Reduction of Smooth Surfaces[1]

PROGRESS REPORT[2]

Jaroslav Semančík[3]
Centre for Computer Graphics and Data Visualization
Department of Computer Science
Faculty of Applied Sciences
University of West Bohemia
Plzeň, Czech Republic

June 28, 2002

[1]This work was supported by the Ministry of Education of the Czech Republic - project MSM 23500005

[2]electronic version on: `http://herakles.zcu.cz/~semancik/progrep.ps`

[3]`jaroslav_semancik@yahoo.com`

Contents

Chapter 1

Introduction

In the very beginning we present some ideas and answers why this research has been initiated, what are the goals and we also specify the particular tasks to be completed.

1.1 Motivation

The first and obvious question should be "Why to use the smooth surfaces?". The motivations to represent geometric models by the smooth surfaces are:

- **accuracy and efficiency** (in the data-size sense).
  Models in the current de facto standard of the boundary representation – triangular mesh commonly present very large data sets reaching up to gigabyte for objects scanned from the real world. Even if the model is not especially complex, a high number of small triangles is needed to preserve its curved shape accurately. It is a kind of inevitable oversampling of the surface; though a single smooth patch could do the job well. Its another advantage is smoothness in any scale level while the triangular mesh has always sharp edges in a close look.

- **advancements in theory of geometric modeling**.
  After the basic rectangular patches have been developed (and hence suitable mainly) for car and aircraft industry in 60- and 70-ties, the recent research resulted in triangular and multisided patches [3], triangular NURBS surfaces [14] or subdivision surfaces and it is closely related to approximation theory.

- **hardware acceleration of the parametric patches rendering can be expected** in the near future.
  It can be implemented as a kind of ray casting or an automatic adaptive tesselation of the patch. The automatic adaptive tesselation on the graphics board helps to prevent the huge number of triangles

streaming into the graphics card through the bus – the bottleneck of current systems.

- **analogy with assembler → higher level programming languages changeover**: triangular mesh → smooth surfaces (with possible sharp edges).
The situation is very similar as in the programming several decades ago. The change from low level assembler programming to higher level languages made possible to develop more complex software on the higher abstract level whose implementation wouldn't be on the lowest level feasible. The higher level of abstraction allows to formulate and solve more sophisticated problems without distraction by 'technical details'. In computer graphics is the role of higher programming languages taken over by mathematically sophisticated geometric elements – the building stones of the models e.g. parametric patches, while triangular meshes remains as an underlying technical layer handled by hardware. In practice – an application should work with smooth surfaces, while the tesselation in the necessary quality and the rendering are provided by a graphics card.

The second question is why to reduce the models represented by smooth surfaces. Complex and detailed models present still too much data to process and they are often inconvenient for interacive applications where the details are unrecongnizable anyway and the simpler overall shape is usually sufficient. Another applications are relatively low-speed network transmissions. There is the area for model reduction (or decimation) level-of-detail (LOD) techniques, progressive model reduction and its reverse refinement.

1.2 Goals

The first and fundamental goal of this work is reduction of a smooth surface model to a simpler one while preserving its overall shape at best. Useful extensions of that goal are: error-driven reduction (i.e. reduction with a given maximal or average error to be satisfied by the resulting model) or a bound to the size of the resulting model.

It would be very appreciated if the results of this work will have also some general charcter – i.e. what is common for the reduction of all kinds of patches or subdivision surfaces. Although the use of smooth surfaces can be expected in the future, it is not clear for now, what kind of them will establish as the standard.

Besides the basic reduction of the model the further goals are progressive reduction and its reverse refinement as an analogy to progressive meshes introduced by Hoppe in [7]. The advantage over the progressive meshes based on triangular meshes is in accuracy and smaller models to both storage and data transmission for local refinement in the rendering process. Further methods from LOD techniques can be adapted, e.g. 'geomorphs' for progressive meshes proposed by Hoppe (see [7]).

Finally, an estimate for error of the reduced model will be helpful for more exact as visual evaluation of the resulting model quality and possibly for driving the reduction level.

1.3 Context of the Problem

The tasks in the context of our problem can be enumerated as follows:

1. **Smooth surface model acquisition/construction**
   This is not the focus of our work, hence we will not devote too much effort to that, but it is a necessary step to obtain a suitable model for the next step. This issue is treated in Chapter 3.

2. **Reduction of the model**
   The core of the work and the primary goal resulting in a reduced model approximating the original one.

3. **Error analysis**
   Formulation of an appropriate error measure for the reduced model, analysis of the error bound and distribution. A decision about the further reduction of the model.

4. **Interactive work with the constructed LOD model**
   An application step. After the reduced model or a hierarchy of models is created, the interactive rendering or transmissions can take place.

Chapter 2

Smooth Surfaces in Computer Graphics

There is a wide variety of smooth surfaces used in computer graphics and geometry modeling. Let us look at the requirements (or viewpoints, criteria for considering a particular representation of a smooth surface) first, and then we examine the alternatives to choose from.

2.1 Requirements

The reasonable requirements in our application are (in the descending importance order):

- **flexibilty** – ability to catch both smooth curved surfaces and sharp edges, open and closed models, model composed of several independent components, possibility to define texture or material properties of the surface, etc.
- **efficiency** in two senses: (1) size of data needed to describe the surface, and (2) computational time for the basic operations related to the surface (e.g. surface point and derivatives evaluation $\rightarrow$ rendering, amount of calculations needed for approximation by that kind of surface, inside/outside test, etc.).
- **connection to technical norms and standards**, e.g. IGES, STEP.
- **difficulty of conversions** from/to the commonly used representations (mainly triangular mesh and NURBS surface).

2.2 Alternatives

We can try to satisfy the above requirements (or at least the most of them) by one of the following alternatives:

- **a single prametric patch**
  The most simple is the Bezier parametric patch, its shape is determined by a grid of control points. There are, however, problems with compatibility of twists (mixed partials) in the patch corners when smooth joints between several neighbouring patches are desired. The Gregory patch was designed to eliminate that drawback and to make the two twists in a corner independent. The most used and the most powerful are NURBS (Non-Uniform Rational B-Spline) surfaces, given by a grid of control points as well. The Coons patches are suitable to interpolate given boundary curves and to fill the patch interior. They are given by arbitrary boundary curves sharing the vertices.

  Regardless of the particular type of the parametric patch they all are limited to a very simple topology, equivalent (homeomorphic) to a rectangle, cylinder or torus in case of a rectangular patch and a triangle or cone in case of triangular patch. That is by far not sufficient for the real models.

- **a net of parametric patches**
  A model with arbitrary topology can be represented in this manner – as a mesh of parametric patches sharing the boundaries. The smoothness across the boundaries between the patches is usually desired and the individual patches must be constructed to satisfy it.

- **a subdivision surface**
  A subdivision surface is a modern type of a smooth surface representation defined as a limit surface of the base (or control) mesh refinement steps. The refinement steps are performed according to a particular scheme (e.g. Doo-Sabin, Catmull-Clark, Loop), resulting in a mesh with higher number of smaller faces. A subdivision surface is a piecewise B-spline surface of arbitrary topology given by the topology of the base mesh. Subdivision surfaces were extended by sharp features by Hoppe in [8].

