Katedra algebry, geometrie a didaktiky matematiky
Fakulta matematiky, fyziky a informatiky
Univerzita Komenského, Bratislava

Bézierove a B-splajnové telesá

Dizertačná práca

RNDr. Martin Samuelčík

Študijný odbor: 11-16-9 Geometria a Topológia
Školiteľ: doc. RNDr. Valent Zaťko, CSc.

Bratislava 2010

Department of Algebra, Geometry and Didactics of Mathematics
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Bézier and B-spline volumes

Dissertation Thesis

RNDr. Martin Samuelčík

Branch: 11-16-9 Geometry and Topology
Supervisor: doc. RNDr. Valent Zaťko, CSc.

Bratislava 2010

I do solemnly and sincerely declare that this dissertation thesis has been written by myself without any unauthorized help, using just the literature listed at the end.

. . . .

Martin Samuelčík

Abstrakt

V tejto práci predkladáme základné definície a vlastnosti parametrických telies. Zameriavame sa na telesá v Bézierovom a B-splajnovom tvare, ktoré su definované pomocou polynomických a po častiach polynomických funkcií. Ukážeme novú parametrickú reprezentáciu telies nazývanú S-telesá. Táto reprezentácia je založená na zovšeobecnených barycentrických súradniciach a Bernsteinových polynómoch. V časti venovanej modelovaniu s pomocou parametrických telies popisujeme algoritmy a konštruckie pre tvorbu a modifikáciu telies v tomto tvare. Na záver prezentujeme náš vlastný modelovací systém GeomForge, ktorý implementuje popísané algoritmy a konštrukcie.

Kľúčové slová: parametrické telesá, Bernsteinove polynómy, zovšeobecnené barycentrické súradnice, B-splajnové funkcie

Abstract

This works presents the basic definitions and properties of parametric volumes. We focus on the volumes in Bézier and B-spline form that are defined by polynomials or piecewise polynomials. We present new parametric representation called S-volumes based on generalization of Bernstein polynomials and barycentric coordinates. In the modeling part, we describe algorithms and constructions for the work with the parametric volumes. As an extension of the given theory, we present an implementation of the described algorithms in our own modeling system called GeomForge.

Keywords: parametric volumes, Bernstein polynomials, generalized barycentric coordinates, B-spline functions

Contents

List of Figures

Introduction

The most used representation of the objects in computer graphics is the boundary representation. There are many ways how to represent a boundary. Geometric modeling works also with boundaries that consist of patches described by parametric functions. These functions are mainly in Bézier or B-spline form, i.e. based on Bernstein or B-spline basis functions. This approach to parametric curves and surfaces is well-described in many works, some of the fundamental works in this area being [Far93], [CRE01] or [FH00].

But in many situations, parametric expression of the interior of the object is also needed. One approach can be a description of a physical parameter inside the object or interpolation of data. The principle of extending parametric surfaces into solids or volumes is used by many authors, for example [uC01], [Las85], [LBS08]. Our goal is to derive several constructions of objects based on Bernstein and B-spline basis functions. We apply a well-known tensor-product paradigm to volumes. As an innovation, we extend the multi-sided generalization of Bézier triangles known as S-patches. We call this extension S-volumes. This work presents basics of parametric volumes in geometric modeling. Our main goal is to use known properties of parametric curves and surfaces to construct and describe parametric volumes based on Bézier and B-spline construction. Among these description are basic definitions and properties of such volumes, modeling techniques and volumes visualization.

To achieve our goal, first we show how the similar modeling part works for parametric curves and surfaces. Most algorithms for parametric volumes are then created as extension of curve and surface cases. For the modeling part, we introduce algorithms for creation of volumes from curves and surfaces, algorithms for decomposition of volumes into smaller parts, algorithms for subdivision and algorithms for connection of volumes with certain order of continuity. For the inclusion of our representation into a modeling system, we propose several possibilities for visualization, mainly for visualization using modern graphics hardware. To achieve useful representation, we created software implementation of the modeling system based on parametric volumes. This modeling system implements all introduced properties and algorithms for parametric volumes and shows usefulness of the parametric representation. The system is called GeomForge.

The first chapter of this work gives an overview of the mostly used 3D object representations of objects in computer graphics. Basic definitions, advantages and disadvantages are presented.

The next chapter describes curves and surfaces as basic objects in geometric modeling. These objects are used in many areas, such as path description or contour representation and are the basic elements for constructing parametric volumes. We present curves and surfaces based on Bézier and B-spline form, their properties, properties of blending functions and also subdivision techniques for curves. Based on curve and surface modeling, parametric volumes are defined using barycentric coordinates, Bernstein polynomials and B-spline polynomials. We introduce are new volumes that can represent multi-sided generalization of Bézier tetrahedron. Several theorems then give many useful properties of Bézier tensor-product volumes, S-volumes and B-spline tensor product volumes. As an addition, rational volumes are introduced.

The third chapter presents several ways how to work with parametric volumes based on their properties. In this modeling part, we describe techniques for creation, decomposition, subdivision, deformation and connection of parametric volumes. The presented algorithms are illustrated using screenshots from the GeomForge system.

In the fourth chapter, the visualization algorithms are given. We present the wireframe (using isoparametric curves) and solid (using isoparametric surfaces) visualization. For insight into interior of an object, visualization using isosurfaces and transparency is given. We also present visualization of volume approximation based on several subdivision techniques (degree elevation, knot insertion, de Casteljau algorithm). At the end of the chapter, we present GeomForge, our own modeling system for work with parametric solids. Basic functionality of the system as well as the description of all supported types of objects is described.

Chapter 1

Object representations

The basic element in 3D computer graphics and geometry is a three-dimensional object. In computer graphics and geometric modeling, there exist many ways to represent three-dimensional objects. Because of the diversity and number of these representations, each of them has its own advantages and disadvantages. The areas that we consider when comparing different representations are:

- *Data effectiveness for modeling.*
- *Space occupied in memory.*
- *Issues for different visualization techniques.*
- *Possibility to represent wide range of basic objects (e.g. objects bounded by conic surfaces).*
- *Possibility of conversion to another representation.*

In this work, we prepare a framework for parametric representation of objects based on observations from the lower degree approaches. First we give introduction to the most used types of parametric curves and surfaces and then the extensions for parametric 3D objects. We call objects represented this way as parametric solids or volumes. Some volumetric representations are presented in [CKY01]. An overall introduction to object representations is given in [Req80].

1.1 Boundary representation (B-Rep)

Boundary representation describes an object using just its boundary. This approach has its own advantages and disadvantages. Because only the border is given, we don't have an information about the inside of the object; on the other hand, the size of the stored data is lower, the computation times are faster and the visualization is simpler with the help of graphics hardware. The border can be described using several approaches:

- *Wireframe* - The model is represented only by boundary edges. It occupies only a small amount of memory and the visualization of the object in wireframe is very fast, so it can be used for the preview. But we don't have information about faces of the object.

- *Polygonal representation* - The border of the object is represented by a the set of polygons. There are many data structures for storing objects in this representation. The mostly used structures are winged edge, half edge, quad edge and indexed polygons. This representation is natively supported in graphics hardware, so the visualization is fast.

- *Parametric surfaces* - The border is described by a bivariate function. Many forms of these functions are used. Later in this work, two usable forms are presented. Using this approach, complex borders can be described using small amount of data (control points), but for the visualization, the surface must be approximated by using a set of polygons. Here, due to the parametrization it is easy to prepare mapping on surface or approximation by polygons, but it is hard to find if point in the space lies on the surface.

- *Subdivision surfaces* - This approach is the bridge between the polygonal and parametric representations. The initial control mesh is refined repeatedly to get finer and smoother representation. We present this approach later in detail.

- *Implicit surfaces* - The boundary surface is represented by a function $F : R^3 \to R$ and the object is a set $O = \{(x, y, z) \in E^3; F(x, y, z) = 0\}$. This representation is good for Boolean operations (union, intersection and difference), or to check if a point in space is on the surface. Visualization is again performed using approximation by a set of polygons or by the approximation in the uniform three-dimensional grid.

1.2 Functional representation (F-Rep)

This approach is similar to the implicit surfaces, but instead of equality, inequality is used. For a given function $F : R^3 \to R$, the object is the set $O = \{(x, y, z) \in E^3; F(x, y, z) \leq 0\}$. The advantage of this approach for modeling purposes is that the Boolean operations are performed very fast and easily. Another advantage is that complex objects can be described using one function.

1.3 Constructive solid geometry (CSG)

In this representation, the object is constructed from basic objects using a defined set of operations such as regularized Boolean operations. The object is then represented as a graph or a tree with basic objects (box, cone, sphere) in leaves of the tree and the operations (union, intersection, difference) in the inner nodes. Many operations and modeling techniques are performed by traversing this construction tree.

1.4 Discrete volume representation

The object is represented as a finite set of volume elements called voxels (usually boxes or tetrahedrons) can be arranged in uniform grid, in tree structure (octree) or in non-uniform grid glued together with neighborhood information in each voxel. Also here, the Boolean operations are performed easily. Due to the discrete representation of the object, many artifacts (like alias) can arise.

1.5 Parametric representation

The object is described as a set of points represented by a trivariate point function. The parametric volumes are described in detail in the next section. This approach uses more storage space for the parameters (control points, weights, knots), also the approximation by discretization uses more memory and computational time, but modern hardware can handle this. There are many degrees of freedom that can be lowered using additional constraints. The representation is good for deformations, mapping on object can be performed easily and we can c0onstruct many useful modeling algorithms. There also exists bridge between the parametric volumes and the discrete representation called subdivision volumes.

Chapter 2

Parametric volumes

In the same way as we can extended parametric curves into parametric surfaces, we can also extend parametric surfaces into parametric volumes. Here, we have a parameter from the three dimensional domain or three real parameters that form similar domain. The parametric volumes can be used for representation of objects, where we need to represent the interior of the object. Deformations can be easily performed on these volumes, so we call them free-form objects [SP86]. The parametric volume is defined by a function that maps three dimensional domain on the points in three dimensional euclidean space.

Definition 2.0.1 *Parametric volume in 3-dimensional Euclidean space E^3 is the set of points $PV = \{X \in E^3; X = F(\mathbf{u}); \mathbf{u} \in D, D \subset \mathbb{R}^3\}$, where $F : R^3 \to E^3$ is the function describing the volume, $\mathbf{u}$ is a parameter, D is the domain of the volume.*

It is obvious that the main contribution to the shape of the volume comes from the function F. We can use a wide variety of functions for modeling. In geometric modeling, we often use combination of control points. This way we can achieve the modulation of the shape by the modulation of the position of these control points. For this, we need blending functions that particularly modulate control points.

Definition 2.0.2 *Let us have a set $D \subset \mathbb{R}^3$ and functions $\phi_i(\mathbf{u}); i = 0, 1, ..., n; \mathbf{u} \in D$ such that $\sum_{i=0}^{n} \phi_i(\mathbf{u}) = 1$ for each $\mathbf{u} \in \mathbf{D}$. Then the parametric volume with blending functions ϕ_i is defined as $F(\mathbf{u}) = \sum_{i=0}^{n} P_i \phi_i(\mathbf{u})$. P_i are given points from E^3 called control points.*

We use this notation for our definitions of parametric volumes. Last thing will be definition of blending functions. We mainly work with blending functions in the form of polynomials, later we extend to spline (piecewise polynomial) functions and rational functions.

2.1 Barycentric coordinates

In the field of linear algebra, one of the basic principles is the linear combination of vectors. But it is also necessary to work with combination of vertices in Euclidean space. Such combination is called affine combination an is well-defined if the sum of combination parameters is equal to 1. Affine combination of points is useful for data interpolation, intersection computation, integration over simplex, triangle rasterization, and many more. If we want to describe each point of m-dimensional Euclidean space as combination of given points, these given points must posses several properties. And then they form a object called simplex. Later we will extend this to more complex set of vertices in E^2 and E^3.

Definition 2.1.1 *Consider the m-dimensional Euclidean space E^m. Let us have $(m+1)$ vertices $A_1, ..., A_{m+1} \in E^m$ that don't lie in one hyperplane. Then the convex hull of vertices $A_1, ..., A_{m+1}$ is called m-dimensional simplex S in E^m. For each point $P \in E^m$ we define its barycentric coordinates with respect to the simplex S as function $U : E^m \rightarrow R^{m+1}, U(P) = (u_1(P), ..., u_{m+1}(P))$ that posses following properties:*

- *Partition of unity:* $\sum_{j=1}^{m+1} u_j(P) = 1$
- *Linear precision:* $\sum_{j=1}^{m+1} u_j(P)A_j = P$

We can also define barycentric coordinates of vectors from the vector space $V(E^m)$. For each vector $\mathbf{v} \in V(E^m)$ we define its barycentric coordinates as function $V : V(E^m) \rightarrow R^{m+1}, V(\mathbf{v}) = (v_1(\mathbf{v}), ..., v_{m+1}(\mathbf{v}))$ that posses following properties:

- $\sum_{j=1}^{m+1} v_j(\mathbf{v}) = 0$
- $\sum_{j=1}^{m+1} v_j(\mathbf{v})A_j = \mathbf{v}$

The barycentric coordinates of point or vector with respect to the simplex can be easily computed as the solution of linear equations system. It is shown in the next theorem.

Theorem 2.1.1 *If we have simplex $S = (A_1, ..., A_{m+1})$ in E^m and point $P \in E^m$, then the barycentric coordinates for point P are unique and we can compute them using following formulas:*

$$u_j(P) = \frac{\begin{vmatrix} A_1 - A_{m+1} \\ \cdots \\ A_{j-1} - A_{m+1} \\ P - A_{m+1} \\ A_{j+1} - A_{m+1} \\ \cdots \\ A_m - A_{m+1} \end{vmatrix}}{\begin{vmatrix} A_1 - A_{m+1} \\ A_2 - A_{m+1} \\ \cdots \\ \\ A_m - A_{m+1} \end{vmatrix}}, \; j = 1, ..., m, \; u_{m+1}(P) = 1 - \sum_{j=1}^{m} u_j$$

This formulas shows the existence and uniques of barycentric coordinates.

Proof: Because points $A_1, ..., A_{m+1}$ don't lie in hyperspace of E^m, consecutive vectors $A_1 - A_{m+1}, ..., A_m - A_{m+1}$ are linearly independent, their number is m, so together they form a basis of the vector space $V(E^m)$. Then because $P - A_{m+1}$ is a vector from the vector space $V(E^m)$, there exist real numbers $e_1, ..., e_m$ such that

$$P - A_{m+1} = \sum_{j=1}^{m} e_j(A_j - A_{m+1}) \; (1)$$

$$P - A_{m+1} = \sum_{j=1}^{m} e_j A_j - \sum_{j=1}^{m} e_j A_{m+1}$$

$$P = \sum_{j=1}^{m} e_j A_j - \sum_{j=1}^{m} e_j A_{m+1} + A_{m+1} = \sum_{j=1}^{m} e_j A_j + (1 - \sum_{j=1}^{m} e_j) A_{m+1} = \sum_{j=1}^{m+1} u_j(P) A_j.$$

Now we can write $u_j(P) = e_j$ for $j = 1, ..., m$ and $u_{m+1}(P) = (1 - \sum_{j=1}^{m} e_j) = (1 - \sum_{j=1}^{m} u_j(P))$. For the computation of e_j and $u_j(P)$, we can use well-known

Cramer's rule to get solution of m linearly independent equations of m variables (1). Using Cramer's rule, we can get the analytical solution for $u_j(P)$. Application of Cramer's rule proves the theorem. $\square$

In the same way as above, we compute the barycentric coordinates of a vector $\mathbf{v} \in V(E^m)$. The only difference is that we use $\mathbf{v}$ instead of $P - A_{m+1}$ and $v_{m+1}(\mathbf{v}) = -\sum_{j=1}^{m} u_j(\mathbf{v})$. It can be easily shown that the points inside the simplex S have all barycentric coordinates nonnegative.

As we have seen in the previous definition and theorem, if we have exactly $m+1$ points in E^m that don't lie in one hyperplane, then these points forms some sort of basis for the points of E^m and the vectors of $V(E^m)$. But in many situations, we have more than $m+1$ points and we want to express the points in E^m as a affine combination of these points. For this, we will reuse the properties of barycentric coordinates to define generalized barycentric coordinates.

Definition 2.1.2 *Consider m-dimensional Euclidean space E^m. Let us have set of n vertices $A = \{A_1, ..., A_n\}; A_1, ..., A_n \in E^m$, where $n \geq (m+1)$ and there is $(m+1)$ points among $A_1, ..., A_n$ that don't lie in one hyperplane. For each point $P \in E^m$ we define its generalized barycentric coordinates as function $G : E^m \to R^n, G(P) = (g_1(P), ..., g_n(P))$ that posses following affine combination properties:*

- *Partition of unity: $\sum_{j=1}^{n} g_j(P) = 1$*
- *Linear precision: $\sum_{j=1}^{n} g_j(P)A_j = P$*

In this general case, we also need another properties of the generalized barycentric coordinates that are true for barycentric coordinates on simplices. The mostly used ones are

- Convex combination: For each P in convex hull of points A_j, all barycentric coordinates are non-negative, i.e. $g_j(P) \geq 0$.
- Lagrange property: if $P = A_j$, then $g_j(P) = 1$ and $g_i(P) = 0$ for $i \neq j$.
- Edge-preserving: If P is on the edge A_jA_{j+1}, then $g_i(P) = 0$ for $i \neq j$, $i \neq (j+1)$ and its generalized barycentric coordinates are reduced to usual linear interpolation between A_j and A_{j+1}

- Smoothness: $g_1(P), g_2(P), ..., g_n(P)$ must be infinitely differentiable with respect to P and the vertices of domain A. This ensures smoothness in the variation of the coefficients g_j when we move any vertex A_j.

When $n > m+1$ and we try to compute these generalized barycentric coordinates for a given point P, following the proof of Theorem 2.1.1, we get a system of m linearly independent equations of $n-1$ variables. Such system have infinite number of solutions, so we have to find some solution of this system with described properties. The computation algorithm for these generalized barycentric coordinates depends also on the dimension of space, we will need mostly generalized barycentric coordinates in E^2 and E^3, specially two kinds of them.

There were several attempts to compute generalized barycentric coordinates. We will give an overview of most used generalized barycentric coordinates without the proof of its properties. Loop and DeRose [LD89] used computation of generalized barycentric coordinates inside regular two-dimensional polygons for modeling multi-sided Bézier patches.

Theorem 2.1.2 *(Coordinates for regular polygons) Let us have regular n-sided polygon given by points $A_i \in E^2, i = 1, ..., n$. For $P \in E^2$, we define $\alpha_i(P) = \frac{V(PA_iA_{i+1})}{V(A_iA_{i+1}A_{i+2})}$ for $i = 1, ..., n$, where V is signed area of a triangle and $A_{n+1} = A_1, A_{n+2} = A_2$. Let $\pi_i(P)$ denote the product of all αs, except for $\alpha_{i-1}(P)$ and $\alpha_i(P)$:*

$$\pi_i(P) = \alpha_1(P)\alpha_2(P)...\alpha_{i-2}(P)\alpha_{i+1}(P)...\alpha_n(P),\ i = 1, ..., n$$

Let

$$g_i(P) = \frac{\pi_i(P)}{\pi_1(P) + \pi_2(P) + ... + \pi_n(P)}.$$

Then $G(P) = (g_1(P), ..., g_n(P))$ posses all properties of generalized barycentric coordinates. They also fulfill edge-preserving property.

Another attempt to define generalized barycentric coordinates for convex polygons was given by Pinkall and Polthier [PP93], but their coordinates lack the Convex combination property.

Theorem 2.1.3 *(Discrete harmonic coordinates) Let us have convex n-sided polygon Q given by points $A_i \in E^2, i = 1, ..., n$. For $P \in E^2$, we define weights $\omega_i(P) = \cot(\beta_{i-1}) + \cot(\gamma_i)$ where $\beta_{i-1} = \angle PA_{i-1}A_i$ and $\gamma_i = \angle PA_{i+1}A_i$, where*

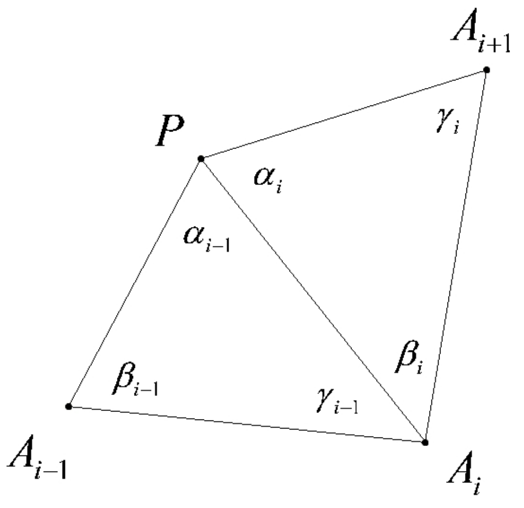


Figure 2.1: Angles in polygon used for computation of barycentric coordinates.

$A_{-1} = A_n$ and $A_{n+1} = A_1$, see Figure 2.1. Then normalization of these weights leads to generalized barycentric coordinates. If

$$g_i(P) = \frac{\omega_i(P)}{\omega_1(P) + \omega_2(P) + ... + \omega_n(P)},$$

then $G(P) = (g_1(P), ..., g_n(P))$ posses all properties of generalized barycentric coordinates except Convex combination property.

Wachspress [Wac75] proposed a construction that leads to analytical formulation of generalized barycentric coordinates for irregular two-dimensional convex polygons using the area of triangles, but for the computation of one a coordinate, almost all areas had to be computed. His work was extended by Meyer at al. [MLBD02] by delivering a local formula using only a few angles. These coordinates are extension of Loop's coordinates (Theorem 2.1.2) for irregular convex polygons.

Theorem 2.1.4 *(Wachspress coordinates) Let us have convex n-sided polygon Q given by points $A_i \in E^2, i = 1, ..., n$. For $P \in E^2$, we define weights*

$$\omega_i(P) = V(A_{i-1}A_iA_{i+1}) \prod_{k \notin (i,i+1)} V(A_{k-1}A_kP)$$

for $i = 1, 2, .., n$, where V is signed area of a triangle, $A_0 = A_n, A_{n+1} = A_1$. Then $G(P) = (g_1(P), ..., g_n(P))$ posses all properties of generalized barycentric coordinates, where $g_i(P) = \frac{\omega_i(P)}{\omega_1(P)+\omega_2(P)+...+\omega_n(P)}$. Weights $\omega_i(P)$ can be defined also locally.

If P is strictly within the polygon Q (does not lie on the border of the polygon), we can rewrite weight's as

$$\omega_i(P) = \frac{V(A_{i-1}A_iA_{i+1})}{V(A_{i-1}A_iP)V(A_iA_{i+1}P)} = \frac{\cot(\gamma_{i-1}) + \cot(\beta_i)}{\|P - A_i\|^2},$$

where $\gamma_{i-1} = \angle A_{i-1}A_iP$ and $\beta_i = \angle PA_iA_{i+1}$, see Figure 2.1.

Previous generalizations of barycentric coordinates work well for regular or for irregular convex polygons. Floater [Flo03] presented simple analytical formula for coordinates that are well defined for some subset of non-convex polygons, the star-shaped polygons, i.e. polygons with non-empty kernel.

Theorem 2.1.5 *(Mean value coordinates) Let us have star shaped n-sided polygon Q defined by points $A_i \in E^2, i = 1, ..., n$. For $P \in E^2$, we define*

$$\omega_i(P) = \frac{\tan(\frac{\alpha_{i-1}}{2}) + \tan(\frac{\alpha_i}{2})}{\|P - A_i\|} \quad i = 1, 2, .., n$$

$$g_i(P) = \frac{\omega_i(P)}{\omega_1(P) + \omega_2(P) + ... + \omega_n(P)},$$

where $\alpha_{i-1} = \angle A_{i-1}PA_i$ and $\alpha_i = \angle A_iPA_{i+1}$, see Figure 2.1. We put $A_0 = A_n$ and $A_{n+1} = A_1$. Then $G(P) = (g_1(P), ..., g_n(P))$ posses all properties of generalized barycentric coordinates.

Warren et al. [WSHD04] constructed an extension of Wachspress coordinates to E^3. These coordinates are well defined for convex polyhedron with arbitrary facets. These coordinates can be computed using polar duals, as was shown in [JSWD05].

Theorem 2.1.6 *(Warren's coordinates in E^3) Let us have closed convex polyhedron Q in E^3 defined by points $A_i \in E^3, i = 1, ..., n, n > 3$ and bounded by a set of 2-dimensional facets with outward unit normal vectors n_i. Let $ind(A_i)$ denote the set of facets indices that contain point A_i. For point $P \in Q$ and for point A_i with valence ($|ind(A_i)|$) equal to 3 (such vertices are called simple), we define*

$$\omega_i(P) = \frac{V(A_i)}{\prod_{j \in ind(A_i)} n_j(A_i - P)},$$

$$g_i(P) = \frac{\omega_i(P)}{\omega_1(P) + \omega_2(P) + ... + \omega_n(P)},$$

where $V(A_i)$ is the volume of the parallelepiped span by the normals to the facets incident with A_i. We can compute $V(A_i)$ as determinant of a matrix whose rows *are the vectors n_j where $j \in ind(A_i)$. For vertices with valence more than 3 (called non-simple vertices), the weight functions are constructed by infinitesimally perturbing the facets touching A_i such that the non-simple vertex is decomposed into simple vertices. The weight function is built by summing the weight function for the simple vertices together. Then normalization of these weights leads to generalized barycentric coordinates $G(P) = (g_1(P), ..., g_n(P))$ that posses all properties of generalized barycentric coordinates.*

Floater at al. [FKR05] and later Ju et al. [JS05] extended the mean value coordinates to work in E^3 for closed triangular meshes (polyhedron with triangular facets).

Theorem 2.1.7 *(Mean value coordinates in E^3) Let Q be a polyhedron, viewed as a closed region of E^3, with triangular facets and vertices $A_i \in E^3, i = 1, ..., n, n > 3$. Let K be the kernel of Q, the open set consisting of all points V in the interior $Int(Q)$ with the property that for all $i = 1, ..., n$, the only intersection between the line segment $[V, A_i]$ and the boundary δQ is A_i. If K is non-empty we say that Q is star-shaped. Assume that we have star shaped polyhedron Q with triangular facets and point $P \in K$. For vertex A_i and triangular facet $T = [A_i, A_j, A_k]$ that contains A_i, we define*

$$\mu_{i,T}(P) = \frac{\beta_{jk} + \beta_{ij}(n_{ij}.n_{jk}) + \beta_{ki}(n_{ki}.n_{jk})}{2(e_i.n_{jk})}$$

where β_{rs} is the angle between the two line segments $[P, A_r]$ and $[P, A_s]$, $e_i = \frac{1}{\|A_i - P\|} \cdot .(A_i - P)$ and n_{rs} denote the unit normal to the facet $[P, A_r, A_s]$, pointing into the tetrahedron $[P, A_i, A_j, A_k]$. See Figure 2.2 for explanation of notation. Now we can define weights

$$\omega_i(P) = \frac{1}{\|A_i - P\|} \sum_{T \ni A_i} \mu_{i,T}(P),$$

where sum is over all triangles that contain vertex A_i. Define normalization as

$$g_i(P) = \frac{\omega_i(P)}{\omega_1(P) + \omega_2(P) + ... + \omega_n(P)}.$$

Then $G(P) = (g_1(P), ..., g_n(P))$ posses all properties of generalized barycentric coordinates.

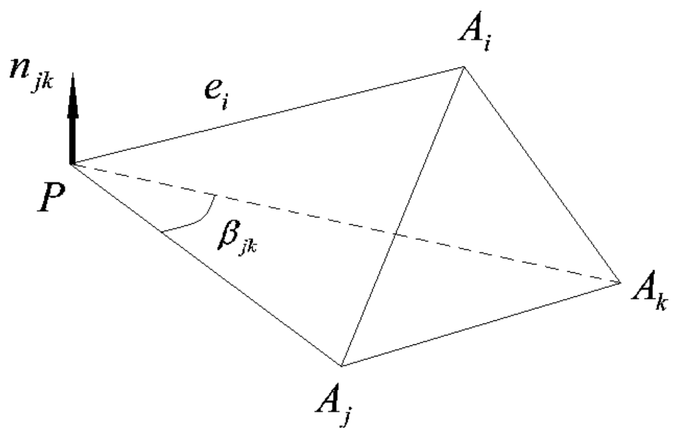


Figure 2.2: Parameters for computation of mean value coordinates in E^3

2.2 Bernstein polynomials

Bernstein polynomials are another way how to work with polynomials because they form a basis for the space of all polynomials with maximal degree n. The basic form of the Bernstein basis polynomial function is $B_i^n(t) = \frac{n!}{i!(n-i)!}t^i(1-t)^{n-i}$ for $0 \leq i \leq n$. Bernstein polynomials were polynomial functions used for approximation of continuous functions. We will use generalized form of Bernstein polynomials as blending functions for the definition of the parametric volumes. For that, we will need the multi-index notation.

Definition 2.2.1 *An n-dimensional multi-index is an n-tuple of non-negative integers: $\mathbf{i} = (i_1, i_2, ..., i_n)$, $i_1, ..., i_n \in \mathbb{N}_0$. For multi-index, we define*

- *Sum and difference:* $\mathbf{i} \pm \mathbf{j} = (i_1 \pm j_1, i_2 \pm j_2, ..., i_n \pm j_n)$
- *Multiplication: For* $k \in \mathbb{N}$, $k.\mathbf{i} = (k.i_1, k.i_2, ..., k.i_n)$.
- *Norm:* $|\mathbf{i}| = \sum_{k=1}^{n} i_k$
- *Basic multi-indices:* $\mathbf{e}_k$ *has components* $i_j = 0$ *for* $j \neq k$ *and* $i_k = 1$. *For example* $\mathbf{e}_1 = (1, 0, ..., 0)$.
- *Binomial coefficient:* $\binom{k}{\mathbf{i}} = \frac{k!}{\prod_{j=1}^{n} i_j!}$ *for* $k = |\mathbf{i}|$. *In computations, if some coefficient of multi-index is less then zero, binomial coefficient is equal to zero.*

We will use multi-index for definition of Bernstein basis functions and later also for definition of control nets for Bézier volumes. In these situations, it is convenient to know the exact number of multi-indices with given exact norm, for example to know the exact number of control points. This number is given in the following theorem.

Theorem 2.2.1 *Let us have n-dimensional multi-index $\mathbf{i}$ with norm $|\mathbf{i}| = k$. Define total number of multi-indices with norm equal to k as $N_{n,k}$. Then $N_{n,k} = \binom{n+k-1}{k}$.*

Proof: We will use mathematical induction for the parameter n. If $n = 1$, we have simple integer index that have to be equal to k, there is only one such index so $1 = N_{1,k}$. This base case is true because $1 = \binom{1+k-1}{k} = \binom{n+k-1}{k}$. Now for the inductive step, assume that the statement holds for n-dimensional multi-index, i.e. $N_{n,k} = \binom{n+k-1}{k}$. Now assume $(n+1)$-dimensional multi-index $(i_1, i_2, ..., i_n, i_{n+1})$ with norm equal to k. If $i_{n+1} = 0$, then $i_1 + i_2 + ... + i_n = k$ and by the induction hypothesis the number of such n-dimensional multi-indices is $\binom{n+k-1}{k}$. Similarly if $i_{n+1} = 1$, then $i_1 + i_2 + ... + i_n = k - 1$ and the number of such multi-indices is $\binom{n+k-2}{k-1}$. We can continue this way until $i_{n+1} = k$, when the number of multi-indices is $\binom{n-1}{0}$. Together we get $N_{n+1,k} = \binom{n+k-1}{k} + \binom{n+k-2}{k-1} + ... + \binom{n-1}{0} = \sum_{j=0}^{k} \binom{n+k-j-1}{k-j}$. Using the well-known identity for binomial coefficient $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ and the fact that $\binom{n-1}{0} = \binom{n}{0}$ we get

$$\begin{aligned}
N_{n+1,k} &= \binom{n-1}{0} + \binom{n}{1} + \binom{n+1}{2} + ... + \binom{n+k-2}{k-1} + \binom{n+k-1}{k} = \\
&= \left(\binom{n}{0} + \binom{n}{1}\right) + \binom{n+1}{2} + ... + \binom{n+k-2}{k-1} + \binom{n+k-1}{k} = \\
&= \left(\binom{n+1}{1} + \binom{n+1}{2}\right) + ... + \binom{n+k-2}{k-1} + \binom{n+k-1}{k} = \\
&= \binom{n+2}{2} + ... + \binom{n+k-2}{k-1} + \binom{n+k-1}{k} = ... = \\
&= \left(\binom{n+k-2}{k-2} + \binom{n+k-2}{k-1}\right) + \binom{n+k-1}{k} = \\
&= \binom{n+k-1}{k-1} + \binom{n+k-1}{k} = \binom{n+k}{k}.
\end{aligned}$$

This concludes the proof of statement for the inductive step. □

Definition 2.2.2 *Let us have k-dimensional multi-index $\mathbf{i} = (i_1, i_2, ..., i_k)$ with norm $n = |\mathbf{i}|$ and a k-tuple of real variables $\mathbf{u} = (u_1, u_2, ..., u_k); u_1, u_2, ..., u_k \in \mathbb{R}$ such that $\sum_{j=1}^{k} u_j = 1$ or $\sum_{j=1}^{k} u_j = 0$. Then k-variate Bernstein polynomial $B^n_\mathbf{i}(\mathbf{u}) : \mathbb{R}^k \to \mathbb{R}$ is a polynomial function of k variables and is defined as*

$$B^n_\mathbf{i}(\mathbf{u}) := \binom{n}{\mathbf{i}} \prod_{j=1}^{k} u_k^{i_k},$$

where $\binom{n}{\mathbf{i}}$ is the binomial coefficient. We define $0^0 = 1$ in the computation of these polynomials. The number n is called the degree of Bernstein polynomial.

In the notation of the Bernstein polynomials we do not mark the number of variables, it is explicitly defined in the dimension of multi-index. If this dimension is 2, i.e. if we deal with basic Bernstein polynomials, we often write Bernstein polynomial as $B^n_i(t) = \binom{n}{i} t^i (1-t)^{n-i}; 0 \le i \le n; t \in \mathbb{R}$.

