FACULTY OF MATHEMATICS, PHYSICS and INFORMATICS
COMENIUS UNIVERSITY
Bratislava

D I P L O M A T H E S I S

Bratislava, April 2002

Michal Farkaš

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

Triangular B-Spline Surfaces in Geometric Modeling

Diploma Thesis

Author:
Michal Farkaš

Diploma Thesis Consultant:
RNDr. Pavel Chalmovianský

I honestly allege that I have developed the diploma thesis independently only with the use of the literature listed in the bibliography.

Bratislava, 28th April 2002

. . . .

Michal Farkaš

This way I have to thank to my diploma thesis consultant RNDr. Pavel Chalmovianský for valuable technical advices, remarks and suggestions, which helped me to realize the final form of this thesis.

Contents

Introduction

Nowadays the geometric modeling represents an important part of information technology industry, especially in machinery, visualisation, advertisement and entertainment areas. In various applications of geometric modeling we can see use of different collections of surfaces and methods representing widely different principles and technologies. Every concrete area of operation requires specific conditions and approach.

In my diploma thesis we will explore specific surface scheme: Triangular B-spline Surfaces. The purpose of my diploma thesis is to investigate triangular B-spline surfaces and to detect pros and cons of implemented surfaces in practice. The focus of my diploma thesis is to explore practical possibilities of triangular B-spline surfaces from the view of computational complexity field. The faster the algorithm is, the more flexibility the designer has. It is usual question concering computational complexity, whether it's better to use more space complexity or more time complexity.

The primary focus of my diploma thesis is decomposed at lower level into more specific goals e.g. devising fast algorithms for points evaluation and discussing more possibilities, creating appropriate data structures, and comparing triangular B-spline surfaces to another surfaces schemes.

The first important step to examine given surface scheme is to start with sufficient theoretical basis. In first pages we introduce necessary theoretical background for triangular B-spline surfaces and we start with polar forms and certain properties and theorems for polar forms.

In the B-patches chapter we introduce triangular B-spline surfaces, we describe certain properties of triangular B-spline surfaces scheme and we present an algorithm for fast points evaluation. Next we define data structures and show certain optimizations. Data structures and optimizations are typical for design and they present a solution for implementation. B-patches definition consists of many parameters and conditions, which are discussed in B-patch Checking section. The B-patch adaptation section describes a method of conrol vertices movement.

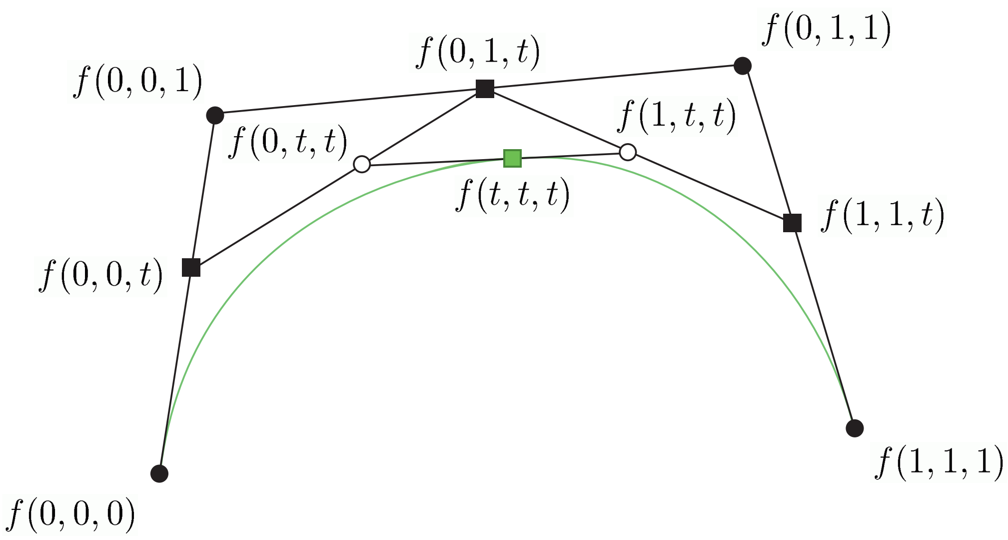
The next chapter represents another approach to triangular B-spline surfaces via bivariate B-splines. We present definition and a set of properties of bivariate B-splines scheme. An effective recurrence computation for Bivariate B-splines is important for implementation and we describe a method for bivariate B-splines computation.

New B-spline scheme combines previous approaches and a powerful surface scheme is developed there. Triangular B-spline surfaces and bivariate B-splines can share their properties. Here we can see a solution to produce smooth and continuous surfaces.

Chapter 1

Polar Forms

Polar forms simplify the construction of polynomial and piecewise-polynomial curves and surfaces and lead to a new surface representations and algorithms. The main idea behind polar forms is explained in the following figure:



In the figure, we show a cubic Bézier curve F defined over the unit interval $[0,1]$ with its Bézier points and all the intermediate points that came up while we used the de Casteljau algorithm to evaluate the curve at a parameter t. What is new in this figure is the labeling scheme. While most standard texts use labels like b_i^j for the de Casteljau algorithm's intermediate points, the labels in the figure are of the form $f(.,.,.)$, where f is the polar form of the polynomial F. Since F is of degree three, its polar form has three arguments. Furthermore, the polar form is symmetric, you can write its three arguments in any order without changing the value of f, and f is related to F by the identity $F(u) = f(u,u,u)$. Finally, the labels reflects the incidency structure of the points and lines in the figure. All points, whose labels share at least two arguments lie on the same line. The exact position of a point on this line is determined by the remaining third label: As t moves with constant speed between 0 and 1, the point $f(0, t, 1)$, for example, moves with constant speed between $f(0, 0, 1)$ and $f(0, 1, 1)$. The point $f(0, t, 1)$ lies of the way from $f(0, 0, 1)$ to $f(0, 1, 1)$. Moving on a line with constant speed means that the polar form is *affine* in each argument, or simply *multiaffine*. Thus, the polar form of a cubic polynomial curve F is symmetric triaffine map that satisfies $F(u) = f(u, u, u)$.

The above properties let us reconstruct the point $F(t)$ from the Bézier points $f(0, 0, 0)$, $f(0, 0, 1)$, $f(0, 1, 1)$ and $f(1, 1, 1)$ as follows: First we interpolate linearly along the edges of the control polygon to get the points $f(0, 0, t), f(0, t, 1)$ and $f(t, 1, 1)$. Then we interpolate linearly between these points to get $f(0, t, t)$ and $f(t, t, 1)$. Finally, the last step of interpolation between these two points yield the point $F(t) = f(t, t, t)$ on the curve. This is exactly what the de Casteljau algorithm does.

1.1. Polynomials and Polar Forms

In this section we introduce the polar form of a surface. We show that polynomials and polar forms are equivalent to each other.

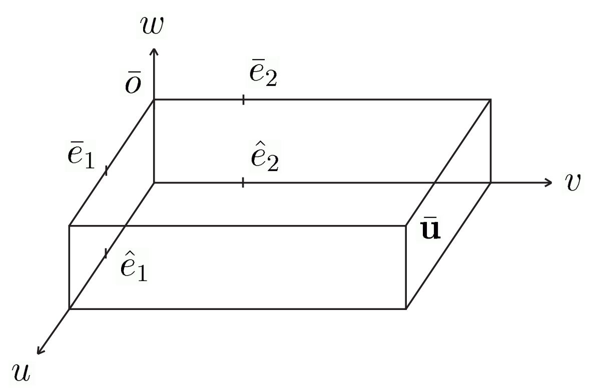
Often it is suitable to use as short symbols as possible and we will define shortcuts in the text. Instead of $i = 0, \ldots, n$ we will write $i = \overline{0, n}$.

Definition 1.1 (Linear Map).
A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is linear iff it preserves linear combinations, i.e. iff f satisfies $f(x\mathbf{u} + y\mathbf{v}) = xf(\mathbf{u}) + yf(\mathbf{v})$, where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, $x, y \in \mathbb{R}$. $\square$

We can rewrite definition of linear map more generally. A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is linear iff f satisfies $f(\sum x_i \mathbf{u}_i) = \sum x_i f(\mathbf{u}_i)$, where $\mathbf{u}_i \in \mathbb{R}^n$, $x_i \in \mathbb{R}$, $i = \overline{0, k}$.

Definition 1.2 (Affine Map).
A map $f : \mathbb{R}^2 \to \mathbb{R}^t$ is affine iff it preserves affine combinations, i.e. iff f satisfies $f(\sum_{i=0}^m \alpha_i \mathbf{u}_i) = \sum_{i=0}^m \alpha_i f(\mathbf{u}_i)$ whenever $\sum_{i=0}^m \alpha_i = 1$. $\square$

Every affine map f has a linearization f_*. In order to define f_* we consider the embeddings as shown in next figure $e_0 : \mathbb{R}^2 \to \mathbb{R}^3 : \mathbf{u} = (u, v) \mapsto \hat{\mathbf{u}} = (0, u, v)$ and $e_1 : \mathbb{R}^2 \to \mathbb{R}^3 : \mathbf{u} = (u, v) \mapsto \bar{\mathbf{u}} = (1, u, v)$. It is easily seen that e_1 is affine map.