- **implicit and volume models**
  We will not it consider here, they are listed just for completness.

The reasonable candidates are the net of patches and the subdivision surface. We can consider further the parametric C^k and geometric G^k smoothness of the surface and the capability of sharp edges and vertices. Moreover the structure of the net of patches or the base mesh for subdivision can vary:

- **triangular vs. quadrangular**
  The historically first were rectangular patches and subdivision surfaces

as a consequence of the origin in car industry[1] (the car parts are usually of rectagular topology). They are rapid to evaluate and the underlying curve algorithms can be employed. The convenient properties are, however, only in the two isoparametric directions. The triangular-domain patches are mathematically more natural generalization of the curvesand though more expensive to evaluate, their properties are equally distributed in every direction. They are based on barycentric coordinates.

  Actually, even the mixed or general polygonal meshes exists but they are not interesting for our work.

- **regular vs. irregular**
  In a regular triangular mesh each vertex (except the boundary ones) is common to exactly 6 triangles. In a reqular quadrangular mesh, each vertex (except the boundary ones) is common to exactly 4 quadrangles. Regular meshes are suitable for simulation processes and they are usually result of a computer aided modeling. The algorithms are simpler and more efficient than the ones for their irregular brothers. On the other hand, irregular meshes are more flexible and the models scanned from the real world are generally irregular.

  Moreover, there are even hybrid meshes (see [5]) containing both regular and irregular areas.

To conclude it, we expect an irregular triangular mesh as the input, consequently the irregular net of triangular patches (or a triangular subdivision surface) was chosen.

[1] see [3]

Chapter 3

Smooth Surface Reconstruction

Surface reconstruction is a fundamental *reverse engineering* problem – for a given set of discrete points in the space (so-called scattered data) find a (virtual) surface those points were sampled from. It is a very well studied problem and a number of methods has been developed (see [16] for an overview of them). The main application is reconstruction of the real objects scanned by a 3D scanners (e.g. laser scanners). The output of the reconstruction process is a triangular mesh (usually very dense to catch all the details). We will not deal with this problem and we assume a triangular mesh as our input. Since also implicit and volume models can be polygonized, we can construct a smooth surface even for them, indirectly.

Thus, the problem specification is: for a given mesh and maybe its normals in vertices find a smooth surface passing through the vertices (and satisfying the given normals there) and having the same topology. More precisely, we differ the surface *interpolation*, when the surface passes the vertices exactly, and the surface *approximation*, when it passes only nearby them.

While in applications like car or aircraft design higher order smoothness is required due to the aerodynamics, in surface reconstruction the meaning of 'smooth' is mostly tangent-plane or derivative continuity - G^1, C^1, up to the curvature continuity G^2.

3.1 Surface Interpolation

Surface interpolation of arbitrary triangulations is not a simple task (see [1]). The main difficulty is to achieve the desired smoothness between adjacent patches. The shape of individual patches is determined from the tangent planes (i.e. from normals) in the mesh vertices, either given or estimated. A network of curves – the future patch boundaries and tangent plane fields along them are usually constructed first, over the mesh edges and filled by the patches afterwards.

The most often used surface interploants are based on so-called *split domain schemes* (see [10]) avoiding the problem of adjacent patches consistency in the vertices by splitting the domain triangle into subtriangles and constructing 3 (Clough-Tucker) or 6 or even 12 (Powel-Sabin) patches per face (see [15]). It is not acceptable for larger models we expect.

Another approach is constructing one patch (satisfying the prescribed boundaries and tangent planes along them) per face as done by *convex combination schemes* (see [10]). Although there is also the third approach – construction of a single patch for a whole group of neigbouring faces providing a direct reduction, our choice remains the 1 : 1 approach due to the progressive reduction in mind. We use the triangular Coons patches, they are thoroughly explained in the next section.

A separate problem is construction of a base mesh for a subdivision surface interpolating the given mesh vertices.

3.2 Triangular Coons Interpolant

Consider three curves sharing their endpoints as shown in Figure 3.1, thus forming a curved triangle. A triangular interpolant is a surface that fills the interior of that triangle and interpolates the given curves (i.e. they become the surface boundaries). Since the given curves are arbitrary curves, it is so-called *transfinite* (i.e. infinitely-dimensional) interpolation. Besides the three curves the input data can incorporate cross boundary derivatives or tangent plane fields along those curves to achieve smooth (C^1 or G^1) interpolation across the boundaries. The constructed patch $\mathcal{P}\mathbf{X}(u, v, w)$ is a parametric surface with barycentric coordinates as parameters and it is an image of a domain triangle $u+v+w = 1,\ \ u, v, w \geq 0$. We denote the vector of barycentric coordinates as $\mathbf{u} = (u, v, w)$ further.

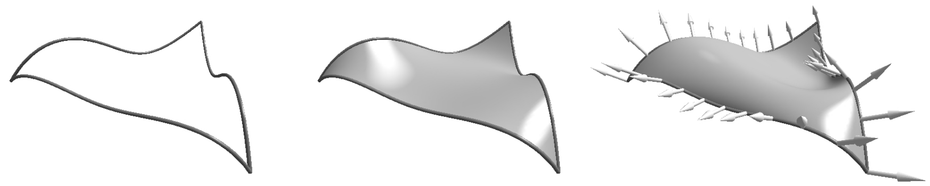

Figure 3.1: Three given curves sharing their endpoints on the left and corresponding Coons C^0 and C^1 interpolants respectively. Given normal fields are shown on the right image.

The basic principle of the triangular Coons interpolant is a construction of three simple auxiliary patches, each of them interpolating a part of input data and blending those patches together in such a clever way, that the result interpolates all input data. There are two main approaches how to construct the auxiliary patches and to blend them: The first one by Barnhill, Birkhoff and Gordon (BBG) constructs the auxiliary patches to interpolate a pair of boundary curves sharing a vertex. In the second one, Nielsons', each auxiliary patch interpolates one boundary curve and the vertex across. We describe both of them in a more general way than in [3].

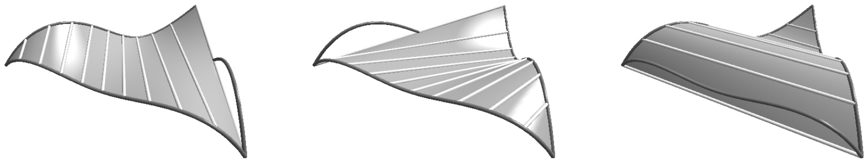

Figure 3.2: Three ruled surfaces $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$, $\mathcal{P}_3\mathbf{X}$ – the auxiliary patches of the Barnhill-Birkhoff-Gordon approach.