Theorem 2.2.2 *k-variate Bernstein polynomials posses the following properties:*

a) If $\mathbf{u} = (u_1, u_2, ..., u_k)$ and $u_j \ge 0, j = 1, 2, ..., k$, then $B^n_\mathbf{i}(\mathbf{u}) \ge 0$ for each multi-index with $|\mathbf{i}| = \mathrm{n}$.

b) If $\mathbf{u} = (u_1, u_2, ..., u_k)$ and $u_j = 0$ for $j = 1, 2, ..., m-1, m+1, .., k$ and $u_m = 1$. Then $B^n_{n.\mathbf{e}_m}(\mathbf{u}) = 1$ and $B^n_\mathbf{i}(\mathbf{u}) = 0$ for every other k-dimensional multi-index $\mathbf{i}$.

c) $B^n_\mathbf{i}(\mathbf{u}) = \sum_{j=1}^{k} u_j B^{n-1}_{\mathbf{i}-\mathbf{e}_j}(\mathbf{u})$, where $|\mathbf{i}| = \mathrm{n}$.

d) $\sum_{|\mathbf{i}|=\mathrm{n}} B^n_\mathbf{i}(\mathbf{u}) = (u_1 + u_2 + ... + u_k)^n$

e) If $\mathbf{u} = (u_1, u_2, ..., u_k)$ and $\mathbf{i} = (i_1, i_2, ..., i_k)$, then $u_j = \sum_{|\mathbf{i}|=n} \frac{i_j}{n} B^n_\mathbf{i}(\mathbf{u})$ for each $j = 1, 2, ..., k$.

f) If $\mathbf{u} = (u_1, u_2, ..., u_k)$; $u_1 + u_2 + ... + u_k = 1$, then k-variate Bernstein polynomial of degree n can be written as the sum of k-dimensional Bernstein polynomials of degree $n+1$:

$$B^n_\mathbf{i}(\mathbf{u}) = \sum_{j=1}^{k} \frac{i_j + 1}{n+1} B^{n+1}_{\mathbf{i}+\mathbf{e}_j}(\mathbf{u})$$

g) If $\mathbf{u} = (u_1, u_2, ..., u_k), u_1 + u_2 + ... + u_k = 1$, the function $B^n_\mathbf{i}(\mathbf{u}) : \mathbb{R}^k \longrightarrow \mathbb{R}$ reaches its maximum for $\mathbf{u}$ with coefficients $u_j = \frac{i_j}{n}, j = 1, ..., k$.

Proof:

a) Binomial coefficient $\binom{n}{\mathbf{i}}$ is always non-negative and also $u_1^{i_1}, ..., u_k^{i_k} \geq 0$, then $B_{\mathbf{i}}^n(\mathbf{u}) := \binom{n}{\mathbf{i}} \prod_{j=1}^k u_k^{i_k} \geq 0$.

b) By definition, if $u_j = 0$ for $j = 1, 2, ..., m-1, m+1, .., k$ and $u_m = 1$, then

$$B_{n.\mathbf{e}_m}^n(\mathbf{u}) = \binom{n}{n.\mathbf{e}_m} u_1^{i_1} u_2^{i_2} .. u_m^{i_m} .. u_k^{i_k} = \binom{n}{n.\mathbf{e}_m} 0^0 0^0 .. 1^n .. 0^0 = 1,$$

because we defined $0^0 = 1$. For each other multi-index $\mathbf{i}$, at least one index from numbers $i_1, ..., i_{m-1}, i_{m+1}, ..., i_k$ is non-zero; let us mark it as i_l. Then

$$B_{\mathbf{i}}^n(\mathbf{u}) = \binom{n}{\mathbf{i}} u_1^{i_1} u_2^{i_2} .. u_m^{i_m} .. u_k^{i_k} = \binom{n}{\mathbf{i}} 0^0 .. 0^{i_l} .. 0^0 = 0.$$

c) By modification of the right side of the equation we get

$$\sum_{j=1}^k u_j B_{\mathbf{i}-\mathbf{e}_j}^{n-1}(\mathbf{u}) = \sum_{j=1}^k u_j \binom{n-1}{\mathbf{i}-\mathbf{e}_j} u_1^{i_1} .. u_j^{i_j-1} .. u_k^{i_k} =$$

$$= \sum_{j=1}^k \frac{(n-1)!}{i_1! .. (i_j-1)! .. i_k!} u_1^{i_1} .. u_j^{i_j} .. u_k^{i_k} = \sum_{j=1}^k \frac{i_j}{n} \frac{n!}{i_1! .. i_j! .. i_k!} u_1^{i_1} .. u_j^{i_j} .. u_k^{i_k} =$$

$$\sum_{j=1}^k \frac{i_j}{n} B_{\mathbf{i}}^n(\mathbf{u}) = B_{\mathbf{i}}^n(\mathbf{u}) \sum_{j=1}^k \frac{i_j}{n}$$

Now because $|\mathbf{i}| = \mathrm{n} = \sum_{\mathrm{j}=1}^{\mathrm{k}} \mathrm{i_j}$, we get

$$\sum_{j=1}^k u_j B_{\mathbf{i}-\mathbf{e}_j}^{n-1}(\mathbf{u}) = B_{\mathbf{i}}^n(\mathbf{u}) \sum_{j=1}^k \frac{i_j}{n} = B_{\mathbf{i}}^n(\mathbf{u})$$

d) We use mathematical induction with respect to n to prove that $\sum_{|\mathbf{i}|=\mathrm{n}} B_{\mathbf{i}}^n(\mathbf{u}) = (u_1 + u_2 + ... + u_k)^n$

1. (Base case) For $n = 0$ we get $\sum_{|\mathbf{i}|=0} B_{\mathbf{i}}^0(\mathbf{u}) = \frac{0!}{0!...0!} u_1^0 ... u_k^0 = 1 = (u_1 + u_2 + ... + u_k)^0$

2. (Inductive step) Assume that the statement holds for n, we show that it holds also for $(n+1)$ using the statement for n:

$$(u_1 + u_2 + ... + u_k)^{n+1} = (u_1 + u_2 + ... + u_k)(u_1 + u_2 + ... + u_k)^n =$$

$$= (u_1+u_2+...+u_k)\sum_{|\mathbf{i}|=\mathrm{n}} B^n_{\mathbf{i}}(\mathbf{u}) = (u_1+u_2+...+u_k)\sum_{|\mathbf{i}|=\mathrm{n}} \frac{n!}{i_1!...i_k!}u_1^{i_1}...u_k^{i_k} =$$

$$= \sum_{|\mathbf{i}|=\mathrm{n}} \frac{n!}{i_1!...i_k!}u_1^{i_1+1}...u_k^{i_k} + ... + \sum_{|\mathbf{i}|=\mathrm{n}} \frac{n!}{i_1!...i_k!}u_1^{i_1}...u_k^{i_k+1} =$$

$$= \sum_{|\mathbf{i}|=\mathrm{n}} \frac{i_1+1}{n+1}\frac{(n+1)!}{(i_1+1)!..i_k!}u_1^{i_1+1}..u_k^{i_k}+..+\sum_{|\mathbf{i}|=\mathrm{n}} \frac{i_k+1}{n+1}\frac{(n+1)!}{i_1!..(i_k+1)!}u_1^{i_1}..u_k^{i_k+1} =$$

$$= \sum_{|\mathbf{i}|=\mathrm{n}} \frac{i_1+1}{n+1}B^{n+1}_{\mathbf{i+e_1}}(\mathbf{u}) + ... + \sum_{|\mathbf{i}|=\mathrm{n}} \frac{i_k+1}{n+1}B^{n+1}_{\mathbf{i+e_k}}(\mathbf{u})$$

Now using the transformation of indices we get:

$$(u_1+u_2+...+u_k)^{n+1} = \sum_{|\mathbf{i}|=\mathrm{n+1}} \frac{i_1}{n+1}B^{n+1}_{\mathbf{i}}(\mathbf{u})+...+\sum_{|\mathbf{i}|=\mathrm{n+1}} \frac{i_k}{n+1}B^{n+1}_{\mathbf{i}}(\mathbf{u}) =$$

$$= \sum_{|\mathbf{i}|=\mathrm{n+1}} B^{n+1}_{\mathbf{i}}(\mathbf{u})\frac{i_1+...+i_k}{n+1} = \sum_{|\mathbf{i}|=\mathrm{n+1}} B^{n+1}_{\mathbf{i}}(\mathbf{u})$$

So the inductive step is proven and the statement holds for each $n \in \mathbb{N}$.

e) Computation of the right side leads to

$$\sum_{|\mathbf{i}|=n} \frac{i_j}{n}B^n_{\mathbf{i}}(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \frac{i_j}{n}\frac{n!}{i_1!i_2!...i_j!...i_k!}u_1^{i_1}u_2^{i_2}...u_k^{i_k} =$$

$$= u_j \sum_{|\mathbf{i}|=n,i_j>0} \frac{(n-1)!}{i_1!i_2!...(i_j-1)!...i_k!}u_1^{i_1}u_2^{i_2}...u_j^{i_j-1}...u_k^{i_k} =$$

$$= u_j \sum_{|\mathbf{i}|=n,i_j>0} B^{n-1}_{\mathbf{i}} = u_j \sum_{|\mathbf{i}|=n-1} B^{n-1}_{\mathbf{i}} = u_j$$

where we used a simple transformation of multi-index and property d) of k-variate Bernstein polynomials.

f) Because $u_1 + u_2 + ... + u_k = 1$, we can write

$$B^n_{\mathbf{i}}(\mathbf{u}) = B^n_{\mathbf{i}}(\mathbf{u})\sum_{j=1}^{k} u_j = \sum_{j=1}^{k} u_j\frac{n!}{i_1!i_2!...i_k!}u_1^{i_1}...u_j^{i_j}...u_k^{i_k} =$$

$$= \sum_{j=1}^{k} \frac{i_j}{n}\frac{(n+1)!}{i_1!...(i_j+1)!...i_k!}u_1^{i_1}...u_j^{i_j+1}...u_k^{i_k} = \sum_{j=1}^{k} \frac{i_j+1}{n+1}B^{n+1}_{\mathbf{i+e}_j}(\mathbf{u})$$

g) If we have $i_j = 0$ for some $j = 0, 1, ..., k$, then the maximum of $B^n_{\mathbf{i}}$ is reached for $u_j = 0 = \frac{i_j}{n}$, because $\binom{n}{\mathbf{i}} u_1^{i_1}..u_j^0..u_k^{i_k} = \binom{n}{\mathbf{i}} u_1^{i_1}..0^0..u_k^{i_k}$. So we can assume that $i_j \neq 0$ for all $j = 0, 1, ..., k$. Using the method of Lagrange multipliers we define function $L(\mathbf{u}) = B^n_{\mathbf{i}}(\mathbf{u}) + \lambda(u_1 + ... + u_k - 1)$ for $\lambda \in R$. We have to find the stationary points for the Lagrange function $L(\mathbf{u})$. Computing the partial derivative we get

$$\frac{\partial}{\partial u_j} L(\mathbf{u}) = \frac{\partial}{\partial u_j} B^n_{\mathbf{i}}(\mathbf{u}) + \frac{\partial}{\partial u_j} \lambda(u_1 + u_2 + ... + u_k - 1) =$$

$$= \frac{\partial}{\partial u_j} \frac{n!}{i_1!...i_j!...i_k!} u_1^{i_1}...u_j^{i_j}...u_k^{i_k} + \lambda = n \frac{(n-1)!}{i_1!...(i_j-1)!...i_k!} u_1^{i_1}...u_j^{i_j-1}...u_k^{i_k} + \lambda =$$

$$= n\Big[B^{n-1}_{\mathbf{i}-\mathbf{e}_j}(\mathbf{u})\Big] + \lambda$$

To find the stationary points of $L(\mathbf{u})$, we put all partial derivatives equal to zero. Then

$$\frac{\partial}{\partial u_1} L(\mathbf{u}) = \frac{\partial}{\partial u_2} L(\mathbf{u}) = ... = \frac{\partial}{\partial u_k} L(\mathbf{u}) = 0$$

$$nB^{n-1}_{\mathbf{i}-\mathbf{e}_1}(\mathbf{u}) + \lambda = nB^{n-1}_{\mathbf{i}-\mathbf{e}_2}(\mathbf{u}) + \lambda = ... = nB^{n-1}_{\mathbf{i}-\mathbf{e_k}}(\mathbf{u}) + \lambda$$

$$n\frac{(n-1)!}{(i_1-1)!...i_k!} u_1^{i_1-1}...u_k^{i_k} = ... = n\frac{(n-1)!}{i_1!..(i_k-1)!} u_1^{i_1}...u_k^{i_k-1}$$

$$i_1 u_1^{i_1-1}...u_k^{i_k} = ... = i_k u_1^{i_1}...u_k^{i_k-1}$$

To get the maximum, all u_i must be non-zero, because if at least one u_i is equal to zero, we get $B^n_{\mathbf{i}}(\mathbf{u}) = 0$, and this is not the maximum. So we can write

$$i_1 \frac{u_1^{i_1}...u_k^{i_k}}{u_1} = ... = i_k \frac{u_1^{i_1}...u_k^{i_k}}{u_k}$$

$$\frac{i_1}{u_1} = ... = \frac{i_k}{u_k} = S,\ S \in R$$

$$u_j = \frac{i_j}{S},\ j = 1, .., k;\ \sum_{j=1}^{k} u_j = 1 = \sum_{j=1}^{k} \frac{i_j}{S} \Longrightarrow S = \sum_{j=1}^{k} i_j = n \Longrightarrow u_j = \frac{i_j}{n}$$

□

Now we can move to the definition and the properties of the directional derivatives for Bernstein polynomials.

Definition 2.2.3 *Le us have vector* $\mathbf{d}$ *with barycentric coordinates* $\mathbf{d} = (v_1, v_2, ..., v_k)$, $\sum_{i=1}^{k} v_i = 0$. *Then we can define the directional derivative of* k*-variate Bernstein polynomial of* r*-th order in the direction* $\mathbf{d}$ *recursively as*

$$D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) = v_1 \frac{\partial}{\partial u_1} D_{\mathbf{d}}^{r-1} B_{\mathbf{i}}^{n}(\mathbf{u}) + v_2 \frac{\partial}{\partial u_2} D_{\mathbf{d}}^{r-1} B_{\mathbf{i}}^{n}(\mathbf{u}) \ + ... + v_k \frac{\partial}{\partial u_k} D_{\mathbf{d}}^{r-1} B_{\mathbf{i}}^{n}(\mathbf{u}),$$

where $r = 1, ..., n$ *and* $D_{\mathbf{d}}^{0} B_{\mathbf{i}}^{n}(\mathbf{u}) = B_{\mathbf{i}}^{n}(\mathbf{u})$.

Theorem 2.2.3 *The formula for the computation of the directional derivative of* k*-variate Bernstein polynomial of* r*-th order in the direction* $\mathbf{d}$ *is:*

$$D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) = \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B_{\mathbf{j}}^{r}(\mathbf{d}) B_{\mathbf{i}-\mathbf{j}}^{n-r}(\mathbf{u})$$

Proof: Using mathematical induction for variable r:

1. For $r = 0$ we have

$$B_{\mathbf{i}}^{n}(\mathbf{u}) = D_{\mathbf{d}}^{0} B_{\mathbf{i}}^{n}(\mathbf{u}) = \frac{n!}{(n-0)!} \sum_{|\mathbf{j}|=0} B_{\mathbf{j}}^{0}(\mathbf{d}) \mathbf{B}_{\mathbf{i}-\mathbf{j}}^{\mathbf{n}-\mathbf{0}}(\mathbf{u})$$

2. From the definition:

$$D_{\mathbf{d}}^{r+1} B_{\mathbf{i}}^{n}(\mathbf{u}) = v_1 \frac{\partial}{\partial u_1} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) + v_2 \frac{\partial}{\partial u_2} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) + ... + v_k \frac{\partial}{\partial u_k} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u})$$

Now for part $\frac{\partial}{\partial u_l} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u})$; $l = 1, ..., k$, we use induction hypothesis:

$$\frac{\partial}{\partial u_l} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) = \frac{\partial}{\partial u_l} \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B_{\mathbf{j}}^{r}(\mathbf{d}) B_{\mathbf{i}-\mathbf{j}}^{n-r}(\mathbf{u}) =$$

$$\frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B_{\mathbf{j}}^{r}(\mathbf{d}) \frac{\partial}{\partial u_l} B_{\mathbf{i}-\mathbf{j}}^{n-r}(\mathbf{u})$$

Using the partial derivative of Bernstein polynomial we get

$$\frac{\partial}{\partial u_l} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) = \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B_{\mathbf{j}}^{r}(\mathbf{d}) (n-r) \left[B_{\mathbf{i}-\mathbf{j}-\mathbf{e}_l}^{n-r-1}(\mathbf{u}) \right] =$$

$$= \frac{n!}{(n-(r+1))!} \sum_{|\mathbf{j}|=r} B_{\mathbf{j}}^{r}(\mathbf{d}) B_{\mathbf{i}-\mathbf{j}-\mathbf{e}_l}^{n-r-1}(\mathbf{u}).$$

Because we evaluated all $\frac{\partial}{\partial u_l} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u})$, we can compute $D_{\mathbf{d}}^{r+1} B_{\mathbf{i}}^{n}(\mathbf{u})$:

$$D_{\mathbf{d}}^{r+1} B_{\mathbf{i}}^{n}(\mathbf{u}) = v_1 \frac{\partial}{\partial u_1} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) + v_2 \frac{\partial}{\partial u_2} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) + ... + v_k \frac{\partial}{\partial u_k} D_{\mathbf{d}}^{r} B_{\mathbf{i}}^{n}(\mathbf{u}) =$$

$$= \frac{n!}{(n-(r+1))!}\Big[v_1 \sum_{|\mathbf{j}|=r} B^r_{\mathbf{j}}(\mathbf{d}) B^{n-r-1}_{\mathbf{i}-\mathbf{j}-\mathbf{e}_1}(\mathbf{u}) + v_2 \sum_{|\mathbf{j}|=r} B^r_{\mathbf{j}}(\mathbf{d}) B^{n-r-1}_{\mathbf{i}-\mathbf{j}-\mathbf{e}_2}(\mathbf{u}) + ...+$$

$$+ v_k \sum_{|\mathbf{j}|=r} B^r_{\mathbf{j}}(\mathbf{d}) B^{n-r-1}_{\mathbf{i}-\mathbf{j}-\mathbf{e}_k}(\mathbf{u})\Big]$$

The transformation of multi-index $\mathbf{j} + \mathbf{e}_l$ for $l = 1, ..., k$ yields to

$$D^{r+1}_{\mathbf{d}} B^n_{\mathbf{i}}(\mathbf{u}) = \frac{n!}{(n-(r+1))!}\Big[v_1 \sum_{|\mathbf{j}|=r+1} B^r_{\mathbf{j}-\mathbf{e}_1}(\mathbf{d}) B^{n-r-1}_{\mathbf{i}-\mathbf{j}}(\mathbf{u}) +$$

$$+ v_2 \sum_{|\mathbf{j}|=r+1} B^r_{\mathbf{j}-\mathbf{e}_2}(\mathbf{d}) B^{n-r-1}_{\mathbf{i}-\mathbf{j}}(\mathbf{u}) + ... + v_k \sum_{|\mathbf{j}|=r+1} B^r_{\mathbf{j}-\mathbf{e}_k}(\mathbf{d}) B^{n-r-1}_{\mathbf{i}-\mathbf{j}}(\mathbf{u})\Big] =$$

$$= \frac{n!}{(n-(r+1))!} \sum_{|\mathbf{j}|=r+1} B^{n-(r+1)}_{\mathbf{i}-\mathbf{j}}(\mathbf{u}) \Big[v_1 B^r_{\mathbf{j}-\mathbf{e}_1}(\mathbf{d}) + ... + v_k B^r_{\mathbf{j}-\mathbf{e}_k}(\mathbf{d})\Big].$$

Using Theorem 2.2.2(c) we get:

$$D^{r+1}_{\mathbf{d}} B^n_{\mathbf{i}}(\mathbf{u}) = \frac{n!}{(n-(r+1))!} \sum_{|\mathbf{j}|=r+1} B^{r+1}_{\mathbf{j}}(\mathbf{d}) B^{n-(r+1)}_{\mathbf{i}-\mathbf{j}}(\mathbf{u})$$

$\square$

2.3 Bézier volumes

We will work with several types of Bézier volumes derived from Bézier tensor-product surfaces, Bézier triangles and generalized S-patches. Their domains sets will have the shape of cuboid, tetrahedron or convex polyhedron. All of them will have several properties that will arise from the usage of k-variate Bernstein polynomials as blending functions. We will follow the work of Lasser [Las85], who introduced the tensor-product volumes in Bernstein-Bézier form. We presented several construction for Bézier volumes in [Sam04b].

2.3.1 Bézier tetrahedron

At the beginning, let us state the definition of Bézier curve and Bézier triangle, on which the principles for the definition of Bézier tetrahedron are based. For Bézier triangles we have

- The domain D of Bézier curve is a line segment AB, for Bézier triangle it is a triangle ΔABC (simplex in E^2), where $A, B, C \in E^2$ are three non-collinear points.
- The degree of Bézier curve or Bézier triangle is an integer n that express the degree of polynomials that define the Bézier object.
- The control net of Bézier curve is a set of points $V_{\mathbf{i}} \in E^3$ that form the shape of Bézier curve. Multi-index $\mathbf{i}$ is 2-dimensional, its norm is $n = |\mathbf{i}|$ and the total number of control points is $n+1$.
- The control net of Bézier triangle is set of points $V_{\mathbf{i}} \in E^3$ that form the shape of Bézier triangle. Multi-index $\mathbf{i}$ is 3-dimensional, its norm is $n = |\mathbf{i}|$ and the total number of control points is $\binom{n+2}{3}$(see Theorem 2.2.1).
- Bézier curve BC^n is defined for each point P in the domain line segment $AB = D$ as a blending of the control points using 2-variate Bernstein polynomials

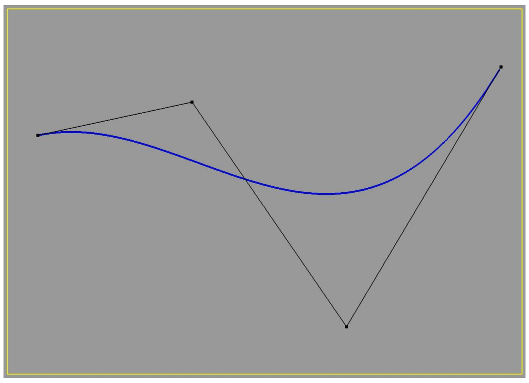
  $$BC^n(P) = \sum_{dim(\mathbf{i})=2, |\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(U(P)),$$

  where U is a function of barycentric coordinates with respect to the line segment (simplex) AB and is computed using the formula from Theorem 2.2.1.
- Bézier triangle BT^n is defined for each point P in the domain triangle $ABC = D$ as a blending of the control points using 3-variate Bernstein polynomials

  $$BT^n(P) = \sum_{dim(\mathbf{i})=3, |\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(U(P)),$$

  where U is a function of barycentric coordinates with respect to the triangle (simplex) ABC and is computed using the formula from Theorem 2.1.1.

This gives us the formula for computation of points that belong to Bézier curve or triangle. The points of objects are well defined, because $\sum_{|\mathbf{i}|=n} B^n_{\mathbf{i}}(U(P)) = 1$ and so we can perform an affine combination of the control points. The domain is a general line segment or triangle so we can construct for example continuous set of Bézier triangles over a triangulation in E^2. The shape or length of the domain does not have impact on the shape of the Bézier object, it is used to get the barycentric coordinates and substitute them into Bernstein polynomials. Figure 2.3 shows examples of Bézier


(a) Bézier curve of degree 3 with control points.

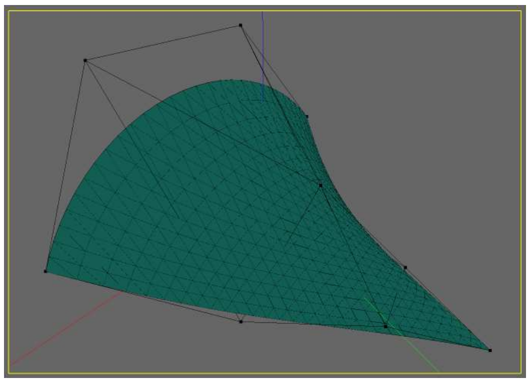

(b) Bézier triangle of degree 3 over net of control points.

Figure 2.3: Bézier curve and Bézier triangle.

curve and triangle. If we take 3-dimensional simplex (tetrahedron) as the domain for the function containing Bernstein blending polynomials, we will get 3-dimensional object just like in case of the triangle. For this we need the appropriate number of control points. Such construction leads to the definition of Bézier tetrahedron (see Figure 2.4).

Definition 2.3.1 *(Bézier tetrahedron) Let us have the degree $n \in N$, a tetrahedron given by four non-coplanar points $A, B, C, D \in E^3$, control points $V_{\mathbf{i}} \in E^3; dim(\mathbf{i}) = 4; |\mathbf{i}| = n$, then a Bézier tetrahedron BTH^n is given analytically for point $P \in ABCD$ as*

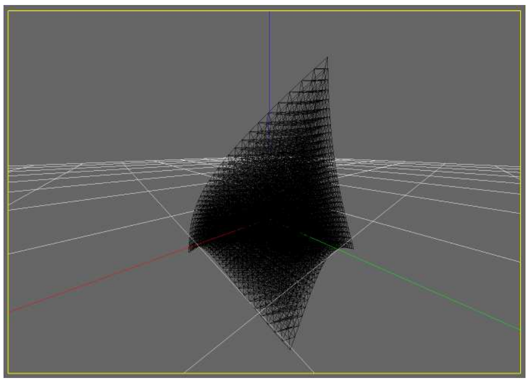
$$BTH^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(U(P)),$$

where U is a function of barycentric coordinates with respect to the tetrahedron (simplex) $ABCD$ and is computed using the formula from Theorem 2.2.1.

Bézier tetrahedron possesses many properties and constructions that are similar to properties of Bézier curves or triangles. We will show many of them in next section for the generalization of Bézier tetrahedra in the form of S-volumes.

2.3.2 S-volume

Loop and DeRose [LD89] proposed a generalization of Bézier triangles with multiple sides, not just three. They used k-variate Bernstein polynomials for blending the net of control points and to get new kind of surface. As a domain they used regular


(a) Wireframe visualization.

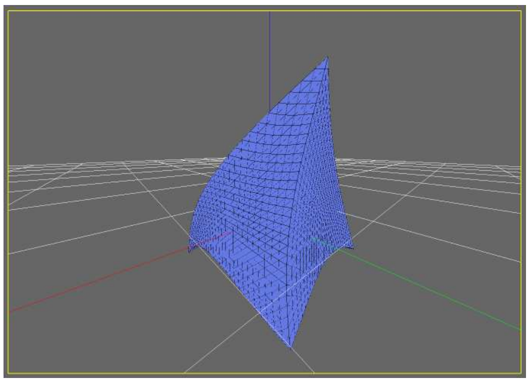

(b) Wireframe and solid visualization.

Figure 2.4: Bézier tetrahedron.

2-dimensional n-sided polygon and for each point inside the domain they computed the barycentric coordinates described in Theorem 2.1.2. They call these representations "S-patches" (Figure 2.5) to emphasize their connection to the theory of Bézier simplexes. Langer and Seidel [LS07] used the mean value coordinates (Theorem 2.1.5) to construct parametric surfaces.

We will use similar approach to formulate the construction of volumes based on polyhedral domain with multiple facets and Bernstein polynomials as blending functions. We call these objects S-volumes.

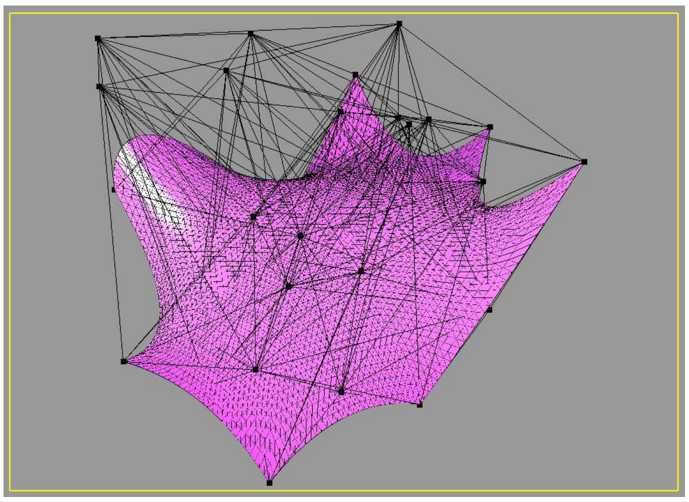
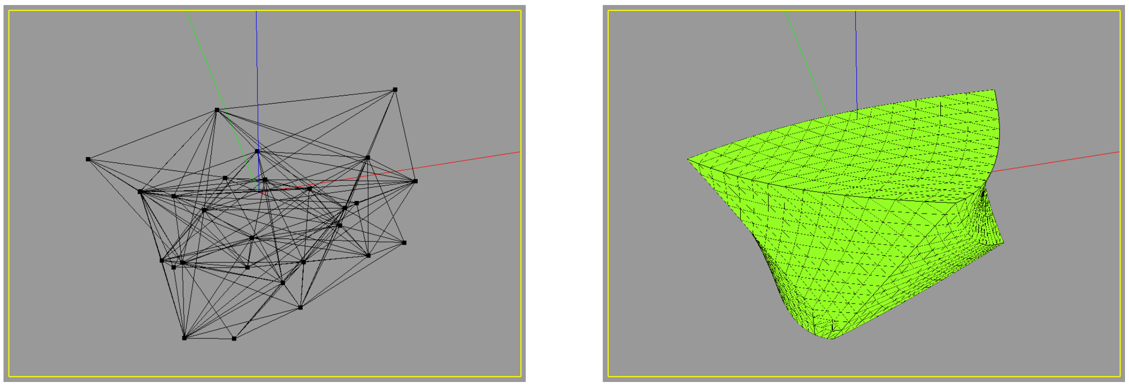


Figure 2.5: 8-sided S-patch of degree 2 with displayed net of control points.

Definition 2.3.2 *(S-volume) Let us have a polyhedron D defined by a set of non-coplanar points $A_1, A_2, ..., A_k \in E^3$ and facets. Let us have the degree $n \in \mathbb{N}$, control*



(a) Net of control points

(b) S-volume visualized using boundary patches.

Figure 2.6: S-volume of degree 2 over domain given by 7 points and 10 triangles.

points $V_{\mathbf{i}} \in E^3; dim(\mathbf{i}) = k; |\mathbf{i}| = n$. Then a S-volume SV^n with domain D, degree n and control points $V_{\mathbf{i}}$ is defined analytically for a point $P \in D$ as

$$SV^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(G(P)),$$

where G is function of barycentric coordinates with respect to the polyhedron D.

Because $G(P)$ holds the partition of unity property, Theorem 2.2.2(d) gives that $SV^n(P)$ is an affine combination of control points and is well defined. The domain D must have sufficient properties to use a proper generalization of barycentric coordinates in E^3. We presented two such barycentric coordinates - Warren's coordinates in Theorem 2.1.6 for convex polyhedron and Mean value coordinates in Theorem 2.1.7 for star-shaped polyhedron with triangular facets. In our work we used Mean value coordinates. The mostly used domains are tetrahedron or convex polyhedron with triangular facets is illustrated. In Figure 2.6 S-volume over polyhedral domain with seven triangular facets. Now we will show several properties of S-volumes. All these theorems are true also for Bézier curves and triangles, so these properties are fulfilled for them as well.

Theorem 2.3.1 *(Affine invariant) S-volume is invariant under an affine transformation, i.e. if $A : E^3 \to E^3$ is an affine transformation, then*

$$A\Big(\sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(G(P)) \Big) = \sum_{|\mathbf{i}|=n} A(V_{\mathbf{i}}) B^n_{\mathbf{i}}(G(P))$$

Proof: We know that an affine combination of points is invariant under affine transformation. But $SV^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(G(P))$ is an affine combination of control points $V_{\mathbf{i}}$, because barycentric coordinates G forms a partition of unity and Theorem 2.2.2 d) gives us that $\sum_{|\mathbf{i}|=n} B^n_{\mathbf{i}}(G(P)) = 1$. $\square$

Previous theorem simply states that to transforming whole volume, all we need to do is to transform its control points.

