, level-of-detail editing, multiresolution analysis.

1 Introduction

Level-of-detail (LoD) editing of free form curves and surfaces is now established as a valuable modeling tool [FS94, SDD95, SDS96]. It is an attractive application of multiresolution methods, because it allows modification of the overall shape of a geometric model at any scale while automatically preserving all fine details.

*corresponding author: sauvage@dpt-info.u-strasbg.fr;
Phone (+33) 390 24 45 67; fax (+33) 390 24 44 55;
LSIIT, Pole API, Boulevard Sebastien Brant, BP 10413, 67412 ILLKIRCH, France.

In contrast to classical control-point-based editing methods where complex detail preserving deformations need to manipulate a large set of control points, multiresolution methods can achieve the same effect by manipulating only a few control points of some low resolution representation.

Nonetheless, there are application areas, including CAGD and computer animation, where constrained deformations are required. Obviously, the constraints offer additional and finer control over the deformations applied to curves and surfaces. In the past many researchers have explored constrained deformations of free form curves and surfaces. Linear constraints such as position, tangency, orthogonality, and symmetry [BB89, Fow92, Gle92, WW92, FB93] are generally related to direct shape manipulation. These constraints offer the advantage of efficient processing, allowing for interactive manipulation of the free form geometry. Non-linear constraints that are commonly considered are the area [Elb01, HSB05], the volume [RSB95], second order differential constraints such as convexity [KS95, PE98], curve constraints [CW92, GL96, PGL$^+$02], and first and second order fairing functionals such as the bending energy of a thin plate [CG91, GC95, FRSW87, HS92, BH94, Hah98]. Satisfying non-linear constraints, however, requires intense computational effort, so that their use for interactive shape manipulation is generally very limited.

In the context of multiresolution, detail preserving editing with only linear and area constraints has been studied for planar curves. [Elb01] showed that the enclosed area can be viewed as a linear constraint. Incorporated into a multiresolution free form editing environment, the method allows the manipulation of non-uniform B-spline curves at different scales. In [HSB05], a wavelet-based multiresolution formula of the area constraint has been developed for uniform B-spline curves. Both methods are designed for real-time deformations with up to a few thousand control points. Yet, a direct generalization to three-dimensional surfaces is not obvious because the growth of the complexity in the volume computations would make it unusable for any interactive manipulation. The latter, however, is essential for any surface editing method. The volume constraint is important for many applications — for example, when one designs an airplane's fuselage that is assumed to hold a fixed volume. Volume constraint also belongs to Lasseter's principles of animations [Las87], which states that volume preservation enhances the realism when deforming characters, in computer animations. Volume preserving shape deformation has been considered for free form solids [RSB95], FFD [HML99], space deformation [vFTS06], meshes [LCOGL07], and multiresolution meshes [BK03, SHB07]. The volume preservation of smooth spline surfaces in a multiresolution basis has not been considered before.

This paper presents a method for interactive detail-preserving editing of B-spline surfaces with volume constraint. Both uniform and non-uniform B-spline basis functions are addressed. It makes the following three main contributions:

- It generalizes [Elb01, HSB05] to three-dimensional B-spline surfaces with volume preservation. The appearance of the deformation is intuitively controlled by two settings: the *scale* and the *extent* of the deformation. The *scale* defines a threshold: if the details are finer than this threshold, they are preserved during the deformation process. The *extent* defines a portion

of the surface that is involved in the deformation: beyond this extent the surface does not change.

- It develops volume formulas that are consistent with the LoD editing of B-spline surfaces. A so-called *two-scale* basis is used for non-uniform B-spline. A wavelet based multiresolution (MR) basis is used for uniform B-splines. First, the volume is expressed in trilinear form in the coefficients describing the surface with respect to these bases. Then, recursive formulas are developed to efficiently compute the trilinear forms.
- The volume computations and constraint solvings are presented as closed-form solutions, which allow interactive editing. It is supported by a precise analysis of the algorithmic issues.

The paper is organized as follows. Section 2 describes our approach. In Section 3 we give the basics on B-spline surfaces and we specify the interaction with uniform and non-uniform MR representations. Section 4 presents the volume computations and explains how to preserve the volume during the deformation process. Finally, we present some results in Section 5 and conclude in Section 6.

In order to make this paper concise and chronological, both uniform and non-uniform frameworks are treated at the same time. Skipping the sections 3.3 and 4.4 at first reading would lead the reader through the non-uniform setting only.

2 Overview

This paper aims to provide a robust model for real-time MR volume preserving deformation of B-spline surfaces. The problem can be stated as follows.
Given **two pieces of information**:

- An initial surface S enclosing a volume Θ_{ref},
- The displacement of a point on the surface or a control point,

we seek to define a deformation Δ satisfying **two constraints**:

- The volume enclosed by the deformed surface $S + \Delta$ equals Θ_{ref},
- The displacement of the point,

while **two settings** control the appearance of the deformation:

- The *scale* below which the details of the shape are preserved,
- The *extent* defines the portion of the surface involved in the deformation.

This problem is generally under-constrained since there are only one scalar constraint and one vector constraint versus many degrees of freedom (the position of the control points). Among the infinitely many possible solutions, the proposed process tend to minimize the impact of the volume constraint: the key step solves the minimum change of control points.

The solution proposed in this paper has been validated by an interactive editing process illustrated in Figure 1. It states the essential issues. Steps i to $i\nu$ (Figure 1) gather the pre-processing and the adjustment of the settings. Steps ν and νi constitute the core of the deformation process where efficiency is decisive:

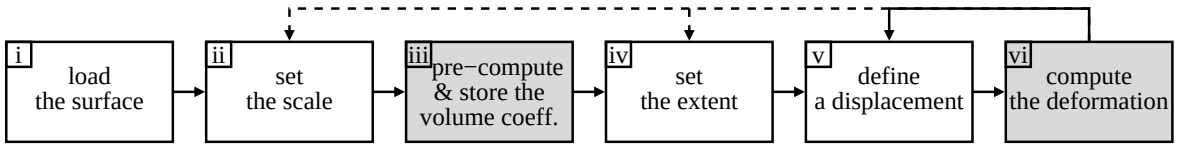

Figure 1: Volume preserving editing process. Expensive steps are shaded.

the backwards arrow $\nu i \rightarrow \nu$ means that these two steps are looped during interactive editing. The computational effort due to the volume preserving is divided between the shaded steps and meets two requirements: The data storage limitation (step *iii*, Figure 1) and the interactivity (step νi, Figure 1). Let us now explain the function of each step:

Step i. The preliminary step consists of loading the surface and building the appropriate data structures.

Step ii. The scale of the deformation is set: the geometric information at finer scales (i.e., the details) will be preserved during the deformation process. Accordingly, the following steps depend on this specified scale. The way of specifying it depends on the surface model. As explained in Section 3, this is done by choosing either some knot sub-sequences (non-uniform B-splines) or an MR level (uniform B-splines).

Step iii. At this stage some coefficients that are used for the volume computation (during step νi) are pre-computed and stored. As shown in Figure 1, this step has to be processed again only if one desires a new deformation scale. Hence, interactivity is not required at this stage and preferably, as much of the whole computation effort as possible should be invested here. This step is discussed in detail in Section 4.

Step iv. In order to control the portion of the surface involved in the editing process, the user sets the extent of the deformation. Our model needs no assumption about the way this is chosen (see discussion Section 4.2).

Step v. The surface is manipulated. Regardless of the surface interaction medium (drag-and-drop, skinning, collision detection), the outcome can be formulated as positional constraints. Denote this interaction outcome as a "displacement": either a control point or a point on the surface is displaced (see Section 3.4).

Step vi. The volume-preserving deformation is computed. It respects the target deformation (step ν) while modifying the surface at the specified scale (step *ii*) and in the specified extent (Step $i\nu$) as well as preserving the volume. We justify and explain, in Section 4, the explicit solution as a constrained minimization problem.

3 Multiresolution deformation for B-spline surfaces

In this section, the framework for the B-spline surfaces' manipulation is presented. In Section 3.1, the basics are reviewed and the notations for patched surfaces are set up. It defines the *fine* or *detailed surfaces*. Since the main issue is to define detail-preserving deformations, two different models are proposed:

- A *multiresolution (MR)* analysis for uniform patches (Section 3.3). This is a uniform B-spline wavelet scheme [SDS96]. As explained in Section 3.3, it provides a unified basis for the surface and the deformation. Then the shape can be intuitively manipulated through an approximating control polyhedron. Moreover, the basis changes using linear filters are efficient. In the ensuing discussion, a superscript e denotes the coarse level of resolution of all the variables.
- A *two-scale* analysis for *non-uniform* patches (Section 3.2). Two basis (*fine* and *coarse*) are defined. This is a simpler but less convenient framework based on knot insertions and removals. This choice is motivated by two factors: non-uniform B-spline wavelet schemes are more complex and less efficient than uniform ones. The advantage of the non-uniform setting lies in its ability to handle more general shapes than the uniform setting, i.e. possibly C^1 discontinuities. In the ensuing discussion, all the variables relative to the coarse basis are denoted with a covering tilde ($\tilde{\ }$).

Eventually, we define how the surface is manipulated (Section 3.4). Following the application we propose to manipulate either the control points or a point on the surface.

In the following bold indices denote double indices, bold letters denote points or vectors in the embedding space $\mathbb{R}^3$, and covering arrows denote high-dimensional vectors.

3.1 Tensor-product B-spline surfaces

A surface S is given by a set $\{S^q\}_q$ of tensor-product B-spline patches. Each patch S^q is defined as an element of a functional space V by

$$S^q(u,v) = \sum_{\mathbf{i}=(0,0)}^{(m_u-1,m_v-1)} \mathbf{p}_\mathbf{i}\ \varphi_\mathbf{i}(u,v)\ , \quad (u,v) \in [\,0,1]^2, \tag{1}$$

where $\mathbf{i} = (i_u, i_v)$ are double indices and where $\varphi_{i_u,i_v}(u,v) = B_{i_u}(u)B_{i_v}(v)$ are the tensor-product B-spline functions of degree $d \times d$ defined over the knot sequences

$$\begin{aligned} \mathcal{U} &= \{u_0, u_1, \ldots, u_{m_u+d}\} \\ \text{and}\ \ \mathcal{V} &= \{v_0, v_1, \ldots, v_{m_v+d}\}. \end{aligned}$$

These basis functions are piecewise polynomial with global C^{d-1} continuity and they span V. With respect to this basis, the coefficients $\mathbf{p}_\mathbf{i} = (x_\mathbf{i}, y_\mathbf{i}, z_\mathbf{i}) \in \mathbb{R}^3$

are called *control points*. Our model remains valid if the degree d differs following the directions u and v but for brevity we assume it does not.