Barnhill, Birkhoff, Gordon

Let the three given curves be the three boundaries $\mathbf{X}(0, v, w)$, $\mathbf{X}(u, 0, w)$ and $\mathbf{X}(u, v, 0)$ of the unknown patch $\mathbf{X}(\mathbf{u})$. We construct three ruled surfaces – one for each pair of boundary curves as shown in Figure 3.2 (using notation from [3])

$$\begin{aligned}
\mathcal{P}_1\mathbf{X}(\mathbf{u}) &= (1-r)\mathbf{X}(u, 0, 1-u) + r\mathbf{X}(u, 1-u, 0) \\
\mathcal{P}_2\mathbf{X}(\mathbf{u}) &= (1-s)\mathbf{X}(1-v, v, 0) + s\mathbf{X}(0, v, 1-v) \\
\mathcal{P}_3\mathbf{X}(\mathbf{u}) &= (1-t)\mathbf{X}(0, 1-w, w) + t\mathbf{X}(1-w, 0, w) \\
\text{denoting } r &= \frac{v}{v+w} \qquad s = \frac{w}{w+u} \qquad t = \frac{u}{u+v}
\end{aligned} \tag{3.1}$$

Note that $v + w = 1 - u$, $w + u = 1 - v$ and $u + v = 1 - w$. The convex combination

$$\mathcal{P}\mathbf{X}(\mathbf{u}) = u\mathcal{P}_1\mathbf{X}(\mathbf{u}) + v\mathcal{P}_2\mathbf{X}(\mathbf{u}) + w\mathcal{P}_3\mathbf{X}(\mathbf{u}) \tag{3.2}$$

interpolate all three boundary curves. Actually the interpolant $\mathcal{P}\mathbf{X}$ is a blend of $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$ and $\mathcal{P}_3\mathbf{X}$ with some variable weight functions. Each patch from $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$, $\mathcal{P}_3\mathbf{X}$ interpolates two boundaries, but not the third one, thus the corresponding weight function must be zero on that boundary. For example $\mathcal{P}_1\mathbf{X}$ interpolates the boundaries respective to $v = 0$ and $w = 0$, but not the one for $u = 0$. Its weight function must be zero for $u = 0$ and the one from (3.2), namely u satisfies it. The weight functions u, v, w from

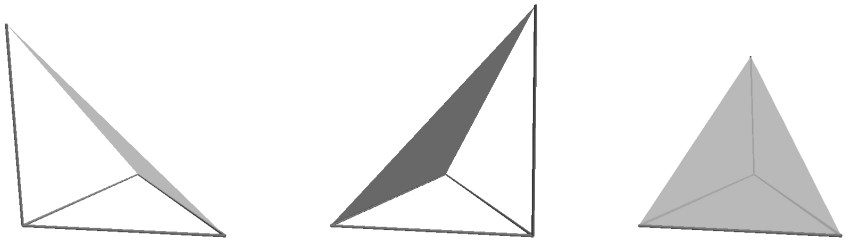

Figure 3.3: Linear weight functions $f(\mathbf{u}) = u$, $g(\mathbf{u}) = v$, $h(\mathbf{u}) = w$ used for C^0 interpolant. The weight of individual patches falls from 1 in a triangle vertex to 0 on the boundary across. Thus the resultant surface is almost identical to a particular patch in the vertex neighbourhood while it is absolutely unaffected by that patch on the boundary across.

the convex combination (3.2) satisfy the above requirement and they are shown in Figure 3.3.

To extend that scheme to C^1 or G^1 and incorporate the cross boundary derivatives or boundary tangent planes, we modify the construction of the auxiliary patches. Instead of the line segment ribs forming a ruled surface we use curved ribs interpolating also the tangent vectors or normals in the endpoints, besides the endpoints themselves (see Figure 3.4). Interpolation of normals (i.e. tangent planes) will be considered further, unlike [3], where interpolation of tangent vectors can be found. Thus the normal fields $\mathbf{X}_n(0, v, w)$, $\mathbf{X}_n(u, 0, w)$ and $\mathbf{X}_n(u, v, 0)$ are required as input besides the boundary curves. Moreover we assume a curve construction operator $\mathbf{C}$ giving for two points $\mathbf{P}_0$, $\mathbf{P}_1$ and their respective normals $\mathbf{n}_0$, $\mathbf{n}_1$ a curve $\mathbf{C}[\mathbf{P}_0, \mathbf{P}_1, \mathbf{n}_0, \mathbf{n}_1](t)$ interpolating them, i.e. $\mathbf{C}(0) = \mathbf{P}_0$, $\mathbf{C}(1) = \mathbf{P}_1$, $\mathbf{C}'(0) \cdot \mathbf{n}_0 = 0$ and $\mathbf{C}'(1) \cdot \mathbf{n}_1 = 0$.[1] We propose some curve constructions in the next section.

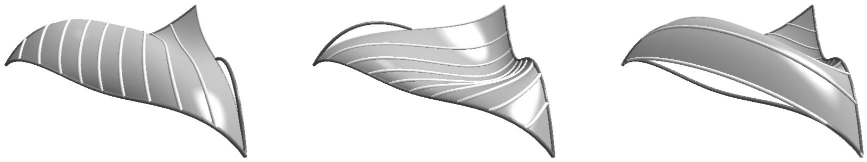

Figure 3.4: Three auxiliary patches of $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$, $\mathcal{P}_3\mathbf{X}$ in BBG approach for C^1 interpolant construction. The curved ribs respect given boundary normal fields.

[1] An analogic operator used in [3] is a cubic Hermit interpolation of the endpoints and tangent vectors.

We construct an auxiliary patch let's say $\mathcal{P}_1\mathbf{X}$ interpolating the curves $\mathbf{X}(u,0,w)$, $\mathbf{X}(u,v,0)$ and the tangent planes along them as follows:

$$\mathcal{P}_1\mathbf{X}(\mathbf{u}) = \mathbf{C}[\mathbf{X}(u,0,1-u), \mathbf{X}(u,1-u,0), \mathbf{X}_n(u,0,1-u), \mathbf{X}_n(u,1-u,0)](r), \tag{3.3}$$

where r is defined as in (3.1). This formula is very similar to the one in (3.1), only the linear segments are replaced by curved ribs constructed by the above operator $\mathbf{C}$. The ribs join points on the boundary curves with the same parameter u and they respect the normals. Thus all three auxiliary patches are given by

$$\begin{aligned}
&\mathcal{P}_1\mathbf{X}(\mathbf{u}) = \\
&\quad \mathbf{C}[\mathbf{X}(u,0,1-u), \mathbf{X}(u,1-u,0), \mathbf{X}_n(u,0,1-u), \mathbf{X}_n(u,1-u,0)](r) \\
&\mathcal{P}_2\mathbf{X}(\mathbf{u}) = \\
&\quad \mathbf{C}[\mathbf{X}(1-v,v,0), \mathbf{X}(0,v,1-v), \mathbf{X}_n(1-v,v,0), \mathbf{X}_n(0,v,1-v)](s) \\
&\mathcal{P}_3\mathbf{X}(\mathbf{u}) = \\
&\quad \mathbf{C}[\mathbf{X}(0,1-w,w), \mathbf{X}(1-w,0,w), \mathbf{X}_n(0,1-w,w), \mathbf{X}_n(1-w,0,w)](t)
\end{aligned} \tag{3.4}$$

where r, s, t are defined as in (3.1). The convex combination blend according to (3.2) of these three patches interpolates the boundary curves, but not the tangent plane fields. We can see it by calculating the derivative vector of the surface (3.2) in direction of parameter u (remember that $w = 1-u-v$):

$$\mathcal{P}_u\mathbf{X} = \mathcal{P}_1\mathbf{X} + u\mathcal{P}_{1u}\mathbf{X} + v\mathcal{P}_{2u}\mathbf{X} - \mathcal{P}_3\mathbf{X} + w\mathcal{P}_{3u}\mathbf{X}, \tag{3.5}$$

and for $u = 0$

$$\mathcal{P}_u\mathbf{X}\big|_{u=0} = \mathcal{P}_1\mathbf{X} + v\mathcal{P}_{2u}\mathbf{X} - \mathcal{P}_3\mathbf{X} + w\mathcal{P}_{3u}\mathbf{X}. \tag{3.6}$$