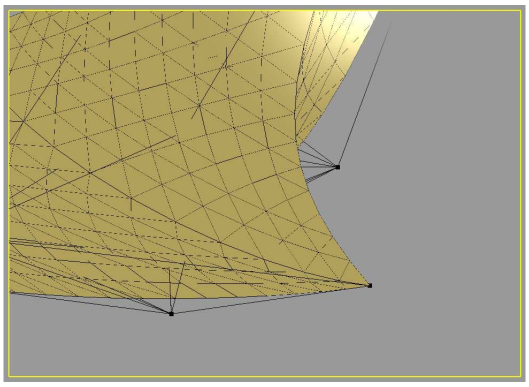
Theorem 2.3.2 *If barycentric coordinates G fulfill the convex combination property and domain D of S-volume is convex, then for each $P \in D$, $SV^n(P)$ lies inside convex hull of control points $V_{\mathbf{i}}$, $|\mathbf{i}| = n$.*

Proof: If barycentric coordinates G fulfill the convex combination property and the domain D of S-volume is convex, then all coordinates of $G(P)$ are non-negative for $P \in D$. Then by Theorem 2.2.2(a) $B^n_{\mathbf{i}}(G(P)) \geq 0$ and then $SV^n(P)$ is a convex affine combination of control points $V_{\mathbf{i}}$, $|\mathbf{i}| = n$. So $B^n(\mathbf{u})$ lies in the convex hull of control points $V_{\mathbf{i}}$, $|\mathbf{i}| = n$. $\square$

Theorem 2.3.3 *If barycentric coordinates G fulfill the Lagrange property, then control points $V_{n.\mathbf{e}_j}$ for $j = 1, 2, ..., n$ are inside of S-volume $SV^n(P)$. We call them corner points (Figure 2.7(a)).*

Proof: Let's take point the A_j from the domain D. Because of the Lagrange property, its barycentric coordinates are $(0, 0, ..., 0, 1, 0, ..., 0)$ with 1 in j-th place and 0 otherwise. Then $SV^n(A_j) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(0, 0, ..., 0, 1, 0, ..., 0)$. Now using the property of k-variate Bernstein polynomial from Theorem 2.2.2(b), we get that all Bernstein polynomials in the given sum are equal to zero except one, for index $\mathbf{i} = n.\mathbf{e}_j$. So we get $SV^n(A_j) = V_{n.\mathbf{e}_j}$. $\square$

Theorem 2.3.4 *If we have S-volume SV^n and if the used generalized barycentric coordinates fulfill the edge-preserving property, then the boundary curves of that S-volume are Bézier curves. If generalized the barycentric coordinates fulfill facet-preserving property, the boundary patches are S-patches (Figure 2.7(b)). The facet-preserving property of the generalized barycentric coordinates $G = (g_1, .., g_k)$ is defined following way: If P is on the facet $A_{p_1}..A_{p_m}$, then $g_i(P) = 0$ for $i \neq p_j$, $i \neq (j + 1)$ and its generalized barycentric coordinates are reduced to the generalized barycentric coordinates with respect to points $A_{p_1}..A_{p_m}$.*


(a) Corner points lies in S-volume

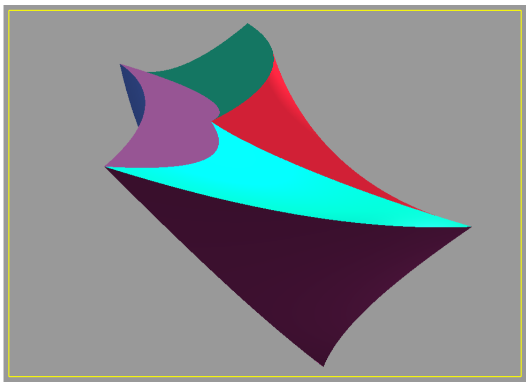

(b) Boundary patches are S-patches (Bézier triangles in this case).

Figure 2.7: Properties of S-volume.

Proof: Let's take the edge A_pA_r from the domain polyhedron D and a point P on that edge. Because the barycentric coordinates $G = (g_1, g_2, ..., g_k)$ fulfill the edge-preserving property, we can write the barycentric coordinates of P as $g_p(P) = t, g_r(P) = 1 - t$ for some p, r and some $t \in \langle 0, 1\rangle$ and all other coordinates are equal to zero. Then the S-volume is reduced to

$$SV^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(G(P)) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \frac{n!}{i_1!..i_k!} g_1(P)^{i_1}...g_k(P)^{i_k} =$$

$$= \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \frac{n!}{i_1!..i_k!} 0^{i_1}..t^{i_p}..(1-t)^{i_r}..0^{i_k} = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \frac{n!}{0!..0!i_p!i_r!} t^{i_p}(1-t)^{i_r} =$$

$$= \sum_{i=0}^{n} V_{(0,..,i,..,n-i,..,0)} \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} = \sum_{i=0}^{n} V_{(0,..,i,..,n-i,..,0)} B^n_i(t).$$

We got the description of the the boundary edge curve in the Bézier form with control points $W_i = V_{(0,..,i,..,n-i,..,0)}; i = 0, ..., n$, where i and $n - i$ are in the multi-index on the p-th and the r-th place.

Similarly if P is on the boundary facet given by points $A_{p_1}A_{p_2}...A_{p_m}; m < k$ of the domain D, then because the barycentric coordinates $G = (g_1, g_2, ..., g_k)$ fulfill the facet-preserving property, we can write P as $P = \sum_{i=1}^{m} g_{p_i}(P)A_{p_i}$, so all coordinates except $g_{p_1}(P), ..., g_{p_m}(P)$ are equal to zero. Then

$$SV^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(G(P)) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \frac{n!}{i_1!..i_k!} g_1(P)^{i_1}...g_k(P)^{i_k} =$$

$$= \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \frac{n!}{\prod_{j=1}^{m} i_{p_j}!} \prod_{j=1}^{m} g_{p_j}(P)^{i_{p_j}} = \sum_{dim(\mathbf{j})=m, |\mathbf{j}|=n} W_{\mathbf{j}} B_{\mathbf{j}}^{n}(g_{p_1}(P), .., g_{p_m}(P)),$$

where $W_{(j_1, j_2, .., j_m)} = V_{j_1.\mathbf{e}_{p_1} + .. + j_m.\mathbf{e}_{p_m}}$. This implies that the boundary patch is a S-patch with m sides, degree n and control points $W_{\mathbf{j}}$. □

Theorem 2.3.5 *(Degree elevation) S-volume is a polynomial function of total degree n. It is also possible to write it as a polynomial function of total degree $n+1$ with new set of control points*

$$SV^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B_{\mathbf{i}}^{n}(G(P)) = \sum_{|\mathbf{i}|=n+1} W_{\mathbf{i}} B_{\mathbf{i}}^{n+1}(G(P)),$$

where $W_{\mathbf{i}} = \sum_{j=1}^{k} \frac{i_j}{n+1} V_{\mathbf{i}-\mathbf{e}_j}$ for $\mathbf{i} = (i_1, i_2, ..., i_k)$; $|\mathbf{i}| = n+1$.

Proof: Because Bernstein polynomials are of degree n, also a linear combination of these polynomials is of degree n. As seen in Theorem 2.2.2 f) we can increase the degree of Bernstein polynomials. Then computation of S-volume leads to

$$SV^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B_{\mathbf{i}}^{n}(G(P)) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \sum_{j=1}^{k} \frac{i_j + 1}{n+1} B_{\mathbf{i}+\mathbf{e}_j}^{n+1}(G(P)) =$$

$$= \sum_{j=1}^{k} \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \frac{i_j + 1}{n+1} B_{\mathbf{i}+\mathbf{e}_j}^{n+1}(G(P)).$$

Now using a transformation of the multi-index we get

$$SV^n(P) = \sum_{j=1}^{k} \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \frac{i_j + 1}{n+1} B_{\mathbf{i}+\mathbf{e}_j}^{n+1}(G(P)) = \sum_{j=1}^{k} \sum_{|\mathbf{i}|=n+1} V_{\mathbf{i}-\mathbf{e}_j} \frac{i_j}{n+1} B_{\mathbf{i}}^{n+1}(G(P)) =$$

$$= \sum_{|\mathbf{i}|=n+1} \Big[\sum_{j=1}^{k} V_{\mathbf{i}-\mathbf{e}_j} \frac{i_j}{n+1} \Big] B_{\mathbf{i}}^{n+1}(G(P)) = \sum_{|\mathbf{i}|=n+1} W_{\mathbf{i}} B_{\mathbf{i}}^{n+1}(G(P)).$$

□

Theorem 2.3.6 *(de Casteljau algorithm) A point of S-volume $SV^n(P)$ for a domain point $P \in D$ can be evaluated using a recursive algorithm. Put $V_{\mathbf{i}}^{0}(P) = V_{\mathbf{i}}$ for each k-dimensional multi-index $\mathbf{i}$ with norm $|\mathbf{i}| = n$. Define new points as*

$$V_{\mathbf{i}}^{r}(P) = \sum_{j=1}^{k} g_j(P) V_{\mathbf{i}+\mathbf{e}_j}^{r-1}(P)$$

for $r = 1, 2, ..., n$ and for $|\mathbf{i}| = n - r$. $g_j(P)$ are the coefficients of the barycentric coordinates G of point P. Then $SV^n(P) = V^r_{\mathbf{0}}(P)$. More generally we can compute the new points using formula

$$V^r_{\mathbf{i}}(P) = \sum_{|\mathbf{j}|=r} V_{\mathbf{i}+\mathbf{j}} B^r_{\mathbf{j}}(G(P)).$$

Proof: Mathematical induction in r leads to the following:

1. For $r = 0$ we have

$$V^0_{\mathbf{i}}(P) = V_{\mathbf{i}} = \sum_{|\mathbf{j}|=0} V_{\mathbf{i}+\mathbf{j}} B^0_{\mathbf{j}}(\mathbf{u}) \quad |\, \mathbf{i} \,| = n.$$

2. Assume that statement holds for r, i.e.

$$V^r_{\mathbf{i}}(P) = \sum_{|\mathbf{j}|=r} V_{\mathbf{i}+\mathbf{j}} B^r_{\mathbf{j}}(G(P)).$$

We show that it holds also for $r + 1$. From definition of $V^r(P)$ we have

$$V^{r+1}_{\mathbf{i}}(P) = \sum_{l=1}^{k} g_l(P) V^r_{\mathbf{i}+\mathbf{e}_l}(P) = \sum_{l=1}^{k} g_l(P) \sum_{|\mathbf{j}|=r} V_{\mathbf{i}+\mathbf{j}+\mathbf{e}_l} B^r_{\mathbf{j}}(G(P)) =$$

$$= \sum_{l=1}^{k} g_l(P) \sum_{|\mathbf{j}|=r+1} V_{\mathbf{i}+\mathbf{j}} B^r_{\mathbf{j}-\mathbf{e}_l}(G(P)) = \sum_{|\mathbf{j}|=r+1} V_{\mathbf{i}+\mathbf{j}} \sum_{l=1}^{k} g_l B^r_{\mathbf{j}-\mathbf{e}_l}(G(P)) =$$

$$= \sum_{|\mathbf{j}|=\mathbf{r+1}} V_{\mathbf{i}+\mathbf{j}} B^{r+1}_{\mathbf{j}}(G(P)),$$

where we used Theorem 2.2.2(c) and a multi-index transformation.

□

The de Casteljau algorithm gives us also a subdivision scheme for breaking S-volume into a set of k smaller volumes. If we have a point P from the domain D, we create k S-volumes SV^n_j such that the domain D_j of the j-th volume is a polyhedron made of points $A_1, A_2, .., A_{j-1}, A_{j+1}, .., A_k, P$ and with control points $W_{\mathbf{i}} = V^{i_j}_{\mathbf{i}-i_j\mathbf{e}_j}(P)$ for $|\mathbf{i}| = n; dim(\mathbf{i}) = k$. Then SV^n_j is restriction of volume SV^n in the domain D_j, i.e. $SV^n(Q) = SV^n_j(Q)$ for all $Q \in D_j$, $j = 1, 2, .., k$.

Theorem 2.3.7 *For the $r-th$ directional derivative of S-volume SV^n in point $P \in D$ and in direction $\mathbf{d}$:*

$$D^r_{\mathbf{d}} SV^n(P) = \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B^r_{\mathbf{j}}(\mathbf{d}) V^{n-r}_{\mathbf{j}}(P) =$$

$$\frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=n-r} B^{n-r}_{\mathbf{j}}(G(P)) V^r_{\mathbf{j}}(\mathbf{d}),$$

where $V^{n-r}_{\mathbf{j}}(P)$ are the points created from de Casteljau algorithm using barycentric coordinates of point P and $V^{n-r}_{\mathbf{j}}(\mathbf{d})$ are the vectors from de Casteljau algorithm using barycentric coordinates of vector $\mathbf{d}$.

Proof: From the analytical expression of S-volume we have

$$D^r_{\mathbf{d}} SV^n(P) = D^r_{\mathbf{d}} \Big[\sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^n_{\mathbf{i}}(G(P)) \Big] = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} D^r_{\mathbf{d}} B^n_{\mathbf{i}}(G(P)).$$

Using Theorem 2.2.3 we get

$$D^r_{\mathbf{d}} SV^n(P) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} \Big[\frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B^{n-r}_{\mathbf{i}-\mathbf{j}}(G(P)) B^r_{\mathbf{j}}(\mathbf{d}) \Big] =$$

$$\frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B^{n-r}_{\mathbf{i}-\mathbf{j}}(G(P)) B^r_{\mathbf{j}}(\mathbf{d}).$$

With the notation $\mathbf{k} := \mathbf{i} - \mathbf{j}$ we get

$$D^r_{\mathbf{d}} SV^n(\mathbf{u}) = \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} \sum_{|\mathbf{k}|=n-r} V_{\mathbf{j}+\mathbf{k}} B^{n-r}_{\mathbf{k}}(G(P)) B^r_{\mathbf{j}}(\mathbf{d}) =$$

$$= \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B^r_{\mathbf{j}}(\mathbf{d}) \sum_{|\mathbf{k}|=n-r} V_{\mathbf{j}+\mathbf{k}} B^{n-r}_{\mathbf{k}}(G(P)) = \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} B^r_{\mathbf{j}}(\mathbf{d}) V^{n-r}_{\mathbf{j}}(P).$$

In the last equation we used Theorem 2.3.6. Similar steps lead to

$$D^r_{\mathbf{d}} SV^n(P) = \frac{n!}{(n-r)!} \sum_{|\mathbf{j}|=r} \sum_{|\mathbf{k}|=n-r} V_{\mathbf{j}+\mathbf{k}} B^{n-r}_{\mathbf{k}}(G(P)) B^r_{\mathbf{j}}(\mathbf{d}) =$$

$$\frac{n!}{(n-r)!} \sum_{|\mathbf{k}|=n-r} B^{n-r}_{\mathbf{k}}(G(P)) \sum_{|\mathbf{j}|=r} V_{\mathbf{j}+\mathbf{k}} B^r_{\mathbf{j}}(\mathbf{d}) = \frac{n!}{(n-r)!} \sum_{|\mathbf{k}|=n-r} B^{n-r}_{\mathbf{k}}(G(P)) V^r_{\mathbf{k}}(\mathbf{d}).$$

$\square$

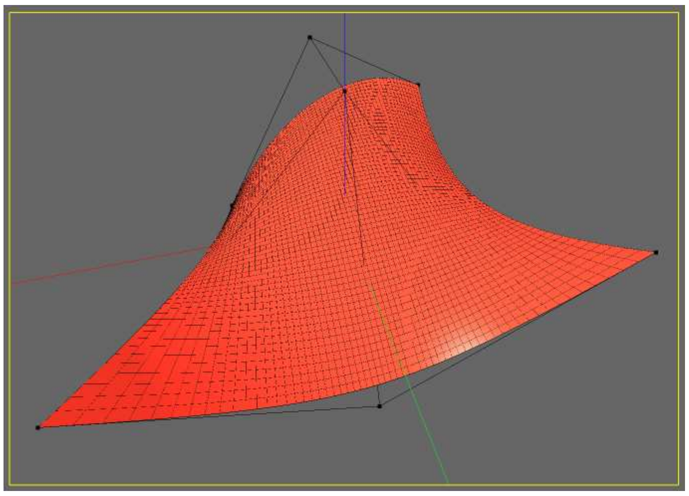


Figure 2.8: Bézier tensor-product surface.

We are aware of the fact, that the computation of directional derivation is not correct. We did not take into account the fact, that the function parameters are not independent, because their sum is equal to 1. However, this simplification does not affect the resulting formula for the computation of directional derivative. Observe the boundary facet of S-volume domain that we have for $u_l(P) = 0$. Let us have vector $\mathbf{d}$ given by the barycentric coordinates $(d_1, d_2, .., d_k)$ such that it is not parallel to that border, i.e. $d_l \neq 0$. Then we can define cross boundary derivative as:

$$D^r_{\mathbf{d}} SV^n(P)\Big|_{u_l(P)=0} = \frac{n!}{(n-r)!} \sum_{|\mathbf{i}|=n-r} B^r_{\mathbf{i}}(\mathbf{d}) V^{n-r}_{\mathbf{i}}(P)\Big|_{u_l(P)=0} =$$

$$\frac{n!}{(n-r)!} \sum_{|\mathbf{i}_l|=n-r} B^r_{\mathbf{i}_l}(\mathbf{d}) V^{n-r}_{\mathbf{i}_l}(P),$$

where $\mathbf{i}_l$ is multi-index with l-th index equal to zero.

2.3.3 Bézier tensor-product volume

Bézier triangles have triangular domain. We can construct very easily patches with rectangular domain. First we define Bézier tensor-product surface that can be used as the basic ground for building similar volumes. Bézier tensor-product surfaces (Figure 2.8) are defined by taking Bernstein polynomials for two parameters independently. This will form a surface created by sweeping one Bézier curve among other Bézier curve.

- The domain of Bézier tensor-product patch is the square $\langle 0,1\rangle \times \langle 0,1\rangle$ in $\mathbb{R}^2$.
- The degrees of Bézier tensor-product patch are the integers n_u, n_v that state degrees of polynomials that define the surface.
- The control net of Bézier tensor-product patch is the set of points $V_{i,j} \in E^3; i = 0,1,..,n_u; j = 0,1,...,n_v$ that will form the shape of surface.
- Bézier tensor-product surface $BTPS^n$ is defined as

  $$BTPS^n(u,v) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v} V_{i,j} B_i^{n_u}(u) B_j^{n_v}(v),$$

  where $u \in \langle 0,1\rangle$ and $v \in \langle 0,1\rangle$.

The definition of Bézier tensor-product volume is similar, we just add one direction and corresponding parameters.

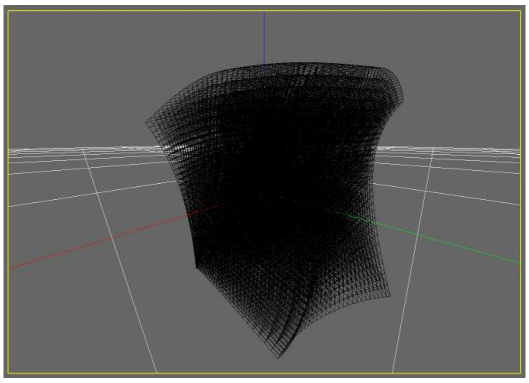
Definition 2.3.3 *For given $n_u, n_v, n_w \in N$ and points $V_{i,j,k} \in E^3; i = 0,1,...,n_u; j = 0,1,...,n_v; k = 0,1,...,n_w$, a Bézier tensor-product volume $BTPV^{n_u,n_v,n_w}$ is analytically defined as*

$$BTPV^{n_u,n_v,n_w}(u,v,w) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} V_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w),$$

where $u,v,w \in \langle 0,1\rangle$, i.e. the domain of volume is $\langle 0,1\rangle \times \langle 0,1\rangle \times \langle 0,1\rangle$, n_u, n_v, n_w are degrees in u, v resp. w direction and $V_{i,j,k}$ are control points arranged in the control net.

Properties of Bézier tensor-product volume are similar to that for S-volumes. We will state them without a proof:

- The Bézier tensor-product volume is invariant under affine transformation.
- The Bézier tensor-product volume approximates the shape of its control net.
- The Bézier tensor-product volume interpolates the control points $V_{0,0,0}$, $V_{n_u,0,0}$, $V_{0,n_v,0}$, $V_{n_u,n_v,0}$, $V_{0,0,n_w}$, $V_{n_u,0,n_w}$, V_{0,n_v,n_w}, V_{n_u,n_v,n_w}.
- Boundary patches are Bézier tensor-product patches, boundary curves are Bézier curves.


(a) Wireframe visualization.

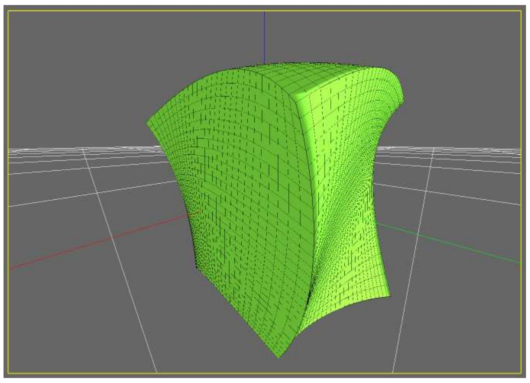

(b) Wireframe and solid visualization.

Figure 2.9: Bézier tensor-product volume.

- For the Bézier tensor-product volume, the degree can be elevated independently in each direction, similarly to the curve case.
- De Casteljau algorithm for curves can be adapted. In the first step, it can be performed in u direction, created points can be then used in de Casteljau algorithm in v direction and then in w direction.

The basic visualizations of Bézier tensor-product volume and its control points are in Figure 2.9.

2.4 B-spline volumes

Pure polynomial representation has also some disadvantages, one important is that the higher number of control points leads to the higher degree. This relation between the number of control points and the degree of Bézier curve has mainly computational issues and also high order polynomials are not sufficient for modeling. In real applications, we need polynomials with maximal degree of three or four. This means that we need independence of the degree from the number of control points, which can be done using piecewise polynomial blending functions.

We will use one well known representation in the form of B-spline blending functions. Because we need a function that consists of polynomial segments, for each segment we have to define the domain interval and these segment domains will be connected by knots. Firstly, we need the interval of the parameter for the whole curve, then we must split this interval into smaller parts so that each polynomial segment has its own domain interval. Of course we have to have given the degree of the polynomial segments.

Definition 2.4.1 *Let $u_0 \leq u_1 \leq \ldots \leq u_m$ be a sequence of real numbers called the knots. For $k = 0, 1, \ldots, m$, and $i = 0, \ldots, m-k-1$, we define i-th B-spline blending function of degree k as*

$$N_i^0(u) = \begin{cases} 1 & \text{for } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

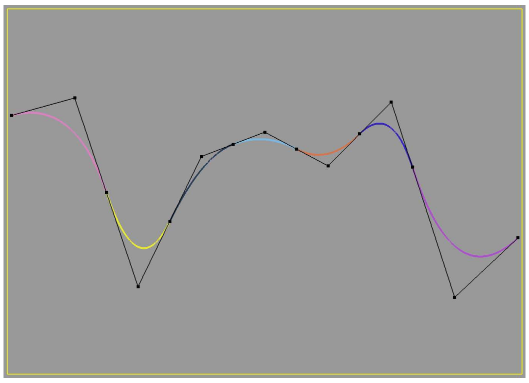
and for $k > 0$

$$N_i^k(u) = \begin{cases} \frac{u-u_i}{u_{i+k}-u_i} N_i^{k-1}(u) + \frac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}} N_{i+1}^{k-1}(u) & u_{i+k} \neq u_i,\ u_{i+k+1} \neq u_{i+1} \\ \frac{u-u_i}{u_{i+k}-u_i} N_i^{k-1}(u) & u_{i+k} \neq u_i,\ u_{i+k+1} = u_{i+1} \\ \frac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}} N_{i+1}^{k-1}(u) & u_{i+k} = u_i,\ u_{i+k+1} \neq u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

These blending functions have many properties that can be easily shown.

Theorem 2.4.1 *B-spline function $N_i^k(u)$ defined over the knot vector $(u_0, u_1, .., u_m)$ fulfills following properties*

- *$N_i^k(u)$ is a piecewise polynomial function, each segment having degree k, with joining points in the knots.*
- *$N_i^k(u) \geq 0$ for each k, i, u.*
- *If $u \geq u_{i+k+1}$ or $u < u_i$, then $N_i^k(u) = 0$, i.e. $N_i^k(u)$ is nonzero only in the interval $\langle u_i, u_{i+k+1})$. In that interval, $N_i^k(u) > 0$ for $u \in (u_i, u_{i+k+1})$.*
- *If $u \in \langle u_j, u_{j+1})$, then at most $k+1$ basis functions are non-zero, namely: $N_{j-k}^k(u), N_{j-k+1}^k(u), N_{j-k+2}^k(u), \ldots, N_j^k(u)$.*
- *If $u \in (u_j, u_{j+1})$ and $i = j-k, j-k+1, \ldots, j$, then $N_i^k(u) > 0$.*
- *If $u \in \langle u_j, u_{j+1})$, then $\sum_{i=0}^{m-k-1} N_i^k(u) = \sum_{i=j-d}^{j} N_i^k(u) = 1$.*


(a) B-spline curve and its polynomial segments.

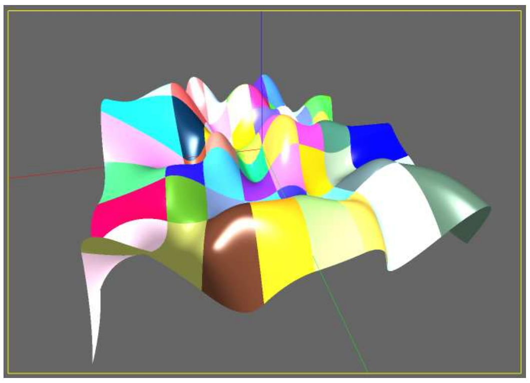

(b) B-spline tensor-product surface and its polynomial segments.

Figure 2.10: B-spline curve and tensor-product surface.

- *If knot vector has special values and length,* $u_0 = u_1 = ... = u_k = 0$ *and* $u_{k+1} = u_{k+2} = ... = u_{2k+1} = 1$*, then B-spline functions* $N_0^k(u), N_1^k(u), ..., N_k^k(u)$ *become Bernstein polynomials* $B_0^k(u), B_1^k(u), ..., B_k^k(u)$ *on interval* $\langle 0, 1\rangle$*.*

- *At a knot* u_j *of multiplicity* r *(there is exactly* r *knots in knot sequence with value* u_j*), basis function* $N_i^k(u)$ *is* C^{k-r} *continuous.*

Using these blending functions, we can state the appropriate definition of parametric curves or tensor-product surfaces.

Definition 2.4.2 *Let us have a given degree* d*, a sequence of knots* $u_0, u_1, ..., u_m$ *and control points* $V_i \in E^3; i = 0, 1, .., n$ *such that* $m = n + d + 1$*. The B-spline curve* BSC^d *is defined as* $BSC^d(u) = \sum_{i=0}^{n} V_i N_i^d(u)$ *for* $u \in \langle u_d, u_{n+1})$*.*
Let us have given degrees d_u, d_v*, two sequences of knots* $u_0, u_1, ..., u_{m_u}$*,*$v_0, v_1, ..., v_{m_v}$ *and control points* $V_{i,j} \in E^3; i = 0, 1, .., n_u; j = 0, 1, .., n_v$ *such that* $m_u = n_u + d_u + 1$*,*$m_v = n_v + d_v + 1$*. The B-spline tensor-product surface* $BSTPS^{d_u,d_v}$ *is defined as* $BSTPS^{d_u,d_v}(u, v) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} V_{i,j} N_i^{d_u}(u) N_j^{d_v}(u)$ *for* $u \in \langle u_{d_u}, u_{n_u+1})$*,* $v \in \langle v_{d_v}, v_{n_v+1})$*.*

One example of B-spline curve is in Figure 2.10(a). We will show the basic properties for B-spline curves. For B-spline tensor-product surfaces, the properties are independent for each direction and similar to the curve case.

Theorem 2.4.2 *Properties of B-spline curve* BSC^d*:*

- *B-spline curve is invariant under an affine transformation. So if* $A : E^3 \to E^3$ *is an affine transformation, then* $A(\sum_{i=0}^{n} V_i N_i^d(u)) = \sum_{i=0}^{n} A(V_i) N_i^d(u)$

- *B-spline curve lies in the convex hull of its control points.*

- *B-spline curve approximates the shape of its control polygon.*

- *B-spline curve has the property of variation diminishing, i.e. a straight line can not intersect a B-spline curve more times than it intersects the control polygon.*

- *If* $u_0 = u_1 = ... = u_d$*, then B-spline curve interpolates the first control point. If* $u_{m-d} = u_{m-d+1} = ... = u_m$*, then B-spline curve interpolates the last control point.*

- *(de Boor algorithm) Let us have B-spline curve* BSC^d *and a parameter* $u \in \langle u_r, u_{r+1})$*. Recursively we compute the auxiliary points*

$$V_i^0(u) = V_i \quad i = 0, 1, .., n$$

$$V_i^j(u) = (1 - \alpha_i^j(u)) V_{i-1}^{j-1}(u) + \alpha_i^j(u) V_i^{j-1}(u),$$

  where $j = 1, .., d$*,* $i = r - d + j, .., r$ *and* $\alpha_i^j(u) = \frac{u - u_i}{u_{i+d-j+1} - u_i}$ *for* $u_{i+d-j+1} \neq u_i$ *and* $\alpha_i^j(u) = 0$ *otherwise. Then* $V_r^d(u) = BSC^d(u)$*.*

- *If* $n = d + 1$*,* $u_0 = u_1 = ... = u_d = 0$ *and* $u_{m-d} = u_{m-d+1} = ... = u_m = 1$*, then B-spline curve is Bézier curve with the same control polygons and de Boor algorithm is reduced to de Casteljau algorithm.*

- *If knot* u_j *has multiplicity* r_j *in the knot vector, then in this knot, B-spline is at least* C^{d-r_j}*.*

- *If each knot has exact multiplicity* r *in the knot vector, then control points of B-spline curve becomes also control points of its segments described in Bézier form. With such knot vector, we get decomposition of B-spline curve into its piecewise Bézier form. In Figure 2.10(a), B-spline curve with its piecewise Bézier segments are illustrated.*

- *When the derivative of B-spline curve exists, it is given by* $\frac{\partial}{\partial u} BSC^d(u) = \sum_{i=0}^{n-1} Q_i N_i^{d-1}(u)$*, where* $Q_i = d\frac{V_i - V_{i-1}}{u_{i+d} - u_i}$*. In this case, the result is a vector, because it is a linear combination of vectors* Q_i*.*

- *Let us have B-spline curve* $BSC^d(u) = \sum_{i=0}^{n} V_i N_i^d(u)$ *with control points* V_i *and knot vector* $\mathbf{u} = (u_0, u_1, ..., u_m)$. *Let* $\overline{u}$ *be a real value such that it* $u_J \leq \overline{u} < u_{J+1}$, *for some index* J. *Then we can prepare a new knot vector* $\mathbf{u} \cup \{\overline{u}\}$ *with corresponding B-spline functions* $\overline{N}_i^d(u)$. *We can write the B-spline curve as another B-spline curve with increased number of control points and knots as* $\sum_{i=0}^{n} V_i N_i^d(u) = \sum_{j=0}^{n+1} Q_j \overline{N}_j^d(u)$, *where*

$$N_i^d(u) = \begin{cases} \overline{N}_i^d & i \leq J-d+1 \\ \frac{\overline{u}-u_i}{u_{i+d+1}-u_i}\overline{N}_i^d(u) + \frac{u_{i+d+2}-\overline{u}}{u_{i+d+2}-u_{i+1}}\overline{N}_{i+1}^d(u) & J-d+2 \leq i \leq J \\ \overline{N}_{i+1}^d & J+1 \leq i \end{cases}$$

$$Q_j = \begin{cases} V_j & j \leq J-d \\ \frac{\overline{u}-u_j}{u_{j+d+1}-u_j}V_j + \frac{u_{j+d+1}-\overline{u}}{u_{j+d+1}-u_j}V_{j-1} & J-d+1 \leq j \leq J \\ V_{j-1} & J+1 \leq j \end{cases}$$

  This algorithm is called the simple knot insertion or Boehm algorithm, because it changes the knot vector and the control polygon without changing the shape and the parametrization of the curve.

- *Let us have B-spline curve* $BSC^d(u)$ *with control points* $V_0, V_1, ..., V_n$, *knot vector* $(u_0, u_1, ..., u_m)$ *and one knot* $u_r; u_r \neq u_{r+1}$ *with multiplicity* s. *We define new control points*

$$W_i = V_i \qquad 0 \leq i < r-d$$
$$W_i = \frac{V_i - (1-\alpha_i)W_{i-1}}{\alpha_i} \qquad i = r-d, r-d+1, ..., \lfloor\frac{1}{2}(2r-d-s-1)\rfloor$$
$$W_j = V_{j+1} \qquad r-s \leq j \leq n-1$$
$$W_j = \frac{V_j - (1-\alpha_j)W_{j+1}}{\alpha_j} \qquad j = r-s, r-s-1, ..., \lfloor\frac{1}{2}(2r-d-s+2)\rfloor,$$

  where $\alpha_i = \frac{u_r-u_i}{u_{i+d+1}-u_i}$ *and* $\lfloor x \rfloor$ *denotes the largest integer not greater than* x. *Then if the curve* $BSC^d(u)$ *is* C^{d-s+1} *continuous in* $u = u_r$, *we can write curve* $BSC^d(u)$ *as a new curve* $\overline{BSC}^d(u)$ *with the knot vector* $(u_0, .., u_{r-1}, u_{r+1}, .., u_m)$ *and the control points* $W_i; i = 0, 1, .., n-1$:

$$BSC^d(u) = \sum_{i=0}^{n} V_i N_i^d(u) = \overline{BSC}^d(u) = \sum_{i=0}^{n-1} W_i \overline{N}_i^d(u)$$

  This algorithm is called simple knot removal, because it changes the knot vector and the control polygon without changing the shape and the parametrization of the curve.

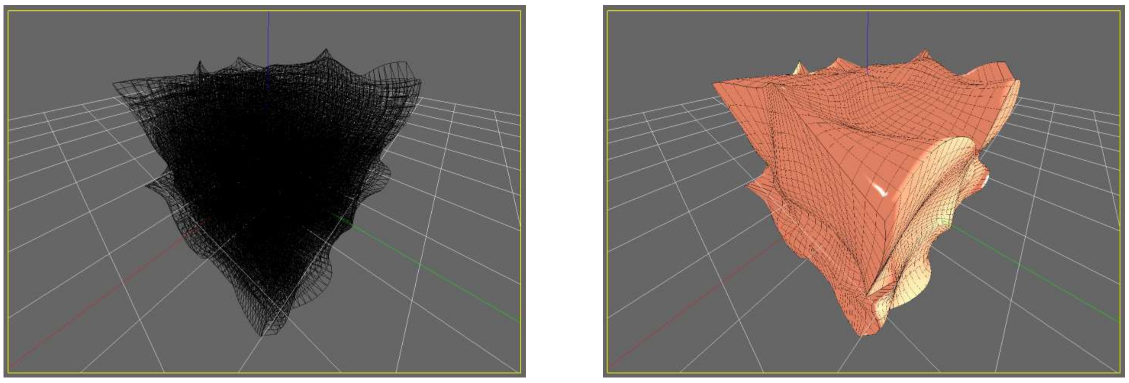
The properties of B-spline curves are necessary for determination of the properties of B-spline tensor-product surfaces and volumes. Several properties are heavily used in modeling, for example de Boor algorithm for evaluation of a point on a curve or a surface. We did not mention the degree elevation algorithm, but it can be done using several steps (as described in [PT95]). If we have a B-spline curve of degree d defined by knot vector $(u_0, ..u_0, u_1, .., u_1, .., u_{m-1}, .., u_{m-1}, u_m, .., u_m)$ with knot multiplicities $s_0 = d + 1,\ s_1, ..,\ s_{m-1},\ s_m = d + 1$ and by control points $V_i; i = 0, 1, .., n;\ n = d + s_1 + s_2 + .. + s_{m-1} + 1$, then we can elevate degree of this curve by one in following way

- First we insert the existing knots $u_1, u_2, .., u_{m-1}$ such that each individual knot except the first one and the last one has multiplicity d in the knot vector. This way we get the segments in Bézier form, i.e. B-spline control points for given segment will be Bézier control points for that segment.
- We perform the degree elevation on each segment in Bézier form (Theorem 2.3.5). This will also increase the multiplicities of the knots by one.
- Now we remove knots such that remaining knot vector have the form $(u_0, ..u_0, u_1, .., u_1, .., u_{m-1}, .., u_{m-1}, u_m, .., u_m)$ and multiplicities $s_0 = d+2, s_1+1, .., s_{m-1}+1, s_m = d+2$. We can do this knot removals, because the necessary continuity conditions are fulfilled in the original curve.