Since we need the surface S to be *a priori* closed, manifold and orientable, we assume the patches are joined along their boundaries. In order to easily manage C^0 continuity between patches, we add the following boundary-interpolating conditions that fix (clamp) the so-called boundary-knots:

$$\begin{aligned} u_0 = u_1 = \ldots = u_d \;&=\; 0 \;=\; v_0 = \ldots = v_d \\ \text{and}\;\; u_{m_u} = u_{m_u+1} = \ldots = u_{m_u+d} \;&=\; 1 \;=\; v_{m_v} = \ldots = v_{m_v+d}. \end{aligned}$$

This implies that the boundary curves only depend on the boundary control points (e.g., $S^q(0,v)$ is completely defined by $(p_{0,i_v})_{0\leq i_v<m_v}$). Moreover, we assume that the degrees and the knot sequences of both patches correspond along their common boundary. Hence, the C^0 continuity between two neighboring patches is ensured by only equating the boundary control points.

3.2 Two-scale editing for non-uniform B-splines

The main objective is to define an arbitrary coarse deformation Δ for some patch S^q in order to compute a deformed surface $S^q + \Delta$. The coarser Δ is, the larger is the *scale* of the deformation. "Two-scale editing" means that a "fine-scale" basis is used for defining the surface S^q while a "coarse-scale" basis is used for defining the deformation Δ. The coarse B-spline basis is defined from the fine one by removing knots in the sequences $\mathcal{U}$ and $\mathcal{V}$.

Following [Elb01], we define Δ as an element of some approximation space $\widetilde{V} \subset V$. It is a spline space based on knot sub-sequences $\widetilde{\mathcal{U}} = \{\widetilde{u}_0, \ldots, \widetilde{u}_{\widetilde{m}_u+d}\} \subset \mathcal{U}$ and $\widetilde{\mathcal{V}} = \{\widetilde{v}_0, \ldots, \widetilde{v}_{\widetilde{m}_v+d}\} \subset \mathcal{V}$, such that:

- The boundary knots are preserved. Thus, the boundary-interpolating property is preserved.
- Some interior knots are removed in order to enlarge the scale of the deformation.

Hence, $\widetilde{V}$ is spanned by a set of coarse B-splines $\tilde{\varphi}_{\mathbf{j}}$ of degree $d \times d$. The deformation $\Delta \in \widetilde{V}$ is written as

$$\Delta(u,v) = \sum_{\mathbf{j}=(0,0)}^{(\widetilde{m}_u-1,\widetilde{m}_v-1)} \widetilde{\boldsymbol{\delta}}_{\mathbf{j}}\; \widetilde{\varphi}_{\mathbf{j}}(u,v) \quad \text{with } \widetilde{\boldsymbol{\delta}}_{\mathbf{j}} \in \mathbb{R}^3. \tag{2}$$

By selecting the knots to be removed, the user chooses the deformation scale (Step *ii*, Figure 1): the more knots s/he removes, the larger is the support of $\tilde{\varphi}_{\mathbf{j}}$, and the coarser is the deformation. Note that

- The *coarse basis* is used to *define* the deformation – i.e., $(\widetilde{\boldsymbol{\delta}}_{\mathbf{j}})_{\mathbf{j}}$ are the unknowns, but
- The *fine basis* is required to *apply* the deformation to the surface – (i.e., to compute $S^q + \Delta$) since $S^q \notin \widetilde{V}$.

Therefore, Δ is expressed in the fine basis as

$$\Delta(u,v) = \sum_{\mathbf{i}} \boldsymbol{\delta}_{\mathbf{i}} \; \varphi_{\mathbf{i}}(u,v) \quad \text{with } (0,0) \leq \mathbf{i} \leq (m_u - 1, m_v - 1),$$

where $\boldsymbol{\delta}_{\mathbf{i}} \in \mathbb{R}^3$ are computed from the $\widetilde{\boldsymbol{\delta}}_{\mathbf{j}}$, using knot-insertions [Far01].

3.3 Multiresolution editing for uniform B-splines

Let S^q be a tensor-product patch as defined by Equation (1), with uniform knot sequences. The uniform framework is nothing but a particular case of the non-uniform one. Hence, the coarse deformation Δ could be defined the same way. Uniform B-spline basis functions, however, are more amendable to wavelet-based MR schemes [FS94]. Therefore, we choose to define Δ in a B-spline wavelet basis because this approach is both more efficient and more convenient:

- Translation-invariant and level-invariant MR filters make the basis changes easy and efficient.
- S^q and Δ lie in the same space. Hence, the deformed surface $S^q + \Delta$ is instantly computed in the MR basis.
- MR analysis provides a meaningful coarse approximation of the shape (namely the control polyhedron, see Figure 6 left), whose manipulation is convenient.
- A single parameter (the MR level) controls the scale of the deformation.

This section is not intended to give a complete presentation of MR surfaces. One-dimensional B-spline wavelets are detailed in [FS94, SDS96] and a tensor-product example is shown in [SDS96].

Building an MR surface analysis with n levels requires the knot sequences to be uniform and the number of intervals to be multiples of 2^n as well:

$$\begin{aligned} & u_{i+1} - u_i = \tfrac{1}{p_u 2^n} \;, \quad d \leq i < p_u 2^n + d = m_u \;, \\ \text{and} \quad & v_{i+1} - v_i = \tfrac{1}{p_v 2^n} \;, \quad d \leq i < p_v 2^n + d = m_v, \end{aligned}$$

where p_u and p_v are arbitrary integer values.

The MR analysis is based on a sequence of nested *approximation spaces* $V^0 \subset V^1 \subset \cdots \subset V^n$. Each space V^j is spanned by some *scaling functions* $(\varphi^j_{\mathbf{i}})_{\mathbf{i}}$ with respect to which the coefficients $\mathbf{p}^j_{\mathbf{i}} \in \mathbb{R}^3$ are called *control points*. For each multiresolution level j three *detail spaces* W_0^{j-1}, W_1^{j-1} and W_2^{j-1} make the complement of V^{j-1} in V^j:

$$V^j = V^{j-1} \oplus W_0^{j-1} \oplus W_1^{j-1} \oplus W_2^{j-1}.$$

The spaces are spanned by the *wavelets* $\psi^j_{0,\mathbf{i}}$, $\psi^j_{1,\mathbf{i}}$ and $\psi^j_{2,\mathbf{i}}$. The coefficients $\mathbf{d}^j_{0,\mathbf{i}}$, $\mathbf{d}^j_{1,\mathbf{i}}$ and $\mathbf{d}^j_{2,\mathbf{i}}$ are called *detail coefficients*. Notations and properties are collected in Table 1.

Space	Dimension	Basis functions	Coefficients
V^j	$(p_u 2^j + d) \times (p_v 2^j + d)$	$\left(\varphi^j_{\mathbf{i}}(u,v)\right)_{\mathbf{i}}$	$\left(\mathbf{p}^j_{\mathbf{i}}\right)_{\mathbf{i}} = \vec{\mathbf{p}}^j$
W^j_0	$p_u 2^j \times (p_v 2^j + d)$	$\left(\psi^j_{0,\mathbf{i}}(u,v)\right)_{\mathbf{i}}$	$\left(\mathbf{d}^j_{0,\mathbf{i}}\right)_{\mathbf{i}} = \vec{\mathbf{d}}^j_0$
W^j_1	$(p_u 2^j + d) \times p_v 2^j$	$\left(\psi^j_{1,\mathbf{i}}(u,v)\right)_{\mathbf{i}}$	$\left(\mathbf{d}^j_{1,\mathbf{i}}\right)_{\mathbf{i}} = \vec{\mathbf{d}}^j_1$
W^j_2	$p_u 2^j \times p_v 2^j$	$\left(\psi^j_{2,\mathbf{i}}(u,v)\right)_{\mathbf{i}}$	$\left(\mathbf{d}^j_{2,\mathbf{i}}\right)_{\mathbf{i}} = \vec{\mathbf{d}}^j_2$

Table 1: Multiresolution spaces.

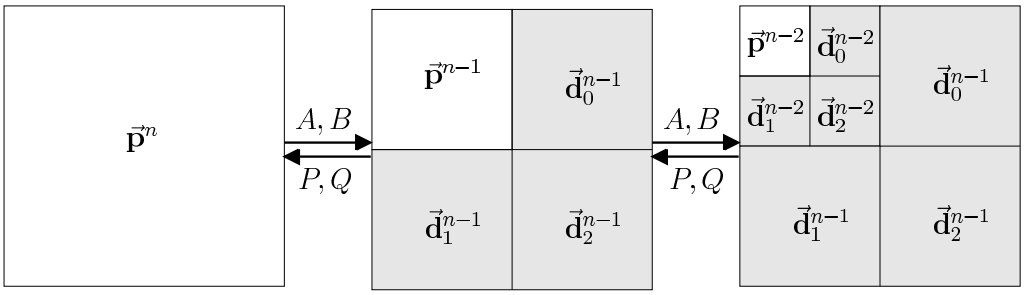

Figure 2: Tensor-product filter bank. From left to right: decomposition process (using filters A and B); From right to left: reconstruction process (using filters P and Q).

The fine space $V^n = V$ contains the detailed patch S^q. Its basis functions $\varphi^n_{\mathbf{i}} = \varphi_{\mathbf{i}}$ are the tensor-product B-spline functions and the coefficients $\mathbf{p}^n_{\mathbf{i}} = \mathbf{p}_{\mathbf{i}}$ are the B-spline control points. Given the fine control points $\mathbf{p}^n_{\mathbf{i}}$ defining $S^q(u,v) = \sum \mathbf{p}^n_{\mathbf{i}} \varphi^n_{\mathbf{i}}(u,v)$, the patch can be written in the MR basis at any level e, $0 \le e \le n$, as follows:

$$S^q(u,v) = \sum_{\mathbf{i}} \mathbf{p}^e_{\mathbf{i}}\ \varphi^e_{\mathbf{i}}(u,v) + \sum_{j=e}^{n-1} \left(\sum_{\mathbf{i}} \mathbf{d}^j_{0,\mathbf{i}}\, \psi^j_{0,\mathbf{i}}(u,v) + \sum_{\mathbf{i}} \mathbf{d}^j_{1,\mathbf{i}}\, \psi^j_{1,\mathbf{i}}(u,v) + \sum_{\mathbf{i}} \mathbf{d}^j_{2,\mathbf{i}}\, \psi^j_{2,\mathbf{i}}(u,v) \right). \quad (3)$$

The coarse control points $\mathbf{p}^e_{\mathbf{i}}$ encode an approximation of the shape (low frequency). They are the vertices of a coarse polyhedron controlling the surface (see Figure 6 left). The details $\mathbf{d}^j_{k,\mathbf{i}}$ encode higher frequencies: the larger j, the higher the frequency and the finer the details.