Theorem 1.1 (Linearization).
Affine maps $f : \mathbb{R}^2 \to \mathbb{R}^t$ and linear maps $f_* : \mathbb{R}^3 \to \mathbb{R}^t$ are equivalent. In particular, given a map of either type, a unique map of the other type exist that satisfies the identity $f(\mathbf{u}) = f_*(\bar{\mathbf{u}})$. In this situation f_* is called the linearization of f.

Proof ► Let $o \in \mathbb{R}^2$ be the origin and let $\mathcal{B} = \{\mathbf{e}_1, \mathbf{e}_2\}$ be a basis for $\mathbb{R}^2$. Since $\bar{\mathcal{B}}_0 = \{\bar{\mathbf{o}}, \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2\}$ is a basis of $\mathbb{R}^3$ there exists a unique linear map f_* satisfying $f_*(\bar{\mathbf{o}}) = f(\mathbf{o})$, $f_*(\bar{\mathbf{e}}_1) = f(\mathbf{e}_1)$, $f_*(\bar{\mathbf{e}}_2) = f(\mathbf{e}_2)$. Writing $\mathbf{u} = (1 - u_1 - u_2)\mathbf{o} + u_1\mathbf{e}_1 + u_2\mathbf{e}_2$ as an affine combination of $\mathbf{o}, \mathbf{e}_1$, and $\mathbf{e}_2$, it is easily checked that $f(\mathbf{u}) = f_*(\bar{\mathbf{u}})$. The converse is straightforward. ◄

Example 1.1 (Linearization).
Consider the affine map $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(\mathbf{u}) = 1 + 2u + v$, where $\mathbf{u} = u\mathbf{e}_1 + v\mathbf{e}_2 \in \mathbb{R}^2$. Let us consider the basis $\mathcal{B}_0 = \{\bar{\mathbf{o}}, \hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2\}$. The linearization f_* of f w.r.t. $\mathcal{B}_0$ is given as $f_*(\mathbf{v}) = w + 2u + v$, where $\mathbf{v} = w\bar{\mathbf{o}} + u\hat{\mathbf{e}}_1 + v\hat{\mathbf{e}}_2 \in \mathbb{R}^3$. This follows from $f_*(\bar{\mathbf{o}}) = 1, f_*(\bar{\mathbf{e}}_1) = (\bar{\mathbf{o}}) + (\hat{\mathbf{e}}_1) = f(\mathbf{e}_1)$ and $f_*(\bar{\mathbf{e}}_2) = (\bar{\mathbf{o}}) + (\hat{\mathbf{e}}_2) = f(\mathbf{e}_2)$. □

Definition 1.3 (Symmetric, Multilinear, Multiaffine).
Let ϕ be an n-variate map. Then ϕ is symmetric iff $\phi(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \phi(\mathbf{u}_{\pi(1)}, \ldots, \mathbf{u}_{\pi(n)})$ for all permutations π of the set $\{1, \ldots, n\}$. The map ϕ is multilinear (multiaffine) iff ϕ is linear (affine) in each argument if the other are held fixed. □

Theorem 1.2 (Multilinearization).
Symmetric multiaffine maps $f : (\mathbb{R}^2)^n \to \mathbb{R}^t$ and symmetric multilinear maps $f_* : (\mathbb{R}^3)^n \to \mathbb{R}^t$ are equivalent. In particular, given a map of either type, a unique map of the other type exists that satisfies identity $f(\mathbf{u}_1, \ldots, \mathbf{u}_n) = f_*(\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_n)$.

Proof ► The proof is almost identical to the proof of Theorem 1.1. Let $\mathbf{o} \in \mathbb{R}^2$ be the origin and let $\mathcal{B} = \{\mathbf{e}_1, \mathbf{e}_2\}$ be a basis of $\mathbb{R}^2$. Since $\bar{\mathcal{B}}_0 = \{\bar{\mathbf{o}}, \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2\}$ is a basis of $\mathbb{R}^3$, a symmetric multilinear map $f_* : (\mathbb{R}^3)^n \to \mathbb{R}^t$ is defined by the $\binom{n+2}{2}$ values $f_*(\bar{\mathbf{o}}, \ldots, \bar{\mathbf{o}}, \bar{\mathbf{e}}_1, \ldots, \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \ldots, \bar{\mathbf{e}}_2)$. Setting $f_*(\bar{\mathbf{o}}, \ldots, \bar{\mathbf{o}}, \bar{\mathbf{e}}_1, \ldots, \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2, \ldots, \bar{\mathbf{e}}_2) = f(\mathbf{o}, \ldots, \mathbf{o}, \mathbf{e}_1, \ldots, \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_2)$ we thus obtain a symmetric multilienar map f_* that satisfies the identity $f(\mathbf{u}_1, \ldots, \mathbf{u}_n) = f_*(\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_n)$. Again, the converse is trivial. ◄

Now we are ready to establish blossoming principle.

Theorem 1.3 (Blossoming Principle).
Polynomials $F : \mathbb{R}^2 \to \mathbb{R}^t$ of degree n, symmetric multiaffine maps $f : (\mathbb{R}^2)^n \to \mathbb{R}^t$, and symmetric multilinear maps $f_* : (\mathbb{R}^3)^n \to \mathbb{R}^t$ are equivalent. In particular, given a map of either type, unique maps of the other two types exist that satisfy the identity $F(\mathbf{u}) = f(\mathbf{u}, \ldots, \mathbf{u}) = f_*(\bar{\mathbf{u}}, \ldots, \bar{\mathbf{u}})$. In this situation f and f_* are called the multiaffine and multilinear polar form or blossom of F. Furthermore, the q-th directional derivative of F with respect to vectors $\xi_1, \ldots, \xi_q \in \mathbb{R}^2$ is given as

$$D_{\xi_1,\ldots,\xi_q} F(\mathbf{u}) = \frac{n!}{(n-q)!} f_*(\bar{\mathbf{u}}, \ldots, \bar{\mathbf{u}}, \hat{\xi}_1, \ldots, \hat{\xi}_q) \tag{1.1}$$

where $\hat{\xi} = \mathbf{e}_0(\xi) \in \mathbb{R}^3$. □

Proof ▶ We start with the existence of f_*. Since $\mathcal{B}_0 = \{\bar{\mathbf{o}}, \hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2\}$ is a basis of $\mathbb{R}^3$, a symmetric multilinear map $f_* : (\mathbb{R}^3)^n \to \mathbb{R}^t$ is completely defined by the $\binom{n+2}{2}$ values $f_*(\bar{\mathbf{o}}, \ldots, \bar{\mathbf{o}}, \hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \ldots, \hat{\mathbf{e}}_2)$. Setting

$$f_*(\underbrace{\bar{\mathbf{o}}, \ldots, \bar{\mathbf{o}}}_{i}, \underbrace{\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_1}_{j}, \underbrace{\hat{\mathbf{e}}_2, \ldots, \hat{\mathbf{e}}_2}_{k}) = \frac{i!}{n!} \frac{\partial^{j+k} F(0)}{\partial_u^j \partial_v^k} \tag{1.2}$$

we thus obtain a multilinear map f_* and writing $\bar{\mathbf{u}} = \bar{\mathbf{o}} + u\hat{\mathbf{e}}_1 + v\hat{\mathbf{e}}_2$ we see that f_* satisfies

$$\begin{aligned} f_*(\bar{\mathbf{u}}, \ldots, \bar{\mathbf{u}}) &= \sum_{i+j+k=n} \binom{n}{ijk} f_*(\underbrace{\bar{\mathbf{o}}, \ldots, \bar{\mathbf{o}}}_{i}, \underbrace{\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_1}_{j}, \underbrace{\hat{\mathbf{e}}_2, \ldots, \hat{\mathbf{e}}_2}_{k}) u^j v^k \\ &= \sum_{i+j+k=n} \frac{\partial^{j+k} F(\mathbf{o})}{\partial_u^j \partial_v^k} \frac{u^j v^k}{j!k!} \\ &= F(\mathbf{u}) \end{aligned}$$

where the last step follows from the Taylor's theorem.