Also for B-spline curves, we can use the tensor-product approach and extend it into tensor-product surfaces. We will skip this extension, the Figure 2.10(b) show such B-spline surface. After extension in one another direction, we get the definition of volumes based on B-spline basis functions.

Definition 2.4.3 *Let us have real numbers $u_0 \leq u_1 \leq ... \leq u_{m_u}$, $v_0 \leq v_1 \leq ... \leq v_{m_v}$, $w_0 \leq w_1 \leq ... \leq w_{m_w}$, degrees $d_u, d_v, d_w \in N; d_u, d_v, d_w \geq 1$ and control points $V_{i,j,k} \in E^3$; $i = 0, 1, ..., n_u; j = 0, 1, ..., n_v; k = 0, 1, ..., n_w$, where $n_u = m_u - d_u - 1, n_v = m_v - d_v - 1, n_w = m_w - d_w - 1$. Then B-spline volume BSV^{d_u,d_v,d_w} of degrees d_u, d_v, d_w over the knot vectors $(u_0, u_1, ..., u_{m_u})$, $(v_0, v_1, ..., v_{m_v})$ and $(w_0, w_1, ..., w_{m_w})$ is defined over the interval $\langle u_{d_u}, u_{n_u+1}) \times \langle v_{d_v}, v_{n_v+1}) \times \langle w_{d_w}, w_{n_w+1})$ as*

$$BSV^{d_u,d_v,d_w}(u, v, w) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{i,j,k} N_i^{d_u}(u) N_j^{d_v}(v) N_k^{d_w}(w)$$



(a) Wireframe visualization. (b) Wireframe and solid visualization.

Figure 2.11: B-spline volume with degrees $(3, 3, 3)$ and $8 \times 8 \times 8$ control points.

$$u \in \langle u_{d_u}, u_{n_u+1}), v \in \langle v_{d_v}, v_{n_v+1}), w \in \langle w_{d_w}, w_{n_w+1}),$$

where $N_i^{d_u}(u), N_j^{d_v}(v), N_k^{d_w}(w)$ *are B-spline blending functions.*

The properties of B-spline volumes in are similar to the properties of B-spline curves and tensor-product surfaces. For each independent direction, we can adapt properties from the curve case. B-spline volume over randomly picked control points can be seen in Figure 2.11.

2.5 Rational volumes

Polynomials and piecewise polynomials represent a wide set of curves, surfaces and volumes, but not all simple and useful objects are part of this set. There are still objects that we want to represent with help of Bernstein or B-spline blending functions. For example, some conics like circle or ellipse can not be described using polynomial or piecewise polynomial functions. A quarter of a circle with the center in the origin of the coordinate system and radius 1 lying in xy coordinate plane can be written as $\{X \in E^3; X = [\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, 0]t \in \langle 0, 1\rangle\}$, i.e. as a ratio of two polynomials. Such functions are called rational functions. Because conics are invariant under projective transformations, we will use coordinates that are invariant under these transformations. We introduce rational blending functions based on Bernstein and B-spline blending functions. For this, we use the projective hyper-plane and homogeneous coordinates.

Definition 2.5.1 *4-dimensional homogeneous coordinates are 4-tuples of real values*

$[x,\ y,\ z,\ w]$ with following properties:

- *If $w = 0$, the tuple represents a point at infinity in augmented E^3 with the direction (x, y, z), otherwise it represents a point in E^3 with Cartesian coordinates $(x/w, y/w, z/w)$.*
- *$[x, y, z, w] = [a, b, c, d] \Leftrightarrow \exists \alpha \neq 0; x = \alpha a, y = \alpha b, z = \alpha c, w = \alpha d$. Homogeneous coordinates are the equivalence classes of coordinates created by this relation.*
- *$[x, y, z, w] + [a, b, c, d] = [wa + dx : wb + dy : wc + dz : wd]$*
- *if $a \in \mathbb{R}; a \neq 0$, then $a[x, y, z, w] = [ax, ay, az, w]$.*

The mapping of homogeneous coordinates to Euclidean space can be treated also as a projection of 4-dimensional Euclidean space to 3-dimensional hyperplane E^3. That is why they are used for describing projective hyperplanes. The barycentric coordinates for simplices introduced in Section 2.1 are also homogeneous coordinates. In computer graphics, the homogeneous coordinate system is used for its unification of the translation, scaling and rotation of geometric objects.

We work with homogeneous coordinates to form a new set of parametric volumes. We use the description of volumes using control points V_i from E^3 and blending functions $f_i(P)$ (see Definition 2.0.2). For each control point $V_i = (x_i, y_i, z_i)$, we add a new parameter value $w_i \in \mathbb{R}$. Using these parameters, we do the mapping from E^3 to the homogeneous coordinates, creating control points $\overline{V}_i$ in homogeneous coordinates, $\overline{V}_i = [wx, wy, wz, w]$. Now in this 4-dimensional space we model a new volume $\overline{V}(P) = (x(P), y(P), z(P), w(P)) = \sum_{i=0}^{n} \overline{V}_i f_i(P)$. To finish with the volume in E^3, we have to finally map the point of volume back to E^3, getting $(x(P)/w(P), y(P)/w(P), z(P)/w(P))$ for $w(P) \neq 0$. We get description of volume in E^3 as $V(P) = \frac{\sum_{i=0}^{n} w_i V_i f_i(P)}{\sum_{i=0}^{n} w_i f_i(P)}$. Using this approach for several blending functions we get following definitions of rational volumes.

Definition 2.5.2 *(Rational Bézier tensor-product volume) Let us have given points $V_{i,j,k} \in E^3$; $i = 0, 1, ..., n$, $j = 0, 1, ..., m$, $k = 0, 1, ..., l$ and real numbers $w_{i,j,k} \in R; w_{i,j,k} \geq 0$; $i = 0, 1, ..., n$, $j = 0, 1, ..., m$, $k = 0, 1, ..., l$. We define rational Bézier tensor-product volume $RBTPV^{n,m,l}$ analytically as*

$$RBTPV^{n,m,l}(u, v, w) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{l} w_{i,j,k} V_{i,j,k} B_i^n(u) B_j^m(v) B_k^l(w)}{\sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{l} w_{i,j,k} B_i^n(u) B_j^m(v) B_k^l(w)}$$

$$u, v, w \in \langle 0, 1 \rangle,$$

where n, m, l are degrees in u resp. v resp. w direction, $\langle 0,1\rangle \times \langle 0,1\rangle \times \langle 0,1\rangle$ is the domain of the volume, $V_{i,j,k}$ are the control points and $w_{i,j,k}$ are the weights.

Definition 2.5.3 *(Rational Bézier tetrahedron) Let us have a degree $n \in N$, a tetrahedron given by four non-coplanar points $A, B, C, D \in E^3$, control points $V_{\mathbf{i}} \in E^3; dim(\mathbf{i}) = 4; |\mathbf{i}| = n$ and for each control point a given weight $w_{\mathbf{i}} \in \mathbb{R}; dim(\mathbf{i}) = 4; |\mathbf{i}| = n$. Then a rational Bézier tetrahedron $RBTH^n$ is given analytically for point $P \in ABCD$ as*

$$RBTH^n(P) = \frac{\sum_{|\mathbf{i}|=n} w_{\mathbf{i}} V_{\mathbf{i}} B^n_{\mathbf{i}}(U(P))}{\sum_{|\mathbf{i}|=n} w_{\mathbf{i}} B^n_{\mathbf{i}}(U(P))},$$

where U is function of barycentric coordinates with respect to tetrahedron $ABCD$.

Definition 2.5.4 *(Rational S-volume) Let us have a polyhedron D defined by set of non-coplanar points $A_1, A_2, ..., A_k \in E^3$ and facets. Let us have a degree $n \in \mathbb{N}$, control points $V_{\mathbf{i}} \in E^3; dim(\mathbf{i}) = k; |\mathbf{i}| = n$, and for each control point a given weight $w_{\mathbf{i}} \in \mathbb{R}; dim(\mathbf{i}) = k; |\mathbf{i}| = n$. Then a rational S-volume RSV^n with the domain D, degree n and control points $V_{\mathbf{i}}$ is defined analytically for a point $P \in D$ as*

$$RSV^n(P) = \frac{\sum_{|\mathbf{i}|=n} w_{\mathbf{i}} V_{\mathbf{i}} B^n_{\mathbf{i}}(G(P))}{\sum_{|\mathbf{i}|=n} w_{\mathbf{i}} B^n_{\mathbf{i}}(G(P))},$$

where G is function of barycentric coordinates with respect to polyhedron D.

Definition 2.5.5 *(Rational B-spline tensor-product volume) Let us have real numbers $u_0 \le u_1 \le ... \le u_{m_u}$, $v_0 \le v_1 \le ... \le v_{m_v}$ and $w_0 \le w_1 \le ... \le w_{m_w}$, degrees $d_u, d_v, d_w \in N$ and control points $P_{i,j,k} \in E^3; i = 0, 1, ..., n_u; j = 0, 1, ..., n_v; k = 0, 1, ..., n_w$, real numbers $w_{i,j,k} \in R; i = 0, 1, ..., n_u; j = 0, 1, ..., n_v; k = 0, 1, ..., n_w$, where $n_u = m_u - d_u - 1, n_v = m_v - d_v - 1, n_w = m_w - d_w - 1$. Not all $w_{i,j,k}$ can be equal to 0. Then the rational B-spline tensor-product volume (NURBS volume) $RBSTP^{d_u,d_v,d_w}$ of degrees d_u, d_v, d_w over the knot vectors $(u_0, ..., u_{m_u})$, $(v_0, ..., v_{m_v})$ and $(w_0, ..., w_{m_w})$ is defined over domain $\langle u_{d_u}, u_{n_u+1}) \times \langle v_{d_v}, v_{n_v+1}) \times \langle w_{d_w}, w_{n_w+1})$ as*

$$RBSTP^{d_u,d_v,d_v}(u, v, w) = \frac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} w_{i,j,k} V_{i,j,k} N_i^{d_u}(u) N_j^{d_v}(v) N_k^{d_w}(w)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} w_{i,j,k} N_i^{d_u}(u) N_j^{d_v}(v) N_k^{d_w}(w)}$$

$$u \in \langle u_{d_u}, u_{n_u+1}), v \in \langle v_{d_v}, v_{n_v+1}), w \in \langle w_{d_w}, w_{n_w+1}).$$

The general rational B-spline tensor-product volume is also called NURBS volume, NURBS is a well-known abbreviation for non-uniform rational B-splines. Using these rational volumes we can describe some useful objects as is illustrated in Figure 3.9.

Because all rational volumes were defined by a mapping of homogeneous coordinates to Euclidean ones, this way we can extend the properties and algorithms from non-rational to rational cases. We will give an example of de Casteljau algorithm for rational S-volume for a point P:

1. Define points in E^4 from the control points of S-volume and weights as $\overline{V}_{\mathbf{i}}^0(P) = (w_{\mathbf{i}}V_{\mathbf{i}}, w_{\mathbf{i}})$.
2. Using these points, recursively define points $\overline{V}_{\mathbf{i}}^r(P) = \sum_{j=1}^k g_j(P)\overline{V}_{\mathbf{i}+\mathbf{e}_j}^{r-1}(P)$ for $r = 1, 2, ..., n$ and for $|\mathbf{i}| = n - r$.
3. At the end, we get the point $\overline{V}_{\mathbf{0}}^r = [V_x, V_y, V_z, V_w]$. After mapping this point back to E^3, we get $RSV^d(P) = (V_x/V_w, V_y/V_w, V_z/V_w)$.

We can write algorithm using the coordinates in E^3 and weights separately, then the algorithm is as follows:

1. Put $V_{\mathbf{i}}^0(P) = V_{\mathbf{i}}$ and $w_{\mathbf{i}}^0(P) = w_{\mathbf{i}}$ for $r = 1, 2, ..., n$ and for $|\mathbf{i}| = n - r$.
2. Because $\overline{V}_{\mathbf{i}}^r(P) = \sum_{j=1}^k g_j(P)\overline{V}_{\mathbf{i}+\mathbf{e}_j}^{r-1}(P)$, we can write

   $$w_{\mathbf{i}}^r(P) = \sum_{j=1}^{k} g_j(P)w_{\mathbf{i}+\mathbf{e}_j}^{r-1}(P)$$

   and

   $$w_{\mathbf{i}}^r(P)V_{\mathbf{i}}^r(P) = \sum_{j=1}^{k} g_j(P)w_{\mathbf{i}+\mathbf{e}_j}^{r-1}(P)V_{\mathbf{i}+\mathbf{e}_j}^{r-1}(P).$$

   From these equations we can compute $w_{\mathbf{i}}^r(P)$ and then $V_{\mathbf{i}}^r(P)$ for $r = 1, 2, ..., n$ and $|\mathbf{i}| = n - r$.
3. Finally $RSV^d(P) = V_{\mathbf{0}}^d(P)$.

Chapter 3

Modeling with parametric volumes

In the previous chapter, we introduced several types of parametric volumes and presented the properties of theses objects. We work with objects in the form of functions that blend the controls points from the given set of points. Using several types of blending functions we presented Bézier tensor-product volumes, S-volumes and B-spline tensor-product volumes together with their rational versions. But if we want to have a representation that is useful, we must also present the algorithms and prescriptions for modeling with such volumes.

In this chapter we will focus on the work and modeling with the described volumes. We will observe the impact of changing the parameters that define the volume. If we want to work with volumes we have to have techniques to create some basic types of objects in the form of parametric volumes. We will use paradigms like ruled objects or object of revolution to create volumes from lower order objects like curves and surfaces. The description of these techniques was given in [Sam05]. We will present basic common objects - parts of a sphere, generalized cylinders, toruses or prisms described as Bézier or B-spline volumes. In many applications, we want to convert complicated volumes like B-splines or S-volumes into sets of simpler objects like Bézier tetrahedra or Bézier tensor-product volumes. We will also present algorithms for decomposition of these volumes into simpler ones. Another recently used technique for the creation of objects is subdivision. We will use this approach to model approximations of parametric volumes. At the end we will take a look at the space deformations given by the set of control points of parametric volume.

3.1 Common volumes

In this section we present several algorithms for generation of parametric volumes from curves and surfaces and construction of some basic well-known objects. First type of geometrically generated objects are the ruled volumes. A ruled volume is constructed as a linear interpolation between two parametric surfaces. Two opposite boundary patches of a ruled volume are the generating patches and the remaining boundary patches are ruled surfaces.

Definition 3.1.1 *Let us have 2 patches $P_1(U), P_2(U)$ with a common domain $D \subset E^2$. The ruled volume RV is the union of all line segments AB such that $A = P_1(U)$ and $B = P_2(U)$ for each point U from the common domain D.*

We can describe the representation of a ruled volume based on the representation of the given patches. In all cases, both generating patches must be compatible, i.e. must have the same representation, domain and degrees. Now we show methods how to construct the ruled volume for several representations:

- If P_1 and P_2 are in the form of rational Bézier tensor-product patches with the same domain, then their expressions are

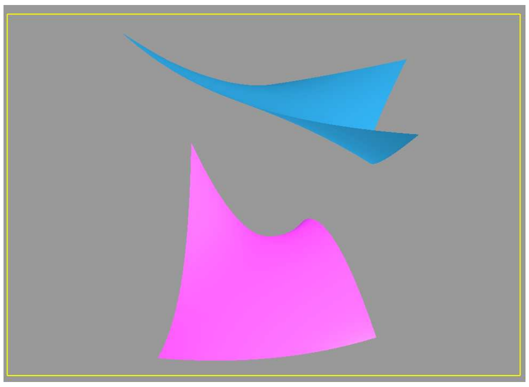
$$P_1(u,v) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j}V_{i,j}B_i^{n_u}(u)B_j^{n_v}(v)}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j}B_i^{n_u}(u)B_j^{n_v}(v)},$$

$$P_2(u,v) = \frac{\sum_{i=0}^{\overline{n}_u}\sum_{j=0}^{\overline{n}_v} \overline{w}_{i,j}\overline{V}_{i,j}B_i^{\overline{n}_u}(u)B_j^{\overline{n}_v}(v)}{\sum_{i=0}^{\overline{n}_u}\sum_{j=0}^{\overline{n}_v} \overline{w}_{i,j}B_i^{\overline{n}_u}(u)B_j^{\overline{n}_v}(v)}.$$

  First we need the degrees of these patches to be equal. If $n_u > \overline{n}_u$ we perform degree elevation in u direction $(n_u - \overline{n}_u)$ times (for degree elevation of Bézier curve, see Theorem 2.3.5) on patch P_2, otherwise we perform degree elevation $(\overline{n}_u - n_u)$ times on patch P_1. Similarly in v direction. We get two rational patches with equal degrees

$$P_1(u,v) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j}V_{i,j}B_i^{n_u}(u)B_j^{n_v}(v)}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j}B_i^{n_u}(u)B_j^{n_v}(v)},$$

$$P_2(u,v) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} \overline{w}_{i,j}\overline{V}_{i,j}B_i^{n_u}(u)B_j^{n_v}(v)}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} \overline{w}_{i,j}B_i^{n_u}(u)B_j^{n_v}(v)}.$$


(a) Two Bézier tensor-product patches.

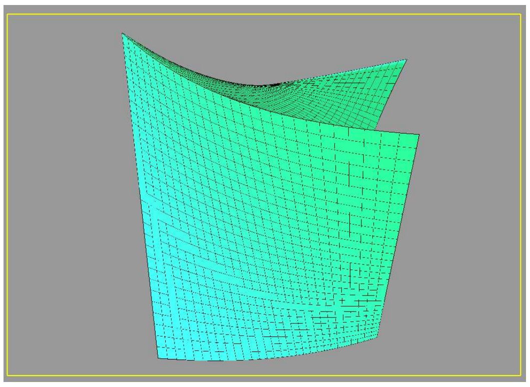

(b) Ruled volume in Bézier tensor-product form that connects two Bézier tensor-product patches.

Figure 3.1: Ruled volume in Bézier tensor-product form.

Because a ruled volume represents a linear interpolation of two patches, we can write the ruled volume given by these two patches as a linear Bernstein polynomial in third direction:

$$RV(u,v,w) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{1} q_{i,j,k} W_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^1(w)}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{1} q_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^1(w)}$$

$$u, v, w \in \langle 0, 1 \rangle$$

where $W_{i,j,0} = V_{i,j}, W_{i,j,1} = \overline{V}_{i,j}, w_{i,j,0} = w_{i,j}, q_{i,j,1} = \overline{w}_{i,j}$. This process is illustrated in Figure 3.1.

- The second case of rational B-spline tensor-product patches is straightforward. The harder part is the creation of two compatible patches. We have to make compatible not only the degrees in each direction, but we also have to make the appropriate knot vectors equal. Let us have two rational B-spline tensor-product patches

$$P_1(u,v) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j} V_{i,j} N_i^{d_u}(u) N_j^{d_v}(v)}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j} N_i^{d_u}(u) N_j^{d_v}(v)}$$

with knot vectors $(u_1, u_2, ..., u_{m_u})$ and $(v_1, v_2, ..., v_{m_v})$, and

$$P_2(u,v) = \frac{\sum_{i=0}^{\overline{n}_u}\sum_{j=0}^{\overline{n}_v} \overline{w}_{i,j} \overline{V}_{i,j} \overline{N}_i^{\overline{d}_u}(u) \overline{N}_j^{\overline{d}_v}(v)}{\sum_{i=0}^{\overline{n}_u}\sum_{j=0}^{\overline{n}_v} \overline{w}_{i,j} \overline{N}_i^{\overline{d}_u}(u) \overline{N}_j^{\overline{d}_v}(v)}$$

with knot vectors $(\overline{u}_0, \overline{u}_1, ..., \overline{u}_{\overline{m}_u})$ and $(\overline{v}_0, \overline{v}_1, ..., \overline{v}_{\overline{m}_v})$. First we have to elevate the degree in each direction so that the appropriate degrees are equal just like in the case of Bézier volumes. For degree elevation of B-spline curves, see the remarks after Theorem 2.4.2. Then for this degree elevated patches with extended knot vectors, we have to make the knot vectors equal in each direction. We need to set the knot vector $(u_0, u_1, ..., u_{m_u}) \cup (\overline{u}_0, \overline{u}_1, ..., \overline{u}_{\overline{m}_u})$ as the knot vector in u direction for both P_1 and P_2. This can be done using the knot insertion mechanism for B-spline curves (see Theorem 2.4.2). We proceed similarly for v direction. After these operations, we get two rational B-spline patches

$$P_1(u,v) = \frac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{i,j} V_{i,j} N_i^{d_u}(u) N_j^{d_v}(v)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{ij} N_i^{d_u}(u) N_j^{d_v}(v)},$$

$$P_2(u,v) = \frac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \overline{w}_{i,j} \overline{V}_{i,j} N_i^{d_u}(u) N_j^{d_v}(v)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \overline{w}_{i,j} N_i^{d_u}(u) N_j^{d_v}(v)}$$

with equal knot vectors $(u_0, u_1, ..., u_{m_u})$ and $(v_0, v_1, ..., v_{m_v})$. Because ruled volume represents linear interpolation of two patches, we can construct ruled volume RV as rational B-spline tensor-product volume using linear (degree 1) B-spline functions in third direction

$$RV(u,v,w) = \frac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{1} q_{i,j,k} W_{i,j,k} N_i^{d_u}(u) N_j^{d_v}(v) N_k^1(w)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{1} q_{i,j,k} N_i^{d_u}(u) N_j^{d_v}(v) N_k^1(w)},$$

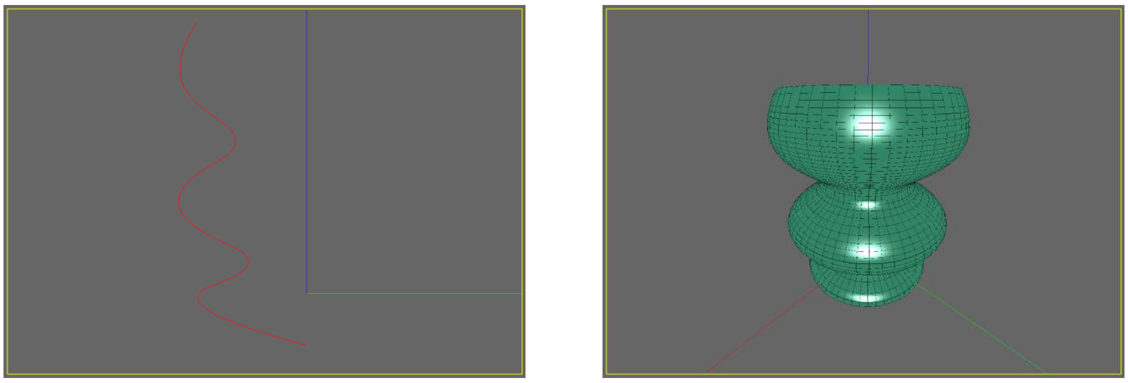
where $W_{i,j,0} = V_{i,j}$,$W_{i,j,1} = \overline{V}_{i,j}$,$q_{i,j,0} = w_{i,j}$,$q_{i,j,1} = \overline{w}_{i,j}$ and with knot vectors $(u_0, u_1, ..., u_{m_u})$, $(v_0, v_1, ..., v_{m_v})$ and $(0, 0, 1, 1)$.

- P_1 and P_2 can be given in the form of rational Bézier triangles with degrees n, $\overline{n}$ and the same domain - triangle ABC:

$$P_1(P) = \frac{\sum_{|\mathbf{i}|=n; dim(\mathbf{i})=3} w_{\mathbf{i}} V_{\mathbf{i}} B_{\mathbf{i}}^n(U(P))}{\sum_{|\mathbf{i}|=n; dim(\mathbf{i})=3} w_{\mathbf{i}} B_{\mathbf{i}}^n(U(P))}$$

$$P_2(P) = \frac{\sum_{|\mathbf{i}|=\overline{n}; dim(\mathbf{i})=3} \overline{w}_{\mathbf{i}} \overline{V}_{\mathbf{i}} B_{\mathbf{i}}^{\overline{n}}(U(P))}{\sum_{|\mathbf{i}|=\overline{n}; dim(\mathbf{i})=3} \overline{w}_{\mathbf{i}} B_{\mathbf{i}}^{\overline{n}}(U(P))}$$

for $P \in ABC$ and simplex barycentric coordinates U. First we perform degree elevation of the Bézier triangle with the smaller degree so both degrees are equal (for a general formula for degree elevation of Bézier triangles, see Theorem 2.3.5). Now we want to express the ruled volume made by P_1 and P_2 as an



(a) B-spline curve that will be rotated around the blue line.

(b) Resulting surface of revolution in B-spline tensor-product form.

Figure 3.2: Surface of revolution.

S-volume of degree n. To achieve this, we have to create the domain and the control points and its weights for this volume. To create the domain D, we simply extrude the triangle ABC into 3-sided prism. Such domain is convex and consists of 6 vertices and 6 facets. If we want to use generalized barycentric coordinates G from Theorem 2.1.7, we have to triangulate boundary of domain and get 8 triangular facets. To create the control points and the weights, we use linear combination of control points from the given Bézier triangles. Then the ruled volume RV as a rational S-volume from Bézier triangles P_1 and P_2 has degree n and is given as

$$RV(P) = \frac{\sum_{|\mathbf{i}|=n; dim(\mathbf{i})=6} q_{\mathbf{i}} W_{\mathbf{i}} B^n_{\mathbf{i}}(G(P))}{\sum_{|\mathbf{i}|=n; dim(\mathbf{i})=6} q_{\mathbf{i}} B^n_{\mathbf{i}}(G(P))},$$

where P is a point from the domain D. Control points and weights are computed as

$$W_{\mathbf{i}} = W_{(\mathbf{k},\mathbf{l})} = \frac{|\mathbf{k}|}{n} V_{\mathbf{k}} + \frac{|\mathbf{l}|}{n} \overline{V}_{\mathbf{l}},$$
$$q_{\mathbf{i}} = q_{(\mathbf{k},\mathbf{l})} = \frac{|\mathbf{k}|}{n} w_{\mathbf{k}} + \frac{|\mathbf{l}|}{n} \overline{w}_{\mathbf{l}},$$

for $|\mathbf{k}| + |\mathbf{l}| = n; dim(\mathbf{k}) = 3; dim(\mathbf{l}) = 3$. This approach can be extended for P_1,P_2 given as S-patches, but we have to be careful when creating appropriate the domain, on which we can use the chosen generalized barycentric coordinates.

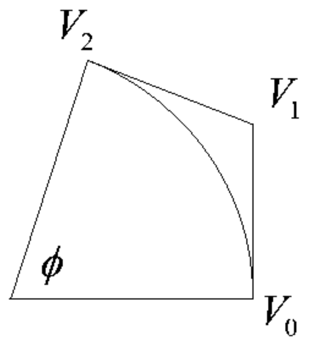
Another useful construction of surfaces is by revolution (Figure 3.2). In this method, we create a surface by rotating the points of a given curve around some axis by a given angle. This way we can represent many useful objects. We can adapt this approach to presents several methods for creation of volumes of revolution, as given in [Sam06].

Definition 3.1.2 *The volume of revolution VR is set of point created by rotating every point of a given patch P_1 around some axis by a given angle α. We can generate similar volume from a given curve C by creating a patch P as a ruled surface created by a curve C and a segment on the axis of revolution.*

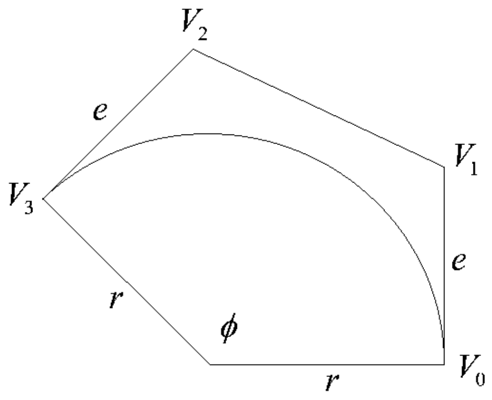
Usually the axis is one of the coordinate axes, we use the z-axis. The rotation of a given patch around this axis is represented by the rotation of its control points. The rotation of the control points is represented by the control points of their rotation trajectories. When rotating, these trajectories are circles or circular arcs. So we need to describe part of a circle or circular arc as Bézier or B-spline curve. As we mentioned before, it has to be rational Bézier or rational B-spline form. There are many ways how to achieve this, we will present several possibilities.

For the rational Bézier form, if we compute derivation at the beginning and at the end of control polygon, we get that the beginning and the end of control polygon is tangent to the circle. Using this property for rational Bézier curve of degree 2, we can represent circular arc with inner angle ϕ up to (but not including) π. The configuration of control points is shown in Figure 3.3(a), the weights are $w_0 = 1$, $w_1 = \cos(\frac{\phi}{2})$, $w_2 = 1$. If we elevate the degree of this curve (using Theorem 2.3.5), we get rational Bézier curve of degree 3, which can represent circular arc with angle up to (but not including) $\frac{3}{2}\pi$. The control points are given in Figure 3.3(b), the weights are $w_0 = 1$, $w_1 = \frac{1+2\cos(\phi/2)}{3}$, $w_2 = \frac{1+2\cos(\phi/2)}{3}$, $w_3 = 1$ and the distance is $e = r\frac{2\sin(\frac{\phi}{2})}{1+2\cos(\phi/2)}$. This way we can continue with the degree elevation. Finally a rational Bézier curve of degree 5 can represent whole circle, see Figure 3.3(c), with weights $w_0 = 1$, $w_1 = w_2 = w_3 = w_4 = \frac{1}{5}$, $w_5 = 1$.

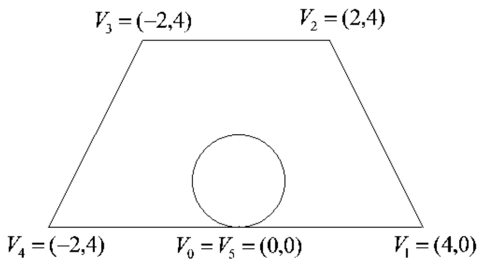
As we can see, to represent the whole circle, we need to use a higher degree rational Bézier curve. For larger arcs, the control polygon is not close to the curve an this is not desired for modeling, as we can see it in Figure 3.3(c). In these cases, it is better to use a curve that consists of several rational Bézier segments with low degree and construct a rational B-spline curve from these segments. It is easy to create such composited rational B-spline curve, we just need to put together the control polygons of segments into one polygon with appropriate weights. To create



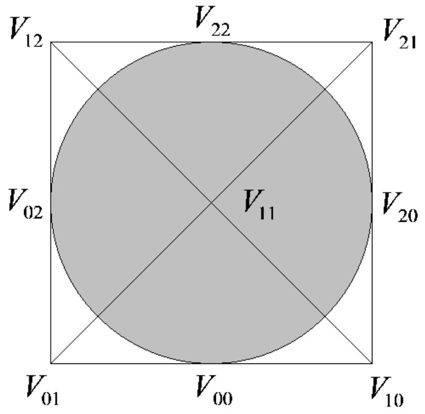
(a) Circular arc as rational Bézier curve of degree 2.



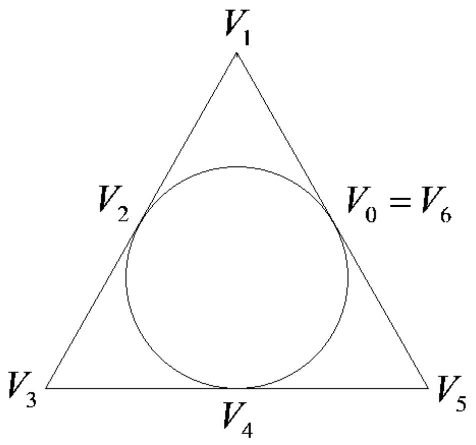
(b) Circular arc as rational Bézier curve of degree 3.



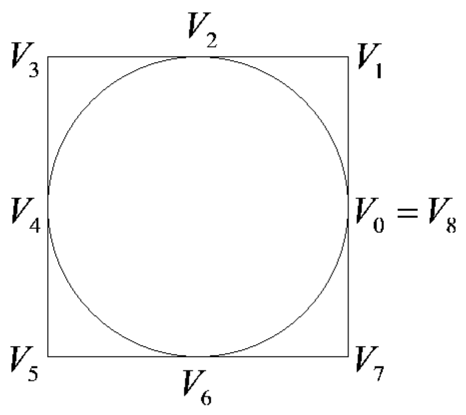
(c) Circle with radius 1 as rational Bézier curve of degree 4.



(d) Disk as rational Bézier tensor-product patch of degrees $(2, 2)$.



(e) Circle as rational B-spline curve of degree 2.



(f) Circle as rational B-spline curve of degree 2.

Figure 3.3: Circular arcs and disk in rational Bézier and B-spline form.

the knot vector, if we have m segments of degree d, then the knot vector has the form $(0, 0, .., 0, 1, 1, .., 1, ..., m, m, .., m)$ where each knot has multiplicity d except the first and the last with multiplicity $(d + 1)$. Figure 3.3(e) and Figure 3.3(f) show rational B-spline curves with given control points that represent circles and were constructed from rational Bézier curves of degree 2 using the above approach. For triangular configuration (Figure 3.3(e)), the knot vector is $(0, 0, 0, 1, 1, 2, 2, 3, 3, 3)$ and weights are $w_0 = w_2 = w_4 = w_6 = 1$, $w_1 = w_3 = w_5 = \frac{1}{2}$. For square configuration (Figure 3.3(f)), the knot vector is $(0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 4)$ and weights are $w_0 = w_2 = w_4 = w_6 = w_8 = 1$, $w_1 = w_3 = w_5 = w_5 = \frac{\sqrt{2}}{2}$.