The main advantage with regard to our objectives is the ability to manipulate the global shape at an arbitrary level e through the control points $\vec{\mathbf{p}}^e$ while preserving the finer details: e sets the scale of the deformation (Step *ii*, Figure 1) – i.e., the lower e is, the coarser the deformation. The deformation Δ lies in the approximation space V^e and is defined by

$$\Delta(u,v) = \sum_{\mathbf{i}} \boldsymbol{\delta}^e_{\mathbf{i}} \varphi^e_{\mathbf{i}}(u,v). \quad (4)$$

Hence, computing $S^q + \Delta$ amounts to adding $\vec{\mathbf{p}}^e + \vec{\boldsymbol{\delta}}^e$, both of them being vectors of coefficients in $\mathbb{R}^3$ for the scaling functions in V^e (see equations (3) and (4)).

The MR coefficients in (3) are computed thanks to the filterbank algorithm [Mal99]. It is a linear, recursive and invertible process computing the change of basis illustrated in Figure 2. It is based on two decomposition filters A^j and B^j, and two reconstruction filters P^j and Q^j for each level j, which are applied in a tensor-product manner on the matrices of coefficients. Given these filters in the shape of sparse matrices, the decomposition step at level j consists of computing the control points and the details at level $j-1$ from the control points at level j:

$$\begin{aligned}
\vec{\mathbf{p}}^{j-1} &= A^j\,\vec{\mathbf{p}}^j\,(A^j)^T,\\
\vec{\mathbf{d}}_0^{j-1} &= B^j\,\vec{\mathbf{p}}^j\,(A^j)^T,\\
\vec{\mathbf{d}}_1^{j-1} &= A^j\,\vec{\mathbf{p}}^j\,(B^j)^T,\\
\vec{\mathbf{d}}_2^{j-1} &= B^j\,\vec{\mathbf{p}}^j\,(B^j)^T;
\end{aligned}$$

the reverse process is called reconstruction step:

$$\vec{\mathbf{p}}^j = P^j\,\vec{\mathbf{p}}^{j-1}\,(P^j)^T \;+\; Q^j\,\vec{\mathbf{d}}_0^{j-1}\,(P^j)^T \;+\; P^j\,\vec{\mathbf{d}}_1^{j-1}\,(Q^j)^T \;+\; Q^j\,\vec{\mathbf{d}}_2^{j-1}\,(Q^j)^T.$$

Note that the deformation (4) could be exactly computed in the non-uniform framework by repeatedly removing every second knot from $\mathcal{U}$ (once for each level of resolution from n to e). Then $\widetilde{V} = V^e$ and equations (2) and (4) are equivalent. This shows that the non-uniform model is more general. In compensation the uniform model provides the convenient decomposition (3) and the ensuing control polyhedron.

3.4 Surface manipulation

The present method is independent of the surface manipulating tool. The deformation may be driven either by direct user interaction in a modeler (e.g., the mouse's displacement in a drag-and-drop operation), by skinning, or via collision detection, etc. Therefore, we propose to formalize it by fixing a displacement $\mathbf{t}$ for either a control-point or a point on the surface.

A simple surface manipulation scheme is the direct displacement of control point $\bar{\boldsymbol{\imath}}$ and is equivalent to fixing one coefficient of the deformation:

- set $\widetilde{\boldsymbol{\delta}}_{\bar{\boldsymbol{\imath}}} = \mathbf{t}$ in the non-uniform framework;
- set $\boldsymbol{\delta}^e_{\bar{\boldsymbol{\imath}}} = \mathbf{t}$ in the uniform framework.

This simple scheme is especially convenient in the uniform framework because the surface may be manipulated through the control polyhedron whose vertices are $\{\mathbf{p}^e_{\mathbf{i}}\}$. Hence, $\boldsymbol{\delta}^e_{\bar{\boldsymbol{\imath}}}$ is meaningful: it corresponds to the displacement of the vertex $\bar{\boldsymbol{\imath}}$.

Alternatively, displacing a point on the surface with parameter $(\bar{u}, \bar{v})$, i.e. moving $S(\bar{u}, \bar{v})$ to a new location $S(\bar{u}, \bar{v}) + \Delta(\bar{u}, \bar{v})$, is equivalent to fixing $\Delta(\bar{u}, \bar{v})$ to a given value $\mathbf{t}$ (e.g. obtained through a click-and-drag operation):

- Constrain $\sum_{\mathbf{j}} \widetilde{\boldsymbol{\delta}}_{\mathbf{j}} \ \widetilde{\varphi}_{\mathbf{j}}(\bar{u}, \bar{v}) = \mathbf{t}$ in the non-uniform framework;
- Constrain $\sum_{\mathbf{j}} \boldsymbol{\delta}^e_{\mathbf{j}} \ \varphi^e_{\mathbf{j}}(\bar{u}, \bar{v}) = \mathbf{t}$ in the uniform framework.

The advantage of this alternative is found in the direct manipulation of the surface. It is more intuitive in the non-uniform framework since there is no control polyhedron. Moreover it allows the enforcement of exact positional constraints on the surface.

4 Volume preserving deformation

This section details the key contributions of our method to maintaining the volume while preserving the details during the surface deformation process. The first contribution is formulas for volume preservation that are consistent with the LoD editing frameworks (see Section 3). The second contribution is efficient algorithms for these formulas and for constraining the surface during the deformation process.

In Section 4.1, the computation of the enclosed volume as a trilinear form is presented. In section 4.2, we define volume-preservation as a constrained minimization problem. In Sections 4.3 and 4.4, efficient solutions are proposed (for one single patch) and the algorithmic issues are discussed. Both surface models (non-uniform and uniform) and both displacements (either a control point or a point on the surface) are considered. Eventually, in Section 4.5, we extend the results to several patches with C^0 continuity.

4.1 Volume enclosed by a fine surface

The signed volume enclosed by a parametric surface may be written as an integral over the domain [GOMP98, Elb01]. The (signed) volume enclosed by a patched surface S equals the sum over the patches:

$$\Theta\left(S\right) = \sum_{\text{patches } q} \Theta\left(S^q\right), \tag{5}$$

where

$$\Theta(S^q) = \int_{v=0}^{1} \int_{u=0}^{1} z \left(\frac{\partial x}{\partial u} \frac{\partial y}{\partial v} - \frac{\partial x}{\partial v} \frac{\partial y}{\partial u} \right) du\, dv, \tag{6}$$

for each patch $S^q(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$. By substituting Formula (1) into (6) the volume becomes a trilinear form with respect to the control points' coordinates:

$$\Theta(S^q) = \sum_{\mathbf{i},\mathbf{j},\mathbf{k}} x_{\mathbf{i}}\, y_{\mathbf{j}}\, z_{\mathbf{k}} \iint_{[0,1]^2} \varphi_{\mathbf{k}} \left(\frac{\partial \varphi_{\mathbf{i}}}{\partial u} \frac{\partial \varphi_{\mathbf{j}}}{\partial v} - \frac{\partial \varphi_{\mathbf{i}}}{\partial v} \frac{\partial \varphi_{\mathbf{j}}}{\partial u} \right) du\, dv \tag{7}$$

where the sum runs over $(0,0) \leq \mathbf{i}, \mathbf{j}, \mathbf{k} \leq (m_u - 1, m_v - 1)$. The integrals cannot be computed on-the-fly (i.e., during step νi, Figure 1), for efficiency reasons; nor can they be pre-computed and stored because of the expected memory size

($m_u^3 m_v^3$ integrals per patch). Hence, we propose to separate the parameters u and v:

$$\iint \varphi_{\mathbf{k}} \left(\frac{\partial \varphi_{\mathbf{i}}}{\partial u} \frac{\partial \varphi_{\mathbf{j}}}{\partial v} - \frac{\partial \varphi_{\mathbf{i}}}{\partial v} \frac{\partial \varphi_{\mathbf{j}}}{\partial u} \right) du\, dv = \theta^u_{j_u k_u i_u} \theta^v_{i_v k_v j_v} - \theta^u_{i_u k_u j_u} \theta^v_{j_v k_v i_v}, \qquad (8)$$

where

$$\theta^u_{ijk} = \int_0^1 B_i(u)\, B_j(u)\, B'_k(u)\, du, \qquad (9)$$

$$\theta^v_{ijk} = \int_0^1 B_i(v)\, B_j(v)\, B'_k(v)\, dv\ . \qquad (10)$$

It is now possible to pre-compute and store m_u^3 values θ^u_{ijk} and m_v^3 values θ^v_{ijk} during the initialization step i (Figure 1). Moreover, the B-spline functions have local support (containing $d+1$ intervals). Hence, these values are stored in sparse data structures with size $O(m_u d^2)$ and $O(m_v d^2)$. The computation through (9) and (10) may be either analytical or via a numerical approximation.

4.2 Problem statement

We now propose a mathematical statement of the problem sketched in Section 2: how to define a relevant deformation that fits the constraints and the settings?

Since we have only **two constraints**, displacement and volume, we propose to solve a constrained minimization problem. We minimize the L_2 norm of the deformation's coefficients ($\tilde{\boldsymbol{\delta}}_{\mathbf{i}}$ or $\boldsymbol{\delta}^e_{\mathbf{i}}$). The displacement constraint was discussed in Section 3.4. The volume constraint is $\Theta(S^q + \Delta) = \Theta(S^q)$. The first challenge is to write this constraint as a function of the free variables ($\tilde{\boldsymbol{\delta}}_{\mathbf{i}}$ or $\boldsymbol{\delta}^e_{\mathbf{i}}$), i.e., finding formulas that fit, respectively, the two-scale basis or the MR basis. The second challenge is to build an algorithm that satisfies both the memory and interactivity requirements. In other words, the pre-computation (Step *iii*, Figure 1) and the on-line processing (Step νi, Figure 1) must be carefully balanced.

Above, we mentioned **two settings**. *Scale* control was discussed in Sections 3.2 and 3.3. In order to control the *extent* of the deformation, the user chooses a set $\mathcal{F}$ of "free" control points that will be modified in order to preserve the volume. In our editing tool, $\mathcal{F}$ is a neighborhood of the displaced point whose size is specified by the user.