The derivative (and hence the tangent plane) on the boundary $u = 0$ is affected by $\mathcal{P}_1\mathbf{X}$ which does not interpolate that boundary and its tangent plane field. We need to find different blending functions to interpolate the tangent plane as well.

Let us write the blend (3.2) in a general fashion:

$$\mathcal{P}\mathbf{X}(\mathbf{u}) = f(\mathbf{u})\mathcal{P}_1\mathbf{X}(\mathbf{u}) + g(\mathbf{u})\mathcal{P}_2\mathbf{X}(\mathbf{u}) + h(\mathbf{u})\mathcal{P}_3\mathbf{X}(\mathbf{u}). \tag{3.7}$$

Consider what are the properties required for the blending functions $f(\mathbf{u})$, $g(\mathbf{u})$ and $h(\mathbf{u})$:

$$(\forall\mathbf{u} = (u,v,w), \quad u+v+w=1) \qquad f(\mathbf{u}) + g(\mathbf{u}) + h(\mathbf{u}) = 1, \tag{1}$$

$$f(\mathbf{u})\big|_{u=0} = 0, \qquad g(\mathbf{u})\big|_{v=0} = 0, \qquad h(\mathbf{u})\big|_{w=0} = 0. \tag{2}$$

The first requirement expresses affine correctness of the blend and the second one boundary interpolation. Example of blending functions satisfying (1) and (2) are $f(\mathbf{u}) = u$, $g(\mathbf{u}) = v$, $h(\mathbf{u}) = w$ used in (3.2). Wee need to add one more property to achieve interpolation of tangent plane besides, i.e. to make the derivatives on a boundary independent from the opposite auxiliary patch. Derivatives of each blending function must be zero in every direction $\mathbf{d}$ on the non-interpolated boundary:

$$\mathrm{D}_{\mathbf{d}} f(\mathbf{u})\big|_{u=0} = 0, \quad \mathrm{D}_{\mathbf{d}} g(\mathbf{u})\big|_{v=0} = 0, \quad \mathrm{D}_{\mathbf{d}} h(\mathbf{u})\big|_{w=0} = 0. \tag{3}$$

The blending functions from (3.2) do not satisfy that, we find different ones satisfying all (1), (2) and (3).

Looking at requirement (1) let us inspire by triangular Bernstein polynomials

$$\mathrm{B}^n_{ijk}(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k, \quad i+j+k=n. \tag{3.8}$$

Observation that $\sum_{i+j+k=n} = \mathrm{B}^n_{ijk}(\mathbf{u}) = 1$ leads to an idea that each of f, g, h would be a part of the above sum, and thus they satisfy (1). Note that the blending functions used in (3.2) are actually the first order triangular Bernstein polynomials. Investigating higher order Bernstein polynomials (see Figure 3.5) we come to blending functions based on 4^{th}-order Bernstein polynomials:

$$\begin{aligned}
f(\mathbf{u}) &= \mathrm{B}^4_{400} + \mathrm{B}^4_{310} + \mathrm{B}^4_{301} + \mathrm{B}^4_{211} + \frac{1}{2}(\mathrm{B}^4_{220} + \mathrm{B}^4_{202}) \\
g(\mathbf{u}) &= \mathrm{B}^4_{040} + \mathrm{B}^4_{031} + \mathrm{B}^4_{130} + \mathrm{B}^4_{121} + \frac{1}{2}(\mathrm{B}^4_{022} + \mathrm{B}^4_{220}) \\
h(\mathbf{u}) &= \mathrm{B}^4_{004} + \mathrm{B}^4_{103} + \mathrm{B}^4_{013} + \mathrm{B}^4_{112} + \frac{1}{2}(\mathrm{B}^4_{202} + \mathrm{B}^4_{022})
\end{aligned} \tag{3.9}$$

or expanded

$$\begin{aligned}
f(\mathbf{u}) &= u^4 + 4u^3 v + 4u^3 w + 12u^2 vw + 3u^2 v^2 + 3u^2 w^2 \\
g(\mathbf{u}) &= v^4 + 4v^3 w + 4uv^3 + 12uv^2 w + 3v^2 w^2 + 3u^2 v^2 \\
h(\mathbf{u}) &= w^4 + 4uw^3 + 4vw^3 + 12uvw^2 + 3u^2 w^2 + 3v^2 w^2
\end{aligned} \tag{3.10}$$

It is not difficult to verify, that above f, g, h satisfy (1), (2) and (3).[2] They are shown in Figure 3.6.

To conclude it, the patch interpolating the three given boundaries and the tangent plane fields along them is built from three auxiliary patches (3.4) by the scheme (3.7) using the blending functions defined as (3.10). Finally we show that the explained technique really interpolates the tangent plane.

[2] Calculate the derivatives in directions of two parameters to verify (3). Two independent zero derivatives implies zero derivative in every direction.