Based on the form of the given curve or surface, we can construct rational Bézier or B-spline tensor-product volume of revolution. Let us have rational Bézier tensor-product surface

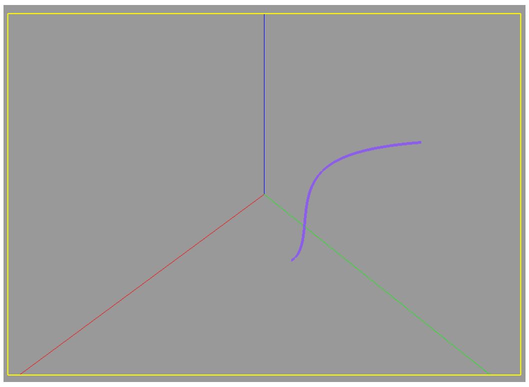
$$P_1(u, v) = \frac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{i,j} V_{i,j} B_i^{n_u}(u) B_j^{n_v}(v)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{i,j} B_i^{n_u}(u) B_j^{n_v}(v)}$$

which we want to rotate around the z-axis by an angle α. To achieve this, we rotate each control point of P_1 around the z-axis, more precisely we describe this rotation in the means of the control points of the rotation trajectories. For each control point $V_{i,j}$ of patch P_1, we prepare a rational Bézier curve $ARC_{i,j}(w) = \frac{\sum_{k=0}^{nw} p_k Q_k^{i,j} B_k^{nw}(w)}{\sum_{k=0}^{nw} p_k B_k^{nw}(w)}$, such that it represents a planar circular arc of angle α in E^3 and is parallel to the coordinate plane xy with the center on the z-axis and starting in point $V_{i,j}$, i.e. $Q_0^{i,j} = V_{i,j}$. Then the volume of revolution based on these parameters has the form

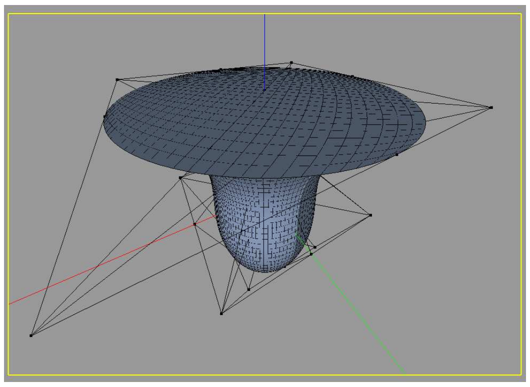
$$VR(u, v, w) = \frac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} q_{i,j,k} W_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} q_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w)},$$

where $W_{i,j,k} = Q_k^{i,j}$ and $q_{i,j,k} = w_{i,j} p_k$. These new control points define the above mentioned rotation of control points of the given patch P_1.

If the rotating object is a curve and we are rotating by the angle 2π, we can use another approach. A disk can be described as rational Bézier tensor-product patch with degrees $(2, 2)$, weights $w_{00} = w_{20} = w_{02} = w_{11} = w_{22} = 1$, $w_{10} = w_{01} = w_{21} = w_{12} = \frac{\sqrt{2}}{2}$ and the control points as seen in Figure 3.3(d). This can be easily shown, because the boundary curves are rational Bézier curves and their descriptions give a quarter of the circle. Now let us have rational Bézier curve $RBC(u) = \frac{\sum_{i=0}^{n_u} w_i V_i B_i^n(u)}{\sum_{i=0}^{n_u} w_i B_i^n(u)}$. Then the volume of revolution that is created by rotating $RBC(u)$ by angle 2π



(a) Rational Bézier curve that will be rotated around the blue line.



(b) Volume of revolution in rational Bézier form with displayed control points.

Figure 3.4: Rational Bézier tensor-product volume of revolution.

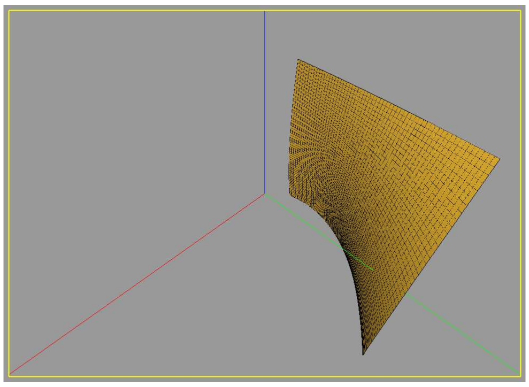
around z-axis has rational Bézier tensor-product volume form

$$VR(u,v,w) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{2}\sum_{k=0}^{2} q_{i,j,k} W_{i,j,k} B_i^{n_u}(u) B_j^2(v) B_k^2(w)}{\sum_{i=0}^{n_u}\sum_{j=0}^{2}\sum_{k=0}^{2} q_{i,j,k} B_i^{n_u}(u) B_j^2(v) B_k^2(w)}.$$

For each control point V_i of the given curve, we construct a planar disk $FC_i(v,w) = \frac{\sum_{j=0}^{2}\sum_{k=0}^{2} r_{j,k} S_{j,k}^i B_j^2(v) B_k^2(w)}{\sum_{j=0}^{2}\sum_{k=0}^{2} r_{j,k} B_j^2(v) B_k^2(w)}$ that is parallel to the coordinate plane xy, has the center on the coordinate axis z and $S_{0,0}^i = V_i$. Then $W_{i,j,k} = S_{j,k}^i$ and $q_{i,j,k} = w_i r_{j,k}$. Rational Bézier tensor-product volume created this way is displayed in Figure 3.4. The described approaches for rational Bézier volumes of revolution can be easily adapted for B-spline volumes (see Figure 3.5). The only difference is in using B-spline basis functions instead of Bernstein polynomials and in describing the circular arcs in B-spline form. The knot vectors are just copied from the given curves or patches and from the description of circular arcs or disk.

Sweep surfaces are created when a given curve is moved along another given curve (Figure 3.6). In general terms, the moving curve can be transformed during sweeping. This way also the surfaces of revolution are general sweep surfaces, because the curve is swept along the circular arc and is rotated when sweeping. We will adapt this concept and present sweep volumes. We sweep the patches along a given curve without transformations of the curve when sweeping.

Definition 3.1.3 *Let us have a given patch P_1 and a curve C. Then the sweep volume SV is generated by the moving each point P of the patch P_1 such that the point P is moved along zhe curve C.*


(a) Rational B-spline tensor-product surface will be rotated around the blue line.

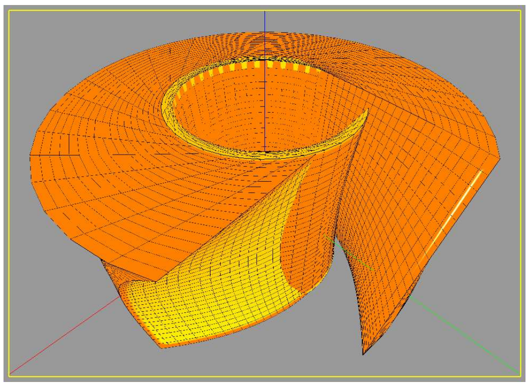
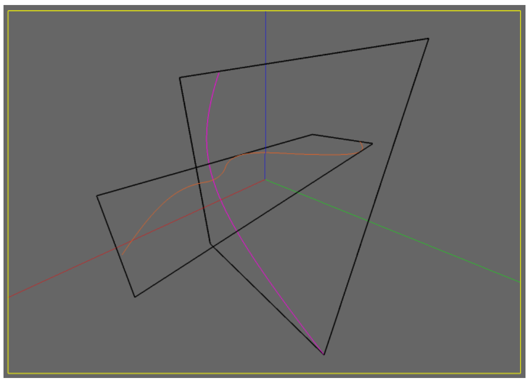

(b) Rational B-spline tensor-product volume of revolution after rotation of patch by 280°.

Figure 3.5: Rational B-spline tensor-product volume of revolution.


(a) Two B-spline curves.

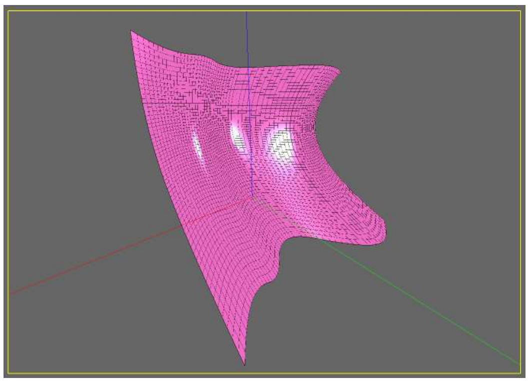
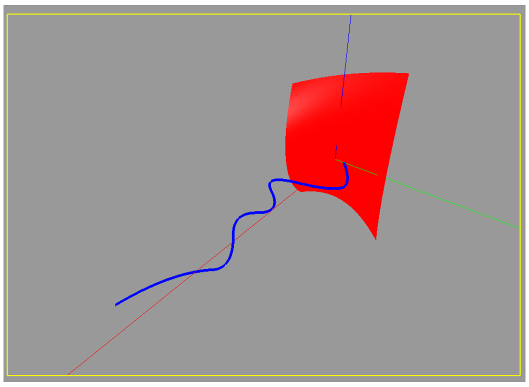

(b) B-spline tensor-product surface created by sweeping one curve by the second one.

Figure 3.6: Sweep surface.


(a) B-spline curve and B-spline tensor-product patch ready for sweeping.

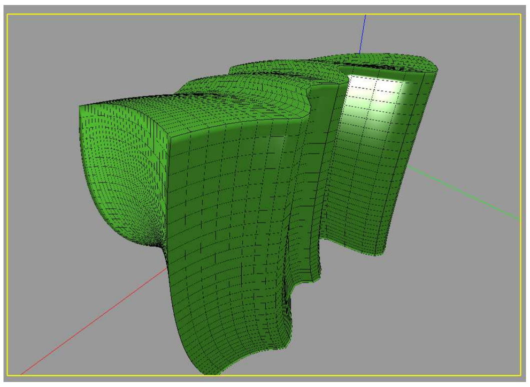

(b) B-spline tensor-product volume created by sweeping the patch along the curve.

Figure 3.7: Sweep volume.

It is easy to express the sweep volume in the form of parametric volume. All we need is to have C and P_1 in the same representation. If both objects are in the rational Bézier form, i.e.

$$P_1(u,v) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j} V_{i,j} B_i^{n_u}(u) B_j^{n_v}(v)}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} w_{i,j} B_i^{n_u}(u) B_j^{n_v}(v)},$$

$$C(w) = \frac{\sum_{k=0}^{n_w} p_k Q_k B_k^{n_w}(w)}{\sum_{k=0}^{n_w} p_k B_k^{n_w}(w)},$$

then SV expressed as rational Bézier tensor-product volume is

$$SV(u,v,w) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} q_{i,j,k} W_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w)}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} q_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w)},$$

where $q_{i,j,k} = w_{i,j} p_k$ and $W_{i,j,k} = V_{i,j} + Q_k - Q_0$. Figure 3.7 shows a sweep volume created this way. For rational B-spline tensor-product sweep volumes, the creation is identical to Bézier volumes. Also a transformation of P_1 can be introduced while sweeping. One example of "twisted" volume is in Figure 3.8, where given surface was also rotated around the z-axis during sweeping. A useful representation of objects must of course represent the basic objects in an easy way. We will show that using parametric volumes, many well-known objects can be easily described by techniques like ruled or sweep volumes. We will describe a procedure how to construct them and express them in the form of parametric volumes:

- Sphere is a volume of revolution, we rotate a half-disk in rational B-spline form or we rotate a half-circle and get rational Bézier form.

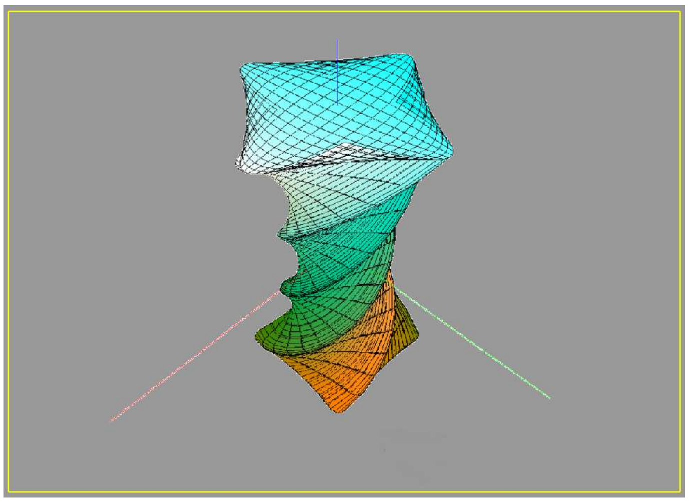


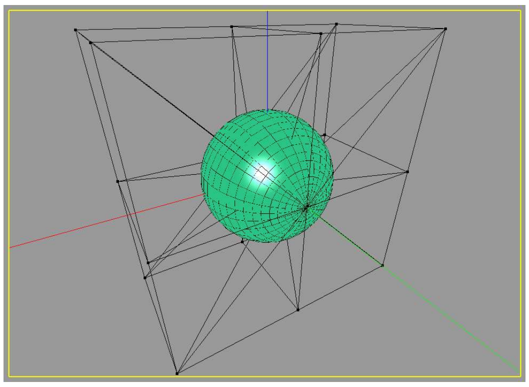
Figure 3.8: Sweep volume with patch rotation during sweeping.

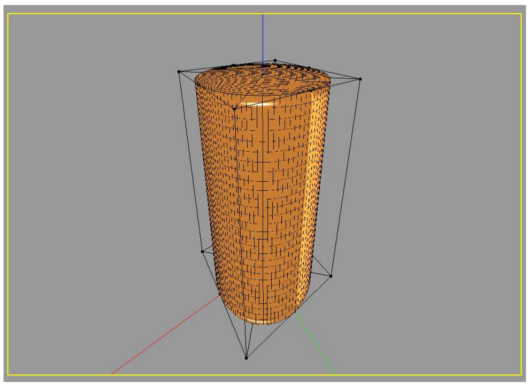
- Cylinder is a ruled volume, when we are joining two circles, or it is a volume of revolution when we rotate a line segment parallel to the axis of rotation.
- Cone can be described as general cylinder or as Bézier tetrahedron of degree 2.
- n-sided prism is an S-volume made as ruled volume connecting two filled n-sided polygons in the form of S-patches.
- Spherical tetrahedron is given by a spherical triangle and middle of the sphere, it can be described as rational Bézier tetrahedron.
- Torus is a volume of revolution when we are rotating disk.

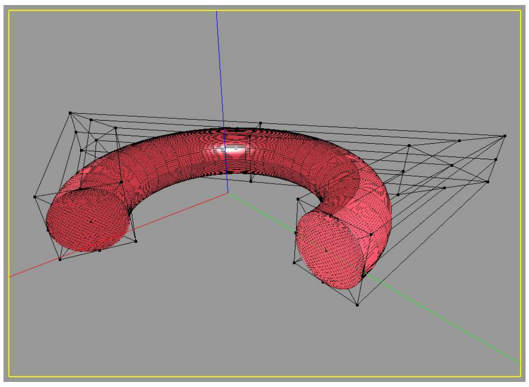
These basic objects are illustrated in Figure 3.9.

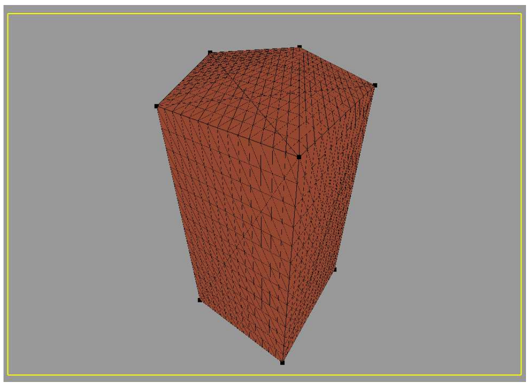
3.2 Decomposition

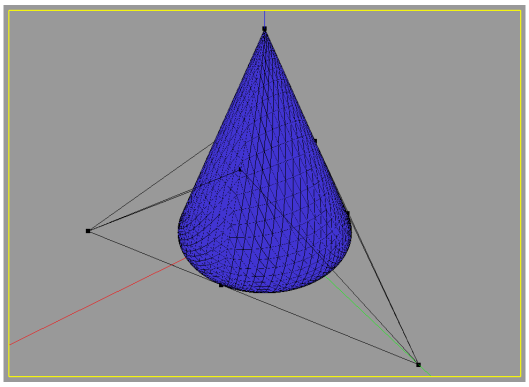
Decomposition of objects into smaller parts is useful in modeling, for example for intersection algorithm or for conversion between representations. For example in the case of S-patches, it is easier to work with Bézier triangles than with S-patches. In many situations we need to break an S-patch into set of Bézier triangles. This example of decomposition algorithm is illustrated in Figure 3.10. This kind of S-patch decomposition is more detailed described [Sam09a].


(a) Sphere as B-spline volume of revolution with degrees $(2, 2, 2)$.


(b) Cylinder as rational Bézier tensor-product volume of degrees $(2, 2, 1)$.


(c) Partial torus as B-spline volume of revolution with degrees $(2, 2, 2)$.


(d) 5-sided prism as S-volume of degree 1.


(e) Cone as Bézier tetrahedron of degree 2.

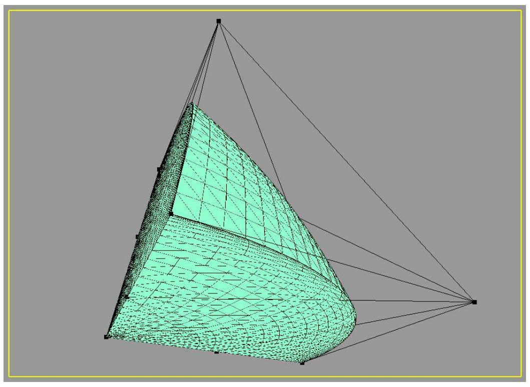
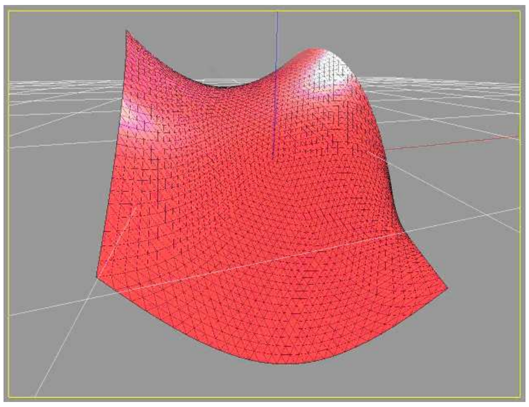

(f) Spherical tetrahedron as Bézier tetrahedron of degree 2.

Figure 3.9: Basic objects as parametric volumes.


(a) 5-sided S-patch.

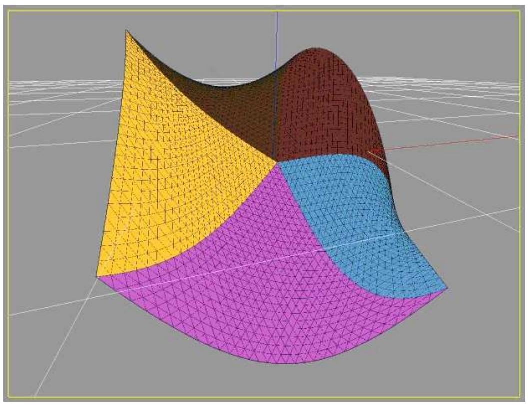

(b) S-patch decomposed into set of Bézier triangles.

Figure 3.10: Decomposition of S-patch.

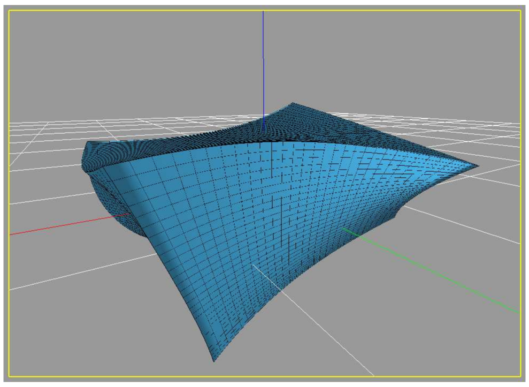
For these reasons, we need decomposition algorithms also for volumes. We define the decomposition as turning parametric volume into smaller parts while these parts are again described as parametric volumes. For example expressing a B-spline tensor-product volume as a union of Bézier tensor-product volumes can be used for the conversion between Bézier and B-spline representations. Another good example is breaking S-volume into set of Bézier tetrahedra. We will present several decomposition algorithms:

- Decomposing Bézier tensor-product volume into set of smaller Bézier tensor-product volumes.
- Decomposing B-spline tensor-product volume into set of Bézier tensor-product volumes.
- Decomposing S-volume into set of smaller S-volume.
- Decomposing S-volume into set of Bézier tetrahedra.

We utilize methods for solving these decomposition problems. The decomposition algorithms are the following:

- Given a Bézier tensor-product volume, we can work with each of its directions independently. We show the algorithm in u direction of volume. If we have Bézier tensor-product volume

$$BTPV(u,v,w) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} V_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w),$$


(a) Bézier tensor-product volume of degrees $(2,2,2)$.

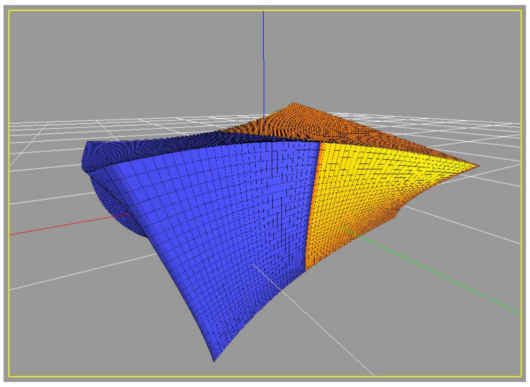

(b) Decomposition of volume into two volumes using de Casteljau algorithm.

Figure 3.11: Decomposition of Bézier tensor-product volume.

we can write it as Bézier curve in parameter u with constant v and w as

$$BTPV(u,v,w) = \sum_{i=0}^{n_u} \left[\sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{i,j,k} B_j^{n_v}(v) B_k^{n_w}(w) \right] B_i^{n_u}(u).$$

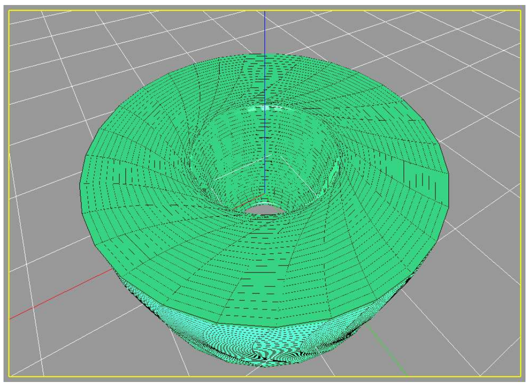
To perform de Casteljau algorithm in u direction, for each $j = 0,1,..,n_v$, $k = 0,1,..,n_w$, we work as in the curve case using control points $\{V_{i,j,k}\}_{i=0}^{n_u}$. Using parameter $\overline{u} \in \langle 0,1 \rangle$, de Casteljau algorithm (Theorem 2.3.6) gives two sets of new control points $\{V_{0,j,k}^{i}(\overline{u})\}_{i=0}^{n_u}$ and $\{V_{i,j,k}^{n-i}(\overline{u})\}_{i=0}^{n_u}$. After performing this algorithm $(n_v+1)(n_w+1)$ times, we get two Bézier tensor-product volumes $BTPV_1$, $BTPV_2$ that form the decomposition of the volume $BTPV$

$$BTPV_1(u,v,w) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{0,j,k}^{i}(\overline{u}) B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w),$$

$$BTPV_2(u,v,w) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{i,j,k}^{n-i}(\overline{u}) B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w).$$

The process of splitting Bézier tensor-product volume into two is illustrated in Figure 3.11. This approach can be repeated in v direction for parameter $\overline{v} \in \langle 0,1 \rangle$ and in w direction for parameter $\overline{w} \in \langle 0,1 \rangle$. This way we get the decomposition of Bézier volume into 8 smaller Bézier tensor-product volumes. Usual splitting values for de Casteljau algorithm are $\overline{u} = \overline{v} = \overline{w} = 0.5$.

- From Theorem 2.4.2, if we want to convert B-spline curve with degree d into piecewise Bézier form, we need to insert the knots so many times that each knot


(a) B-spline tensor-product volume created by rotating patch around z-axis.

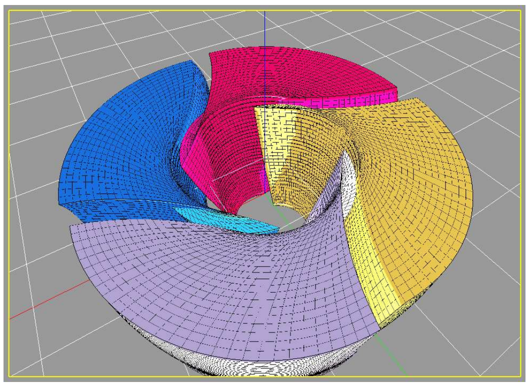
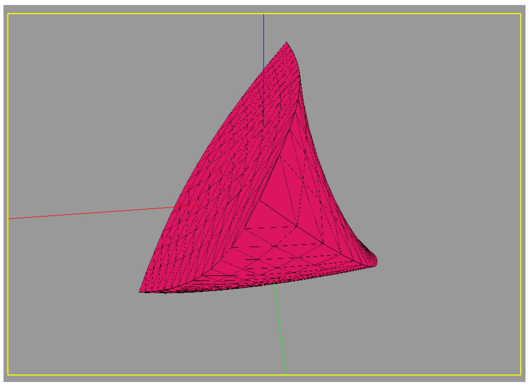

(b) Decomposition of volume into set of Bézier volumes.

Figure 3.12: Decomposition of B-spline tensor-product volume into set of Bézier tensor-product volumes.

will have multiplicity d with first and last knots having multiplicity $(d+1)$. In the case of volumes, all we need is to insert the knots in all three knot vectors so that this multiplicity condition is fulfilled. The decomposition of B-spline tensor-product volume into set of Bézier tensor-product volumes using knot insertions is illustrated in Figure 3.12.

- By the remark after Theorem 2.3.6, if we take a point P from the domain D of S-volume $SV^n(P) = \sum_{|\mathbf{i}|=n} V_\mathbf{i} B^n_\mathbf{i}(G(P))$, then de Casteljau algorithm for a point $Q \in D$ creates auxiliary points $V^k_\mathbf{j}(Q)$ for $k = 0, 1, .., n$; $|\mathbf{j}| = n-k$. From these auxiliary points, we can create k new S-volumes, where r-th S-volume is of degree n and has control points $W^r_\mathbf{i} = V^{i_r}_{\mathbf{i}-i_r\mathbf{e}_r}(Q)$. Figure 3.13 shows decomposition of Bézier tetrahedron into 4 smaller Bézier tetrahedra.

- For the decomposition of S-volume into set of Bézier tetrahedra, we used numerical solution. The control points are computed using system of a linear equations. Generally, for S-volume $SV^n(P) = \sum_{|\mathbf{i}|=n} V_\mathbf{i} B^n_\mathbf{i}(G(P))$ with domain D, we want to find new S-volume $\overline{SV}^m(P) = \sum_{|\mathbf{i}|=m} \overline{V}_\mathbf{i} B^m_\mathbf{i}(\overline{G}(P))$ as a restriction of SV^n on a subset $\overline{D}$ of domain D, i.e. $SV^n(P) = \overline{SV}^m(P)$ for each $P \in \overline{D} \subset D$. The subset $\overline{D}$ must be again some polyhedron given by points $\overline{A}_1, \overline{A}_2, .., \overline{A}_{\overline{k}}$ from D, such that we can apply generalized barycentric coordinates $\overline{G}$ within $\overline{D}$, where the total number of these points is $\overline{k} = dim(\mathbf{j})$. Loop and de deRose [LD89] showed that the degree of the new volume must


(a) S-volume of degree 2 and with 4 domain triangular facets (Bézier tetrahedron).

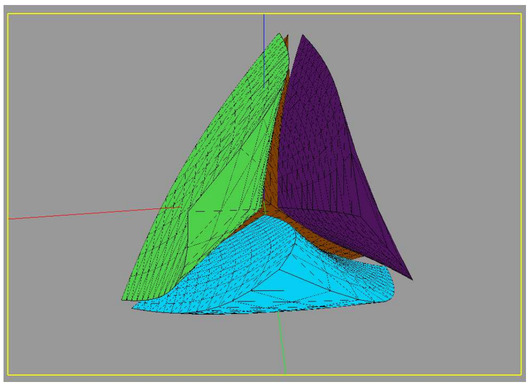

(b) Decomposition of volume into set of 4 S-volumes (Bézier tetrahedra).

Figure 3.13: Decomposition of S-volume into set of smaller S-volumes.

be at least $n(k-2)=m$, where $k=dim(\mathbf{i})$ is number of points defining D. Then the only thing that must be evaluated are the control points of volume $\overline{SV}^m$. It is known that

$$SV^n(P)=\sum_{|\mathbf{i}|=n} V_\mathbf{i} B^n_\mathbf{i}(G(P))=\overline{SV}^m(P)=\sum_{|\mathbf{j}|=m}\overline{V}_\mathbf{j} B^m_\mathbf{j}(\overline{G}(P)).\ \ \forall P\in\overline{D}$$

in this equation the control points $\overline{V}_\mathbf{j}$ are unknown variables. Because $|\mathbf{j}|=m$ and $dim(\mathbf{j})=\overline{k}$, the total number of these unknown control points is $N=\binom{m+\overline{k}-1}{\overline{k}}$ (see Theorem 2.2.1). We need N equations for solving the system with these unknowns. To generate these equations, we have to choose N points from $\overline{D}$ and insert them into the previous equation. The easiest way to generate domain points is as

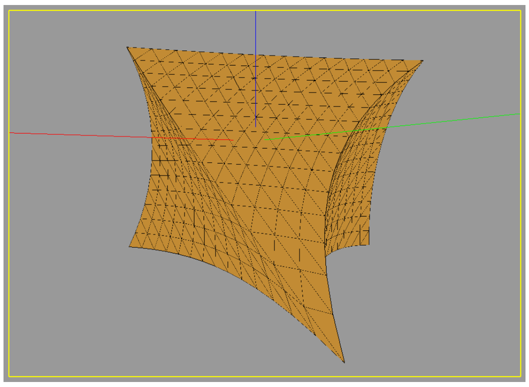
$$P_\mathbf{l}=\sum_{r=1}^{\overline{k}}\frac{l_r}{m}\overline{A}_r\ \ \mathbf{l}=(l_1,l_2,..,l_{\overline{k}}), |\mathbf{l}|=m.$$

The total number of points $P_\mathbf{l}$ is N, so we get a system of N linear equations with N unknowns

$$\sum_{|\mathbf{i}|=n} V_\mathbf{i} B^n_\mathbf{i}(G(P_\mathbf{l}))=\sum_{|\mathbf{j}|=m}\overline{V}_\mathbf{j} B^m_\mathbf{j}(\overline{G}(P_\mathbf{l})),$$

$$\mathbf{l}=(l_1.l_2,..,l_{\overline{k}}), |\mathbf{l}|=m.$$

By solving this system, we find the representation for the restriction in S-volume form. We use this approach for decomposition of S-volumes into set


(a) S-volume of degree 2 and with 6 domain triangular facets.

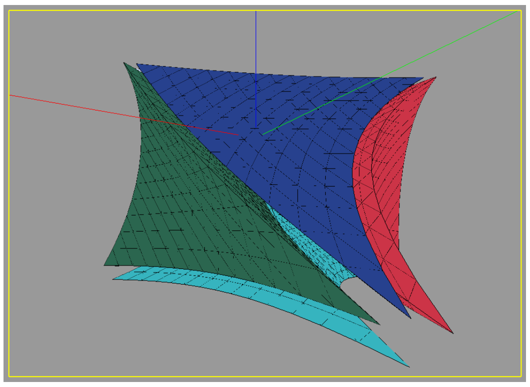

(b) Decomposition of volume into set of 6 Bézier tetrahedra.

Figure 3.14: Decomposition of S-volume into set of Bézier tetrahedra.

of Bézier tetrahedra. We work with convex domain D containing triangular boundary facets. We pick the point P as the barycenter and then for each facet ABC of the domain D, we create a tetrahedron $ABCP = \overline{D}$. Restricting the S-volume on that tetrahedron $\overline{D}$ we get Bézier tetrahedron defined on $\overline{D}$. Applying this process for each facet of domain D leads to the set of Bézier tetrahedra as the decomposition of given S-volume (3.14).

Further generalization of the decomposition algorithms is the computation of new volume over a subset of original volume domain. For example we can get the subvolume in the S-volume form from the interior of given S-volume.

3.3 Subdivision volumes

Subdivision curves and surfaces are very popular in current stage of computer graphics and geometric modeling. This popularity comes from the easy implementation of such subdivision algorithms and since the subdivision works on polygons and polygonal meshes, it is also easy to render. Basic introduction to subdivision surfaces can be found in [ZS00]. Description of basic schemes on triangular meshes is given in [Sam04a]. Subdivision objects are not exactly parametric objects, they generate their own special class, but they are related to parametric objects. In the limit process, they converge to parametric objects like curves, surfaces or volumes.