While the setting of this "active domain" that can undergo deformation is orthogonal to the work presented here, it is a crucial step in any MR/LOD editing environment. The specification of this active domain should be interactive while hiding the internal representation (i.e. parametric domain) as much as possible from the end user. One such possibility will allow the user to sketch a closed region on the surface, region that will be employed internally, in the parametric domain of the surface, to identify all control points (i.e. degrees of freedom) that affects this region.

Volume is a non-linear constraint. For efficiency purposes and in order to avoid the use of non-linear optimizations, linearization techniques will be used but without loosening mathematical exactness. Following [Elb01] the volume

constraint is linearized by separating the deformation according to the x, y and z-axes. This makes the processing more efficient. In the following sections we detail only the x-axis deformation. Symmetric formulas are obtained for the y and z-axes.

An alternative linearization of the volume constraint is a first-order Taylor expansion, as proposed in [HSB05]. One single minimization is needed (instead of three in the present method) but the volume is just approximated. Both methods differ in some specific geometric configurations. Most of our tests, however, show very similar results.

4.3 Non-uniform B-splines

4.3.1 Volume constraint in the two-scale basis

In order to define Δ such that the volume is preserved we regard the volume $\Theta(S^q + \Delta)$ as a function of the variables $\widetilde{\boldsymbol{\delta}}_{\mathbf{i}}$. Since the function $\Theta(.)$ is trilinear, separating the deformation by axes makes it linear in each axis and hence much easier to manage. Consider the case $\widetilde{\boldsymbol{\delta}}_{\mathbf{i}} = (\widetilde{\boldsymbol{\delta}} x_{\mathbf{i}}, 0, 0)$ and $\mathbf{t} = (\mathbf{t}x, 0, 0)$. Other axes are deduced by symmetry. By substituting Formulas (1) and (2) into Equation (6), we get

$$\Theta(S^q + \Delta) = \Theta(S^q) + \sum_{\mathbf{i}} \widetilde{\Theta}_{YZ}(\mathbf{i})\, \widetilde{\boldsymbol{\delta}} x_{\mathbf{i}}, \qquad (0,0) \leq \forall\ \mathbf{i} \leq (\widetilde{m}_u - 1, \widetilde{m}_v - 1),$$

where

$$\widetilde{\Theta}_{YZ}(\mathbf{i}) = \sum_{\mathbf{j},\mathbf{k}} y_{\mathbf{j}} z_{\mathbf{k}} \iint \varphi_{\mathbf{k}} \Big(\frac{\partial \widetilde{\varphi}_{\mathbf{i}}}{\partial u} \frac{\partial \varphi_{\mathbf{j}}}{\partial v} - \frac{\partial \widetilde{\varphi}_{\mathbf{i}}}{\partial v} \frac{\partial \varphi_{\mathbf{j}}}{\partial u} \Big)\, du\, dv. \tag{11}$$

Then, the volume constraint $\Theta(S^q + \Delta) = \Theta(S^q)$ for the x-axis becomes

$$\sum_{\mathbf{i}} \widetilde{\Theta}_{YZ}(\mathbf{i})\, \widetilde{\boldsymbol{\delta}} x_{\mathbf{i}} = 0, \tag{12}$$

where the sum is in fact limited to the specified *extent* i.e. it runs over $\mathbf{i} \in \mathcal{F}$.

Thanks to the linearity of the integral and the sum operators, one can use the pre-computed θ^u and θ^v to compute the coefficients $\widetilde{\Theta}_{YZ}(\mathbf{i})$: Firstly compute

$$\Theta_{YZ}(\mathbf{i}) = \sum_{\mathbf{j},\mathbf{k}} y_{\mathbf{j}} z_{\mathbf{k}} \left(\theta^u_{j_u k_u i_u} \theta^v_{i_v k_v j_v} - \theta^u_{i_u k_u j_u} \theta^v_{j_v k_v i_v} \right) \tag{13}$$

for $(0,0) \leq \mathbf{i} \leq (m_u - 1, m_v - 1)$, from which the $\widetilde{\Theta}_{YZ}(\mathbf{i})$ are then computed, using a *knot-removal* algorithm based on Boehm's recurrence. The details are given in Appendix A.

4.3.2 Solving the minimization problem

All the terms of the minimization problem are now known. One has to solve it for both manipulation models discussed in Section 3.4: Displacing a control point or displacing a point on the surface. In both cases a closed form of the resulting deformation $\{\widetilde{\boldsymbol{\delta}} x_{\mathbf{i}}\}$ is derived.

Displacing a control point

When displacing the point $\bar{\boldsymbol{\imath}}$, one fixes $\widetilde{\boldsymbol{\delta}}x_{\bar{\boldsymbol{\imath}}} = \mathbf{t}x$. Excluding $\bar{\boldsymbol{\imath}}$ from the set $\mathcal{F}$ of free control points, the volume constraint (12) becomes:

$$\widetilde{\Theta}_{YZ}(\bar{\boldsymbol{\imath}})\,\mathbf{t}x + \sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\Theta}_{YZ}(\mathbf{i})\,\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = 0.$$

Hence, the minimization is

$$\min_{\{\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\}} \sum_{\mathbf{i}\in\mathcal{F}} |\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}|^2 \quad \text{subject to} \quad \widetilde{\Theta}_{YZ}(\bar{\boldsymbol{\imath}})\,\mathbf{t}x + \sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\Theta}_{YZ}(\mathbf{i})\,\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = 0.$$

Using the Lagrange multiplier λ [Cia88], the minimization problem is equivalent to solving the unconstrained min-max problem of

$$\max_{\lambda}\min_{\{\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\}} \sum_{\mathbf{i}\in\mathcal{F}} |\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}|^2 + \lambda\left(\widetilde{\Theta}_{YZ}(\bar{\boldsymbol{\imath}})\,\mathbf{t}x + \sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\Theta}_{YZ}(\mathbf{i})\,\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\right)$$

$$\Leftrightarrow \quad \overrightarrow{\bigtriangledown}\left(\sum_{\mathbf{i}\in\mathcal{F}} |\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}|^2 + \lambda\left(\widetilde{\Theta}_{YZ}(\bar{\boldsymbol{\imath}})\,\mathbf{t}x + \sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\Theta}_{YZ}(\mathbf{i})\,\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\right)\right) = \vec{0}$$

$$\Leftrightarrow \quad \begin{cases} 2\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} + \lambda\widetilde{\Theta}_{YZ}(\mathbf{i}) = 0, & \forall \mathbf{i}\in\mathcal{F}, \\ \widetilde{\Theta}_{YZ}(\bar{\boldsymbol{\imath}})\,\mathbf{t}x + \sum \widetilde{\Theta}_{YZ}(\mathbf{i})\,\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = 0. \end{cases}$$

Having

$$\sum_{\mathbf{j}\in\mathcal{F}} |\widetilde{\Theta}_{YZ}(\mathbf{j})|^2 \neq 0, \tag{14}$$

the system has a full-rank and the unique solution is

$$\lambda = \frac{2\,\mathbf{t}x\ \widetilde{\Theta}_{YZ}(\bar{\boldsymbol{\imath}})}{\sum_{\mathbf{j}\in\mathcal{F}} |\widetilde{\Theta}_{YZ}(\mathbf{j})|^2},$$

and

$$\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = \frac{-\mathbf{t}x\ \widetilde{\Theta}_{YZ}(\bar{\boldsymbol{\imath}})}{\sum_{\mathbf{j}\in\mathcal{F}} |\widetilde{\Theta}_{YZ}(\mathbf{j})|^2}\ \widetilde{\Theta}_{YZ}(\mathbf{i}), \qquad \forall \mathbf{i}\in\mathcal{F}. \tag{15}$$

Equation (14) approaches zero in the singular case where the set of support functions, in $\mathcal{F}$, all provide vanishing volumetric values to the total volume and hence can clearly have no affect on the final shape's volume, not to say preserve it.

Displacing a point on the surface

When displacing a point with parameter $(\bar{u}, \bar{v})$, the minimization is subject to the volume (12) and position constraints (see Section 3.4):

$$\min_{\{\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\}} \sum_{\mathbf{i}\in\mathcal{F}} |\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}|^2 \quad \text{subject to} \quad \begin{cases} \sum \widetilde{\Theta}_{YZ}(\mathbf{i})\,\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = 0 \\ \sum \widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v})\,\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = \mathbf{t}x \end{cases}.$$

Using, once again, the Lagrange multipliers (λ and μ), the problem is reduced to

$$\max_{\lambda,\mu} \min_{\{\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\}} \sum_{\mathbf{i}\in\mathcal{F}} |\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}|^2 + \lambda \sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\Theta}_{YZ}(\mathbf{i})\, \widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} + \mu \left(\sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v})\, \widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} - \mathbf{t}x \right)$$

$$\Leftrightarrow \quad \overrightarrow{\bigtriangledown} \left(\sum_{\mathbf{i}\in\mathcal{F}} |\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}|^2 + \lambda \sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\Theta}_{YZ}(\mathbf{i})\, \widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} + \mu \left(\sum_{\mathbf{i}\in\mathcal{F}} \widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v})\, \widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} - \mathbf{t}x \right) \right) = \vec{0}$$

$$\Leftrightarrow \quad \begin{cases} 2\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} + \lambda\widetilde{\Theta}_{YZ}(\mathbf{i}) + \mu\widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v}) = 0, & \forall \mathbf{i} \in \mathcal{F}, \\ \sum \widetilde{\Theta}_{YZ}(\mathbf{i})\, \widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = 0, \\ \sum \widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v})\, \widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = \mathbf{t}x. \end{cases}$$

Let

$$\alpha = \sum_{\mathbf{j}\in\mathcal{F}} |\widetilde{\varphi}_{\mathbf{j}}(\bar{u},\bar{v})|^2 \sum_{\mathbf{j}\in\mathcal{F}} |\widetilde{\Theta}_{YZ}(\mathbf{j})|^2 - \left(\sum_{\mathbf{j}\in\mathcal{F}} \widetilde{\varphi}_{\mathbf{j}}(\bar{u},\bar{v})\, \widetilde{\Theta}_{YZ}(\mathbf{j}) \right)^2 .$$

The system has a unique full-rank solution when $\alpha \neq 0$, a solution that is equal to

$$\lambda = \frac{2\,\mathbf{t}x}{\alpha} \sum_{\mathbf{j}\in\mathcal{F}} \widetilde{\varphi}_{\mathbf{j}}(\bar{u},\bar{v})\, \widetilde{\Theta}_{YZ}(\mathbf{j}) \ ,$$

$$\mu = \frac{-2\,\mathbf{t}x}{\alpha} \sum_{\mathbf{j}\in\mathcal{F}} |\widetilde{\Theta}_{YZ}(\mathbf{j})|^2 \ ,$$

and

$$\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}} = \frac{-\lambda}{2}\widetilde{\Theta}_{YZ}(\mathbf{i}) - \frac{\mu}{2}\widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v}), \qquad \forall \mathbf{i} \in \mathcal{F}. \tag{16}$$

α can vanish in three different case:

1. when $\widetilde{\varphi}_{\mathbf{j}}(\bar{u},\bar{v})$ vanish for all $\mathbf{j}$,
2. when $\widetilde{\Theta}_{YZ}(\mathbf{j})$ vanish for all $\mathbf{j}$, or
3. when $\widetilde{\varphi}_{\mathbf{j}}(\bar{u},\bar{v})$, $\forall \mathbf{j}$ and $\widetilde{\Theta}_{YZ}(\mathbf{j})$, $\forall \mathbf{j}$, seen as two vectors of size $|\mathcal{F}|$, are colinear vectors.