The above argument also shows that some symmetric multiaffine map f satisfying $f(\mathbf{u}, \ldots, \mathbf{u}) = F(\mathbf{u})$ must satisfy (1.3), hence f is unique. Finally, (1.2) follows from the fact that the differential operator D_ξ is linear in ξ. Together with Theorem 1.2 this completes the proof of Theorem 1.3. ◀

Corollary 1.1 (Monomials).
Consider a monomial

$$F(\mathbf{u}) = \sum_{j+k \le n} \mathbf{a}_{jk} u^j v^k, \mathbf{a}_{jk} \in \mathbb{R}^t. \tag{1.3}$$

Then the coefficients $\mathbf{a}_{jk}$ satisfy

$$\mathbf{a}_{jk} = \binom{n}{ijk} f_*(\underbrace{\bar{\mathbf{o}}, \ldots, \bar{\mathbf{o}}}_{i}, \underbrace{\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_1}_{j}, \underbrace{\hat{\mathbf{e}}_2, \ldots, \hat{\mathbf{e}}_2}_{k}). \tag{1.4}$$

The multilinear polar form f_* of F w.r.t. $\mathcal{B}_o = \{\bar{\mathbf{o}}, \hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2\}$ is given as

$$f_*(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sum_{\Gamma_n = \{I,J,K\}} \binom{n}{|I||J||K|}^{-1} \mathbf{a}_{|J||K|} \prod_{i \in I} w_i \prod_{j \in J} u_j \prod_{k \in K} v_k \tag{1.5}$$

where $\mathbf{v}_i = w_i \bar{\mathbf{o}} + u_i \hat{\mathbf{e}}_1 + v_i \hat{\mathbf{e}}_2 \in \mathbb{R}^3$. The multiaffine polar form f of F with respect to $\mathcal{B} = \{e_1, e_2\}$ is given as

$$f(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \sum_{\Gamma_n=\{I,J,K\}} \binom{n}{|I||J||K|}^{-1} \mathbf{a}_{|J||K|} \prod_{j\in J} u_j \prod_{k\in K} v_k. \tag{1.6}$$

where $\mathbf{u}_1 = u_1\mathbf{e}_1 + v_1\mathbf{e}_2 \in \mathbb{R}^2$. Here $\mathcal{P}_n = \{I, J, K\}$ denotes a partition of $\{1, \ldots, n\}$ into the sets I, J, K.

Proof ▶ Using Taylor's Theorem, formula (1.4) follows from (1.2). Writing $\mathbf{v}_i = w_i\bar{\mathbf{o}} + + u_i\hat{\mathbf{e}_1} + v_i\hat{\mathbf{e}_2}$, the multinomial theorem then shows

$$f_*(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sum_{\Gamma_n=\{I,J,K\}} f_*(\underbrace{\bar{\mathbf{o}}, \ldots, \bar{\mathbf{o}}}_{i}, \underbrace{\hat{\mathbf{e}_1}, \ldots, \hat{\mathbf{e}_1}}_{j}, \underbrace{\hat{\mathbf{e}_2}, \ldots, \hat{\mathbf{e}_2}}_{k}) \prod_{i\in I} w_i \prod_{j\in J} u_j \prod_{k\in K} v_k,$$

and (1.5) follows. Specialization (1.5) to $\mathbf{v}_i = \bar{\mathbf{u}}_i = (1, u_i, v_i)$ yields (1.6). ◀

Chapter 2

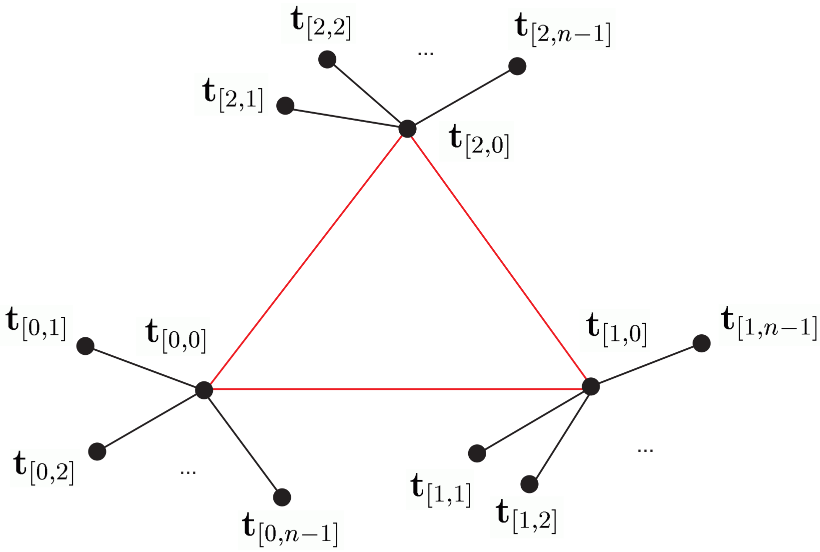
B-Patches

In this section we generalize triangular Bézier patches by assigning a family of usually different knots to each vertex of the triangle $\triangle$. The resulting representation is called a *B-patch*.

2.1. The B-Patch Representation

Definition 2.1 (Knot Arrangement).
Let $\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n-1}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n-1}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n-1} \in \mathbb{R}^2$ be given and assume that every triple of knots $(\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k})$, $0 \le i+j+k \le n-1$ is affinely independent, i.e. that $\triangle_{i,j,k} = [\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k}]$ forms a proper triangle. Then $\mathcal{A} = \{\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n-1}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n-1}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n-1}\}$ is called a *knot arrangement associated with the triangle* $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2] := [\mathbf{t}_{0,0}, \mathbf{t}_{1,0}, \mathbf{t}_{2,0}]$. $\square$

Typical knot arrangement example:



Definition 2.2 (Normalized B-Weights).
Let $\mathcal{A}$ be a knot arrangement associated with a triangle $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$ and let $\lambda_{ijk,d}(\mathbf{u}),\ d = 0, 1, 2$ be the barycentric coordinates of $\mathbf{u}$ w.r.t. $\triangle_{ijk} = [\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k}]$. The polynomials $B^{\mathcal{A}}_{ijk}(\mathbf{u}), i + j + k = n$, that are defined recursively as

$$B^{\mathcal{A}}_{000}(\mathbf{u}) = 1 \tag{2.1}$$

and

$$B^{\mathcal{A}}_{ijk}(\mathbf{u}) = \lambda_{i-1,j,k,0} B^{\mathcal{A}}_{i-1,j,k}(\mathbf{u}) + \lambda_{i,j-1,k,1} B^{\mathcal{A}}_{i,j-1,k}(\mathbf{u}) + \lambda_{i,j,k-1,2} B^{\mathcal{A}}_{i,j,k-1}(\mathbf{u}), \tag{2.2}$$

where all the terms with negative indices are equal to zero, are called the *normalized B-weights over* $\mathcal{A}$. □

Definition 2.3 (B-Patches).
Let $\mathcal{A}$ be a knot arrangement associated with a triangle $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$, and let $\mathcal{F} : \mathbb{R}^2 \to \mathbb{R}^t$ be a polynomial. The representation

$$\mathcal{F}(\mathbf{u}) = \sum_{i+j+k=n} B^{\mathcal{A}}_{ijk}(\mathbf{u}) \mathbf{c}_{ijk} \tag{2.3}$$

is called B-patch representation of $\mathcal{F}$ over $\mathcal{A}$. The restriction of $\mathcal{F}$ to $\triangle$ is called a *B-patch*. The coefficients $\mathbf{c}_{ijk} \in \mathbb{R}^t$ are called *B-patch control points*. They form the *B-patch control net*. □

Theorem 2.1.
Let $\mathcal{A}$ be a knot arrangement associated with a triangle $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$. Then every polynomial $\mathcal{F} : \mathbb{R}^2 \to \mathbb{R}^t$ can be represented as B-patch over $\mathcal{A}$. Moreover, the B-patch control points are given as

$$c_{ijk} = f(\mathbf{t}_{0,0} \dots \mathbf{t}_{0,i-1}\mathbf{t}_{1,0} \dots \mathbf{t}_{1,j-1}\mathbf{t}_{2,0} \dots \mathbf{t}_{2,k-1}) \tag{2.4}$$

where f is the multiaffine polar form of $\mathcal{F}$.