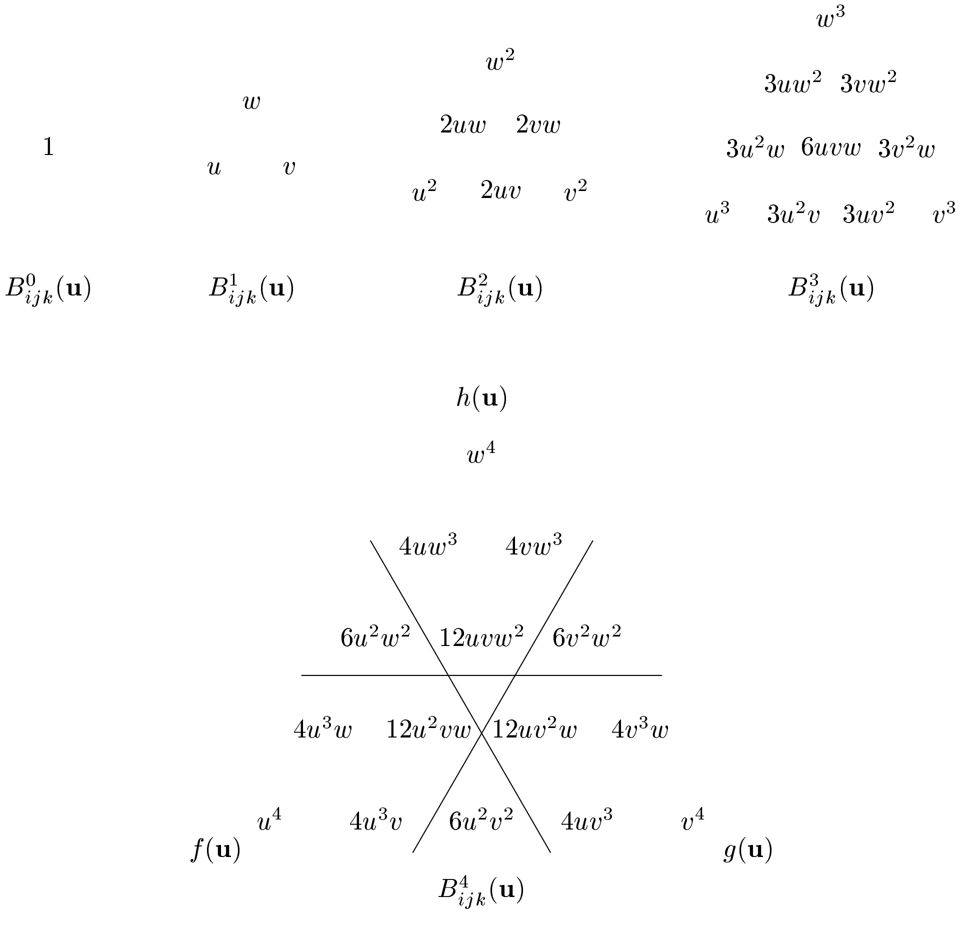

Figure 3.5: Triangular Bernstein polynomials up to degree 4. The weight functions $f(\mathbf{u})$, $g(\mathbf{u})$, $h(\mathbf{u})$ for C^1 interpolation are defined as a part of sum $\sum_{i+j+k=4} B^4_{ijk}(\mathbf{u})$.

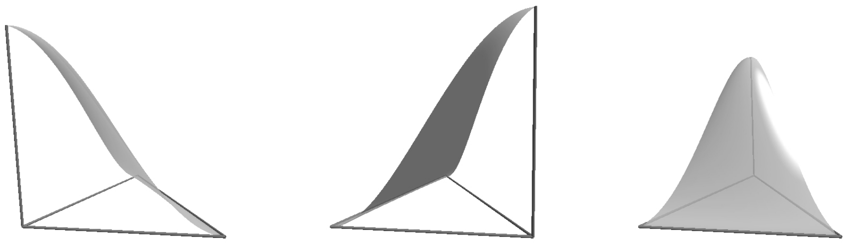

Figure 3.6: Weight functions f, g, h given by (3.10). Derivatives on the non-interpolated boundary are zero.

Derivative of the surface defined by (3.7) in direction of vector $\mathbf{d}$ is

$$\begin{aligned}\mathrm{D}_{\mathbf{d}}\mathcal{P}\mathbf{X} = \mathrm{D}_{\mathbf{d}}f \cdot \mathcal{P}_1\mathbf{X} + f \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_1\mathbf{X} + \mathrm{D}_{\mathbf{d}}g \cdot \mathcal{P}_2\mathbf{X} + g \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_2\mathbf{X} \\ + \mathrm{D}_{\mathbf{d}}h \cdot \mathcal{P}_3\mathbf{X} + h \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_3\mathbf{X}.\end{aligned} \tag{3.11}$$

Look at the derivatives on the boundary $u = 0$. Using properties (2) and (3) for f yields

$$\mathrm{D}_{\mathbf{d}}\mathcal{P}\mathbf{X}\big|_{u=0} = \mathrm{D}_{\mathbf{d}}g \cdot \mathcal{P}_2\mathbf{X} + g \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_2\mathbf{X} + \mathrm{D}_{\mathbf{d}}h \cdot \mathcal{P}_3\mathbf{X} + h \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_3\mathbf{X}. \tag{3.12}$$

Since both auxiliary patches $\mathcal{P}_2\mathbf{X}$ and $\mathcal{P}_3\mathbf{X}$ interpolate the boundary $u = 0$, i.e. $\mathcal{P}_2\mathbf{X}|_{u=0} = \mathcal{P}_3\mathbf{X}|_{u=0}$, we rewrite the previous formula to

$$\mathrm{D}_{\mathbf{d}}\mathcal{P}\mathbf{X}\big|_{u=0} = (\mathrm{D}_{\mathbf{d}}g + \mathrm{D}_{\mathbf{d}}h)\mathcal{P}_2\mathbf{X} + g \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_2\mathbf{X} + h \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_3\mathbf{X}. \tag{3.13}$$

$f + g + h$ is a constant function due to (1), its derivative is always zero and $\mathrm{D}_{\mathbf{d}}f\big|_{u=0} = 0$. Then $(\mathrm{D}_{\mathbf{d}}g + \mathrm{D}_{\mathbf{d}}h)\big|_{u=0} = 0$ and finally

$$\mathrm{D}_{\mathbf{d}}\mathcal{P}\mathbf{X}\big|_{u=0} = g \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_2\mathbf{X} + h \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_3\mathbf{X}, \quad g + h = 1. \tag{3.14}$$

It says that the tangent vector of the constructed patch is a linear combination of the $\mathcal{P}_2\mathbf{X}$ and $\mathcal{P}_3\mathbf{X}$ tangent vectors whose lies in the given tangent plane and so does their combination then. That is true also for the adjacent patch across the boundary, i.e. they have a common tangent plane on the boundary yielding the G^1-smoothness. The same reasoning can be performed for the other two boundaries.

We have explained construction of the triangular Coons patch for arbitrary given boundary curves and their tangent plane fields. In surface interpolation, however, only three vertices and normals of these vertices are given and the boundary curves and the tangent plane fields are defined from two endpoints and their normals. If we use the same curve construction operator $\mathbf{C}$ for both boundary curves and the 'ribs' of the three auxiliary patches, each of these patches will interpolate besides two boudaries and their tangent plane also the third one (but not its tangent plane!). The first introduced blending functions u, v and w are sufficient for smooth interpolation in this case. Justification: since $\mathcal{P}_1\mathbf{X}|_{u=0} = \mathcal{P}_2|\mathbf{X}_{u=0} = \mathcal{P}_3|\mathbf{X}_{u=0}$, formula (3.11) can be rewritten to

$$\begin{aligned}\mathrm{D}_{\mathbf{d}}\mathcal{P}\mathbf{X}\big|_{u=0} = (\mathrm{D}_{\mathbf{d}}f + \mathrm{D}_{\mathbf{d}}g + \mathrm{D}_{\mathbf{d}}h)\mathcal{P}_1\mathbf{X} \\ + f \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_1\mathbf{X} + g \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_2\mathbf{X} + h \cdot \mathrm{D}_{\mathbf{d}}\mathcal{P}_3\mathbf{X}.\end{aligned} \tag{3.15}$$

The sum in brackets is always zero as is explained above and due to property (2) for f we obtain the same result (3.14), but without using requirement (3). Hence blending functions satisfying only (1) and (2) are sufficient in case of the same construction of the boundary curves and the ribs.