In case 1, clearly the selection of the functions in $\mathcal{F}$ cannot support the displacement constraint. Similarly, in case 2, the selection of the functions in $\mathcal{F}$ cannot support the volume constraint. In case 3, these two sets of support functions are linearly dependent and hence cannot simultaneously support the two linearly independent constraints of the displacement and volume.

4.3.3 Algorithmic issues

Here we list the successive computations done during interactive Step νi (Figure 1) and discuss the complexity. We go into details for the x-axis only. The complete algorithm successively treats the x, y and z coordinates. The costs are expressed with respect to the degree d of the surface, the total number $m_u m_v$ of control points and the number $\#\mathcal{F}$ of free control points.

- Computing $\{\Theta_{YZ}(\mathbf{i})\}$ costs $O(m_u m_v d^2)$ using Formula (13). This low cost is due to the sparsity of θ^u and θ^v.
- Computing $\{\widetilde{\Theta}_{YZ}(\mathbf{i})\}$ from $\{\Theta_{YZ}(\mathbf{i})\}$ costs $O(m_u m_v d)$ using the knots-removal algorithm (see Appendix A). This is due to the efficiency of the knots-removal step.
- Computing $\{\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\}_{\mathbf{i}\in\mathcal{F}}$ costs $O(\#\mathcal{F})$ through Formula (15) or (16). Having closed forms (15) and (16) solving the minimization problem is decisive here for efficiency.
- Computing $\{\boldsymbol{\delta}x_{\mathbf{i}}\}$ from $\{\widetilde{\boldsymbol{\delta}}x_{\mathbf{i}}\}$ using the knots-insertion algorithm costs $O(m_u m_v d)$.
- Adding the deformation to the surface $x_{\mathbf{i}} \leftarrow x_{\mathbf{i}} + \boldsymbol{\delta}x_{\mathbf{i}}$ costs $O(m_u m_v)$.

Moreover, $\#\mathcal{F} \leq m_u m_v$ and we can assume that degree d is bounded (in most cases $d \leq 5$ is sufficient). Hence, the global complexity of Step νi (Figure 1) is $O(m_u m_v)$. In other words, it is linear with respect to the number of control points. This is essential for interactivity. In a typical drag-and-drop operation every small displacement of the mouse (step ν, Figure 1) implies the computation of a new volume preserving deformation (Step νi, Figure 1). Three key points allow the linear cost during the interaction: sparse data structures, closed forms solution to the minimization problems, and the ability to conduct the pre-computations.

Pre-computing $\{\widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v})\}_{\mathbf{i}\in\mathcal{F}}$ is also needed when displacing a point on the surface. It costs $O(\widetilde{m}_u d^2 + \widetilde{m}_v d^2 + \#\mathcal{F})$, during Step iii (Figure 1):

- Computing $\{\widetilde{B}_{i_u}(\bar{u})\}_{i_u}$ and $\{\widetilde{B}_{i_v}(\bar{v})\}_{i_v}$ with the one-dimensional De Boor algorithm costs $O(\widetilde{m}_u d^2 + \widetilde{m}_v d^2)$.
- Computing $\widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v}) = \widetilde{B}_{i_u}(\bar{u})\widetilde{B}_{i_v}(\bar{v})$ for $\mathbf{i} \in \mathcal{F}$ costs $O(\#\mathcal{F})$.

4.4 Uniform B-splines

Uniform B-splines are a less general framework than non-uniform ones, for surface design (e.g., sharp features are not possible). However, it enables the use of efficient MR schemes based on wavelets. This section shows that the MR basis is both efficient and convenient for defining and controlling the volume preserving deformation. The values needed for the volume constraint can be pre-computed through a recursive process using the reconstruction filters P and Q.

4.4.1 Volume constraint in the MR basis

In the uniform framework, Δ and S^q are written in the same MR basis. Rewriting Equations (7) and (8) with superscripts denoting the level of resolution (see Section 3.3), the volume in the fine basis of V^n is given by

$$\Theta(S^q) = \sum_{\mathbf{i},\mathbf{j},\mathbf{k}} x^n_{\mathbf{i}}\, y^n_{\mathbf{j}}\, z^n_{\mathbf{k}} \left(\theta^{n,u}_{j_u k_u i_u} \theta^{n,v}_{i_v k_v j_v} - \theta^{n,u}_{i_u k_u j_u} \theta^{n,v}_{j_v k_v i_v} \right). \tag{17}$$

Assume the surface is decomposed at some fixed level e. A similar formula exists:

$$\Theta(S^q) = \sum_{\mathbf{i},\mathbf{j},\mathbf{k}} x^e_{\mathbf{i}}\, y^e_{\mathbf{j}}\, z^e_{\mathbf{k}} \left(\theta^{e,u}_{j_u k_u i_u}\, \theta^{e,v}_{i_v k_v j_v} - \theta^{e,u}_{i_u k_u j_u}\, \theta^{e,v}_{j_v k_v i_v} \right), \tag{18}$$

where $\vec{x}^e$, $\vec{y}^e$ and $\vec{z}^e$ are the coordinates of *all* the MR coefficients, i.e., of $\vec{\mathbf{p}}^e$, $\vec{\mathbf{d}}^e$, $\vec{\mathbf{d}}^{e+1}, \ldots, \vec{\mathbf{d}}^{n-1}$. This formula is obtained by substituting the MR representation (3) of S^q into Equation (7).

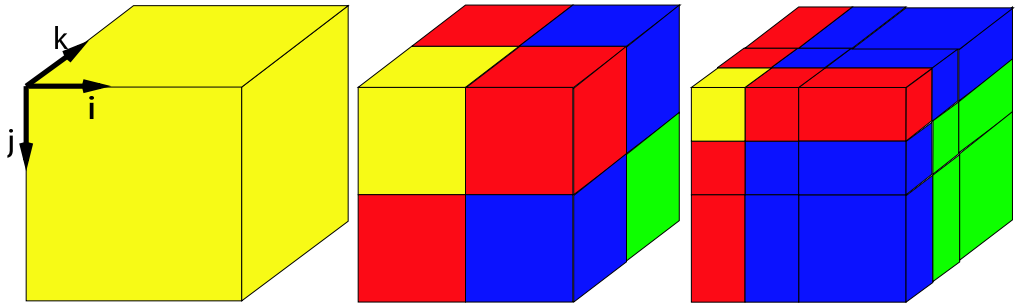

Figure 3: Recursive processing of the tensors.
From left to right: $\theta^{n,u}$, $\theta^{n-1,u}$ and $\theta^{n-2,u}$ (or $\theta^{n,v}$, $\theta^{n-1,v}$ and $\theta^{n-2,v}$).

The sets $\theta^{e,u} = (\theta^{e,u}_{i_u,j_u,k_u})_{i_u,j_u,k_u}$ and $\theta^{e,v} = (\theta^{e,v}_{i_v,j_v,k_v})_{i_v,j_v,k_v}$ are rank 3 tensors, and can represented by three-dimensional arrays (see Figure 3). The fine level tensors $\theta^{n,u}$ and $\theta^{n,v}$ (see Figure 3 left) have been pre-computed through Formulas (9) and (10), and stored during initialization (Step i, Figure 1). Then, $\theta^{e,u}$ and $\theta^{e,v}$ are computed through a recursive process (from left to right). The transposed filter $(P^j)^T$ and $(Q^j)^T$ are applied in a tensor product manner:

- Following the 3 axes on the yellow (light) block,
- following the 2 axes on the red (mid grey) blocks,
- following the 1 axis on the blue (dark) blocks,
- and green blocks (on the right, down and below) are invariant.

Proving that this process makes Equations (17) and (18) identical is simple but tedious. An intuitive explanation considers the transition from Equation (17) to Equation (18):

- $\vec{x}^e$, $\vec{y}^e$ and $\vec{z}^e$ are obtained from $\vec{x}^n$, $\vec{y}^n$ and $\vec{z}^n$ through the filterbank algorithm (using filters A^j and B^j),
- $\theta^{e,u}$ and $\theta^{e,v}$ are obtained from $\theta^{n,u}$ and $\theta^{n,v}$ through this process (using filters $(P^j)^T$ and $(Q^j)^T$),
- and the filters are related by

$$\begin{bmatrix} A^j \\ B^j \end{bmatrix} = \begin{bmatrix} P^j & Q^j \end{bmatrix}^{-1}.$$

Computing $\theta^{e,u}$ and $\theta^{e,v}$ through this process is quite expensive but it is made only once during Step *iii* (Figure 1, pre-computation and storage).

Consider an x-axis deformation $\boldsymbol{\delta}^e_{\mathbf{i}} = (\boldsymbol{\delta} x^e_{\mathbf{i}}, 0, 0)$. By combining Equations (18) and (4), the volume constraint $\Theta(S^q + \Delta) = \Theta(S^q)$ becomes

$$\sum_{\mathbf{i}} \boldsymbol{\delta} x^e_{\mathbf{i}}\, \Theta^e_{YZ}(\mathbf{i}) = 0, \tag{19}$$

where for $(0,0) \leq \mathbf{i} \leq (p_u 2^e + d - 1, p_v 2^e + d - 1)$

$$\Theta^e_{YZ}(\mathbf{i}) = \sum_{\mathbf{j},\mathbf{k}} y^e_{\mathbf{j}} z^e_{\mathbf{k}} \left(\theta^{e,u}_{j_u k_u i_u} \theta^{e,v}_{i_v k_v j_v} - \theta^{e,u}_{i_u k_u j_u} \theta^{e,v}_{j_v k_v i_v} \right). \tag{20}$$

4.4.2 Solving the minimization problem

The solutions in the present uniform framework are close to the previous non-uniform framework. Hence, we do not detail the calculations.