Proof ▶ Let $\mathcal{F} : \mathbb{R}^2 \to \mathbb{R}^t$ be given, and let f be its multiaffine polar form. Define

$$c_{ijk}^l(\mathbf{u}) = f(\mathbf{t}_{0,0} \dots \mathbf{t}_{0,i-1}\mathbf{t}_{1,0} \dots \mathbf{t}_{1,j-1}\mathbf{t}_{2,0} \dots \mathbf{t}_{2,k-1}\mathbf{u}^l),\ i+j+k+l=n$$

We show that

$$\mathcal{F}(\mathbf{u}) = \sum_{i+j+k+l=n} B_{ijk}^{\mathcal{A}}(\mathbf{u}) c_{ijk}^l(\mathbf{u})$$

for $l = \overline{0,n}$. We use induction on $v = n-l$. The case $v = 0$ is trivial. Suppose that our assertion holds for $v = n - l$. Representing $\mathbf{u} = \lambda_{ijk,0}(\mathbf{u})\mathbf{t}_{0,i} + \lambda_{ijk,1}(\mathbf{u})\mathbf{t}_{1,j} + \lambda_{ijk,2}(\mathbf{u})\mathbf{t}_{2,k}$ in barycentric coordinates w.r.t. $\triangle_{ijk} = [\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k}]$ we obtain

$$c_{ijk}^l(\mathbf{u}) = \lambda_{ijk,0}(\mathbf{u}) c_{i+1,j,k}^{l-1} + \lambda_{ijk,1}(\mathbf{u}) c_{i,j+1,k}^{l-1} + \lambda_{ijk,2}(\mathbf{u}) c_{i,j,k+1}^{l-1},$$

and using the induction hypothesis we get

$$\begin{aligned}
\mathcal{F}(\mathbf{u}) &= \sum_{i+j+k+l=n} B_{ijk}^{\mathcal{A}}(\mathbf{u}) c_{ijk}^l(\mathbf{u}) \\
&= \sum_{i+j+k+l-1=n} (\lambda_{i-1,j,k,0}(\mathbf{u}) B_{i-1,j,k}^{\mathcal{A}}(\mathbf{u}) + \lambda_{i,j-1,k,1}(\mathbf{u}) B_{i,j-1,k}^{\mathcal{A}}(\mathbf{u}) + \\
&\qquad\qquad + \lambda_{i,j,k-1,2}(\mathbf{u}) B_{i,j,k-1}^{\mathcal{A}}(\mathbf{u})) c_{ijk}^{l-1}(\mathbf{u}) \\
&= \sum_{i+j+k+l-1=n} B_{ijk}^{\mathcal{A}}(\mathbf{u}) c_{ijk}^{l-1}(\mathbf{u})
\end{aligned}$$

where the last step follows from from 2.2. This proves our claim. In the special case $l = 0$ we thus obtain $\mathcal{F}(\mathbf{u}) = \sum B_{ijk}^{\mathcal{A}}(\mathbf{u}) c_{ijk}$, and the proof of Theorem 2.1 is complete. ◀

Remark 2.1 (Linear Independence).
In particular, Theorem 2.1 shows, that every real valued polynomial can be written as linear combination of the normalized B-weights $B_{ijk}^{\mathcal{A}}(\mathbf{u})$. Hence, the normalized B-weights $B_{ijk}^{\mathcal{A}}(\mathbf{u}),\ i+j+k=n$, represent a basis of the space $\prod^n(\mathbb{R}^2)$ of real valued polynomials up to degree n on $\mathbb{R}^2$. □

The following result is sometimes useful for representing the graph of a real-valued polynomial in B-patch form:

Corollary 2.1 (Greville Abscissae).
The B-patch control points of the polynomial $\mathcal{F}(\mathbf{u}) = \mathbf{u}$ over a knot arrangement $\mathcal{A}$ are given as

$$c_{ijk} = \frac{1}{n}(\mathbf{t}_{0,0} + \cdots + \mathbf{t}_{0,i-1} + \mathbf{t}_{1,0} + \cdots + \mathbf{t}_{1,j-1} + \mathbf{t}_{2,0} + \cdots + \mathbf{t}_{2,k-1}) \tag{2.5}$$

Proof ▶ The multiaffine blossom f of $\mathcal{F}$ is given by $f(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \frac{1}{n}(\mathbf{u}_1 + \cdots + \mathbf{u}_n)$, and Theorem 2.1 yields $c_{ijk} = f(\mathbf{t}_{0,0} \ldots \mathbf{t}_{2,k-1}) = \frac{1}{n}(\mathbf{t}_{0,0} + \cdots + \mathbf{t}_{2,k-1})$. ◀

Remark 2.2 (Multiple Knots).
Suppose that all knots $\mathbf{t}_{0,0} = \mathbf{t}_{0,n-1} = \mathbf{t}_0$, $\mathbf{t}_{1,0} = \mathbf{t}_{1,n-1} = \mathbf{t}_0$, and $\mathbf{t}_{2,0} = \mathbf{t}_{2,n-1} = \mathbf{t}_0$ appear with multiplicity n. Then the recurrence for the normalized B-weights specializes to the recurrence for the Bernstein polynomials, and the B-patch representation specializes to the Bézier representation. □

The previous remark shows, that B-patches subsume triangular Bézier patches in much the same way as the segments of a B-spline curve subsume Bézier curves.

Theorem 2.2 (Properties).
Let $F(\mathbf{u}) = \sum B^{\mathcal{A}}_{ijk}(\mathbf{u})\mathbf{c}_{ijk}$ be a B-patch over $\mathcal{A}$. Then the shape of the surface is related to the shape of its control net as follows:

- **Convex Hull Property:** Suppose that $\triangle_{000} \subseteq \triangle_{ijk},\ \forall i+j+k \leq (n-1)$. If $\mathbf{u} \in \triangle_{000}$, then $F(\mathbf{u})$ lies inside the closed convex hull of the control net $\{\mathbf{c}_{ijk} \mid i+j+k = n\}$.
- **Corner Vertex Interpolation and Tangency at Multiple Knots:** If n knots $\mathbf{t}_{0,0} = \cdots = \mathbf{t}_{0,n-1} = \mathbf{t}_0$ coincide, then $F(\mathbf{t}_0) = \mathbf{b}_{n00}$, and F is tangent to its control net at this point.
- **Affine Invariance:** The relationship between the surface F and its control net is affinely invariant: If $\phi : \mathbb{R}^t \to \mathbb{R}^t$ is an affine map, then the B-patch control points of the image surface $\bar{F} = F \circ \phi$ are the images $\phi(\mathbf{c}_{ijk})$ of the B-patch control points $\mathbf{c}_{ijk}$.

2.2. The B-Patch Computation

In this subsection we introduce an algorithm, which computes B-patch vertices.

input	B-patch control net $C[i,j,k]$ B-patch knot arrangement $t[i,j,k]$ 2D vertex $u,\ u \in \triangle$
output	vertex v

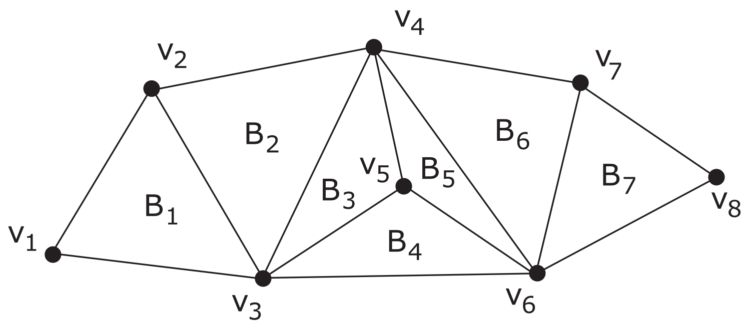
Algorithm 2.1 (The B-Patch Computation).

```
 1  SetBarycentricCoordinatesToZero(L[i,j,k,d]);
 2  for i:=0 to n-1do
 3   for j:=0 to n-1 do
 4    for k:=0 to n-1 do
 5     if i+j+k<=n-1 then ComputeBarycentricCoordinates(L[i,j,k,d],u);
 6  SetBToZero(B[i,j,k]);
 7  B[0,0,0]:=1;
 8  v:=[0,...,0];
 9  for i:=1 to n do
10   for j:=i+1 downto 1 do
11    for k:=j-1 downto 0 do
12     begin
13      B[i+1-j,j-1-k,k]:=L[i+1-j-1,j-1-k  ,k  ,0]*B[i+1-j-1,j-1-k  ,k  ]+
                          L[i+1-j  ,j-1-k-1,k  ,1]*B[i+1-j  ,j-1-k-1,k  ]+
                          L[i+1-j  ,j-1-k  ,k-1,2]*B[i+1-j  ,j-1-k  ,k-1];
14      if i=n then v:=v+B[i+1-j,j-1-k,k]*C[i+1-j,j-1-k,k];
15     end;
```

$\mathcal{O}(n^3)$ time and space complexity immediately follows from the definitions 2.1, 2.2. Computing v vertices implies $\mathcal{O}(n^3 v)$ time complexity. In lines $\overline{1,5}$ barycentric coordinates are computed, in lines $\overline{6,15}$ B-weights are computed. The main cycle for B-weights computation (line 9) running n times represents the depth of the recursive computation by definition 2.2. Note, that introduced algorithm is based on a dynamic programming paradigm.

2.3. The B-Patch Optimization

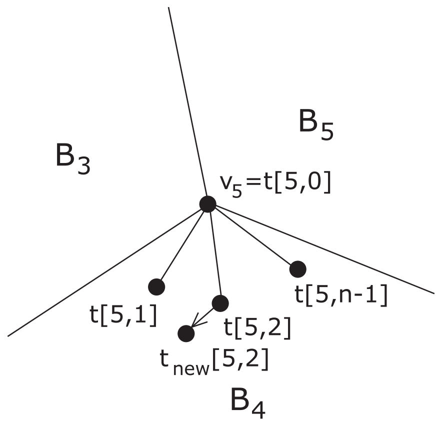
Assume a B-patch net $\mathcal{B} = \{B_1, \ldots, B_t\}$ over triangulation T in $\mathbb{R}^2$, where t is number of triangles in T. Note that all B-patches are about the same degree since adjacent patches share their knots.