Nielson

Another approach by Nielson constructs the three auxiliary patches interpolating one boundary (with its tangent field) and the vertex across that boundary (with its tangent plane). We skip the linear case and describe the C^1-smooth interpolation immediately.

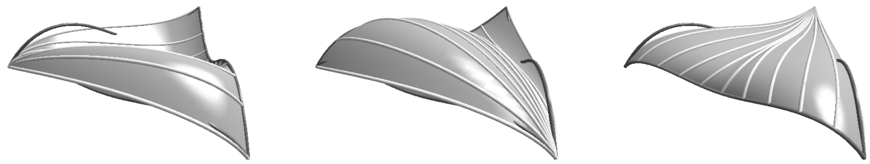

Figure 3.7: Three auxiliary patches $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$, $\mathcal{P}_3\mathbf{X}$ for Nielsons' C^1 interpolant. The radial curves respect normals on the interpolated boundary and the vertex across.

The ribs of auxiliary patches join the points of a boundary curve with the vertex across (so-called *radial curves*) as shown in Figure 3.7. The above curve construction operator $\mathbf{C}$ is fed by a boundary point, its normal derived from a boundary tangent plane field, the vertex across the boundary and its normal. The three patches are defined

$$\begin{aligned}\mathcal{P}_1\mathbf{X}(\mathbf{u}) &= \mathbf{C}[\mathbf{X}(1,0,0), \mathbf{X}(0,r,1-r), \mathbf{X}_n(1,0,0), \mathbf{X}_n(0,r,1-r)](u)\\ \mathcal{P}_2\mathbf{X}(\mathbf{u}) &= \mathbf{C}[\mathbf{X}(0,1,0), \mathbf{X}(1-s,0,s), \mathbf{X}_n(0,1,0), \mathbf{X}_n(1-s,0,s)](v)\\ \mathcal{P}_3\mathbf{X}(\mathbf{u}) &= \mathbf{C}[\mathbf{X}(0,0,1), \mathbf{X}(t,1-t,0), \mathbf{X}_n(0,0,1), \mathbf{X}_n(t,1-t,0)](w)\end{aligned} \tag{3.16}$$

with r, s, t as in (3.1).

If we use the same operator $\mathbf{C}$ for the boundary curves $\mathbf{X}(0,v,w)$, $\mathbf{X}(u,0,w)$, $\mathbf{X}(u,v,0)$ and the ribs as is usual in interpolation (each auxiliary patch thus interpolates all three boundaries), the following blending functions are suitable for G^1 interpolation (see [11]):

$$f(\mathbf{u}) = \frac{vw}{uv+vw+wu} \qquad g(\mathbf{u}) = \frac{wu}{uv+vw+wu} \qquad h(\mathbf{u}) = \frac{uv}{uv+vw+wu} \tag{3.17}$$

The shape of the above blending functions is shown in Figure 3.8.

Nielsons' approach is a bit more efficient for implementation since one point of the ribs is fixed. We are using it in our implementation.

3.3 Boundary Curves Construction

In triangular Coons interpolant we assume three given boundary curves or a curve construction operator $\mathbf{C}$. Our input is only a triangular mesh (with

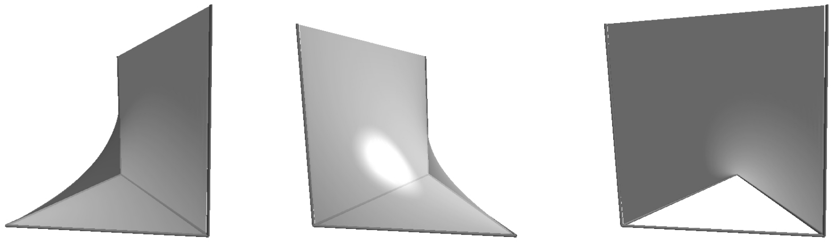

Figure 3.8: Weight functions f, g, h in Nielson's approach given by (3.17).

vertex normals), however. Here we explain how to construct those boundary curves over the mesh edges using normals in the vertices. Further we show how to define normal fields along the boundary curves in form of a normal field construction operator $\mathbf{C}_n$.

All we know about the desired curve over an edge are its endpoints $\mathbf{P}_0$, $\mathbf{P}_1$ – the edge endpoints and normals $\mathbf{n}_0$, $\mathbf{n}_1$ in the endpoints. The first idea is to use a cubic Hermit curve given by two endpoints and the tangent vectors in these endpoints as

$$\mathbf{C}(t) = \mathbf{P}_0 H_0^3(t) + \mathbf{P}_1 H_1^3(t) + \mathbf{n}_0 H_2^3(t) + \mathbf{n}_1 H_3^3(t) \tag{3.18}$$

where $H_i^3(t)$ are cubic Hermit polynomials

$$\begin{aligned} H_0^3(t) &= 2t^3 - 3t^2 + 1 \\ H_1^3(t) &= -2t^3 + 3t^2 \\ H_2^3(t) &= t^3 - 2t^2 + t \\ H_3^3(t) &= t^3 - t^2 \end{aligned} \tag{3.19}$$

The tangent vectors $\mathbf{t}_0$, $\mathbf{t}_1$ of the boundary curves can be derived from the normals $\mathbf{n}_0$, $\mathbf{n}_1$ as follows. The normals determine a tangent plane of the surface in the edge endpoints. Tangent vectors $\mathbf{t}_0$ and $\mathbf{t}_1$ must lie in these tangent planes and we define their directions as an orthogonal projection of the edge into the respective tangent plane as shown in Figure 3.9. The directions are given by

$$\begin{aligned} \mathbf{t}_0 &= \big(\mathbf{n}_0 \times (\mathbf{P}_1 - \mathbf{P}_0)\big) \times \mathbf{n}_0 \\ \mathbf{t}_1 &= \big(\mathbf{n}_1 \times (\mathbf{P}_1 - \mathbf{P}_0)\big) \times \mathbf{n}_1 \end{aligned} \tag{3.20}$$

A right question is: what is the length of the tangent vectors? We basically set it to the length of the edge, or we may introduce moreover a "tension" parameter η and define the tangent vector length as length of the edge divided by η. Thus the constructed curve will approach to the edge as η raises and hence the surface will approach to the faceted mesh. $\eta = 1$ does not affect the curve and when η is dropping to zero, the curve is

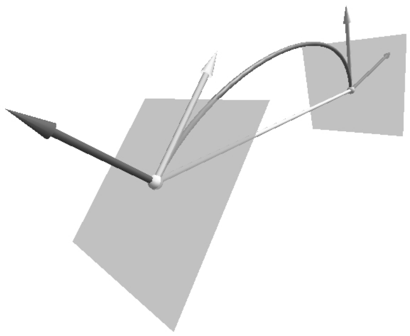

Figure 3.9: Calculation of tangent vectors from normals as an orthogonal projection of the edge vector into the tangent plane given by the normal.