Displacing a control point

Displacing the point $\bar{\boldsymbol{\imath}}$ fixes $\boldsymbol{\delta} x^e_{\bar{\boldsymbol{\imath}}} = \mathbf{t}x$. Excluding $\bar{\boldsymbol{\imath}}$ from $\mathcal{F}$, the volume constraint (19) becomes:

$$\Theta^e_{YZ}(\bar{\boldsymbol{\imath}})\, \mathbf{t}x + \sum_{\mathbf{i} \in \mathcal{F}} \Theta^e_{YZ}(\mathbf{i})\, \boldsymbol{\delta} x^e_{\mathbf{i}} = 0.$$

Hence, the minimization is

$$\min_{\{\widetilde{\boldsymbol{\delta}} x_{\mathbf{i}}\}} \sum_{\mathbf{i} \in \mathcal{F}} |\boldsymbol{\delta} x^e_{\mathbf{i}}|^2 \quad \text{subject to} \quad \Theta^e_{YZ}(\bar{\boldsymbol{\imath}})\, \mathbf{t}x + \sum_{\mathbf{i} \in \mathcal{F}} \Theta^e_{YZ}(\mathbf{i})\, \boldsymbol{\delta} x^e_{\mathbf{i}} = 0,$$

and the solution is

$$\widetilde{\boldsymbol{\delta}} x_{\mathbf{i}} = \frac{-\mathbf{t}x\ \Theta^e_{YZ}(\bar{\boldsymbol{\imath}})}{\sum |\Theta^e_{YZ}(\mathbf{j})|^2} \Theta^e_{YZ}(\mathbf{i}), \qquad \forall \mathbf{i} \in \mathcal{F}, \tag{21}$$

where the sum runs over $\mathbf{j} \in \mathcal{F}$ and it is assumed not to equal zero.

Displacing a point on the surface

Including the position constraint (Section 3.4) and the volume constraint (19), one gets

$$\min_{\{\widetilde{\boldsymbol{\delta}} x_{\mathbf{i}}\}} \sum_{\mathbf{i} \in \mathcal{F}} |\boldsymbol{\delta} x^e_{\mathbf{i}}|^2 \quad \text{subject to} \quad \begin{cases} \sum \Theta^e_{YZ}(\mathbf{i})\, \boldsymbol{\delta} x^e_{\mathbf{i}} = 0, \\ \sum \varphi^e_{\mathbf{i}}(\bar{u}, \bar{v})\, \widetilde{\boldsymbol{\delta}} x_{\mathbf{i}} = \mathbf{t}x. \end{cases}$$

Having

$$\alpha = \sum_{\mathbf{j} \in \mathcal{F}} |\varphi^e_{\mathbf{j}}(\bar{u}, \bar{v})|^2 \sum_{\mathbf{j} \in \mathcal{F}} |\Theta^e_{YZ}(\mathbf{j})|^2 - \left(\sum_{\mathbf{j} \in \mathcal{F}} \varphi^e_{\mathbf{j}}(\bar{u}, \bar{v})\, \Theta^e_{YZ}(\mathbf{j}) \right)^2,$$

never vanish, the unique solution is $\forall \mathbf{i} \in \mathcal{F}$,

$$\boldsymbol{\delta} x^e_{\mathbf{i}} = \frac{\mathbf{t}x}{\alpha} \left(\left(\sum_{\mathbf{j} \in \mathcal{F}} |\Theta^e_{YZ}(\mathbf{j})|^2 \right) \varphi^e_{\mathbf{i}}(\bar{u}, \bar{v}) - \left(\sum_{\mathbf{j} \in \mathcal{F}} \varphi^e_{\mathbf{j}}(\bar{u}, \bar{v})\, \Theta^e_{YZ}(\mathbf{j}) \right) \Theta^e_{YZ}(\mathbf{i}) \right). \tag{22}$$

4.4.3 Algorithmic issues

The pre-processing (Step *iii*, Figure 1) includes the computation of $\theta^{e,u}$ and $\theta^{e,v}$ and of $\{\widetilde{\varphi}_{\mathbf{i}}(\bar{u},\bar{v})\}_{\mathbf{i}\in\mathcal{F}}$ (when displacing a point on the surface).

Interactive Step νi (Figure 1) treats the x, y and z-axes successively. Treating the x-axis involves:

- Computing $\{\Theta^e_{YZ}(\mathbf{i})\}$ that costs $O(m_u m_v \#\mathcal{F})$ through Formula (20).
- Computing $\{\boldsymbol{\delta} x^e_{\mathbf{i}}\}_{\mathbf{i}\in\mathcal{F}}$ that costs $O(\#\mathcal{F})$ through Formula (21) or (22).
- Adding the deformation to the surface $x^e_{\mathbf{i}} \leftarrow x^e_{\mathbf{i}} + \boldsymbol{\delta} x^e_{\mathbf{i}}$ that costs $O(\#\mathcal{F})$.

Hence, the global cost is $O(m_u m_v \#\mathcal{F})$. However, it is not an optimal bound because the computation of $\{\Theta^e_{YZ}(\mathbf{i})\}$ depends on the level e and on the sparsity of $\theta^{e,u}$ and $\theta^{e,v}$. If $e = n$ it costs $O(\#\mathcal{F})$ only.

4.5 Multi-patched surfaces

Modeling a closed surface with a single tensor-product patch restricts the topology. Let $\{S^q\}_q$ be a set of patches modeling a closed surface. We extended the previous results while preserving C^0 continuity between the patches. For example, in the uniform setting one has to just modify the volume constraint (19) in the following way. If the control point $\mathbf{i}$ belongs to more than one patch, i.e., it is a boundary or a corner point, the coefficient $\Theta^e_{YZ}(\mathbf{i})$ is replaced by the sum over the patches to which the points belongs:

$$\sum_{\text{patches } q} \Theta^e_{YZ}(\mathbf{i}_q, q),$$

where $\mathbf{i}_q$ is the index of the control point with respect to the patch q, and $\Theta^e_{YZ}(\mathbf{i}_q, q)$ is computed through Formula (20). The deformation $\boldsymbol{\delta} x^e_{\mathbf{i}}$ is then applied on both sides of the join. The complexity of the resulting algorithm remains the same. Figures 5 and 6 for instance involve 3 patches.

While the presented scheme handles only linear constraints that preserve C^0 continuity across surfaces, additional linear constraints could be added to preserve higher order continuity across the surfaces. The problem of posing the necessary constraints to preserve G^1 or C^1 (or even higher order) continuity is beyond this work and was addressed by many, including the authors [Pet95, HB00, HB03]. We foresee no major obstacle to overcome in supporting higher order continuity across surfaces, once these additional constraints are imposed, or any other additional linear constraints.

5 Results

Figure 4 shows the deformation of a unit cube (6 bi-cubic patches). The surface on the left illustrates the selection of the extent of the deformation (red/dark region) as a neighborhood of the displaced point across the joins. The point's parameter in the upper patch is $(\bar{u},\bar{v}) = (0.7, 0.8)$ and the displacement vector is $\vec{d} = (0.2, 0.2, 0.9)^T$. The two views of the deformed surface (middle and right)

illustrate the linearization by separating the 3 axis: the added volume caused by the bump displacement is balanced by a cavity in the opposite direction.

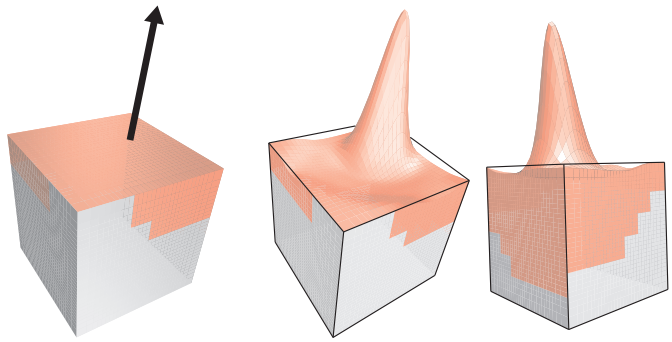

Figure 4: Deformation of the unit cube. Each of the six faces is a 15×15 uniform bi-cubic patch. Left: initial surface with the extent (red/dark region) and the displacement (arrow). Middle and right: two views after volume preserving deformation. Wireframe shows the initial position.

We now compare the different methods presented in the previous section by deforming surfaces representing knight chess pieces. The surfaces are composed of three bi-quadratic patches (the front, back and bottom of the knight).

Figure 5: Deformation of a non-uniform B-spline surface.
Left: Initial surface. Middle: The volume is free. Right: The volume is preserved.

Figure 5 shows the deformation of a non-uniform B-spline surface. The initial surface (left) is deformed by pulling a point on the surface (on the back of the neck). When the volume is unconstrained (middle), the result lacks realism if we expect the knight to be made of some soft non-stretch material. Volume

preserving (right) provides a more intuitive behavior.