In the B-patch implementation, there rise questions about knot vertices manipulation and B-patch re-computation.

We can define a BPatchKnotIndex and KnotBpatchIndex structures. BPatchKnotIndex gives an information, to which vertices of the triangulation given B-patch corresponds. KnotBPatchIndex shows, which B-patches correspond to given triangulation vertex. KnotBPatchIndex can be represented as binary balanced tree.

Assume a concrete vertex $\mathbf{v}$ in triangulation T. According to the B-patch definition there are knots $\mathbf{t}_0 \ldots \mathbf{t}_{n-1}$ corresponding to the vertex $\mathbf{v}$.



Barycentric coordinates computation time for one B-patch and one point is $\mathcal{O}(n^3)$. There are corresponding n^2 *knot-triangles* with each of n knots . When manipulating one of knots $\mathbf{t}_1 \ldots \mathbf{t}_{n-1}$, it is sufficient to recompute $\mathcal{O}(n^2 t')$ values of barycentric coordinates, where n is common B-patch degree and t' is number of triangles corresponding to point v. This result is relevant in computation for small n, that means, when computing a vertex, we also need to compute $B_{[i,j,k]}$ values and this computation time is $\mathcal{O}(n^3 t')$.

When manipulating knot $\mathbf{t}_0$, we can allow designer to move knots $\mathbf{t}_1 \ldots \mathbf{t}_{n-1}$ in the same direction. In this case, computing goes slower, barycentric coordinates computation and vertex computation complexity is $\mathcal{O}(n^3 t')$.

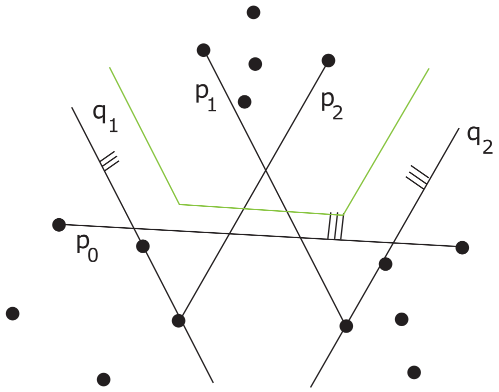
In this section we have introduced simple optimizations. The B-patch optimalization is very important for implementation. There are possibilities, we can quickly implement an unoptimized B-patches or we can spend a lot of time to implemet perfect optimized structures and algorithms. The more optmized the structures and algorithms are, the more complex objects and surfaces we can model.

2.4. The B-Patch Checking

2.4.1. Algorithm with complexity $O(n \log n)$

When manipulating knots, desirable effort is to avoid unstable patches. When changing knots coordinates, we have to verify the B-patch definition conditions. Verification time is $\mathcal{O}(n^2 t')$ for one knot and t' corresponding triangles as developed in section (2.3).

We describe an useful approximation running in $\mathcal{O}(nt' \ln n)$. All the idea behind is shown in the following figure:



Our goal is to create lines p_0, q_1, q_2. Assume knot $t_{[0,r]}$ is manipulated for some r. Knots sets are defined as $T_j = \{t_{[j,i]}, i = \overline{0, n-1}\}, j = \overline{0,2}$. We assign a line p_j to each set $T_j, j = \overline{0,2}$.

The first step of line p_j construction is computation of convex hull $H_j = [h_{j,0}, h_{j,1}, \ldots$ $\ldots, h_{j,q}]$ containing vertices $(\bigcup_{i=0}^{2} T_i) - T_j$ where notation $[h_{j,0}, h_{j,1}, \ldots, h_{j,q}]$ means ordered n-tuple. Note that $q \leq 2n-1$.

For each H_j there exist two pairs $[h_{j,a_{00}}, h_{j,a_{01}}], [h_{j,a_{10}}, h_{j,a_{11}}]$ of four different vertices satisfying condition

$$(a_{00} = a_{01} - 1 \ \vee \ (a_{00} = 0 \ \wedge \ a_{01} = q)) \ \wedge$$

$$\wedge \ (a_{10} = a_{11} - 1 \ \vee \ (a_{10} = 0 \ \wedge \ a_{11} = q)) \ \wedge$$

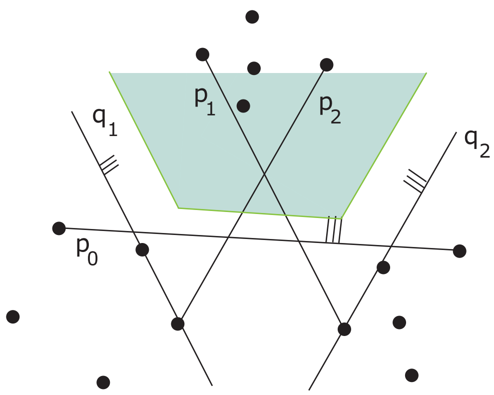
$$\wedge \ (\{h_{j,a_{00}}, h_{j,a_{11}}\} \subset T_a) \ \wedge \ (\{h_{j,a_{01}}, h_{j,a_{10}}\} \subset T_b)$$

for some a, b where $a \neq b$, $j \neq a$, $j \neq b$. Note that $\{a, b, j\} = \{0, 1, 2\}$. Two pairs of vertices $[h_{j,a_{00}}, h_{j,a_{01}}], [h_{j,a_{10}}, h_{j,a_{11}}]$ define two lines $p_{j,0}$, $p_{j,1}$. The second step is to decide which one of them defines the line p_j. Intuitive p_j is the line farther from the set T_j. This we define correctly in the following. We 'randomly' select line $p_{j,0}$. If

$$|d(h_{j,a_{00}}, h_{j,a_{01}}, h_{j,a_{11}})| = |d(h_{j,a_{00}}, h_{j,a_{01}}, t_{j,0})|$$

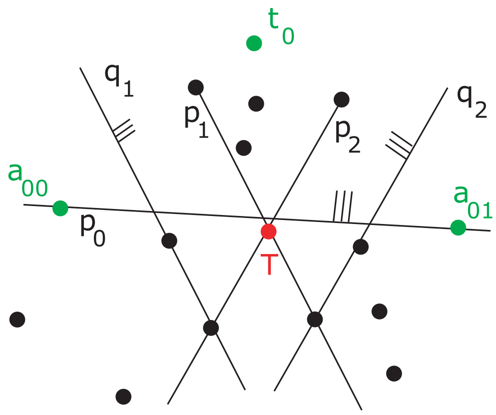
then $p_j = [h_{j,a_{10}}, h_{j,a_{11}}]$ otherwise $p_j = [h_{j,a_{00}}, h_{j,a_{01}}]$. Further we assign lines q_1, q_2 to lines p_1, p_2. We take the nearest vertex $t_{[i,nearest]}$ from T_i to the line p_i. Line q_i is parallel to p_i and passes through vertex $t_{[i,nearest]}$ where $i \in \{1, 2\}$.

Lines p_0, q_1, q_2 form α-region. Symbol α in the term α-region represents parameter that influences the size of the α-region. Algorithmic-defined α-region is a region where designer can move knots $t_{[0,i]}$ for one B-patch and this B-patch stays stable. By stable we mean the B-patch is not stretching to infinity. Computational complexity of convex hull is $\mathcal{O}(n \ln n)$ and the same the complexity of sorting. From this fact follows the complexity $\mathcal{O}(nt' \ln n)$ of α-region computation, where t' is number of B-patches corresponding to the manipulating knot $t_{[0,r]}$. Next we can define sensitivity parameter, which makes α-region closer to the knot set. This is illustrated in the following figure by green lines.



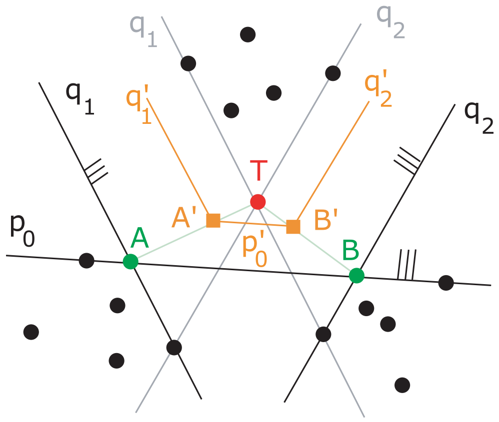
It is recommended that all knots $t_{[0,i]}$, $i = \overline{1, n-1}$ are in ε-neighbourhood of knot $t_{[0,0]}$. We will define ε more precisely.

First, we create point of intersection T, where $T = p_1 \cap p_2$. Next, if $|d(a_{00}, a_{01}, t_0)| \neq$ $\neq |d(a_{00}, a_{01}, T)|$ than there is no possible reduction of α-region. The following figure demonstrates this case:



Now we set $A = q_1 \cap p_0$, $B = q_2 \cap p_0$ and we choose value of ε from interval $[0, 1]$. We divide lines AT and BT by the ratio $\varepsilon : (1-\varepsilon)$, and we create points A' and B'. Formally: $|A'T| = \varepsilon|AT|$, $|AA'| = (1-\varepsilon)|AT|$, $|B'T| = \varepsilon|BT|$, $|BB'| = (1-\varepsilon)|BT|$.