We will look first at curve case. We already have some examples of subdivision process for Bézier curves - degree elevation and de Casteljau algorithm created new control polygon from the old one resulting in better approximation of the original curve. Suppose that we have given a quadratic B-spline curve $N^2(u) = \sum_{i=0}^{n} V_i^0 N_i^2(u)$ with a knot vector $(0, 1, .., m)$, $m = n+3$ and with a control polygon $\mathbf{V^0} = (V_0^0, V_1^0, .., V_n^0)$. Now we insert new knots into each non-empty segment from the domain $\langle 2, n+1)$, in our case inserted knots are $2.5, 3.5, .., \frac{2n+1}{2}$. Inserting these knots does not change the original curve $N^2(u)$, it only changes the control polygon and the knot vector. Because the number of the original control points is $n+1$ and the number of inserted knots is $n-1$, the total number of points in the new control polygon is $2n$. We mark this new polygon as $\mathbf{V^1} = (V_0^1 V_1^1 .. V_{2n-1}^1)$. By Theorem 2.4.2, we can compute the new control points from the old ones as:

$$V_0^1 = \frac{3}{4}V_0^0 + \frac{1}{4}V_1^0$$

$$V_{2i+1}^1 = \frac{3}{4}V_{i+1}^0 + \frac{1}{4}V_{i+2}^0 \quad i = 0, ..., n-2$$

$$V_{2i}^1 = \frac{1}{4}V_i^0 + \frac{3}{4}V_{i+1}^0 \quad i = 1, ..., n-1$$

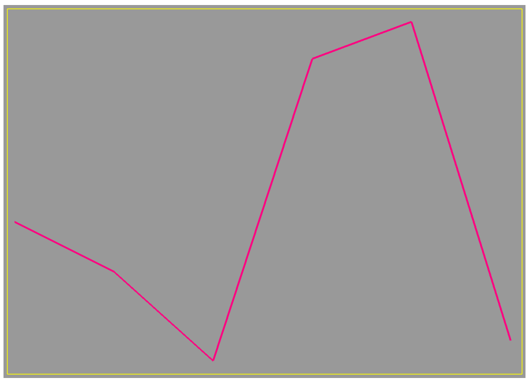
$$V_{2n-1}^1 = \frac{1}{4}V_{n-1}^0 + \frac{3}{4}V_n^0$$

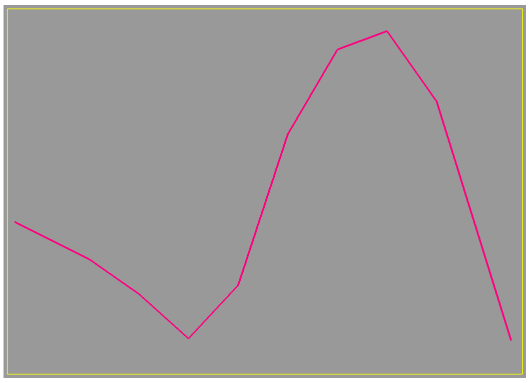
This way we get the new control polygon $\mathbf{V^1}$, which is a better approximation of the curve $N^2(u)$ than $\mathbf{V^0}$. The computation of $\mathbf{V^1}$ from $\mathbf{V^0}$ is the first step of subdivision algorithm. After computation of $\mathbf{V^1}$, we can repeat this algorithm on polygon $\mathbf{V^1}$ and get even better approximation of the original curve by polygon $\mathbf{V^2}$ and so on. for quadratic B-spline curves, this process is called corner-cutting algorithm or Chaikin subdivision process [Cha74], [Rie75] and it converges to the original quadratic B-spline curve $N^2(u)$. In real applications, only few steps of the algorithm are preformed to get a good approximation of the original uniform B-spline curve, as can be seen in Figure 3.15.

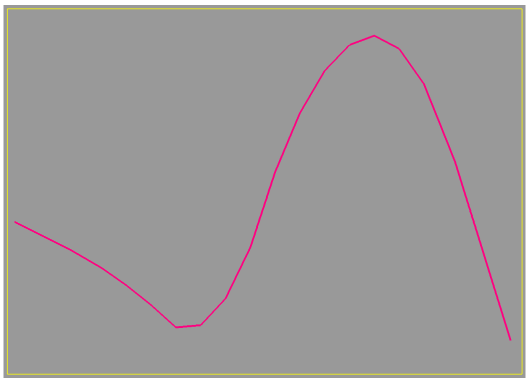
We can use described process also for a cubic B-spline curve $N^3(u) = \sum_{i=0}^{n} V_i^0 N_i^3(u)$ on domain $\langle 3, n+1)$ with control polygon $\mathbf{V^0} = (V_0^0 V_1^0 .. V_n^0)$ and knot vector $(0, 1, .., n+4)$. Again we will insert knots as middle values of non-empty intervals in domain. Then the new control polygon $\mathbf{V^1}$ has vertices $V_0^1, V_1^1, .., V_{2n}^1$, where

$$V_{2i+1}^1 = \frac{1}{2}V_i^0 + \frac{1}{2}V_{i+1}^0 \quad i = 0, 1, .., n-1,$$

$$V_{2i}^1 = \frac{1}{8}V_{i-1}^0 + \frac{3}{4}V_i^0 + \frac{1}{8}V_{i+1}^0 \quad i = 0, 1, .., n\ ,$$


(a) Control polygon.


(b) Polygon after one step of subdivision process.


(c) Polygon after two steps of subdivision process.

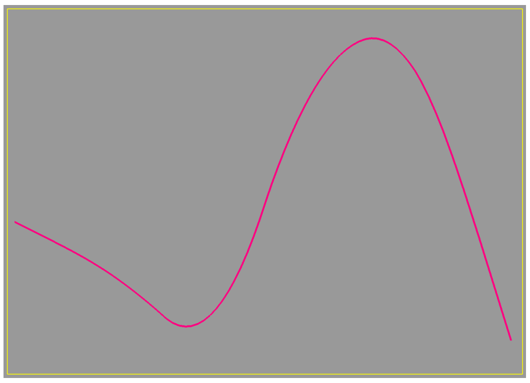

(d) Polygon after four steps of subdivision process.

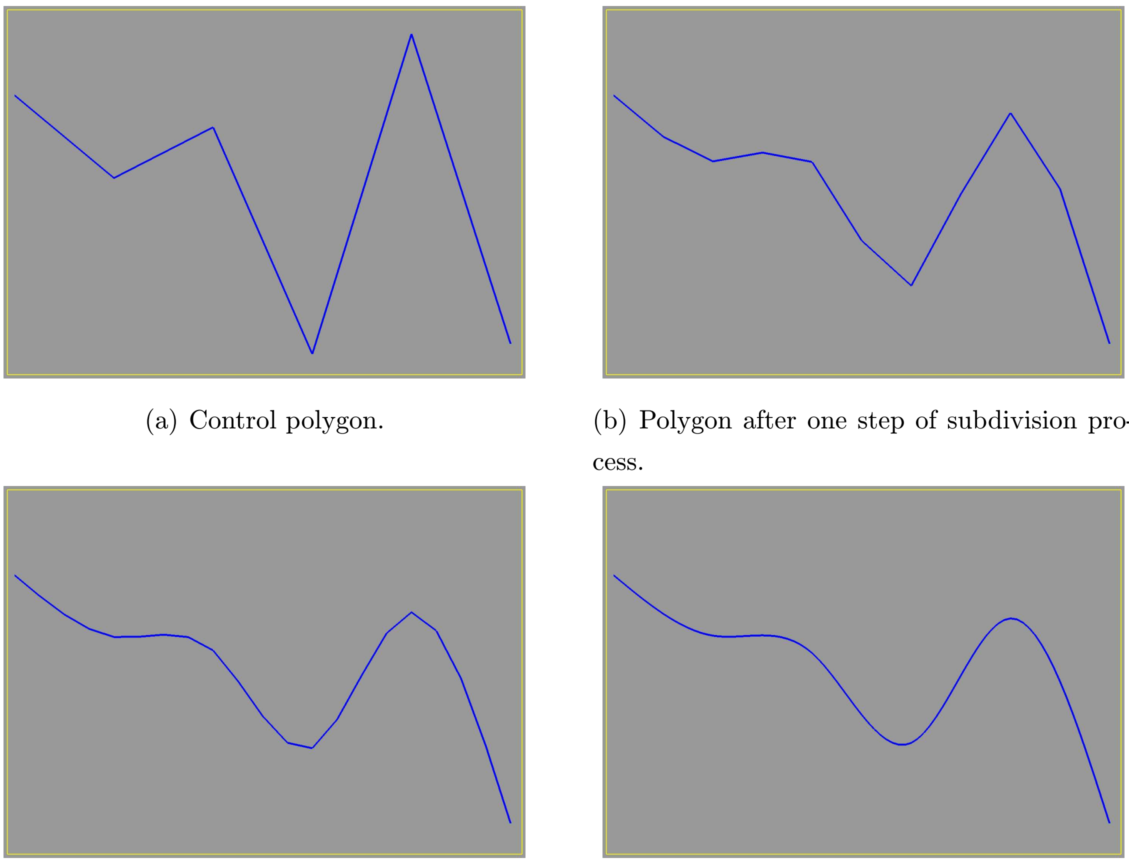
Figure 3.15: Chaikin subdivision process.

where $V_{-1}^0 = V_0^0$ and $V_{n+1}^0 = V_n^0$. This subdivision step can be described in geometrical way

- For each segment $V_i^0 V_{i+1}^0$ of polygon $\mathbf{V^0}$ create an edge point E_i^1 as the average of the end points V_i^0, V_{i+1}^0.
- For each point V_i^0 of polygon $\mathbf{V^0}$, create a new vertex point $V_i^1 = \frac{1}{2}V_i^0 + \frac{1}{4}E_{i-1}^1 + \frac{1}{4}E_i^1$, where E_{i-1}, E_i are previously created edge points of segments that are connected to the point V_i^0 (if particular segment does not exist, such edge point is equal to V_i^0).
- New polygon $\mathbf{V^1}$ is created by alternating vertex and edge points, $\mathbf{V^1} = V_0^1 E_0^1 V_1^1 E_1^1 ... E_{n-1}^1 V_n^1$.

This process converges to a cubic uniform B-spline curve, as illustrated in Figure

3.16.



(a) Control polygon.

(b) Polygon after one step of subdivision process.

(c) Polygon after two steps of subdivision process.

(d) Polygon after five steps of subdivision process.

Figure 3.16: Catmull-Clark subdivision process for curves.

The tensor-product surface case is then straightforward. If we have a uniform quadratic (in both directions) B-spline tensor-product surface over a net of control points $V^0_{i,j}$; $i = 0, 1, .., n_u$, $j = 0, 1, .., n_v$ with knot vectors $(0, 1, .., n_u + 3)$, $(0, 1, .., n_v + 3)$, we can perform Chaikin curve subdivision process in each direction. For each $j = 0, 1, .., n_v$ we do the subdivision of polygons $\{V^0_{i,j}\}_{i=0}^{n_u}$, getting new polygons $\{V'_{i,j}\}_{i=0}^{2n_u-1}$. Now for these points and for each $i = 0, 1, .., 2n_u - 1$ we do the curve subdivision step for polygons $\{V'_{i,j}\}_{j=0}^{n_v}$ getting $\{V^1_{i,j}\}_{j=0}^{2n_v-1}$. The equations for the new control points are

$$V^1_{2i,2j} = \frac{9}{16}V^0_{i,j} + \frac{3}{16}V^0_{i+1,j} + \frac{3}{16}V^0_{i,j+1} + \frac{1}{16}V^0_{i+1,j+1} \quad i = 0, .., n_u - 1; j = 0, .., n_v - 1,$$

$$V^1_{2i+1,2j} = \frac{9}{16}V^0_{i+1,j} + \frac{3}{16}V^0_{i,j} + \frac{3}{16}V^0_{i+1,j+1} + \frac{1}{16}V^0_{i,j+1} \quad i = 0, .., n_u - 1; j = 0, .., n_v - 1,$$

$$V^1_{2i,2j+1} = \frac{9}{16}V^0_{i,j+1} + \frac{3}{16}V^0_{i,j} + \frac{3}{16}V^0_{i+1,j+1} + \frac{1}{16}V^0_{i+1,j} \quad i = 0,..,n_u - 1; j = 0,..,n_v - 1,$$

$$V^1_{2i+1,2j+1} = \frac{9}{16}V^0_{i+,j+1} + \frac{3}{16}V^0_{i+1,j} + \frac{3}{16}V^0_{i,j+1} + \frac{1}{16}V^0_{i,j} \quad i = 0,..,n_u - 1; j = 0,..,n_v - 1.$$

Regular meshes generated using this subdivision process are called Doo-Sabin subdivision surfaces. The equations can be described in geometrical way and extended to work on meshes of arbitrary topology, not only for regular meshes [DS75]. Let us have given a control mesh $\mathbf{P_0}$ with arbitrary topology. Then Doo-Sabin subdivision surface over the control mesh $\mathbf{P_0}$ is defined in the following way: If $\mathbf{P_i}$ is the mesh after i-th step of the process, then mesh after one additional step it is $\mathbf{P_{i+1}}$ and is created using the following rules:

- For each face $F = (V_0V_1...V_K)$ of mesh $\mathbf{P_i}$ and for each vertex $V_i; i = 0, 1, ..., K$ of that face, create a new face-vertex point F_i. If $K = 3$, the face F is quadrilateral and $F_i = \frac{9}{16}V_i + \frac{3}{16}V_{(i-1) \bmod 4} + \frac{3}{16}V_{(i+1) \bmod 4} + \frac{1}{16}V_{(i+2) \bmod 4}$. If $K \neq 3$, then $F_i = \sum_{k=0}^{K} \alpha_k V_{(i+k) \bmod (K+1)}$, where $\alpha_0 = \frac{1+5K}{4}$ and $\alpha_k = \frac{1}{K}(3 + 2cos\frac{2\pi k}{K})$ for $k = 1, 2, ..., K$. These points become the points of mesh $\mathbf{P_{i+1}}$.
- For each vertex of mesh $\mathbf{P_i}$, create new face from all face-vertex points belonging to the given vertex.
- For each edge of mesh $\mathbf{P_i}$, create new face from all face-vertex points that were created from faces adjacent to the edge and end vertices of the edge.
- For each face of mesh $\mathbf{P_i}$, create new face from all face-vertex belonging to the given face.
- All newly created face-vertex points and faces determine new mesh $\mathbf{P_{i+1}}$.

This subdivision algorithm is also called corner the cutting algorithm because the new mesh is created from the old one by cutting the old mesh near its vertices and edges.

For approximation of a uniform cubic B-spline tensor-product surface, we can derive subdivision scheme based on Catmull-Clark subdivision for curves. Just like in case of Doo-Sabin surfaces, we use the curve case independently for u and then for v direction. For a given net of control points $V^0_{i,j}$, $i = 0, 1, .., n_u$, $j = 0, 1, .., n_v$, we get a new net of control points $V^1_{i,j}$, $i = 0, 1, .., 2n_u$, $j = 0, 1, .., 2n_v$ given by following formulas:

$$V^1_{2i+1,2j+j} = \frac{1}{4}V^0_{i,j} + \frac{1}{4}V^0_{i+1,j} + \frac{1}{4}V^0_{i,j+1} + \frac{1}{4}V^0_{i+1,j+1},$$
$$i = 0,1,..,n_u - 1 \quad j = 0,1,..,n_v - 1,$$
$$V^1_{2i,2j+1} = \frac{3}{8}V^0_{i,j+1} + \frac{3}{8}V^0_{i,j} + \frac{1}{16}V^0_{i-1,j} + \frac{1}{16}V^0_{i-1,j+1} + \frac{1}{16}V^0_{i+1,j} + \frac{1}{16}V^0_{i+1,j+1},$$
$$i = 0,1,..,n_u \quad j = 0,1,..,n_v - 1,$$
$$V^1_{2i+1,2j} = \frac{3}{8}V^0_{i+1,j} + \frac{3}{8}V^0_{i,j} + \frac{1}{16}V^0_{i,j-1} + \frac{1}{16}V^0_{i+1,j-1} + \frac{1}{16}V^0_{i,j+1} + \frac{1}{16}V^0_{i+1,j+1},$$
$$i = 0,1,..,n_u - 1 \quad j = 0,1,..,n_v,$$
$$V^1_{2i,2j} = \frac{9}{16}V^0_{i,j} + \frac{3}{32}V^0_{i-1,j} + \frac{3}{32}V^0_{i,j-1} + \frac{3}{32}V^0_{i+1,j} + \frac{3}{32}V^0_{i,j+1} +$$
$$+\frac{1}{64}V^0_{i-1,j-1} + \frac{1}{64}V^0_{i+1,j-1} + \frac{1}{64}V^0_{i-1,j+1} + \frac{1}{64}V^0_{i+1,j+1},$$
$$i = 0,1,..,n_u \quad j = 0,1,..,n_v.$$

The missing vertices are defined as $V^0_{-1,j} = V^0_{0,j}$, $V^0_{n_u+1,j} = V^0_{n_u,j}$, $V^0_{i,-1} = V^0_{i,0}$, $V^0_{i,n_v+1} = V^0_{i,n_v}$. This algorithm is a well-known Catmul-Clark subdivision algorithm for regular meshes and is heavily used in computer graphics or geometric modeling. One of its advantages is that it can be extended for meshes with arbitrary topology [CC78]. If we have given a control mesh $\mathbf{P_0}$ with arbitrary topology and if $\mathbf{P_i}$ is mesh after i-th step of Catmull-Clark subdivision process, then the mesh in the next step is $\mathbf{P_{i+1}}$ and is created using the following rules:

- For each face of mesh $\mathbf{P_i}$, create new face point F as the average of that face vertices.
- For each edge of mesh $\mathbf{P_i}$, create new edge point E as the average of that edge end vertices and face points of adjacent faces. If the edge is on the boundary, E is just the average of end vertices of the edge.
- For each vertex V of mesh $\mathbf{P_i}$, create new vertex point $\overline{V}$ as $\overline{V} = \frac{1}{n}Q + \frac{2}{n}R + \frac{n-3}{n}V$, where Q is the average of the new face points surrounding the vertex V, R is the average of the midpoints of the edges that share the vertex V and n is the number of edges that share the vertex V.
- All new face, edge and point vertices are vertices of mesh $\mathbf{P_{i+1}}$.

- For each face $F \in \mathbf{P_i}$ and for each vertex V of face F, create quadrilateral $ABCD$. A is the new vertex point from V, B and D are edge points of edges adjacent to V and C is face point of given face. All faces created this way are faces of the new mesh $\mathbf{P_{i+1}}$

This algorithm works on meshes of arbitrary topology, but after the first step of the process, the mesh contains only quads. The process produces meshes that approximate the control mesh, an example of the process is in Figure 3.17.

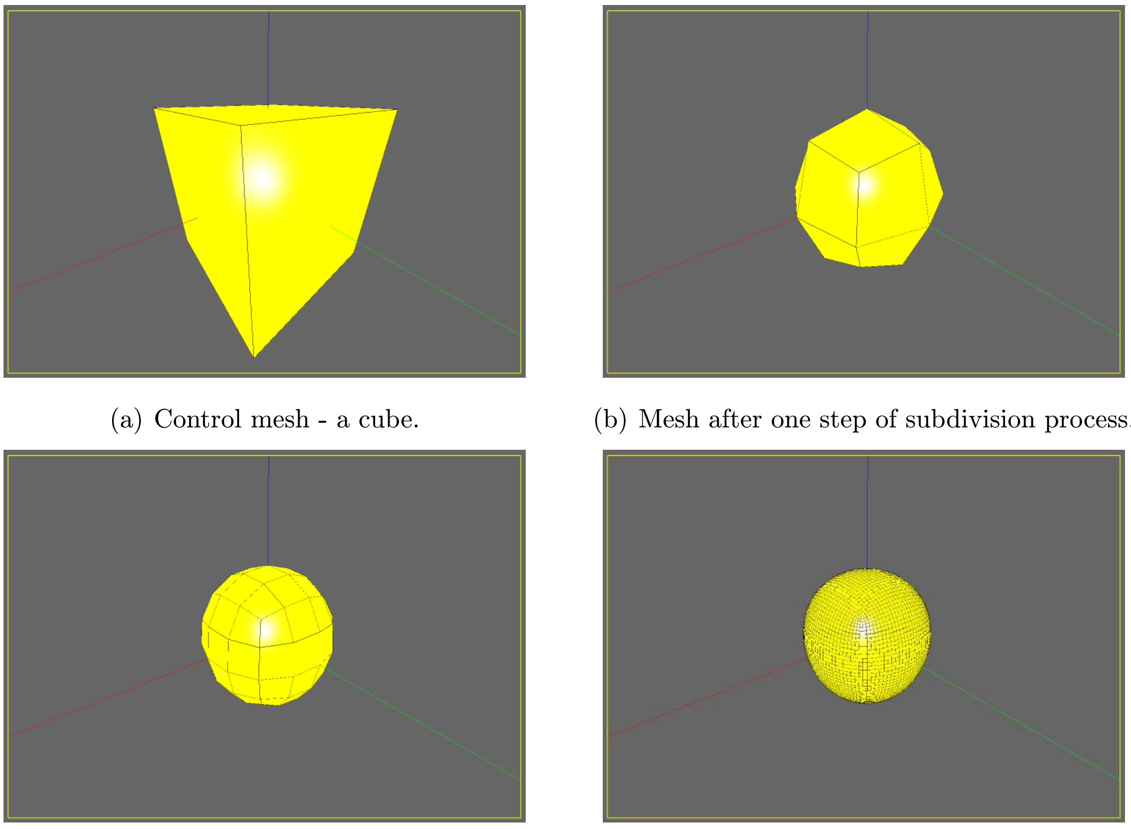

(a) Control mesh - a cube.
(b) Mesh after one step of subdivision process.
(c) Mesh after two steps of subdivision process.
(d) Mesh after five steps of subdivision process.

Figure 3.17: Catmull-Clark subdivision process.

Now we can move to parametric volumes. For uniform quadratic B-spline tensor-product volume $BSTPV^{2,2,2}(u,v,w) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V^0_{i,j,k} N^2_i(u) N^2_j(v) N^2_k(w)$ we can perform the insertion of knots into the knot vector like in the curve and surface cases. We get a new regular net of control points $V^1_{i,j,k}$; $i = 0,1,..,2n_u - 1$; $j = 0,1,..,2n_v - 1$; $k = 0,1,..,2n_w - 1$ given by eight formulas, we present just two of them:

$$V^1_{2i,2j,2k} = \frac{27}{64} V^0_{i,j,k} + \frac{9}{64} V^0_{i+1,j,k} + \frac{9}{64} V^0_{i,j+1,k} + \frac{9}{64} V^0_{i,j,k+1} +$$

$$+\frac{3}{64}V^0_{i+1,j+1,k}+\frac{3}{64}V^0_{i+1,j,k+1}+\frac{3}{64}V^0_{i,j+1,k+1}+\frac{1}{64}V^0_{i+1,j+1,k+1},$$

$$V^1_{2i+1,2j,2k}=\frac{27}{64}V^0_{i+1,j,k}+\frac{9}{64}V^0_{i,j,k}+\frac{9}{64}V^0_{i+1,j,k+1}+\frac{9}{64}V^0_{i+1,j+1,k}+$$
$$+\frac{3}{64}V^0_{i+1,j+1,k+1}+\frac{3}{64}V^0_{i,j+1,k}+\frac{3}{64}V^0_{i,j,k+1}+\frac{1}{64}V^0_{i,j+1,k+1},$$

All eight formulas can be expressed in one general formula. For $\alpha,\beta,\gamma=0,1$ we can write

$$V^1_{2i+\alpha,2j+\beta,2k+\gamma}=\frac{27}{64}V^0_{i+\alpha,j+\beta,k+\gamma}+\frac{9}{64}V^0_{i+((\alpha+1)\bmod 2),j+\beta,k+\gamma}+\frac{9}{64}V^0_{i+\alpha,j+((\beta+1)\bmod 2),k+\gamma}+$$
$$+\frac{9}{64}V^0_{i+\alpha,j+\beta,k+((\gamma+1)\bmod 2)}+\frac{3}{64}V^0_{i+((\alpha+1)\bmod 2),j+((\beta+1)\bmod 2),k+\gamma}+$$
$$+\frac{3}{64}V^0_{i+((\alpha+1)\bmod 2),j+\beta,k+((\gamma+1)\bmod 2)}+\frac{3}{64}V^0_{i+\alpha,j+((\beta+1)\bmod 2),k+((\gamma+1)\bmod 2)}+$$
$$+\frac{1}{64}V^0_{i+((\alpha+1)\bmod 2),j+((\beta+1)\bmod 2),k+((\gamma+1)\bmod 2)},$$
$$i=0,1,..,n_u-1,\quad j=0,1,..,n_v-1,\quad k=0,1,..,n_w-1.$$

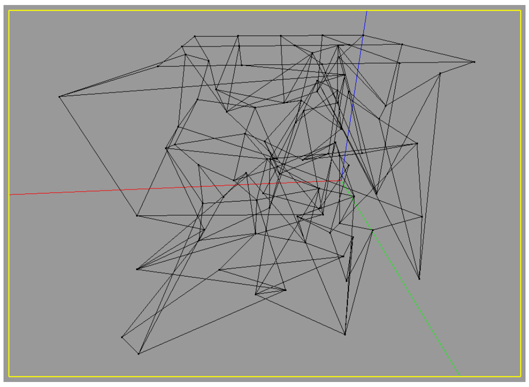
These formulas define one step of Doo-Sabin subdivision algorithm for regular volumes (Figure 3.18).

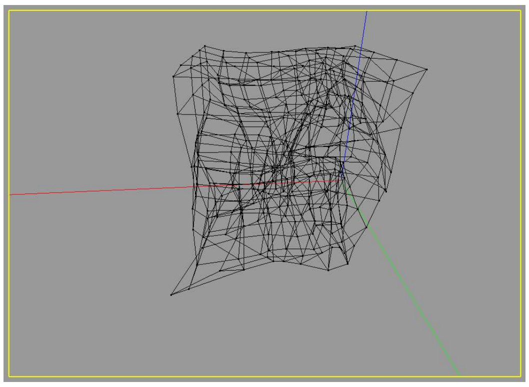
For Catmull-Clark subdivision process, we get the subdivision rules by inserting the knots uniformly in the knot vectors of cubic uniform B-spline tensor-product volume $BSTPV^{3,3,3}(u,v,w)=\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w}V^0_{i,j,k}N^3_i(u)N^3_j(v)N^3_k(w)$. Based on the curve and surface case, after one step of the subdivision process, we get new control points $V^1_{i,j,k}$; $i=0,1,..,2n_u$; $j=0,1,..,2n_v$; $k=0,1,..,2n_w$ from control points $V^0_{i,j,k}$; $i=0,1,..,n_u$; $j=0,1,..,n_v$; $k=0,1,..,n_w$. After extending the formulas in the third direction, we get formulas

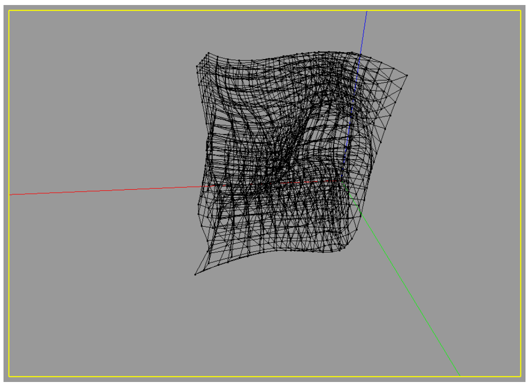
$$V^1_{2i+1,2j+1,2k+1}=\frac{1}{8}V^0_{i,j,k}+\frac{1}{8}V^0_{i+1,j,k}+\frac{1}{8}V^0_{i,j+1,k}+\frac{1}{8}V^0_{i,j,k+1}+$$
$$+\frac{1}{8}V^0_{i+1,j+1,k}+\frac{1}{8}V^0_{i+1,j,k+1}+\frac{1}{8}V^0_{i,j+1,k+1}+\frac{1}{8}V^0_{i+1,j+1,k+1},$$

where $i=0,1,..,n_u-1; j=0,1,..,n_v-1; k=0,1,..,n_w-1$.

$$V^1_{2i,2j+1,2k+1}=\frac{3}{16}V^0_{i,j,k}+\frac{3}{16}V^0_{i,j+1,k}+\frac{3}{16}V^0_{i,j,k+1}+\frac{3}{16}V^0_{i,j+1,k+1}+\frac{1}{32}V^0_{i-1,j,k}+$$
$$+\frac{1}{32}V^0_{i-1,j+1,k}+\frac{1}{32}V^0_{i-1,j,k+1}+\frac{1}{32}V^0_{i-1,j+1,k+1}+\frac{1}{32}V^0_{i+1,j,k}+\frac{1}{32}V^0_{i+1,j+1,k}+$$


(a) Initial parallerpiped lattice.


(b) Lattice after one step of subdivision process.


(c) Lattice after two steps of subdivision process.

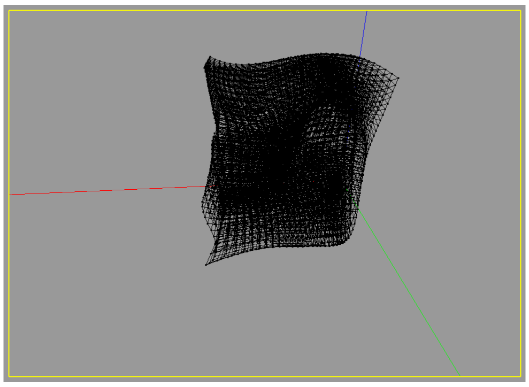

(d) Lattice after three of subdivision process.

Figure 3.18: Doo-Sabin subdivision volume.

$$+\frac{1}{32}V^0_{i+1,j,k+1}+\frac{1}{32}V^0_{i+1,j+1,k+1},$$

where $i = 0, 1, .., n_u; j = 0, 1, .., n_v - 1; k = 0, 1, .., n_w - 1.$

$$V^1_{2i+1,2j,2k+1} = \frac{3}{16}V^0_{i,j,k} + \frac{3}{16}V^0_{i+1,j,k} + \frac{3}{16}V^0_{i,j,k+1} + \frac{3}{16}V^0_{i+1,j,k+1} + \frac{1}{32}V^0_{i,j-1,k}+$$

$$+\frac{1}{32}V^0_{i+1,j-1,k} + \frac{1}{32}V^0_{i,j-1,k+1} + \frac{1}{32}V^0_{i+1,j-1,k+1} + \frac{1}{32}V^0_{i,j+1,k} + \frac{1}{32}V^0_{i+1,j+1,k}+$$

$$+\frac{1}{32}V^0_{i,j+1,k+1}+\frac{1}{32}V^0_{i+1,j+1,k+1},$$

where $i = 0, 1, .., n_u - 1; j = 0, 1, .., n_v; k = 0, 1, .., n_w - 1.$

$$V^1_{2i+1,2j+1,2k} = \frac{3}{16}V^0_{i,j,k} + \frac{3}{16}V^0_{i+1,j,k} + \frac{3}{16}V^0_{i,j+1,k} + \frac{3}{16}V^0_{i+1,j+1,k} + \frac{1}{32}V^0_{i,j,k-1}+$$

$$+\frac{1}{32}V^0_{i+1,j,k-1} + \frac{1}{32}V^0_{i,j+1,k-1} + \frac{1}{32}V^0_{i+1,j+1,k-1} + \frac{1}{32}V^0_{i,j,k+1} + \frac{1}{32}V^0_{i+1,j,k+1}+$$

$$+\frac{1}{32}V^0_{i,j+1,k+1}+\frac{1}{32}V^0_{i+1,j+1,k+1},$$

where $i=0,1,..,n_u-1; j=0,1,..,n_v-1; k=0,1,..,n_w$.

$$V^1_{2i+1,2j,2k}=\frac{9}{32}V^0_{i,j,k}+\frac{9}{32}V^0_{i+1,j,k}+\frac{3}{64}V^0_{i,j,k-1}+\frac{3}{64}V^0_{i,j-1,k}+\frac{3}{64}V^0_{i,j,k+1}+$$
$$+\frac{3}{64}V^0_{i,j+1,k}+\frac{3}{64}V^0_{i+1,j,k-1}+\frac{3}{64}V^0_{i+1,j-1,k}+\frac{3}{64}V^0_{i+1,j,k+1}+\frac{3}{64}V^0_{i+1,j+1,k}+$$
$$+\frac{1}{128}V^0_{i,j-1,k-1}+\frac{1}{128}V^0_{i,j+1,k-1}+\frac{1}{128}V^0_{i,j-1,k+1}+\frac{1}{128}V^0_{i,j+1,k+1}+$$
$$+\frac{1}{128}V^0_{i+1,j-1,k-1}+\frac{1}{128}V^0_{i+1,j+1,k-1}+\frac{1}{128}V^0_{i+1,j-1,k+1}+\frac{1}{128}V^0_{i+1,j+1,k+1},$$

where $i=0,1,..,n_u-1; j=0,1,..,n_v; k=0,1,..,n_w$.

$$V^1_{2i,2j+1,2k}=\frac{9}{32}V^0_{i,j,k}+\frac{9}{32}V^0_{i,j+1,k}+\frac{3}{64}V^0_{i,j,k-1}+\frac{3}{64}V^0_{i-1,j,k}+\frac{3}{64}V^0_{i,j,k+1}+$$
$$+\frac{3}{64}V^0_{i+1,j,k}+\frac{3}{64}V^0_{i,j+1,k-1}+\frac{3}{64}V^0_{i-1,j+1,k}+\frac{3}{64}V^0_{i,j+1,k+1}+\frac{3}{64}V^0_{i+1,j+1,k}+$$
$$+\frac{1}{128}V^0_{i-1,j,k-1}+\frac{1}{128}V^0_{i+1,j,k-1}+\frac{1}{128}V^0_{i-1,j,k+1}+\frac{1}{128}V^0_{i+1,j,k+1}+$$
$$+\frac{1}{128}V^0_{i-1,j+1,k-1}+\frac{1}{128}V^0_{i+1,j+1,k-1}+\frac{1}{128}V^0_{i-1,j+1,k+1}+\frac{1}{128}V^0_{i+1,j+1,k+1},$$

where $i=0,1,..,n_u; j=0,1,..,n_v-1; k=0,1,..,n_w$.