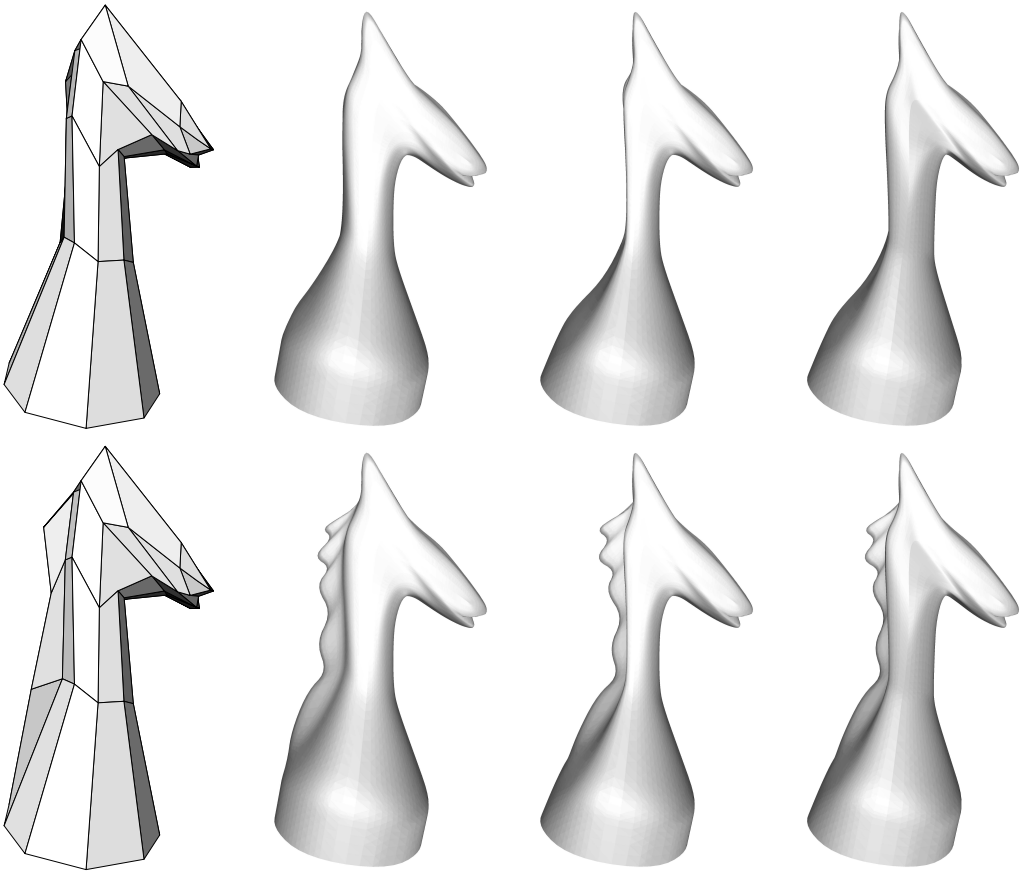

Figure 6: Deformation of a uniform B-spline surface.
Frome left to right : control polyhedrons; initial surfaces; the volume is free; the volume is preserved.

Figure 6 shows the deformation of uniform B-spline surfaces. The initial surfaces (left) are deformed by pushing a control point on the back of the neck. Results without volume preserving (middle) and with volume preserving (right) are compared. One can see that even a small deformation affects the appearance. The bottom row presents the same deformation with some fine details added to the surface. Since these details are basically encoded by the MR details, they are preserved during the deformation.

Figure 7 illustrates both the detail and volume preservation for non-uniform surfaces made up of two 14×15 bi-cubic patches. Three initial surfaces (left) differ in their details only. Manipulation is processed by constraining only two points on the surface. The same deformations (b,c) and (d,e) are applied to all of them, without (b,d) and with (c,e) volume preservation. The preservation of the details is convincing while the global shape is interactively modified to enforce the volume constraint.

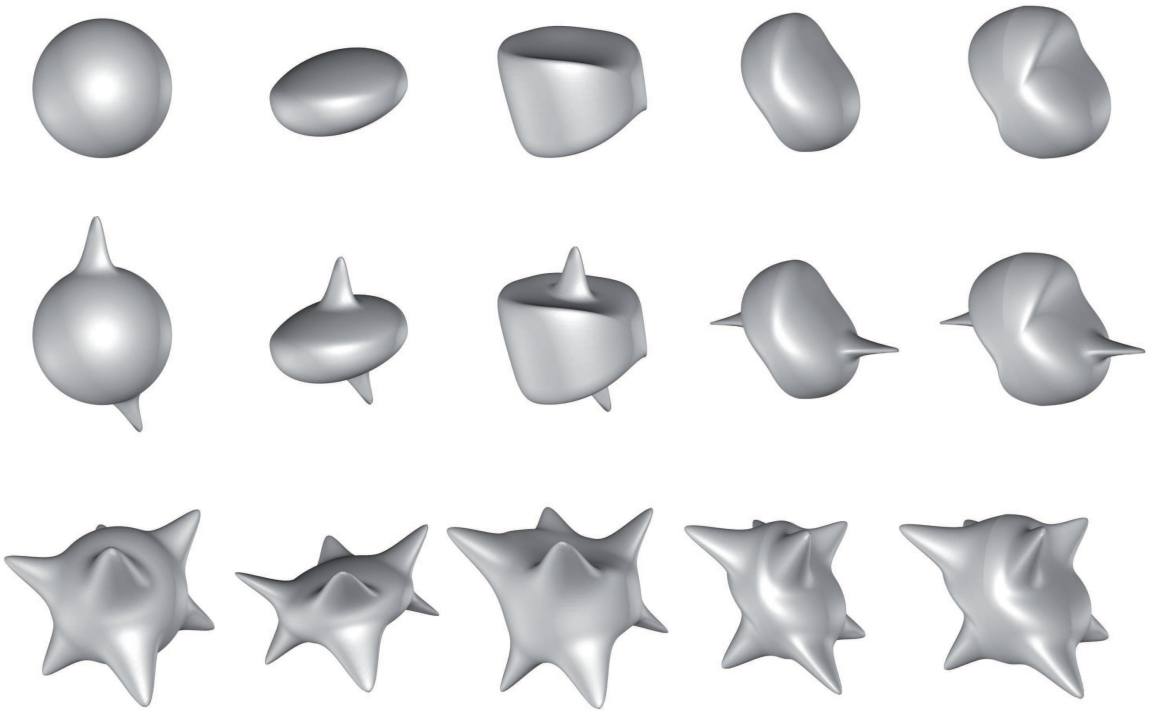

Figure 7: Column (a) shows the original shapes. In columns (b) and (d) two deformations are applied to the object in column (a) without volume preservation. Columns (c) and (e) shows the same deformations but this time with volume preservation.

6 Conclusion and future work

A method for the precise preservation of the volume enclosed by tensor product B-spline surfaces during an MR editing process has been presented. Both uniform and non-uniform settings have been addressed. Volume formulas for two-scale non-uniform B-spline deformation and for MR uniform B-spline deformation have been developed. The portion of the surface involved in the deformation is easily selected. Moreover, the fine details of the shape can be preserved through an appropriate choice of basis: a two-scale basis in the non-uniform setting and an MR basis in the uniform setting. A computation of the volume with respect to these bases is proposed. The volume constraint is then linearized and included in a minimization process. A closed form of the solution allows interactive editing. It is supported by a precise analysis of the algorithmic issues. The results show that the volume constraint is of interest in order to enhance the deformation's realism. This makes our method suitable for the animation of soft objects.

There are several aspects that in our opinion are worth further investigation. We presented a volume preserving deformation method for the non-uniform case that rests on a two-scale basis. It would be interesting to investigate the use of

a non-uniform B-spline wavelet basis [KE97, LM92].

Another direction for future work could be the extension of our method to G^1 continuous surfaces. It is well known that G^1 continuity for surfaces of arbitrary topology is a quite difficult problem. In fact, at a patch corner with more than four patches, the so-called "vertex compatibility problem" has to be solved [Pet95, HB03]. In a classical tensor-product setting, where four patches meet at a corner, the G^1 continuity can be expressed by linear equations with respect to the control points. In all the other cases this constraint might be non-linear.

Acknowledgments

The work was partially supported by the AIM@SHAPE Network of Excellence (FP6 IST NoE 506766) and the IMAG project MéGA.

References

[BB89] R. Bartels and J. Beatty. A technique for the direct manipulation of spline curves. *Graphics Interface '89*, pages 33–39, 1989.

[BH94] G. P. Bonneau and H. Hagen. Variational design of rational bézier curves and surfaces. In L. Laurent and L. Schumaker, editors, *Curves and Surfaces*, volume II, pages 51–58. AK Peters, 1994.

[BK03] Mario Botsch and Leif Kobbelt. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum*, 22(3):483–491, 2003. (Proceedings Eurographics '03).

[Boe80] Wolfgang Boehm. Inserting new knots into a bspline curve. *Computer-Aided Design*, 12:199–201, 1980.

[CG91] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics (SIGGRAPH '91 proceedings)*, 25:257–266, July 1991.

[Cia88] Philippe G. Ciarlet. *Introduction to Numerical Linear Algebra and Optimization*. Cambridge University Press, 1988.

[CW92] G. Celniker and W. Welch. Linear constraints for deformable b-spline surfaces. *1992 Symposium on Interactive 3D Graphics*, pages 165–170, 1992.

[Elb01] G. Elber. Multiresolution curve editing with linear constraints. In *6th ACM/IEEE Symposium on Solid Modeling and Applications*, pages 109–119. Ann Arbor, Michigan, June 2001.

[Far01] Gerald Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., 2001.

[FB93] B. Fowler and R. Bartels. Constraint-based curve manipulation. *IEEE Computer Graphics and Applications*, 13(5):43–49, 1993.

[Fow92] B. Fowler. Geometric manipulation of tensor product surfaces. In *1992 Symposium on Interactive 3D Graphics*, pages 101–108, 1992.

[FRSW87] G. Farin, G. Rein, N. Sapidis, and A. J. Worsey. Fairing cubic b-spline curves. *Computer Aided Geometric Design*, 4:91–103, 1987.

[FS94] Adam Finkelstein and David H. Salesin. Multiresolution curves. *SIGGRAPH Computer Graphics*, pages 261–268, 1994.

[GC95] S. Gortler and M. Cohen. Hierarchical and variational geometric modeling with wavelets. In *1995 Symposium on 3D Interactive Graphics*, pages 35–41, 1995.

[GL96] G. Greiner and J. Loos. Data dependent thin plate energy and its use in interactive surface modeling. *Computer Graphics Forum*, 15:176–185, 1996. (Proceedings Eurographics '96).

[Gle92] M. Gleicher. Integrating constraints and direct manipulation. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 171–174. ACM Press, 1992.

[GOMP98] Carlos Gonzalez-Ochoa, Scott McCammon, and Jörg Peters. Computing moments of objects enclosed by piecewise polynomial surfaces. *ACM Transactions on Graphics*, 17(3):143–157, 1998.

[Hah98] S. Hahmann. Shape improvement of surfaces. *Computing Suppl.*, 13:135–152, 1998.

[HB00] S. Hahmann and G.-P. Bonneau. Triangular G1 interpolation by 4-splitting domain triangles. *Computer-Aided Geometric Design (CAGD)*, 17(8):731–757, 2000.

[HB03] Stefanie Hahmann and Georges-Pierre Bonneau. Polynomial surfaces interpolating arbitrary triangulations. *IEEE Transactions on Visualisation and Computer Graphics*, 9(1):99–109, 2003.