Using points A' and B' we create lines q_1', q_2', p_0' as shown in the following figure:



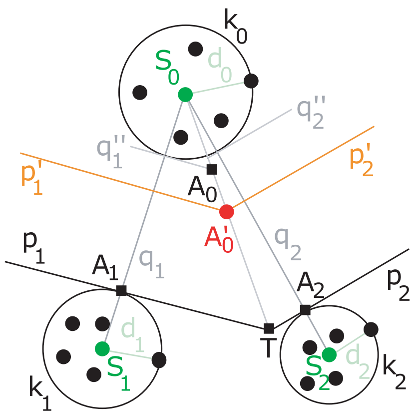
Formally: $q_1'||q_1$, $q_2'||q_2$, $p_0'||p_0$, $A' = q_1' \cap p_0'$, $B' = q_2' \cap p_0$. Lines q_1', q_2' and p_0' form the reduced α-region. Note that: if $\varepsilon = 1$ than the α-region is minimized, if $\varepsilon = 0$ than the α-region is maximized.

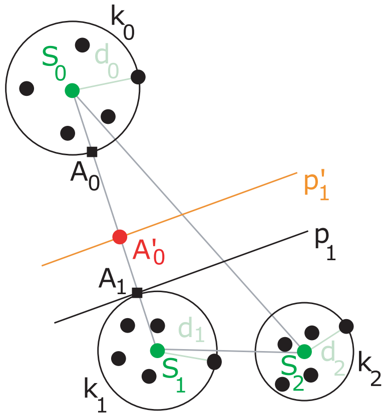
2.4.2. Algorithm with complexity $O(n)$

Again, assume knot $t_{[0,r]}$ is manipulated for some r. Knots sets are defined as $T_j = \{t_{[j,i]}, i = \overline{0, n-1}\}, j = \overline{0, 2}$.

For sets $T_j, j = \overline{0, 2}$ we compute points S_j. The point S_j is the center of all points $t_{[j,i]}, i = \overline{0, n-1}\}, j = \overline{0, 2}$. For every j we compute maximum distance d_j, $d_j = \max_{i=\overline{0,n-1}}\{d(S, t_{[j,i]})\}$. Then we create circles $k_j = k(S_j, d_j)$.

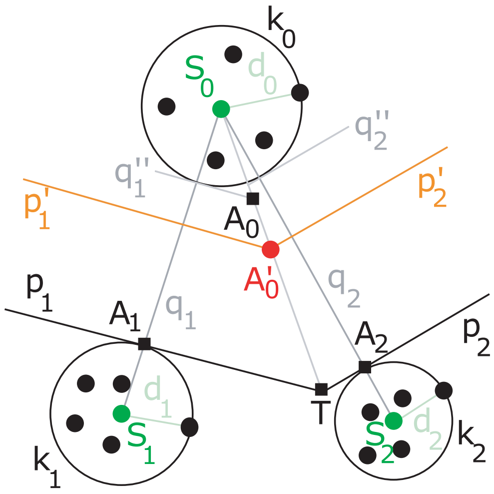
Let $A_1 = S_0S_1 \cap k_1$, $A_2 = S_0S_2 \cap k_2$, $A_1 \in p_1$, $A_2 \in p_2$, $|\measuredangle S_0S_1p_1| = 90^o$, $|\measuredangle S_0S_2p_2| = 90^o$, $T = p_1 \cap p_2$. The α-region is defined by lines p_1 and p_2. There are two cases. The case: triangle $S_0S_1S_2$ is acute-angled or the size of the angle $S_1S_0S_2 > 90^o$. The case: the size of the angle $S_0S_1S_2 > 90^o$ or the size of the angle $S_0S_2S_1 > 90^o$. These cases are illustrated in the following figures:





In the case 2, the α-region is formed by S_0 and the line p_1 or p_2 depending on the location of the obtuse angle.

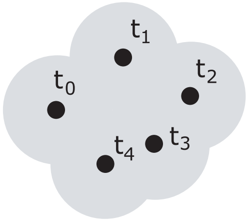
If we have defined stable B-patch then $T \notin k_0$. Otherwise we cannot reduce our α-region. As in the first method, we will select the value of ε. Point is either $A_0 = S_0T \cap q_1''$ or $A_0 = S_0T \cap q_2''$ (depends on which point of intersection is closer to the point T). By the ratio $\varepsilon : (1-\varepsilon)$ we divide the line A_0T. Formally: $|A_0'A_0| = \varepsilon|A_0T|$, $|A_0'T| = (1-\varepsilon)|A_0T|$. By the point A_0' we create lines p_1' and p_2' as showed in the next figure:



Lines p_1' and p_2' form the reduced α-region. Note that: if $\varepsilon = 0$ than the α-region is minimized, if $\varepsilon = 1$ than the α-region is maximized.

2.4.3. Algorithm with complexity $O(1)$

This method is simple, usually it is sufficient. What we do is we restrict a knot move into particular distance. The complexity is then $O(1)$. The following figure pictures this situation:



2.4.4. The B-Patch Checking Summary

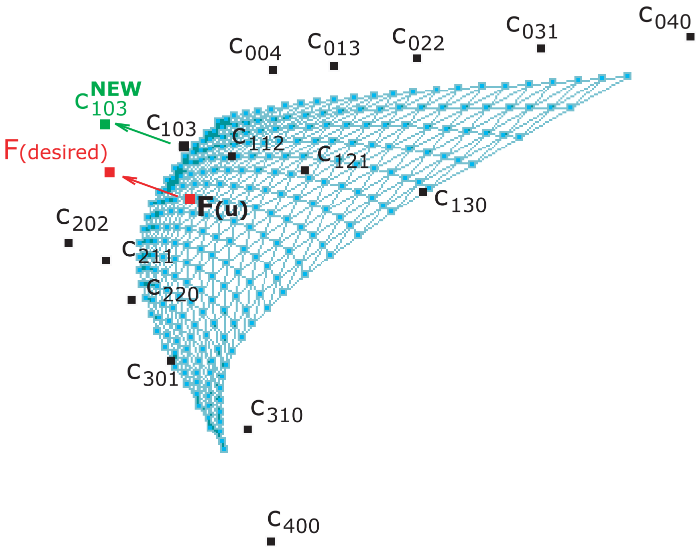
We have discussed three methods of creating α-region, a region where we can move our knots. The complexities of these methods are $O(n \log n)$, $O(n)$ and $O(1)$. The reliability of first two methods is guaranteed using $\varepsilon < \frac{3}{4}$. That means, there exist sufficient boundary among knots and the B-patch stays stable. Otherwise, the actual B-patch is starting to stretch to infinity and forms undesirable surface. The reliability of the third method depends on designer.

2.5. The B-Patch Adaptation

2.5.1. Exact point

For a specific B-patch $\mathcal{F}$ and particular point $\mathbf{u}$ we get a projected point $\mathcal{F}(\mathbf{u})$. In this section we will discuss situation where the desired point $\mathcal{F}(\mathbf{u}) = \mathbf{v}$. We will discuss two situations: first, we will change one control point $\mathbf{c}_{[i,j,k]}$, and second, we will change more control points $\mathbf{c}_{[x,y,z]}$.

The first situation is showed in this picture:



Using the B-patch definition (2.3) we can use the following solution:

$$\mathbf{c}_{xyz} = \left(\mathcal{F}(\mathbf{desired}) - \sum_{i+j+k=n, <ijk> \neq <xyz>} \mathbf{c}_{ijk} B_{ijk}(\mathbf{actual})\right) / B_{xyz}(\mathbf{actual}) \tag{2.6}$$

In the second situation we will move more than one control point. Control point $\mathbf{c}_{xyz}$ has at most six neighbours $neighbour_j(\mathbf{c}_{xyz}), j = \overline{0,5}$.