$$V^1_{2i,2j,2k+1}=\frac{9}{32}V^0_{i,j,k}+\frac{9}{32}V^0_{i,j,k+1}+\frac{3}{64}V^0_{i-1,j,k}+\frac{3}{64}V^0_{i,j-1,k}+\frac{3}{64}V^0_{i+1,j,k}+$$
$$+\frac{3}{64}V^0_{i,j+1,k}+\frac{3}{64}V^0_{i-1,j,k+1}+\frac{3}{64}V^0_{i,j-1,k+1}+\frac{3}{64}V^0_{i+1,j,k+1}+\frac{3}{64}V^0_{i,j+1,k+1}+$$
$$+\frac{1}{128}V^0_{i-1,j-1,k}+\frac{1}{128}V^0_{i-1,j+1,k}+\frac{1}{128}V^0_{i+1,j-1,k}+\frac{1}{128}V^0_{i+1,j+1,k}+$$
$$+\frac{1}{128}V^0_{i-1,j-1,k+1}+\frac{1}{128}V^0_{i-1,j+1,k+1}+\frac{1}{128}V^0_{i+1,j-1,k+1}+\frac{1}{128}V^0_{i+1,j+1,k+1},$$

where $i=0,1,..,n_u; j=0,1,..,n_v; k=0,1,..,n_w-1$.

$$V^1_{2i,2j,2k}=\frac{27}{64}V^0_{i,j,k}+\frac{9}{128}V^0_{i-1,j,k}+\frac{9}{128}V^0_{i+1,j,k}+\frac{9}{128}V^0_{i,j-1,k}+\frac{9}{128}V^0_{i,j+1,k}+$$
$$+\frac{9}{128}V^0_{i,j,k-1}+\frac{9}{128}V^0_{i,j,k+1}+\frac{3}{256}V^0_{i,j-1,k-1}+\frac{3}{256}V^0_{i,j+1,k-1}+\frac{3}{256}V^0_{i,j-1,k+1}+$$
$$+\frac{3}{256}V^0_{i,j+1,k+1}+\frac{3}{256}V^0_{i-1,j,k-1}+\frac{3}{256}V^0_{i+1,j,k-1}+\frac{3}{256}V^0_{i-1,j,k+1}+\frac{3}{256}V^0_{i+1,j,k+1}+$$
$$+\frac{3}{256}V^0_{i-1,j-1,k}+\frac{3}{256}V^0_{i+1,j-1,k}+\frac{3}{256}V^0_{i-1,j+1,k}+\frac{3}{256}V^0_{i+1,j+1,k}+\frac{1}{512}V^0_{i-1,j-1,k-1}+$$

$$+\frac{1}{512}V^0_{i-1,j+1,k-1}+\frac{1}{512}V^0_{i-1,j-1,k+1}+\frac{1}{512}V^0_{i-1,j+1,k+1}+\frac{1}{512}V^0_{i+1,j-1,k-1}+$$
$$+\frac{1}{512}V^0_{i+1,j+1,k-1}+\frac{1}{512}V^0_{i+1,j-1,k+1}+\frac{1}{512}V^0_{i+1,j+1,k+1},$$

where $i = 0, 1, .., n_u; j = 0, 1, .., n_v; k = 0, 1, .., n_w$. The control points with undefined indices are defined as $V^0_{-1,j,k} = V^0_{0,j,k}$, $V^0_{n_u+1,j,k} = V^0_{n_u,j,k}$, $V^0_{i,-1,k} = V^0_{i,0,k}$, $V^0_{i,n_v+1,k} = V^0_{i,n_v,k}$, $V^0_{i,j,-1} = V^0_{i,j,0}$, $V^0_{i,j,n_w+1} = V^0_{i,j,n_w}$. This process for regular nets of control points converges to cubic B-spline tensor-product volume (see Figure 3.19). Joy and MacCracken [JM99], [MJ96] extended this approach to work on lattices of arbitrary topology, not just the regular one, by giving a geometry-based rules for the subdivision step. For a given 3-dimensional the lattice $\mathbf{P_0}$ with arbitrary topology, Catmull-Clark subdivision volume over the control lattice $\mathbf{P_0}$ is defined: If $\mathbf{P_i}$ is lattice after i-th step of the process, then the lattice of the next step is $\mathbf{P_{i+1}}$ and it is created using the following rules:

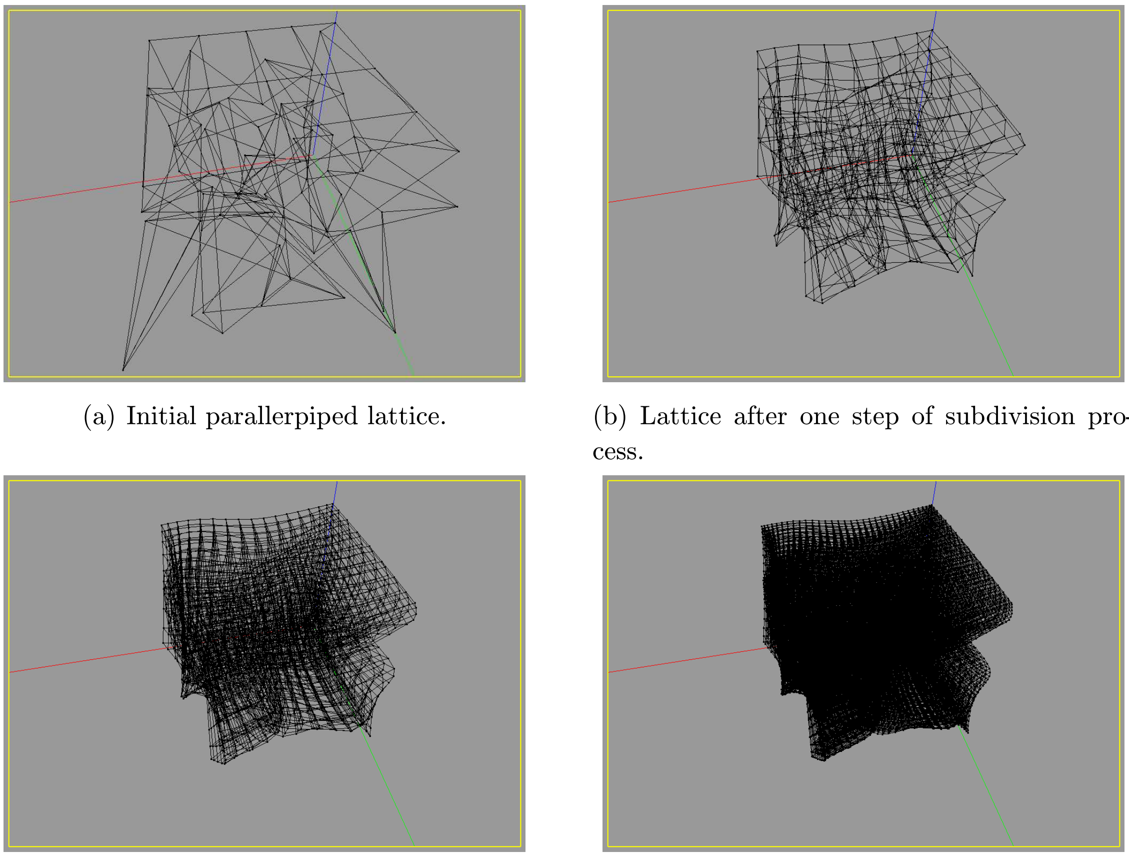
- Solid points - the points are the averages of the control points in the lattice that bound the cell containing this point.

- Face points - the points can be written as $\frac{1}{4}S_1 + \frac{1}{4}S_2 + \frac{1}{2}F$ where S_1 and S_2 are the solid points of the two cells that contain the face and F is the face point calculated by averaging the control points that form the face.

- Edge points - the points can be written as $\frac{1}{4}S+\frac{1}{4}E+\frac{1}{2}F$, where S is the average of the solid points for those cells that contain the edge, F is the average of the face points (calculated as the average of the control points forming the face) for those faces that contain the edge, and E is the midpoint of the edge.

- Vertex points - the points can be written as $\frac{1}{8}S + \frac{3}{8}F + \frac{3}{8}E + \frac{1}{8}V$, where S is the average of the solid points for each of the cells that contain the vertex, F is the average of the face points (average of the vertices that surround the face) for the faces that contain the vertex, E is the average of the edge points (midpoints of the edges) for the edges incident to vertex V, and V is the vertex itself.

- The lattice $\mathbf{P_{i+1}}$ is created by connecting the neighboring created points. Solid points are connected by edges with neighboring face points, face points are connected with nearest edge points and edge points are connected with the nearest vertex points.



(a) Initial parallerpiped lattice.

(b) Lattice after one step of subdivision process.

(c) Lattice after two steps of subdivision process.

(d) Lattice after three of subdivision process.

Figure 3.19: Catmull-Clark subdivision volume.

In this section we presented basic subdivision volumes with regular and arbitrary topology. In the other works [MQ01], [CMQ03], the reader can find other subdivision algorithms for several topological lattices.

3.4 Deformations

Intuitive methods for three-dimensional object design, modification, and animation are very important in computer graphics. If we want to deform an object, we can use several techniques. The first approach is based on changes of parameters that define the object in some representation. But model designers had to consider mathematical model when making the desired modifications, and the shape design could be difficult – making simple changes to the surface requires the modification of many surface parameters. The process gets more difficult when local changes, such as adding arbitrarily shaped bumps, are necessary. On the other hand, free-form deformations embed an object in a deformable region of space such that each point of the object has a unique parameterization that defines its position in the region. Then the region is altered, causing recalculation of the positions of points based upon their initial parameterization. To have an efficient method of deformation, the deformable space should be defined with great flexibility and with few control points relative to the number of points of the surface model. Sederberg and Parry [SP86] presented an initial deformation lattice on a parallelepiped, and define the deformable space as the trivariate Bézier tensor-product volume. This method is widely used because of its ease of use and the power to create many types of deformations with little user-interaction. Griessmair and Purgathofer [GP89] extended this technique by utilizing a trivariate B-Spline representation over regular a parallelepiped lattice. It was later extended using rational B-spline volumes [LJ94]. For a more general lattice structure, Coquillart introduced Extended Free-Form Deformations (EFFD) [Coq90]. This method uses the initial lattice points to define an arbitrary trivariate Bézier volume, and allows the combining of many lattices to form arbitrary shaped spaces. MacCracken and Joy [MJ96] extended the deformations for lattices of arbitrary topology. For the representation of the deformable space, they used Catmull-Clark subdivision volumes (see the previous section). In other works [CKY01], parametric trivariate volumes are initially treated as a deformation of 3-dimensional space.

We now give an overview how the free-form deformations work. Let us have an object Q (for example triangular mesh) in E^3 given by points $A_1, A_2, .., A_m$ embedded in axis-aligned bounding box $B = [X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min}, Z_{max}]$. Initially we have a regular uniform parallelepiped lattice given by control points $V_{i,j,k} = [X_{min} + (X_{max} - X_{min})\frac{i}{n_u}, Y_{min} + (Y_{max} - Y_{min})\frac{j}{n_v}, Z_{min} + (Z_{max} - Z_{min})\frac{k}{n_w}]$; $i = 0, 1, .., n_u$, $j = 0, 1, .., n_v$, $k = 0, 1, .., n_w$. Let the deformation $T : E^3 \rightarrow E^3$ be based on Bézier tensor-product volume. If we transform a point $P = [x, y, z]$ from B using this initial lattice, we get

$$u = \frac{x - X_{min}}{X_{max} - X_{min}} \quad v = \frac{y - Y_{min}}{Y_{max} - Y_{min}} \quad w = \frac{z - Z_{min}}{Z_{max} - Z_{min}}$$

$$T(x, y, z) = [T_x(x, y, z), T_y(x, y, z), T_z(x, y, z)] =$$
$$= \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{i,j,k} B_i^{n_u}(u) B_j^{n_v}(v) B_k^{n_w}(w).$$

Now when computing the transformation for each coordinate separately, we get

$$T_x(x,y,z)=\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w}(X_{min}+(X_{max}-X_{min})\frac{i}{n_u})B_i^{n_u}(u)B_j^{n_v}(v)B_k^{n_w}(w)=$$

$$=X_{min}\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w}B_i^{n_u}(u)B_j^{n_v}(v)B_k^{n_w}(w)+$$

$$+(X_{max}-X_{min})\sum_{j=0}^{n_v}\sum_{k=0}^{n_w}B_j^{n_v}(v)B_k^{n_w}(w)\sum_{i=0}^{n_u}\frac{i}{n_u}B_i^{n_u}(u).$$

Now using the properties of Bernstein polynomials (Theorem 2.2.2)

$$T_x(x,y,z)=X_{min}+(X_{max}-X_{min})u=X_{min}+(X_{max}-X_{min})\frac{x-X_{min}}{X_{max}-X_{min}}=x$$

Similarly $T_y(x,y,z)=y$ and $T_z(x,y,z)=z$. The initial lattice does not deform anything, it leaves the object Q intact. To deform the object, the modeler moves the control points $V_{i,j,k}$ of the initial lattice. The movement of control point deforms the initial lattice and the deformation is transformed to the points of the object by function T. If object Q consists of points $A_1, A_2, .., A_m$ and the original point A_l has coordinates $[x_l, y_l, z_l]$, then the deformed point $A_l^{'}$ has coordinates

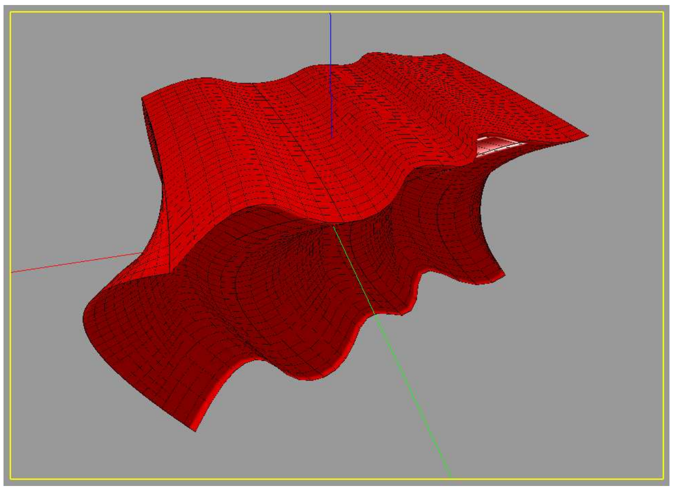
$$A_l^{'}=T(x_l,y_l,z_l)=$$

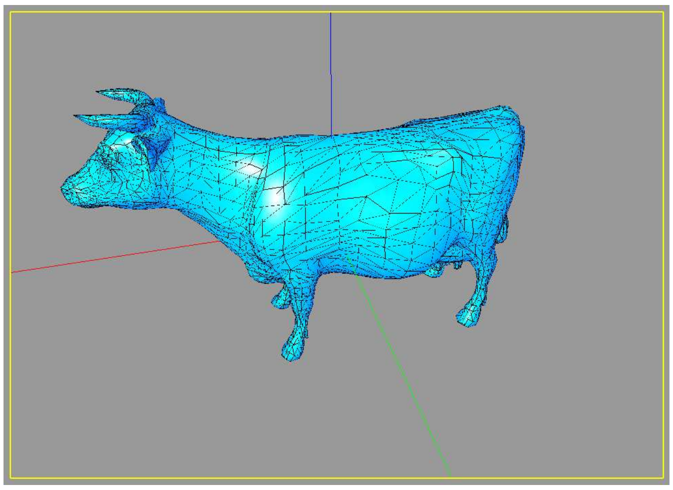
$$=\sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w}V_{(i,j,k)}B_i^{n_u}(\frac{x_l-X_{min}}{X_{max}-X_{min}})B_j^{n_v}(\frac{y_l-Y_{min}}{Y_{max}-Y_{min}})B_k^{n_w}(\frac{z_l-Z_{min}}{Z_{max}-Z_{min}}).$$

Instead of Bézier tensor-product volumes, as the deformation function, B-spline tensor-product volume (Figure 3.20) or S-volume can be used. It is determined by preferred shape of lattice that defines the deformation.

3.5 Continuity

If we want to compose an object from several parametric volumes, it is necessary to glue them together with some order of continuity. The basic order is C^0 continuity, which means that at the end of one volume the another one starts. But in many situations we need higher level of continuity. We will show some basic configurations of control points for two volumes so at the connection of the volumes we get desired level.


(a) B-spline tensor-product volume used as deformation function.


(b) Triangular mesh. Model of cow from [Bur10].

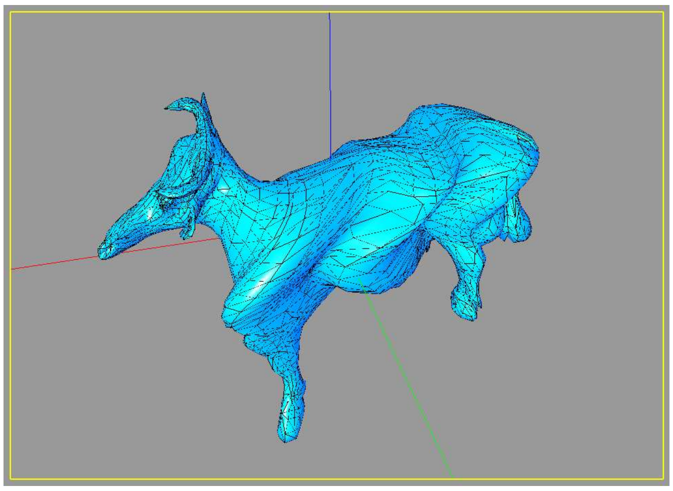

(c) Deformed mesh.

Figure 3.20: Deformation of a triangular mesh using B-spline volume.

In the case of tensor-product volumes, the continuity problem is reduced to the curve case. For example let us have two Bézier tensor-product volumes

$$BTPV_1^{n_u,n_v,n_w} = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} V_{i,j,k}B_i^{n_u}(u)B_j^{n_v}(v)B_k^{n_w}(w)$$

and

$$BTPV_2^{n_u,n_v,n_w} = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} W_{i,j,k}B_i^{n_u}(u)B_j^{n_v}(v)B_k^{n_w}(w)$$

with equal degrees. We want to connect these two volumes at the border $u = 1$ for $BTPV_1^{n_u,n_v,n_w}$ and at the border $u = 0$ for $BTPV_2^{n_u,n_v,n_w}$ with C^r continuity. Because of independence of directions in tensor-product volume, Bézier curve given by points $V_{i,j,k}$; $i = 0, 1, .., n_u$ and Bézier curve given by points $W_{i,j,k}$; $i = 0, 1, .., n_u$ must meet with C^r continuity for each $j = 0, 1, .., n_v$ and $k = 0, 1, .., n_w$. We state the conditions for continuity of Bézier curves together with continuity conditions for Bézier tetrahedra, because of the similar expression of both Bézier objects in the form $BO^n(P) = \sum_{|\mathbf{i}|=\mathbf{n}} V_{\mathbf{i}}B_{\mathbf{i}}^n(U(P))$.

To connect two Bézier tetrahedra on the common boundary with C^r continuity, both object have to have the directional derivatives equal for all directions and for each order up to r. Let us have Bézier tetrahedron $BTH_1^n(P)$ with control points $V_{\mathbf{i}}$ and domain tetrahedron $ABCD$ and Bézier tetrahedron $BTH_2^n(P)$ with control points $W_{\mathbf{i}}$ and domain tetrahedron $EBCD$. To achieve C^r continuity, the cross boundary directional derivatives in direction v on the common border must be equal. Then

$$D_{\mathbf{v}}^j BTH_1^n(P) = D_{\mathbf{v}}^j BTH_2^n(P)$$

for each P from the common boundary of domains, i.e. for each point P from the triangle BCD and for each $j = 0, 1, .., r$. If U_{ABCD} are barycentric coordinates with respect to tetrahedron $ABCD$ and U_{EBCD} are barycentric coordinates with respect to tetrahedron $EBCD$, then using Theorem 2.3.7, we have

$$D_{\mathbf{v}}^j BTH_1^n(P) = \sum_{|\mathbf{i}|=n-j} V_{\mathbf{i}}^j(U_{ABCD}(\mathbf{v}))B_{\mathbf{i}}^{n-j}(U_{ABCD}(P)) =$$

$$= \sum_{|\mathbf{i}|=n-j} W_{\mathbf{i}}^j(U_{EBCD}(\mathbf{v}))B_{\mathbf{i}}^{n-j}(U_{EBCD}(P)) = D_{\mathbf{v}}^j BTH_2^n(P).$$

When P is on the common boundary of both tetrahedra, both barycentric coordinate functions are reduced to the triangle case for that boundary, $U_{ABCD}(P) = U_{EBCD}(P)$

and the first barycentric coordinate is equal to zero. Then

$$\sum_{|\mathbf{i}|=n-j} \Big[V^j_{\mathbf{i}}(U_{ABCD}(\mathbf{v})) - W^j_{\mathbf{i}}(U_{EBCD}(\mathbf{v}))\Big] B^{n-j}_{\mathbf{i}}(U_{ABCD}(P)) = 0$$

for each P from triangle BCD and multi-index $\mathbf{i}$ with the first coordinate equal to zero. Because $B^{n-j}_{\mathbf{i}}$ is a polynomial function, all coefficients must be equal to zero, i.e. $V^j_{\mathbf{i}}(U_{ABCD}(\mathbf{v})) = W^j_{\mathbf{i}}(U_{EBCD}(\mathbf{v}))$. Now let $U_{ABCD}(\mathbf{v}) = (d_1, d_2, d_3, d_4)$, $U_{EBCD}(\mathbf{v}) = (e_1, e_2, e_3, e_4)$ and let $U_{ABCD}(E) = (w_1, w_2, w_3, w_4)$. Then

$$E = w_1A + w_2B + w_3C + w_4D$$

$$\mathbf{v} = e_1E + e_2B + e_3C + e_4D = e_1(w_1A + w_2B + w_3C + w_4D) + e_2B + e_3C + e_4D =$$

$$= (e_1w_1)A + (e_1w_2 + e_2)B + (e_1w_3 + e_3)C + (e_1w_4 + e_4)D.$$

But because $\mathbf{v} = d_1A + d_2B + d_3C + d_4D$, we can write

$$V^j_{\mathbf{i}}(e_1w_1, e_1w_2 + e_2, e_1w_3 + e_3, e_1w_4 + e_4) = W^j_{\mathbf{i}}(e_1, e_2, e_3, e_4)$$

and these formulas are fulfilled for all $e_1, e_2, e_3, e_4 \in (R)$; $e_1 + e_2 + e_3 + e_4 = 0$; $e_1 \neq 0$. Because $V^j_{\mathbf{i}}$ and $W^j_{\mathbf{i}}$ can be written as combination of polynomial functions (see Theorem 2.3.6), the equation must be fulfilled for each $e_1, e_2, e_3, e_4 \in \mathbb{R}$. If we take $e_1 = 1$ and $e_2 = e_3 = e_4 = 0$, we get $V^j_{\mathbf{i}}(w_1, w_2, w_3, w_4) = W^j_{\mathbf{i}}(1, 0, 0, 0)$. Then using Theorem 2.3.6 we get the conditions for C^r continuity of two Bézier tetrahedra on common boundary:

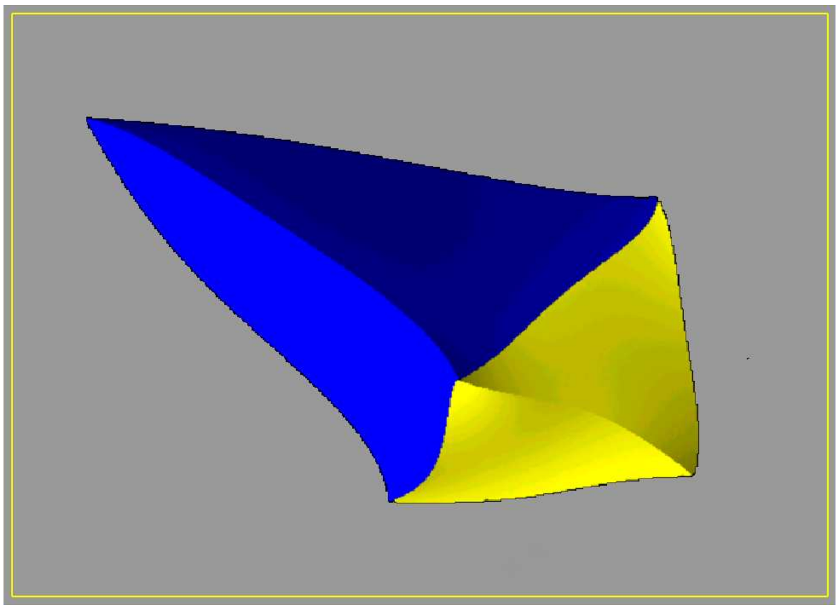
$$W_{j\mathbf{e}_1+\mathbf{i}} = V^j_{\mathbf{i}}(U_{ABCD}(E)) = \sum_{|\mathbf{k}|=j} V_{\mathbf{i}+\mathbf{k}} B^j_{\mathbf{k}}(U_{ABCD}(E)) \quad j = 0, 1, .., r; |\mathbf{i}| = n - j; i_1 = 0$$

Using the above equations, we can guarantee a certain order of continuity when putting together two Bézier tetrahedra. Such connection is illustrated in Figure 3.21(a). It can be also used for composition of Bézier curves and patches, i.e. also for composition of Bézier tensor-product volumes.

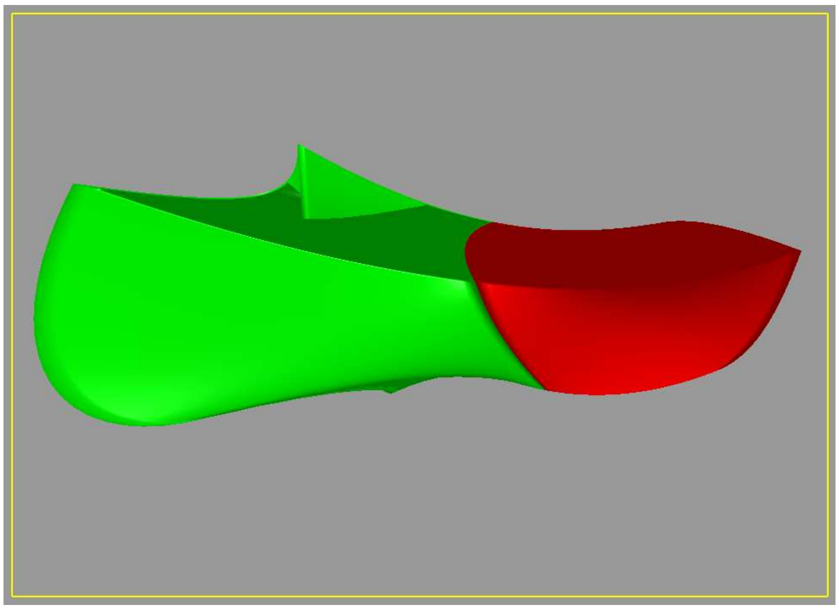
We stated earlier that for C^r continuity of two Bézier tensor-product volumes on common boundary in u direction, we need to have C^r continuity of two Bézier curves defined by points $V_{i,j,k}$; $i = 0, 1, .., n_u$ and $W_{i,j,k}$; $i = 0, 1, .., n_u$. All we need is to get the barycentric coordinates of end point of the second curve's domain related to the domain of the first curve. If the length of second curve domain is 1, we can choose these coordinates as $(2, -1)$. Then the continuity conditions are

$$W_{i,j,k} = V^i_{n_u-i,j,k}(2, -1) = \sum_{l=0}^{i} V_{n_u-i+l,j,k} B^i_l(2) \quad i = 0, 1, .., r$$

Figure 3.21(b) illustrates the composition of two Bézier tensor-product volumes connected with C^2 continuity.



(a) Two Bézier tetrahedra connected with C^1 continuity.



(b) Two Bézier tensor-product volumes connected with C^2 continuity.

Figure 3.21: Continuity of two parametric volumes.

Chapter 4

Visualization of parametric volumes

In previous chapters we presented mathematical properties of volumes and introduced modeling algorithms to work with them. For completeness, we now present several ways for visualization of parametric volumes. We use visualization algorithms similar to the work of Lasser [Las90]. Because volume representation describes also the interior of the object, we want to visualize parts of the interior. For visualization, we use the capabilities of modern graphics hardware, that can visualize only few geometric primitives like points, lines or triangles. We have to approximate the volume by a set of these graphics primitives. To achieve this, we sample the domain of the volume using points that form some net of domain points. Based on the presented modeling and visualization algorithms, we prepared a system, in which we implemented modeling and visualization of several types of parametric volumes. We presented the described algorithms for visualization of volumes in [Sam09b].

4.1 Control net

The basic input parameters of parametric volumes that can be visualized are control points. Because the set of control points forms an organized sequence, we can show this structure using connections between neighboring control points. This connections will form a control net of control points. In our work, we have two types of control nets. Their shape is determined by the type of the volume and used indices. We work with two basic shapes of control net:

- For tensor-product volumes, we have control points $V_{i,j,k}$; $i = 0, 1, .., n_u$; $j = 0, 1, .., n_v$; $k = 0, 1, .., n_w$. Such control net can be displayed as regular box structure. We connect two control points $V_{i,j,k}$ and $V_{a,b,c}$ in the control net only if $(|i - a| = 1) \wedge (j = b) \wedge (k = c)$ or $(i = a) \wedge (|j - b| = 1) \wedge (k = c)$ or $(i = a) \wedge (j = b) \wedge (|k - c| = 1)$. The example of regular box control net structure is in Figure 4.1(a).

- For S-volume, we have points $V_{\mathbf{i}}$; $dim(\mathbf{i}) = k$; $|\mathbf{i}| = n$. To express the connectivity inside this structure, we again connect some neighboring points. If we have two control points $V_{\mathbf{i}}$ and $V_{\mathbf{j}}$, we will connect them if $\mathbf{i} + \mathbf{e}_l = \mathbf{j} - \mathbf{e}_m$ for some $0 \leq l, m \leq k$, $l \neq m$. This way we get a polyhedral structure of control points. One example of control net of Bézier tetrahedron ($k = 4$) is in Figure 4.1(b).

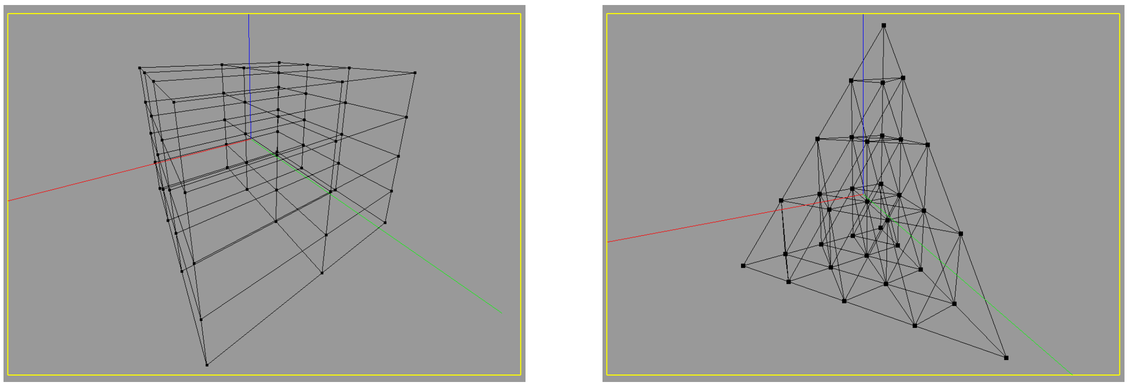
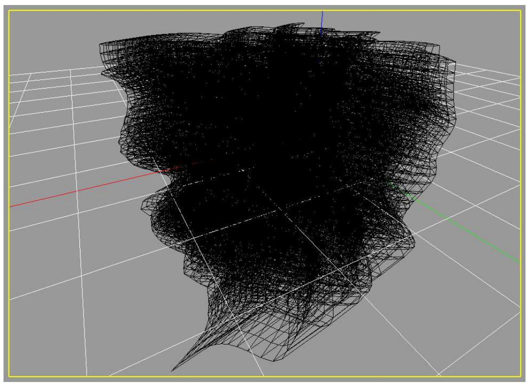

(a) Box structure. (b) Tetrahedral structure.

Figure 4.1: Regular control nets.

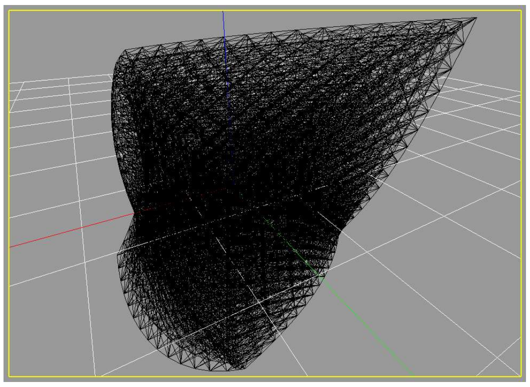
The visualization of control net can be used also for visualization of parametric volumes. Suppose we have Bézier tensor-product volume $BTPV^{n_u,n_v,n_w}$ with control points $V_{i,j,k}$; $i = 0, 1, .., n_u$; $j = 0, 1, .., n_v$; $k = 0, 1, .., n_w$ and domain $\langle 0, 1\rangle \times \langle 0, 1\rangle \times \langle 0, 1\rangle$. For given integer number m, we can sample the direction domain $\langle 0, 1\rangle$ of the volume using values $\frac{l}{m}$; $l = 0, 1, ..m$. By sampling the domain and computing the point of the volume for the sampled values, we get

$$W_{p,q,r} = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{(i,j,k)} B_i^{n_u}(\frac{p}{m}) B_j^{n_v}(\frac{q}{m}) B_k^{n_w}(\frac{r}{m})$$

$$p = 0, 1, .., m \qquad q = 0, 1, .., m \qquad r = 0, 1, .., m.$$



(a) B-spline tensor-product volume.



(b) S-volume.

Figure 4.2: Visualization of parametric volumes using isoparametric curves.

Points $W_{p,q,r}$ form a box structure and can be used as the approximation of the parametric volume. Then the visualization of box structure can be used as visualization of $BTPV^{n_u,n_v,n_w}$. More precisely, it displays the approximations of the isoparametric curves. The isoparametric curve of a volume is defined by setting two parameters of volume to constants. For example if we put $v = w = \frac{1}{m}$, from Bézier tensor-product volume we get the isoparametric curve $IC(u) = \sum_{i=0}^{n_u} \left[\sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{i,j,k} B_j^{n_v}(\frac{1}{m}) B_k^{n_w}(\frac{1}{m}) \right] B_i^{n_u}(u)$. The process of visualization for B-spline tensor-product volumes by isoparametric curves is the same, we just replace the computations of $W_{p,q,r}$ by the expression with B-spline functions (Figure 4.2(a)). The parameter m controls the precision of approximation, higher m will result in better approximation and in a more dense set of points connected with lines.