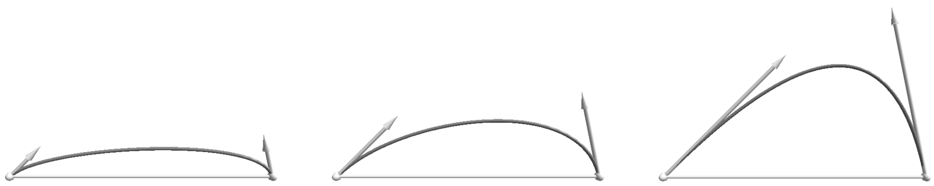

Figure 3.10: Shape of the Hermits' cubic for tension parameter 2, 1 and 0.5 respectively.

more and more attracted to the tangents of the endpoints (see Figure 3.10). The tension parameter η needs to be set according to the surface character.

The surface interpolation with Hermit boundary cubics, however, yields "too flat" interiors of the patches while the curvature is concentrated near the patch boundaries. It is concluded in [10], that the flatness of the patches is caused by the flatness of the Hermit boundary curves, i.e. the curvature is concentrated around the endpoints, while the middle part of the curve is more flat. This flatness is then propagated to the patch interior. A way to obtain a better quality surface is to construct the boundary curves with a more uniform curvature distribution. We propose such a curve construction.

The most uniform curvature distribution – constant, has a circular arc. But for two given points $\mathbf{P}_0$, $\mathbf{P}_1$ and their normals $\mathbf{n}_0$, $\mathbf{n}_1$ generally does not exist a circular arc from $\mathbf{P}_0$ to $\mathbf{P}_1$ with those given normals. Two circular arcs $\mathbf{A}_0(t)$ and $\mathbf{A}_1(t)$ can be constructed, however, interpolating $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{n}_0$ and $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{n}_1$ respectivelly. And, as one can easily verify, a linear blend of these two arcs

$$\mathbf{C}(t) = (1-t)\mathbf{A}_0(t) + t\mathbf{A}_1(t) \tag{3.21}$$

interpolates all $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{n}_0$, $\mathbf{n}_1$. Centers of the arcs are found as an intersection of the line given by an endpoint with its desired normal and a plane perpendicular to the edge P_0P_1 crossing the edge midpoint (see Figure 3.11).

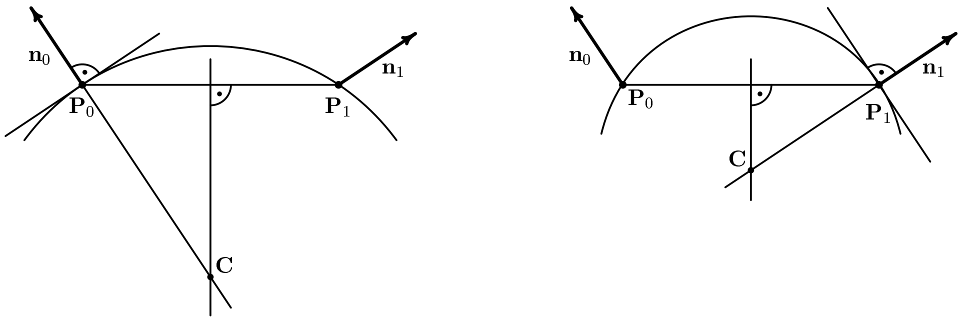

Figure 3.11: Construction of circular arcs interpolating $(\mathbf{P}_0, \mathbf{P}_1, \mathbf{n}_0)$ and $(\mathbf{P}_0, \mathbf{P}_1, \mathbf{n}_1)$ respectively.

Curvature of the above circular arcs blend is more uniform, as shown in Figure 3.12 and hence it yields a more pleasing surface. Even better curvature distribution and surface fairness is achieved using a cubic blend

$$\begin{aligned} \mathbf{C}(t) &= f(t)\mathbf{A}_0(t) + (1-f(t))\mathbf{A}_1(t) \\ f(t) &= 2t^3 - 3t^2 + 1 \end{aligned} \tag{3.22}$$

or a rational blend for

$$f(t) = \frac{t^2 - 2t + 1}{2t^2 - 2t + 1} \tag{3.23}$$

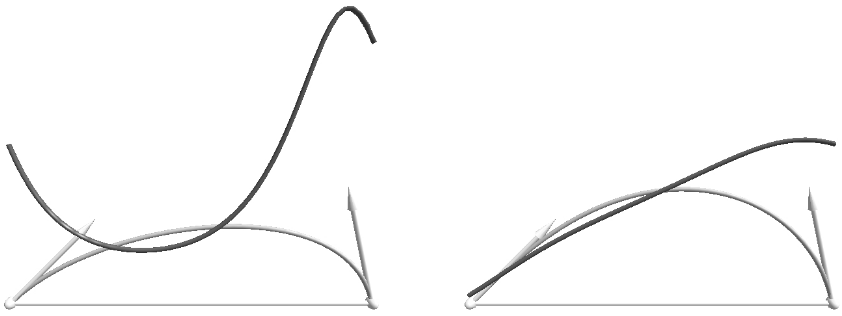

Figure 3.12: Curvature comparison of Hermits' cubic (left) and a blend of two circular arcs (right). The dark curve reflects curvature values along the ligth curve as a variable height.

The last issue, omitted so far, is the normal field construction. The normal field $\mathbf{C}_n(t)$ must return a normal to the curve $\mathbf{C}(t)$ for all $t \in \langle 0, 1\rangle$ and it must also interpolate the end-normals, i.e. $\mathbf{C}_n(0) = \mathbf{n}_0$ and $\mathbf{C}_n(1) = \mathbf{n}_1$. A simple interpolation of $\mathbf{n}_0$ and $\mathbf{n}_1$ is not guaranteed to be perpendicular to $\mathbf{C}'(t)$ for all t. We use a little 'trick'. 'Binormals'[3] $\mathbf{b}_0$, $\mathbf{b}_1$ in the endpoints are calculated as $\mathbf{b}_0 = \mathbf{n}_0 \times \mathbf{C}'(0)$ and $\mathbf{b}_1 = \mathbf{n}_1 \times \mathbf{C}'(1)$. To get a normal $\mathbf{C}_n(t)$ for a particular t the $\mathbf{b}_0$, $\mathbf{b}_1$ are linearly interpolated $\mathbf{b}_t = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$ and the normal is calculated then as $\mathbf{C}_n(t) = \mathbf{C}'(t) \times \mathbf{b}_t$. This normal is clearly perpendicular to the curve tangent and it interpolates $\mathbf{n}_0$, $\mathbf{n}_1$ as well due to the 'binormal' interpolation. Better results are achieved by quadratic interpolation of the 'binormal' adding one more 'binormal' sample in the curve midpoint derived from the edge normal defined above.

3.4 Surface Approximation

Although surface interpolation provides a smooth surface passing through a given set of points, its shape is often not so pleasing as one wishes. An undesired undulation can appear as a consequence of slightly inaccurate positions of the real surface points due to a noise in the scanning process. Inaccuracies are not smoothed out but even emphasized by interpolation. A more pleasing surface can be obtained by relaxing the requirement of passing exactly through the given points for the constructed surface to minimization of some chosen error norm.