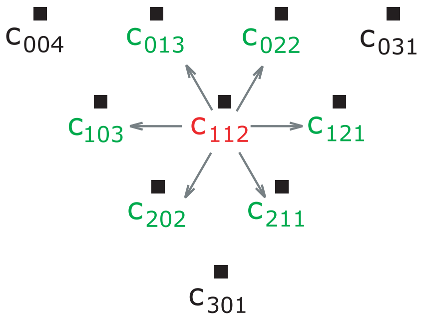
$neighbour_0(\mathbf{c}_{xyz})$	$=$	$\mathbf{c}_{x(y-1)(z+1)}$	$neighbour_1(\mathbf{c}_{xyz})$	$=$	$\mathbf{c}_{x(y+1)(z-1)}$
$neighbour_2(\mathbf{c}_{xyz})$	$=$	$\mathbf{c}_{(x-1)y(z+1)}$	$neighbour_3(\mathbf{c}_{xyz})$	$=$	$\mathbf{c}_{(x+1)y(z-1)}$
$neighbour_4(\mathbf{c}_{xyz})$	$=$	$\mathbf{c}_{(x-1)(y+1)z}$	$neighbour_5(\mathbf{c}_{xyz})$	$=$	$\mathbf{c}_{(x+1)(y-1)z}$

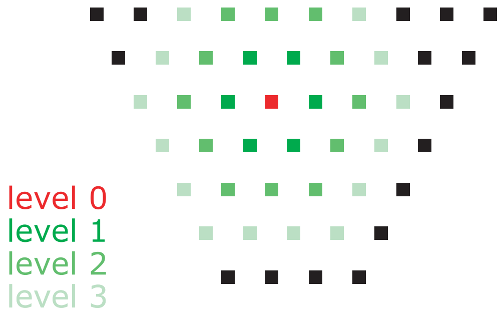
This definition is valid only if $x > 0 \wedge x < n \wedge y > 0 \wedge y < n \wedge z > 0 \wedge z < n$. Otherwise the corresponding set may be empty. Example: $neighbour_4(\mathbf{c}_{032}) = \emptyset$. Direct neighbours of point $\mathbf{c}_{xyz}$ are *level* 1 neighbours. Formally:

$$N_{-1}(\mathbf{c}_{xyz}) = \emptyset \qquad N_0(\mathbf{c}_{xyz}) = \{\mathbf{c}_{xyz}\}$$

$$N_i(\mathbf{c}_{xyz}) = \left(\bigcup_{\forall \mathbf{c} \in N_{i-1}, j \in \overline{0,5}} neighbour_j(\mathbf{c}) \right) - N_{i-1}(\mathbf{c}_{xyz}) - N_{i-2}(\mathbf{c}_{xyz}), \;\; i > 0$$

Note that $\forall \mathbf{c}, \forall i > 0 : \mathbf{c}_{xyz}, \; |N_i(\mathbf{c}_{xyz})| \leq 6i$. The following figures demonstrate these concepts:





Now the question is how to move more than one control vertex $\mathbf{c}$. For example we need to move selected control vertex $\mathbf{c}_{xyz}$ and corresponding levels 1 and 2. First we use *weight function* $\mathcal{W}$. Let m be the maximal level involved. In this case $m = 2$. Control vertex $\mathbf{c}_{xyz}$ and control vertices in sets $N_1(\mathbf{c}_{xyz})$, $N_2(\mathbf{c}_{xyz})$ have to influence the B-patch $\mathcal{F}$ in a way that $\mathcal{F}(u) = v$.

Definition 2.4 (Weight function set $\mathcal{W}$).
Let m be the maximal desired level of control points to be moved. The set of functions $\mathcal{W} =$ $= \{w_i, i = \overline{0, m}\}$ where $w_i : i \to (0, 1) \subset \mathbb{R}$ is called *weight function set* iff $\sum_{i=\overline{0,m}} w_i = 1$.
□

Weight function set represents the influential power of control vertices. Mostly it is sufficient to use $\forall i, w_i = \frac{1}{m+1}$. In our example ($m = 2$) we need to solve this equation:

$$\sum_{\substack{i+j+k=n, <ijk> \neq <xyz> \\ \mathbf{c}_{ijk} \notin N_1(\mathbf{c}_{xyz}) \cup N_2(\mathbf{c}_{xyz})}} \mathbf{c}_{ijk} B_{ijk}(\mathbf{u}) + \sum_{l=\overline{0,2}} \frac{w_l}{|N_l(\mathbf{c}_{xyz})|} \sum_{\mathbf{c}_{ijk} \in N_l(\mathbf{c}_{xyz})} \mathbf{c}_{ijk} B_{ijk}(\mathbf{u}) = \mathcal{F}(\mathbf{desired}) \tag{2.7}$$

We will systematically recompute desired control vertices. Next we will prove that the sum using new control vertices is equal to the desired value. First we will compute new control vertex $\mathbf{c}'_{xyz}$ by the following formula:

$$\mathbf{c}'_{xyz} = w_0 \left(\mathcal{F}(\mathbf{desired}) - \sum_{i+j+k=n, <ijk> \neq <xyz>} \mathbf{c}_{ijk} B_{ijk}(\mathbf{u}) \right) / B_{xyz}(\mathbf{u}) + (1 - w_0) \mathbf{c}_{xyz} \tag{2.8}$$

Suppose that the control vertices in levels $0, 1, \ldots, p$ are already recomputed. Now we proceed to level $p+1$. Let $q_i = |N_i(\mathbf{c}_{xyz})|$. We will recompute first control vertex $\mathbf{c}_{abc} \in$ $\in N_{p+1}(\mathbf{c}_{xyz})$.

$$\mathbf{c}'_{abc} = \frac{w_{p+1}}{q_{p+1}}\Big(\mathcal{F}(\mathbf{desired}) - \sum_{i+j+k=n, <ijk> \neq <abc>} \mathbf{c}_{ijk}B_{ijk}(\mathbf{u})\Big)/B_{abc}(\mathbf{u}) + \Big(1 - \frac{w_{p+1}}{q_{p+1}}\Big)\mathbf{c}_{abc} \tag{2.9}$$

Suppose that s of q control vertices in level $p+1$ are already computed. Now we proceed to control vertex $s+1$. For a new vertex we use previous formula (2.9).

Using this algorithmic recipe we distribute the influence of desired control vertices via the weight function. Now it is necessary to prove the "checksum":

$$\sum_{i+j+k=n \wedge \mathbf{c}_{ijk} \notin N_0(\mathbf{c}_{xyz}) \cup \cdots \cup N_m(\mathbf{c}_{xyz})} \mathbf{c}_{ijk}B_{ijk} + \sum_{i+j+k=n \wedge c_{ijk} \in N_0(\mathbf{c}_{xyz}) \cup \cdots \cup N_m(\mathbf{c}_{xyz})} \mathbf{c}'_{ijk}B_{ijk} = \mathcal{F}(\mathbf{desired}) \tag{2.10}$$

▶ For a selected level p and selected control vertex $\mathbf{c}_{abc}$ we get:

$$\mathbf{c}'_{abc} = \frac{w_p}{q_p}\Big(\mathcal{F}(\mathbf{desired}) - \sum_{i+j+k=n, <ijk> \neq <abc>} \mathbf{c}_{ijk}B_{ijk}(\mathbf{u})\Big)/B_{abc}(\mathbf{u}) + \Big(1 - \frac{w_p}{q_p}\Big)\mathbf{c}_{abc} =$$

$$= \frac{w_p}{q_p}\Big(\mathcal{F}(\mathbf{desired}) - \sum_{i+j+k=n \wedge \mathbf{c}_{ijk} \notin N_0(\mathbf{c}_{xyz}) \cup \cdots \cup N_m(\mathbf{c}_{xyz})} \mathbf{c}_{ijk}B_{ijk}(\mathbf{u}) -$$

$$- \sum_{i+j+k=n \wedge \mathbf{c}_{ijk} \in N_0(\mathbf{c}_{xyz}) \cup \cdots \cup N_m(\mathbf{c}_{xyz})} \mathbf{c}_{ijk}B_{ijk}(\mathbf{u}) + \mathbf{c}_{abc}B_{abc}(\mathbf{u})\Big)/B_{abc}(\mathbf{u}) + \Big(1 - \frac{w_p}{q_p}\Big)\mathbf{c}_{abc} \tag{2.11}$$

By applying (2.11) on (2.10) we have prooved the selection correctness of formula (2.9) and the fact that the sum using new control vertices is equal to the desired value. The proof completion is an exercise in algebra. ◀

Instead of typical weight functions $w_i = \frac{1}{m+1}$ we can use other functions. This depends on the requirements emerged in practice. It is useful to use for example functions $w_i = \frac{1}{2^{i+1}}$, $i = \overline{0, m}$, $w_{m+1} = \frac{1}{2^{m+1}}$. It is easily seen that the more far the influenced control vertices are, the less important their weight is.

2.5.2. More points

In previous subsection we have discussed situation in which we required $\mathcal{F}(u) = v$. In this subsection we will require more contact points. Formally: $\mathcal{F}(u_i) = v_i,\ i = \overline{0,d}$. We will use notions developed in previous subsection and especially formula (2.9). To satisfy $\mathcal{F}(u_i) = v_i,\ i = \overline{0,d}$ we must move at least $d+1$ control vertices.

We have to solve a system of equations:

$$\mathcal{F}(\mathbf{desired_m}) = \sum_{i+j+k=n \wedge <i,j,k> \neq <a_{l,i},a_{l,j},a_{l,k}>, l=\overline{0,d}} \mathbf{c}_{ijk} B_{ijk}(\mathbf{u}_m) + \sum_{l=\overline{0,d}} \mathbf{c}_{a_{l,i},a_{l,j},a_{l,k}} B_{a_{l,i},a_{l,j},a_{l,k}}(\mathbf{u}_m) \tag{2.12}$$

where $m = \overline{0,d}$.