If we have S-volume SV^n, we can also do the visualization using isoparametric curves and control nets. If the domain D of S-volume is a polyhedron given by points $A_1, A_2, .., A_k$, we can sample D using the integer parameter m and points

$$P_{\mathbf{j}} = \sum_{i=1}^{k} \frac{j_i}{m} A_i \quad \mathbf{j} = (j_1, j_2, .., j_k).$$

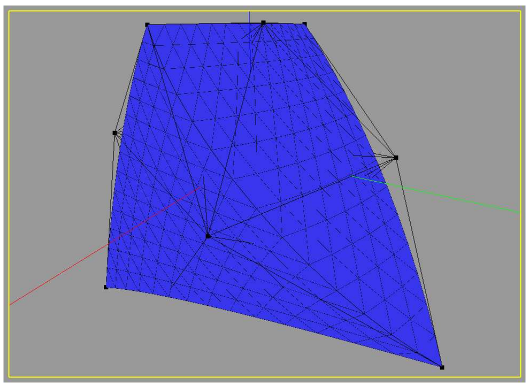
We get a new polyhedral structure of points $P_{\mathbf{j}}$ as samples of domain. The computation of S-volume points for these domain values gives

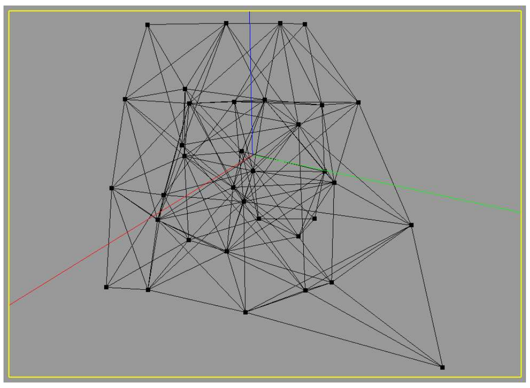
$$W_{\mathbf{j}} = SV^n(P_{\mathbf{j}}) = \sum_{|\mathbf{i}|=n} V_{\mathbf{i}} B_{\mathbf{i}}^n(G(P_{\mathbf{j}})).$$

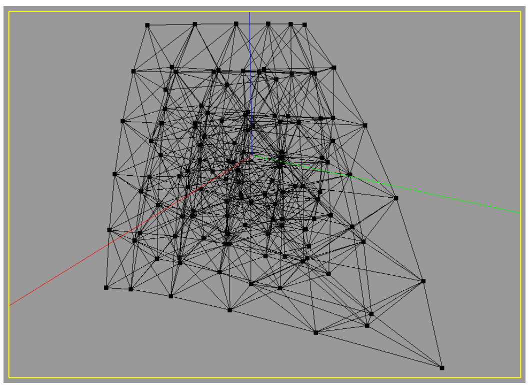
Displaying the structure of $W_{\mathbf{j}}$ leads to visualization of S-volume by using the approximation of isoparametric curves sometimes called also as wireframe visualization

(Figure 4.2(b)). Here again the parameter m controls the precision of the approximation.

Using these nets of control points, we can visualize the steps of subdivision algorithms, where the denser net of control points is generated without changing the original volume. In previous sections we presented algorithms based on knot insertion for B-spline tensor-product volumes (Figure 3.18, Figure 3.19), based on de Casteljau algorithm (Figure 3.11) or based on elevation of degree or degrees. We presented degree elevation algorithms for tensor-product volumes and for S-volumes (Theorem 2.3.5). After degree elevation, we get a denser net, and multiple repetition of the algorithm step converges to the parametric volume (see Figure 4.3 and Figure 4.4).


(a) Initial S-volume with net of control points.


(b) Control net after 1 step of degree elevation algorithm.


(c) Control net after 3 steps of degree elevation algorithm.

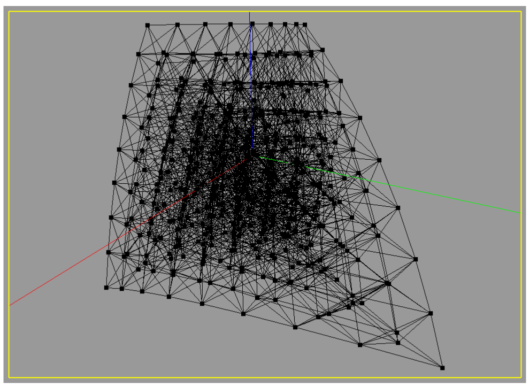
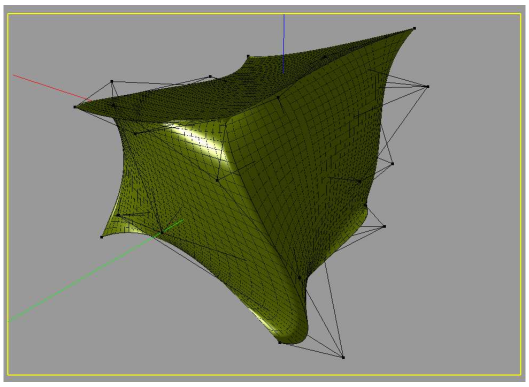
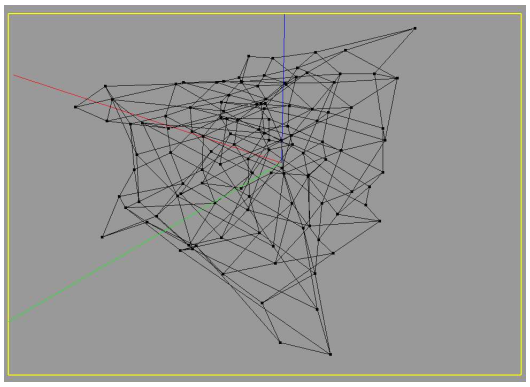

(d) Control net after 6 steps of degree elevation algorithm.

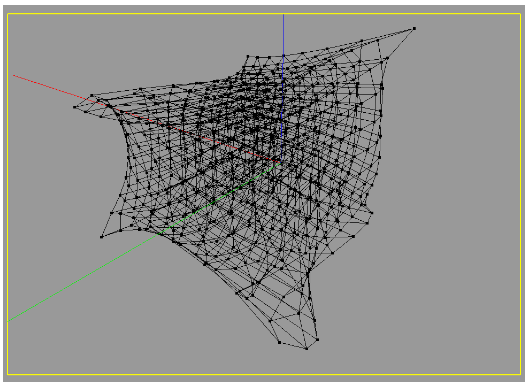
Figure 4.3: Degree elevation for S-volume.



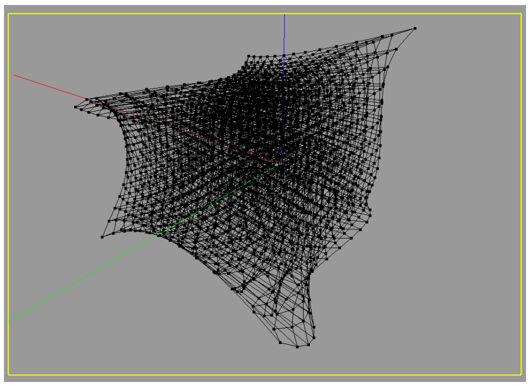
(a) Initial Bézier tensor-product volume with net of control points.



(b) Control net after 1 step of degree elevation algorithm.



(c) Control net after 5 steps of degree elevation algorithm.



(d) Control net after 10 steps of degree elevation algorithm.

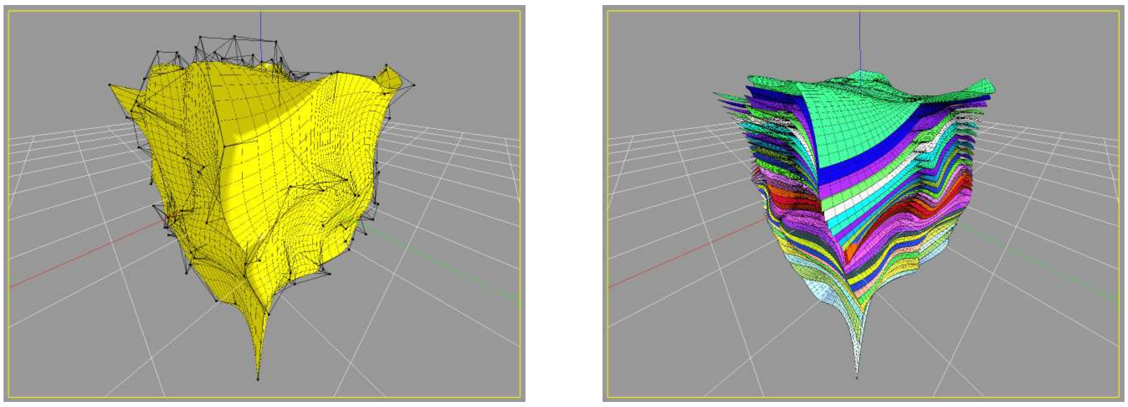
Figure 4.4: Degreee elevation for Bézier tensor-product volume.

4.2 Isoparametric surfaces

Just like in the case of isoparametric curves, isoparametric surfaces are generated from volume's parametric expression by setting one of the parameters to a constant. Because this constant parameter is from a continuous interval, we get an infinite number of surfaces. We need to choose some finite number of representative surfaces, for example by simple uniform sampling of the definition interval. To sample the u direction, we choose uniform sampling $u = \frac{l}{m}$; $l = 0, 1, .., m$. For Bézier tensor-product volume we choose and visualize the following surfaces for u direction (for v and w direction, the surfaces can be determined similarly):

$$IS_l(v, w) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} V_{i,j,k} B_i^{n_u}(\frac{l}{m}) B_j^{n_v}(v) B_k^{n_w}(w)$$

$$l = 0, 1, .., m \qquad v, w \in \langle 0, 1 \rangle$$



(a) B-spline tensor-product volume. (b) Isopatches of volume.

Figure 4.5: Visualization of parametric volume using isoparametric surfaces.

If we want to visualize also the interior of the volume, we will use the isoparametric surfaces only in one direction. In other situations, we render the isoparametric surfaces in all directions. We use isosurfaces in all directions for almost all visualizations of volumes, sometimes extending the visual quality with usage of isoparametric curves. For rendering of surfaces, we used the approximation by set of quadrilaterals. The process of the visualization of B-spline tensor-product volume using isoparametric surfaces only in one direction is illustrated in Figure 4.5.

4.3 Boundary evaluation

Not always is the visualization of the volume interior necessary. In many cases, the boundary of volume is sufficient for visualization. For almost all configurations of control points, the boundary is given by the boundary patches that are computed for the boundary values of the domain. But if control points from the interior of the control net are moved, the mass of the volume from interior can cross the border patches. Joy et al. [JD99] and [MC01] worked with this scenario and proposed the determination of the boundary points. They determine the extra boundary points by putting the Jacobi determinant of trivariate volume equal to zero. The boundary of trivariate object $TO : \mathbb{R}^3 \to E^3$; $TO(u, v, w) = (TO_u(u, v, w), TO_v(u, v, w), TO_w(u, v, w))$ consists of boundary isoparametric patches (for extremal parameters) together with

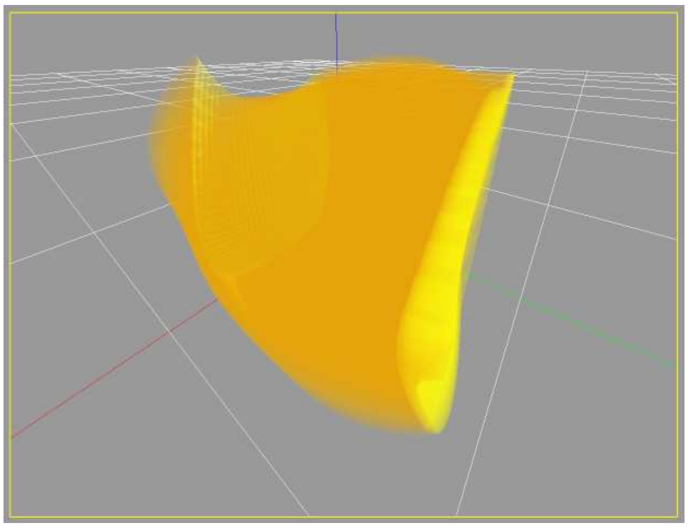


Figure 4.6: B-spline tensor-product volume rendered as transparent object.

the points of the volume where the Jacobi determinant is zero:

$$J(u,v,w) = \begin{vmatrix} \frac{\partial TO_u}{\partial u}(u,v,w) & \frac{\partial TO_u}{\partial v}(u,v,w) & \frac{\partial TO_u}{\partial w}(u,v,w) \\ \frac{\partial TO_v}{\partial u}(u,v,w) & \frac{\partial TO_v}{\partial v}(u,v,w) & \frac{\partial TO_v}{\partial w}(u,v,w) \\ \frac{\partial TO_w}{\partial u}(u,v,w) & \frac{\partial TO_w}{\partial v}(u,v,w) & \frac{\partial TO_w}{\partial w}(u,v,w) \end{vmatrix} = 0$$

The Jacobi determinant defines the implicit surface of boundary points. We had to approximate such implicit surface by a set of triangles. When we combine these this implicit surface with the boundary patches, we can render the whole boundary of a trivariate object.

4.4 Transparency

The common configuration for visualization is to render isoparametric surfaces for all defining parameters. In that case, only the approximated boundary is visible. To see the interior of object, we can use transparent isoparametric surfaces. The rendering of transparency is done using blending functionality of graphics hardware. Figure 4.6 shows B-spline volume rendered using enabled transparency. This way we see also the boundaries and the interior structures of the volume that were not visible without transparency.

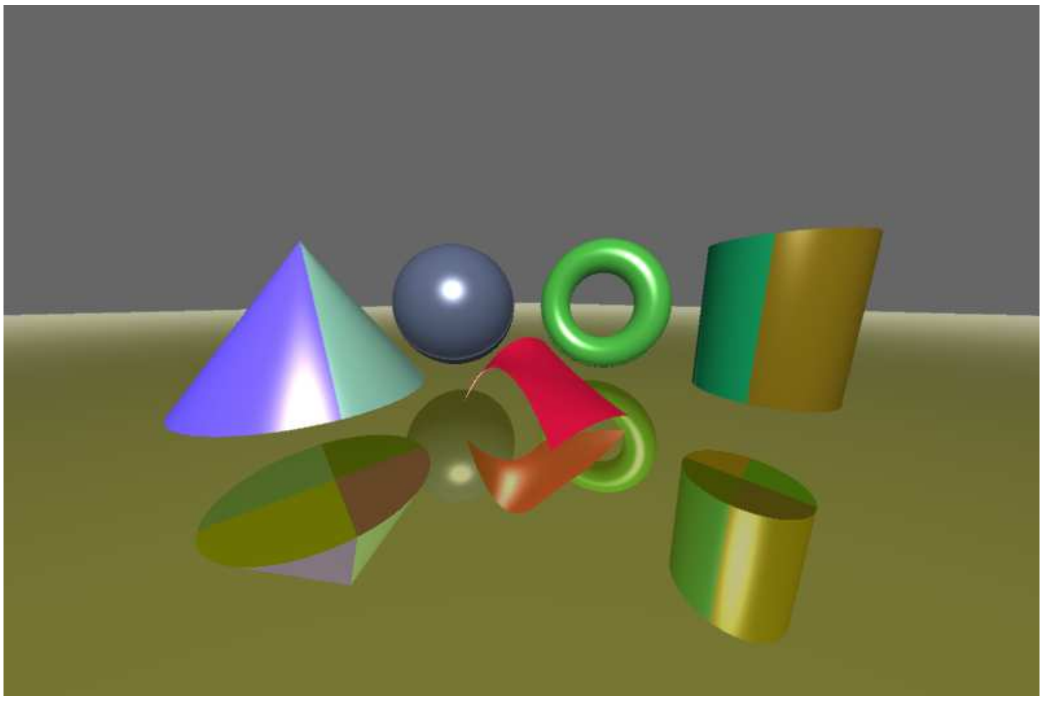
4.5 Raytracing

Other option how to visualize objects is to use the technique called raytracing. It is a well-known technique of visualization, where for each picture element, one ray is created that represents the pixel and intersections of this ray and objects are searched. The elementary algorithm in this technique is to find the intersection of the parametric volume and the ray. For low degree surfaces, the intersection can be found by using direct computation. In the case of parametric surfaces, the intersection between ray and surface of arbitrary degree can be found by using numerical methods ([Tot85]), methods that use properties of Bézier patches [NSK90] or by decomposing the surface into small Bézier patches and then searching for the intersection between the ray and the control net. This method (shown in Figure 4.7(a)) creates reflections and shadows more precisely. For parametric volumes, we just converted the volume into set of boundary patches and then use algorithms for raytracing of patches (see Figure 4.7(b)). For proper determination of boundary, also zero set of Jacobi determinant should be included, as in [MC01]. In the future, we are planning to the extend numerical root finding algorithms and Bézier clipping algorithm for parametric volumes.

4.6 GeomForge

To show that proposed representation of objects in the form of parametric volumes can be used in modeling system and actually useful, we designed and implemented our own complex system for modeling and visualization of curves, surfaces and volumes. At the beginning, we defined the goals for the system. We realized that we need to implement almost all proposed constructions and algorithms. The system should handle following:

- Basic free-form curves, surfaces and volumes based on Bernstein and B-spline functions
- Polygons for definition of control points of curves
- Two-dimensional lattices for definition of control points for tensor-product patches, Bézier triangles and S-patches.


(a) Raytracing of NURBS surfaces.

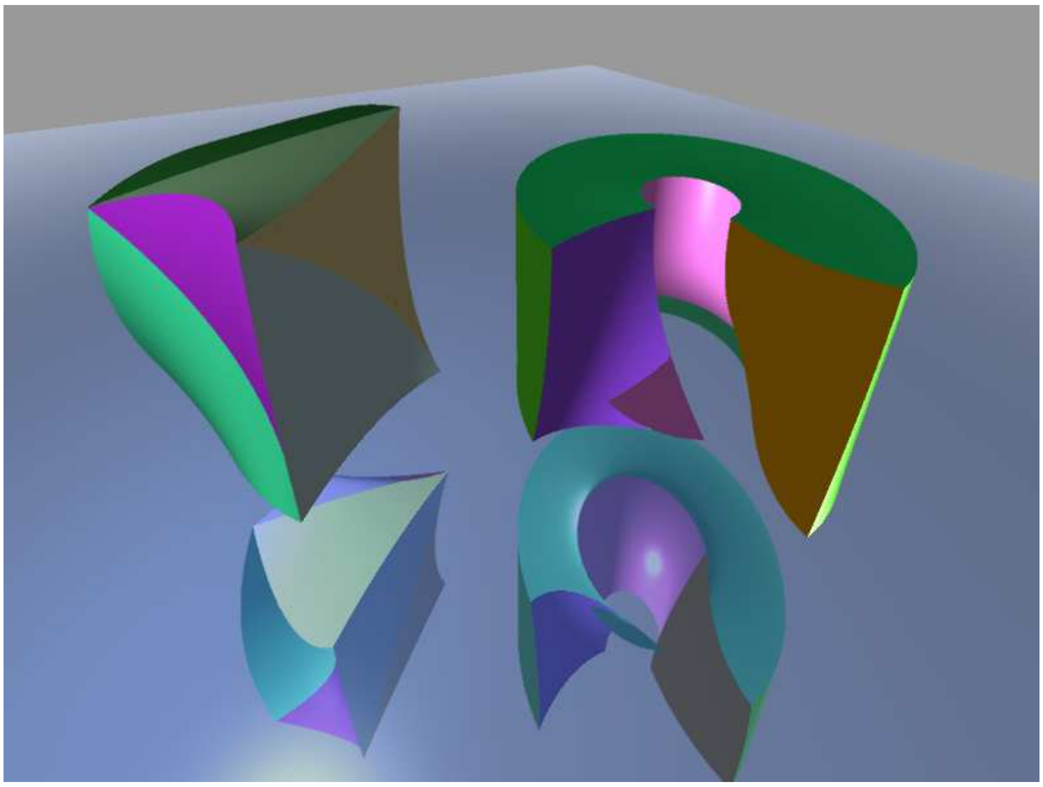
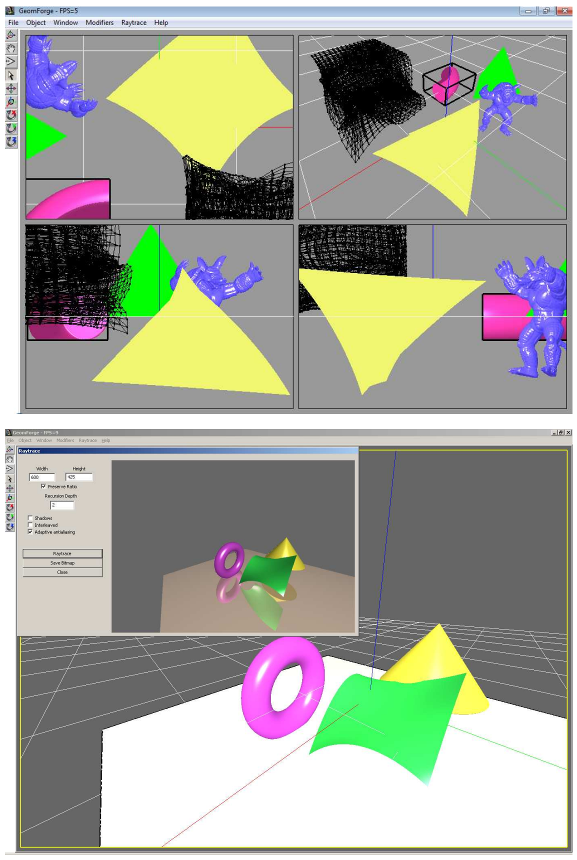

(b) Raytracing of boundary patches of S-volume and B-spline tensor-product volume.

Figure 4.7: Raytracing of parametric objects.

- Triangular meshes for boundary representation of objects and its deformations
- Three-dimensional lattices for definition of control nets of volume
- Basic transformations of objects (scaling, rotating, moving)
- Efficient real-time visualization of objects using cache mechanism
- Visualization of input parameters (control points, weights)
- Visualization of objects by their approximation using lines, filled triangles or filled quadrilaterals
- Visualization using raytracing
- Construction of basic objects (spheres, cylinders,...) in some representations
- Creation of higher order objects from lower order objects (for example volumes from surfaces)
- Decomposition of objects into set of smaller objects
- Picking system to choose one object in scene
- Modification of object's material used in visualization
- Tools for navigation in the scene

As the platform for implementation, we choose Win32 for its good accessibility. Because we want to render objects in three-dimensional space using modern graphics hardware, we had to choose graphics API (Application programming interface). We had two options for graphics API, OpenGL [Gro10] and DirectX [Cor10]. We choose OpenGL for its easy to use specification, multi-platform and multi-language availability and wide acceptance within the academic community. We name the implemented system GeomForge to express its usage with creation and modification of geometric objects. The main part of GeomForge are the modification tools that are working with several geometrical objects. We used several optimalized rendering algorithms and data structures described in [PT95]. The rendering stage of system we used several visualization algorithms and techniques to achieve real-time rendering times. We adopted several of them from [AMH02]. We fulfilled all given goals and implemented algorithms for creation and modification of parametric volumes. The implemented objects and the modeling tools for geometrical objects are:

- **Polygon**: initialization as regular polygon, performation of Chaikin subdivision step, performation of Catmull-Clark subdivision step, conversion to Bézier curve, conversion to B-spline curve
- **B-spline curve**: initialization as circular arc, initialization as random Bézier curve, decomposition to set of Bézier curves, elevation of degree, decomposition of Bézier curve to two Bézier curves using de Casteljau algorithm,
- **Triangular mesh**: initialization as cube, loading from .ply file, performation of Loop subdivision, performation of Catmull-Clark subdivision, deformation using B-spline volume, deformation by S-volume
- **Bézier triangle**: initialization using random control points, initialization as regular subdivision
- **S-patch**: initialization of S-patch based on arbitrary polygon and degree, decomposition of patch to set of Bézier triangles, addition of C^r continuous S-patch, creation of sub-patch
- **B-spline tensor-product surface**: initialization as boundary of basic objects (sphere, torus, cylinder, cone), initialization as Bézier patch, decomposition to piecewise Bézier form, decomposition of Bézier patch to two using de Casteljau algorithm
- **Parallerpiped 3D lattice**: initialization as uniform or random net, performation of Doo-Sabin subdivision, performation of Catmull-Clark subdivision, conversion to B-spline volume
- **Bézier tetrahedron**: initialization as cone, initialization as spherical tetrahedron,
- **S-volume**: initialization of S-volume based on arbitrary triangular domain polyhedron and degree, elevation of degree, decomposition to Bézier tetrahedron, creation of ruled volume from two Bézier triangles, creation of boundary S-patches


GeomForge - FPS=5
File Object Window Modifiers Raytrace Help
GeomForge - FPS=9
File Object Window Modifiers Raytrace Help
Raytrace
Raytrace
Save Bitmap
Close

Width 600
Height 425
[✓] Preserve Ratio
Recursion Depth 2
[ ] Shadows
[ ] Interleaved
[✓] Adaptive antialiasing

Figure 4.8: Screenshots of GeomForge system. Armadillo model from [Lab10].

- **B-spline tensor-product volume**: creation of volume of revolution from B-spline curve, creation of volume of revolution from B-spline surface, decomposition to set of Bézier volumes, creation of boundary B-spline patches, creation of ruled volume from two B-spline patches, creation of sweep volume from B-spline curve and B-spline patch, creation of B-spline isopatches

We work with rational versions of objects when possible. Each object can be transformed using rotation, scaling and translation. The material properties can be changed for each object. To be able to work with selected objects, we implemented a picking functionality. We used the computation of intersection between a ray and object for picking. This functionality was used also for raytracing, we implemented intersection algorithms for all curves and surfaces in system. We are planning to release binaries and source code of GeomForge application to public via webpage [Sam10].

Conclusion

In this work, we have presented complex theory and applications of parametric volumes described as combination of given control points. We have introduces a new general expression of parametric volumes based on Bézier form using generalization of Bernstein polynomials. We call this general Bézier form S-volumes. For definition of S-volumes, we used recent additions in the theory of generalized barycentric coordinates and generalized Bernstein polynomials. Besides S-volumes, we have given an insight into the theory of parametric volumes based on tensor-product paradigm. All volumes use as blending functions Bézier or B-spline functions. We have presented several mathematical properties of blending functions and related mathematical properties of the constructed volumes.

Based on the given properties of volumes, we have prepared modeling techniques for useful work with parametric volumes. We have presented modeling paradigms like sweep volumes and algorithms for construction of several types of basic objects as parametric volumes. We have also shown subdivision and decomposition algorithms. For connection of volumes, we have derived rules for certain order of continuity between the connected parametric volumes. We have proved the usefulness of parametric volumes by implementation of proposed modeling and visualization algorithms in our modeling system. Using this system, called GeomForge, we have illustrated almost all presented volumes, techniques, algorithms and visualizations.

There are many possibilities how to extend this work. In the future, we want to focus more on S-volumes and their modeling techniques. Because a multi-sided generalization of B-spline tensor-product is possible, we want to extend also B-spline volumes into their multi-sided generalization. For raytracing, we want to prepare algorithms for exact computation of intersection between a parametric volume and a ray.

Bibliography

[AMH02] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering.* A.K. Peters, Natick, MA, USA, second edition, 2002.

[Bur10] John Burkardt. Ply files - an ascii polygon format. `http://people.sc.fsu.edu/~burkardt/data/ply/ply.html`, 2010.

[CC78] Edwin Catmull and Jim Clark. Recursively generated b-spline surfaces on arbitrary topological surfaces. *Computer-Aided Design*, 10:350–355, 1978.

[Cha74] George M. Chaikin. An algorithm for high speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.

[CKY01] Min Chen, Arie Kaufman, and Roni Yagel, editors. *Volume Graphics.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[CMQ03] Yu-Sung Chang, Kevin T. McDonnell, and Hong Qin. An interpolatory subdivision for volumetric models over simplicial complexes. In *SMI '03: Proceedings of the Shape Modeling International*, pages 143–152, Washington, DC, USA, 2003. IEEE Computer Society.

[Coq90] Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3D geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA, 1990. ACM.

[Cor10] Microsoft Corporation. DirectX homepage. `http://www.microsoft.com/windows/directx/`, 2010.

[CRE01] Elaine Cohen, Richard F. Riesenfeld, and Gershon Elber. *Geometric Modeling with Splines.* A.K. Peters, Natick, MA, USA, 2001.

[DS75] Daniel Doo and Malcolm Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:304–310, 1975.

[Far93] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide.* Academic Press, New York, NY, USA, third edition, 1993.

[FH00] Gerald E. Farin and Dianne Hansford. *The Essentials of CAGD.* A. K. Peters, Ltd., Natick, MA, USA, 2000.

[FKR05] Michael S. Floater, Géza Kós, and Martin Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22(7):623–631, 2005.

[Flo03] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[GP89] J. Griessmair and Werner Purgathofer. Deformation of solids with trivariate B-splines. In W. Strasser FRA Hopgood, editor, *Proceedings for Eurographics*, pages 137–148, 1989.

[Gro10] Khronos Group. OpenGL homepage. `http://www.opengl.org`, 2010.

[JD99] Kenneth I. Joy and Mark A. Duchaineau. Boundary determination for trivariate solids. In *PG '99: Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*, pages 82–91, Washington, DC, USA, 1999. IEEE Computer Society.

[JM99] Kenneth I. Joy and Ron Maccracken. The refinement rules for Catmull-Clark solids. Technical report, University of California, Davis, 1999.

[JS05] Tao Ju and Scott Schaefer. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph*, 24:561–566, 2005.

[JSWD05] T. Ju, S. Schaefer, J. Warren, and M. Desbrun. A geometric construction of coordinates for convex polyhedra using polar duals. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 181, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[Lab10] Stanford Computer Graphics Laboratory. The stanford 3D scanning repository. `http://www.graphics.stanford.edu/data/3Dscanrep/`, 2010.

[Las85] Dieter Lasser. Bernstein - Bézier representation of volumes. *Computer Aided Geometric Design*, 2(1-3):145–149, 1985.

[Las90] Dieter Lasser. Visualization of free-form volumes. In *IEEE Visualization*, pages 379–387, 1990.

[LBS08] Torsten Langer, Alexander G. Belyaev, and Hans-Peter Seidel. Mean value Bézier maps. In *GMP*, pages 231–243, 2008.

[LD89] Charles T. Loop and Tony D. DeRose. A multisided generalization of Bézier surfaces. *ACM Trans. Graph.*, 8(3):204–234, 1989.

[LJ94] Henry J. Lamousin and Warren N. Waggenspack Jr. NURBS-based free-form deformations. *IEEE Comput. Graph. Appl.*, 14(6):59–65, 1994.

[LS07] Torsten Langer and Hans-Peter Seidel. Mean value Bézier surfaces. In *Mathematics of Surfaces XII*, pages 263–274. Springer, 2007.

[MC01] William Martin and Elaine Cohen. Representation and extraction of volumetric attributes using trivariate splines: a mathematical framework. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 234–240, New York, NY, USA, 2001. ACM.

[MJ96] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23-rd annual conference on Computer graphics and interactive techniques*, pages 181–188, New York, NY, USA, 1996. ACM Press.

[MLBD02] Mark Meyer, Haeyoung Lee, Alan Barr, and Mathieu Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7:13–22, 2002.

[MQ01] Kevin T. McDonnell and Hong Qin. FEM-based subdivision solids for dynamic and haptic interaction. In *SMA '01: Proceedings of the sixth*

ACM symposium on Solid modeling and applications, pages 312–313, New York, NY, USA, 2001. ACM Press.

[NSK90] Tomoyuki Nishita, Thomas W. Sederberg, and Masanori Kakimoto. Ray tracing trimmed rational surface patches. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 337–345, New York, NY, USA, 1990. ACM Press.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[PT95] Les Piegl and Wayne Tiller. *The NURBS book*. Springer-Verlag, London, UK, 1995.

[Req80] Aristides G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv.*, 12(4):437–464, 1980.

[Rie75] Richard F. Riesenfeld. On Chaikin's algorithm. *Computer Graphics and Image Processing*, 4:304–310, 1975.

[Sam04a] Martin Samuelčík. Basic subdivision schemes on triangular meshes. In *Symposium on Computer Geometry*, pages 113–118, Kočovce, Slovakia, 2004.

[Sam04b] Martin Samuelčík. Bézier solids. (poster). In *SCCG 2004*, pages 27–28, Budmerice, Slovakia, 2004.

[Sam05] Martin Samuelčík. Modeling with rational Bézier solids. (poster). In *WSCG 2005*, pages 67–68, Plzeň, Czech Republic, 2005.

[Sam06] Martin Samuelčík. NURBS solids of revolution. (poster). In *SCCG 2006*, pages 75–76, Častá-Papiernička, Slovakia, 2006.

[Sam09a] Martin Samuelčík. Subdivision of generalized S-patches. In *SCCG 2009: Conference Materials and Posters*, pages 27–30, Bratislava, Slovakia, 2009.

[Sam09b] Martin Samuelčík. Visualization of trivariate NURBS volumes. In *Aplimat 2009*, Bratislava, Slovakia, 2009.

[Sam10] Martin Samuelčík. Homepage. `http://www.sccg.sk/~samuelcik/`, 2010.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, New York, NY, USA, 1986. ACM Press.

[Tot85] Daniel L. Toth. On ray tracing parametric surfaces. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 171–179, New York, NY, USA, 1985. ACM Press.

[uC01] Roman Ďurikovič and Silvester Czanner. Interior modelling and object metamorphosis with parametric solids. In *Proceedings of the IASTED International Conference Modelling and Simulation*, pages 324–329, 2001.

[Wac75] E. L. Wachspress. *A Rational Finite Element Basis*. Academic Press, New York, NY, USA, 1975.

[WSHD04] Joe Warren, Scott Schaefer, Anil N. Hirani, and Mathieu Desbrun. Barycentric coordinates for convex sets. Technical report, Advances in Computational and Applied Mathematics, 2004.

[ZS00] Denis Zorin and Peter Schröder. Subdivision for Modeling and Animation, 2000. Course Notes.