The most used is the l_2 error norm minimizing the sum of squared deviations of the given discrete points to the constructed surface inducing the name the *least squares method* (LSM). The problem leads to solving a system of linear equations. The use of a different norm, e.g. l_1 or l_∞ may

[3] They are not the real binormals, in fact, since the $\mathbf{n}_0$, $\mathbf{n}_1$ are generally not the main normals of the curve.

result in a better error distribution (see [6]) and leads to a *linear programming* problem.

Approximation is used usually to approximate a high number of points by a lower degree function; specially a number of mesh faces by a single smooth patch. The regular meshes are favourable here, since it is more efficient to approximate structured samples benefitting from the underlying curve algorithms. Hence authors construct a regular mesh from the original one to determine the shape of the approximating patch in [9], [2], [12]. Moreover the *constrained LSM* is a very usefull in this context – here some conditions must be satisfied exactly (e.g. on patch boundaries to ensure the smoothness) while the remaining ones are only approximated (patch interiors).

Chapter 4

Implementation

A simple implementation is being developed continuously to test and experiment with the previously described methods. It is written in C++ language, rendering is provided by OpenGL functions, hence it is rather portable. Currently it runs under MS Windows. The input mesh is loaded from a text file in simple .tri format. The rendered images in this paper come from our implementation.

Chapter 5

Conclusion and Results

We had not intended to spend much time with the smooth surface reconstruction stage, but it turned out to be a difficult task to achieve a really "fair" surface. Although the reconstruction surface is tangent-plane smooth, using the Hermit boundary curves the patch interiors are relatively flat and the curvature is concentrated near the patch boundaries making composition of the surfaces from individual patches visible. Even a decrease of the introduced tension parameter η does not solve the "flatness problem". In that case the flat regions appear around the mesh vertices. Better situation is with the second type of boundary curves, the blend of two circular arcs. See Figure 5.2 for comparison. The really pleasing surface usually needs some fairing process as used in [9], [2], typically an iterative one. We avoid of such process due to its global nature, since we expect really large models.

The smooth surface reconstruction needs to be extended to make the representation of sharp edges and vertices possible, using multiple normals. Moreover, our technique of Coons patches with the curves constructed in a sophisticated way is not quite suitable for integrating to the graphics hardware due to its algorithmic nature. It will be very useful to convert them to (or to use directly) a more standard and compact patches, e.g. Bezier, Gregory or B-spline patches. It is also the area of the necessary future work. Gregory patches look like a suitable way.

Finally some images from our implementation follows.

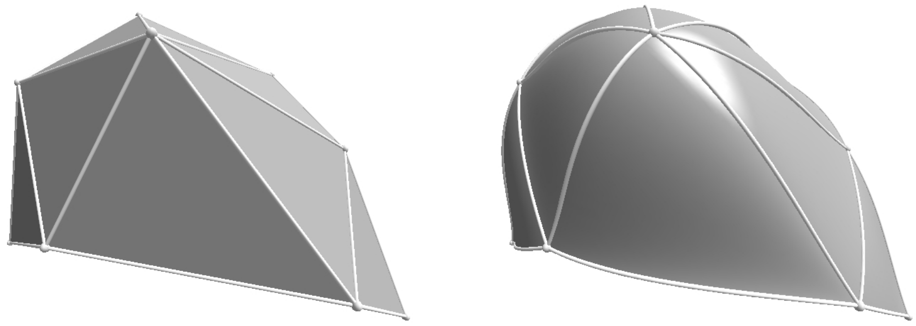

Figure 5.1: Illustration of a smooth surface reconstruction from a mesh portion. The patch boundaries are blends of circular arcs.

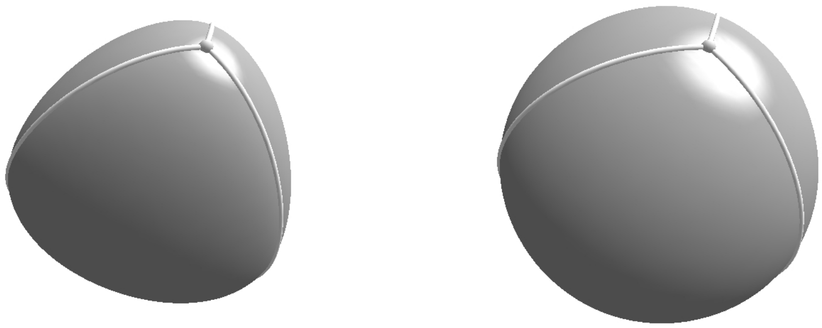

Figure 5.2: Comparison of reconstruction from a tetrahedron using cubic Hermit boundary curves (left) vs. circular arcs blending curves (right). The right surface is closer to a sphere.

Figure 5.3: Example of a larger model – torus with 641 faces.

Figure 5.4: Another example of a larger model – a cow with 5804 faces. The problems with interpolation mentioned in section 3.4 appears here – the surface approximates the original mesh too exactly.

Bibliography

[1] Chandrajit L. Bajaj and Susan Evans. Splines and geometric modeling. *CRC Handbook of Discrete and Computational Geometry*, 1997.

[2] Matthias Eck and Hugues Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. *Computer Graphics*, 30(Annual Conference Series):325–334, 1996.

[3] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, 4^{th} edition, 1996.

[4] Dianne Hansford Gerald E. Farin. *The Geometry Toolbox for Graphics and Modeling*. A K Peters Ltd, 1^{st} edition, 1998.

[5] Igor Guskov, Andrei Khodakovsky, Peter Schröder, and Wim Swelden. Hybrid meshes. 2001.

[6] W. Heidrich, R. Bartels, and G. Labahn. Fitting uncertain data with NURBS. In A. Le Mehaute, C. Rabut, and L. L. Schumaker, editors, *Curves and Surfaces in Geometric Design*. Vanderbilt University Press, 1997.

[7] Hugues Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.

[8] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28(Annual Conference Series):295–302, 1994.

[9] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. *Computer Graphics*, 30(Annual Conference Series):313–324, 1996.

[10] S. Mann, C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan. A survey of parametric scattered data fitting using triangular interpolants, 1992.

[11] G. M. Nielson. A transfinite, visually continuous, triangular interpolant. *Geometric modeling: Algorithms and new trends*, pages 235–246, 1987.

[12] In Kyu Park, Sang Uk Lee, and Il Dong Yun. Constructing NURBS surface model from scattered and unorganized range data. *Proceedings of the Second International Conference on 3-D Imaging and Modeling*, 1998.

[13] Les A. Piegl and Wayne Tiller. *The NURBS Book*. Springer Verlag, 2^{nd} edition, 1997.

[14] Hong Qin and Demetri Terzopoulos. Triangular NURBS and their dynamic generalizations. *Computer Aided Geometric Design*, 14(4):325–347, 1997.

[15] Malcolm Sabin. Computer aided geometric design: An elementary and superficial course.

[16] Inge Soderkvist. Introductory overview of surface reconstruction methods, 1999.