[HML99] Gentaro Hirota, Renee Maheshwari, and Ming C. Lin. Fast volume-preserving free form deformation using multi-level optimization. In *SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 234–245, New York, NY, USA, 1999. ACM Press.

[HS92] H. Hagen and P. Santarelli. Variational design of smooth b-spline surfaces. In H. Hagen, editor, *Topics in Geometric Modeling*, pages 85–94. SIAM Philadelphia, 1992.

[HSB05] S. Hahmann, B. Sauvage, and G.-P. Bonneau. Area preserving deformation of multiresolution curves. *Computer-Aided Geometric Design (CAGD)*, 22(4):349–367, 2005.

[KE97] R. Kazinnik and G. Elber. Orthogonal decomposition of non-uniform bspline spaces using wavelets. *Computer Graphics Forum*, 16(3):27–38, 1997. (Proceedings Eurographics '97).

[KS95] P. D. Kaklis and N. S. Sapidis. Convexity-preserving interpolatory parametric splines of nonuniform polynomial degree. *Comput. Aided Geom. Des.*, 12(1):1–26, 1995.

[Las87] J. Lassiter. Principles of traditional animation applied to 3d computer animation. In *Computer Graphics (SIGGRAPH 87 Proceedings)*, pages 45–44, 1987.

[LCOGL07] Yaron Lipman, Daniel Cohen-Or, Ran Gal, and David Levin. Volume and shape preservation via moving frame manipulation. *ACM Transactions on Graphics*, to appear, 2007.

[LM92] T. Lyche and K. Morken. Spline wavelets of minimal support. In D. Braess and L. Schumaker, editors, *Numerical Methods of Approximation Theory*, pages 177–194. Birkhäuser Verlag, Basel, 1992.

[Mal99] Stéphane Mallat. *A wavelet Tour of Signal Processing*. Academic Press, 1999.

[PE98] M. Plavnik and G. Elber. Surface design using global constraints on total curvature. In *The VIII IMA Conference on Mathematics of Surfaces*, September 1998.

[Pet95] Jörg Peters. Biquartic c1-surface splines over irregular meshes. *Computer-Aided Design*, 27(12):895–903, 1995.

[PGL+02] J. P. Pernot, S. Guillet, J. C. Leon, F. Giannini, B. Falcidieno, and E. Catalano. A shape deformation tool to model character lines in the early design phases. In *Proceedings Shape Modeling International 2002, Banff, Canada*, 2002.

[RSB95] A. Rappoport, A. Sheffer, and M. Bercovier. Volume-preserving free-form solids. In *Proceedings of Solid Modeling 95*, pages 361–372, May 1995.

[SDD95] E. Stollnitz, T. DeRose, and D.Salesin. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, 15(3):76–84, 1995.

[SDS96] Eric J. Stollnitz, Tony DeRose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1996.

[SHB07] Basile Sauvage, Stefanie Hahmann, and Georges-Pierre Bonneau. Volume preservation of multiresolution meshes. *Computer Graphics Forum*, 26(3), to appear 2007. (Proceedings Eurographics '07).

[vFTS06] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. *SIGGRAPH Computer Graphics*, pages 1118–1125, 2006.

[WW92] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics (SIGGRAPH '92 proceedings)*, 26:157–166, July 1992.

A Knot removal algorithm

The volume preserving deformation of non-uniform B-spline surfaces as proposed in Section 4.3 requires efficient computation of $\widetilde{\Theta}_{YZ}(\mathbf{i})$ from $\Theta_{YZ}(\mathbf{i})$. We propose

a recursive algorithm based on Boehm's recurrence [Boe80, Far01] that relates between the B-spline basis functions before and after a knot is inserted or removed.

For the purpose of clarity, first assume $\widetilde{\mathcal{V}} = \mathcal{V}$ and one single knot u_l is removed from $\mathcal{U}$, *i.e.* $\mathcal{U} = \widetilde{\mathcal{U}} \cup \{u_l\}$. Thus, the coarse B-splines $\widetilde{B}_k$ defined over the knot sequence $\widetilde{\mathcal{U}}$ are related to the fine B-splines B_k defined over $\mathcal{U}$ by

$$\widetilde{B}_k(u) = \frac{u_{l+1} - u_k}{u_{k+d+1} - u_k} B_k^d(u) + \frac{u_{k+d+2} - u_{l+1}}{u_{k+d+2} - u_{k+1}} B_{k+1}^d(u) \quad \text{for} \quad l - d \leq k \leq l,$$

and $\widetilde{B}_k = B_k$ otherwise. Thanks to the linearity of the integral and of the sums (Equations (11) and (13)), this relation can be transfered between $\widetilde{\Theta}_{YZ}(\mathbf{i})$ and $\Theta_{YZ}(\mathbf{i})$. The integral in Equation (11) may be written as

$$\begin{aligned}
&\iint \varphi_{\mathbf{k}} \Big(\frac{\partial \widetilde{\varphi}_{\mathbf{i}}}{\partial u} \frac{\partial \varphi_{\mathbf{j}}}{\partial v} - \frac{\partial \widetilde{\varphi}_{\mathbf{i}}}{\partial v} \frac{\partial \varphi_{\mathbf{j}}}{\partial u} \Big) \, du \, dv \\
&= \int B_{j_u} B_{k_u} \widetilde{B}'_{i_u} du \int B_{i_v} B_{k_v} B'_{j_v} dv - \int \widetilde{B}_{i_u} B_{k_u} B'_{j_u} du \int B_{j_v} B_{k_v} B'_{i_v} dv \\
&= \left(\frac{u_{l+1} - u_{i_u}}{u_{i_u+d+1} - u_{i_u}} \, \theta^u_{j_u,k_u,i_u} + \frac{u_{i_u+d+2} - u_{l+1}}{u_{i_u+d+2} - u_{i_u}} \, \theta^u_{j_u,k_u,(i_u+1)} \right) \theta^v_{i_v k_v j_v} \\
&\quad - \left(\frac{u_{l+1} - u_{i_u}}{u_{i_u+d+1} - u_{i_u}} \, \theta^u_{i_u,k_u,j_u} + \frac{u_{i_u+d+2} - u_{l+1}}{u_{i_u+d+2} - u_{i_u}} \, \theta^u_{(i_u+1),k_u,j_u} \right) \theta^v_{j_v k_v i_v} \\
&= \frac{u_{l+1} - u_{i_u}}{u_{i_u+d+1} - u_{i_u}} \, \left(\theta^u_{j_u,k_u,i_u} \theta^v_{i_v k_v j_v} - \theta^u_{i_u,k_u,j_u} \theta^v_{j_v k_v i_v} \right) \\
&\quad + \frac{u_{i_u+d+2} - u_{l+1}}{u_{i_u+d+2} - u_{i_u}} \, \left(\theta^u_{j_u,k_u,(i_u+1)} \theta^v_{i_v k_v j_v} - \theta^u_{(i_u+1),k_u,j_u} \theta^v_{j_v k_v i_v} \right)
\end{aligned}$$

Hence, comparing Equations (11) and (13) implies

$$\widetilde{\Theta}_{YZ}(i_u, i_v) = \frac{u_{l+1} - u_{i_u}}{u_{i_u+d+1} - u_{i_u}} \, \Theta_{YZ}(i_u, i_v) + \frac{u_{i_u+d+2} - u_{l+1}}{u_{i_u+d+2} - u_{i_u+1}} \, \Theta_{YZ}(i_u+1, i_v). \quad (23)$$

Using formula (23) for all i_v one can compute each column $\widetilde{\Theta}_{YZ}(., i_v)$ (relative to the coarse basis) from the column $\Theta_{YZ}(., i_v)$ (relative to the fine basis).

Then assume $\widetilde{\mathcal{V}} = \mathcal{V}$ and $\widetilde{\mathcal{U}} \subset \mathcal{U}$. To each column $\widetilde{\Theta}_{YZ}(., i_v)$ separately one has to apply Formula (23) for each removed knot, i.e., each knot in $\mathcal{U} \setminus \widetilde{\mathcal{U}}$. The iterative Algorithm 1 is an example where no extra memory is needed.

Finally, whatever $\widetilde{\mathcal{V}}$ and $\widetilde{\mathcal{U}}$: i) remove the knots $u \in \mathcal{U} \setminus \widetilde{\mathcal{U}}$ as explained above and ii) remove symmetrically the knots $v \in \mathcal{V} \setminus \widetilde{\mathcal{V}}$ for each row $\widetilde{\Theta}_{YZ}(i_u, .)$.

Algorithm 1 costs $O(m_u d)$. Hence, computing all the $\widetilde{\Theta}_{YZ}(\mathbf{i})$ from the $\Theta_{YZ}(\mathbf{i})$ costs $O(m_u m_v d)$.

Algorithm 1 Knot removal without extra memory allocation.
Input: $n = m_u$, vector u contains $\mathcal{U}$ and $\Theta(0 \dots n-1)$ contains $\Theta_{YZ}(., i_v)$
Output: $m = \widetilde{m}_u$, $\mathrm{u}(0 \dots m-1)$ contains $\widetilde{\mathcal{U}}$ and $\Theta(0 \dots m-1)$ contains $\widetilde{\Theta}_{YZ}(., i_v)$.

$m \leftarrow d+1$ *// boundary knots are preserved*
for $l = d+1$ to $n-1$ **do**
  if $\mathrm{u}(l)$ is preserved **then**
    $\Theta(m) \leftarrow \Theta(l)$
    $\mathrm{u}(m) \leftarrow \mathrm{u}(l)$
    $m \leftarrow m+1$
  else *// $\mathrm{u}(l)$ is removed*
    for $r = 0$ to $d-1$ **do** *// then $m-d-1 \leq m-d-1+r \leq m-2$*

$$\Theta(m{-}d{-}1{+}r) \leftarrow \frac{\mathrm{u}_{(l)} - \mathrm{u}_{(m-d-1+r)}}{\mathrm{u}_{(l+r)} - \mathrm{u}_{(m-d-1+r)}}\Theta(m{-}d{-}1{+}r) + \frac{\mathrm{u}_{(l+r+1)} - \mathrm{u}_{(l)}}{\mathrm{u}_{(l+r+1)} - \mathrm{u}_{(m-d+r)}}\Theta(m{-}d{+}r)$$

    end for

    // special case $r = d$

$$\Theta(m{-}1) \leftarrow \frac{\mathrm{u}_{(l)} - \mathrm{u}_{(m-1)}}{\mathrm{u}_{(l+d)} - \mathrm{u}_{(m-1)}}\Theta(m{-}1) + \Theta(l)$$

  end if
end for