The complexity of the solution of the system of n linear equations is generally $O(n^3)$. This problem is reducible to the problem of matrices multiplication. However, this complexity could be reduced more and the last known result to year 1990 was complexity $O(n^{2.34})$. Generally a hypothesis is accepted that for every ε there exist and algorithm for matrices multiplication with complexity $O(n^{2+\varepsilon})$. For more information see [5].

The operation of vertices movement is necessary for design. It is usual operation and it is considered to be one of the basic operations with surfaces. While we are studying new surface scheme, it is important to take the operation of vertices movement into account.

Chapter 3

Bivariate B-Splines

A single polynomial surface is nice, but is naturally not sufficient for any serious applications. What we need are piecewise polynomial surfaces instead. First thing we would like to avoid is stitching together the individual pieces by hand. What we are looking for are blending functions that produce smooth piecewise polynomial surfaces automatically. Such blending functions for piecewise polynomial surfaces over arbitrary triangulations can be obtained using *bivariate B-splines.*

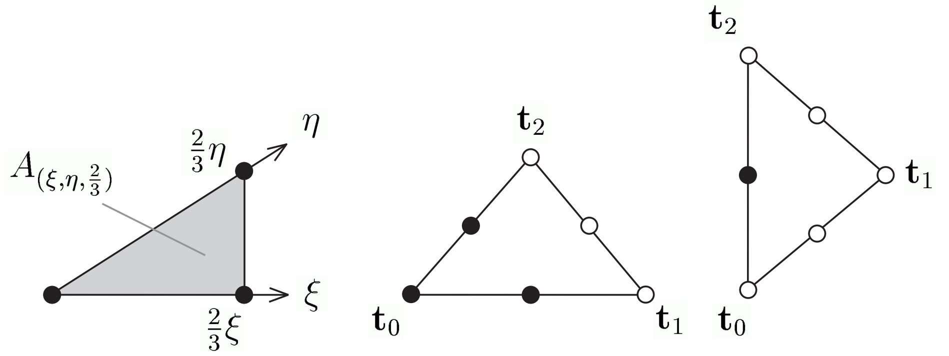
3.1. Bivariate B-Splines Definition

Bivariate B-splines can be defined in a variety of different ways. Their original geometric definition is obtained by projecting a high-dimensional simplex into the plane and by taking the volume of the fibre over each point. For this reason they are also called *simplex splines.* Another possible definitions is by means of reccurence. Although this definition runs into troubles for points on the knot lines, these difficulties can easily be overcome by a slight modification of the area of support for the piecewise constant B-splines. This phenomenon is completely analogous to the univariate case, where the piecewise constant B-spline blending functions are non-zero on the half-open interval $[t_i, t_{i+1})$ instead of closed interval $[t_i, t_{i+1}]$. This observation motivates to the following definition:

Definition 3.1 (Half-Open Convex Hull).
Given $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^2$, we denote by $\triangle[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$ their closed convex hull, and by $\triangle^o =$ $=$int $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$ its interior. The *half-open convex hull* $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$ is then defined as follows: Let ξ be the horizontal unit vector in $\mathbb{R}^2$. A point $\mathbf{u} \in \mathbb{R}^2$ belongs to the half-open convex hull $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$ if and only if there exists a vector η with positive slope and a positive scalar $\varepsilon > 0$, such that the set

$$A_{(\xi,\eta,\varepsilon)}(\mathbf{u}) = \{s\xi + t\eta \mid 0 < s, t, 0 < s + t < \varepsilon\} \tag{3.1}$$

is completely contained in $\triangle^o$. Note that this definition implies $\triangle^o \subset [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2) \subset \triangle$. The definition of the set $A_{(\xi,\eta,\varepsilon)}(\mathbf{u})$ and the definition of the half-open convex hull is illustrated by the following figure: □



After the definition of the half-open convex hull in place we are ready to define bivariate B-splines by means of the following recurrence relation:

Definition 3.2 (Bivariate B-Splines).
Let $V = \{\mathbf{t}_0, \ldots, \mathbf{t}_m\}$ be a finite set of points in $\mathbb{R}^2$.

The *bivariate B-spline* $M(\mathbf{u}|V) = M(\mathbf{u}|\mathbf{t}_0, \ldots, \mathbf{t}_m)$ is defined recursively as follows: For $V = \{\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2\}$ we set

$$M(\mathbf{u}|\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2) = \frac{\chi_{[\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2)}}{|d(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)|}, \tag{3.2}$$

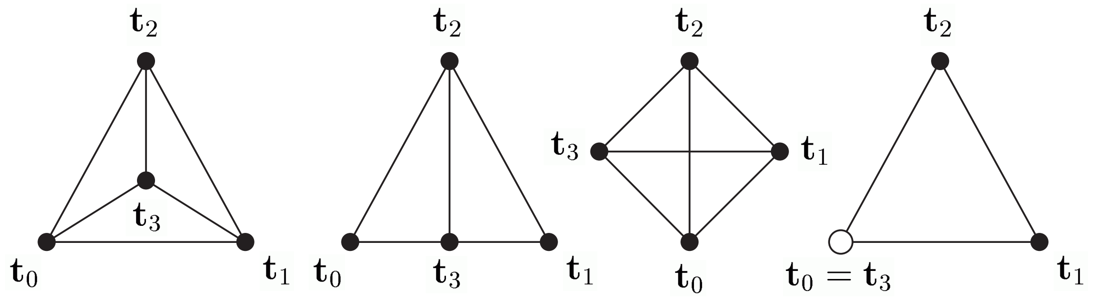
where $\chi_{[\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2)}$ is the characteristic function on the half-open convex hull $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$.

For $V = \{\mathbf{t}_0, \ldots, \mathbf{t}_m\}, m > 2$, we set

$$M(\mathbf{u}|\mathbf{t}_0, \ldots, \mathbf{t}_m) = \sum_{d=0}^{2} \lambda_{i_d}(\mathbf{u}) M(\mathbf{u}|\mathbf{t}_0, \ldots, \widehat{\mathbf{t}_{i_d}}, \ldots, \mathbf{t}_m) \tag{3.3}$$

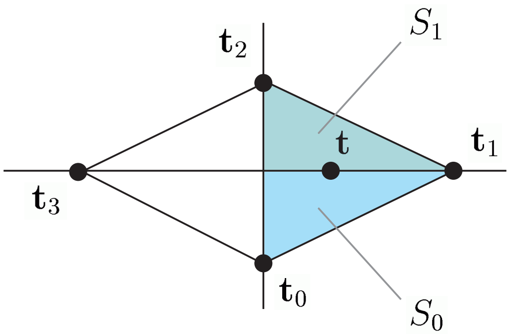
where $W = \{\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}\}$ is any subset of affinely independent points in V, and $\mathbf{u} = \sum_{d=0}^{2} \lambda_{i_d}(\mathbf{u})\mathbf{t}_{i_d}$ is the representation of $\mathbf{u}$ in barycentric coordinates w.r.t. W. In this situation, the points $\mathbf{t}_0, \ldots, \mathbf{t}_m$ are called *knots*. □

In the following figure we show the four essentially different knot configurations for a linear B-spline $M(\mathbf{u}|\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)$.



The following example will help us to clarify our concepts:

Example 3.1 (Linear B-Splines).



Let $\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3 \in \mathbb{R}^2$ be given according to above figure. We consider evaluation of the linear B-spline $M(\mathbf{u}|\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)$. Setting $W = \{\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2\}$, and using (3.3), we get

$$M(\mathbf{u}|\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3) = \lambda_0(\mathbf{u})M(\mathbf{u}|\mathbf{t}_1\mathbf{t}_2\mathbf{t}_3) + \lambda_1(\mathbf{u})M(\mathbf{u}|\mathbf{t}_0\mathbf{t}_2\mathbf{t}_3) + \lambda_2(\mathbf{u})M(\mathbf{u}|\mathbf{t}_0\mathbf{t}_1\mathbf{t}_3).$$

We first consider $\mathbf{u}_0 \in int\ S_0$. Then $M(\mathbf{u}_0|\mathbf{t}_1\mathbf{t}_2\mathbf{t}_3) = M(\mathbf{u}_0|\mathbf{t}_0\mathbf{t}_2\mathbf{t}_3) = 0$, and we get

$$M(\mathbf{u}_0|\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3) = \lambda_2(\mathbf{u}_0)M(\mathbf{u}_0|\mathbf{t}_0\mathbf{t}_1\mathbf{t}_3) = \frac{d(\mathbf{t}_0, \mathbf{t}_1, \mathbf{u}_0)}{d(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)d(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_3)}.$$

Similarly, for $\mathbf{u}_1 \in$int S_1 we get

$$M(\mathbf{u}_1|\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3) = \lambda_0(\mathbf{u}_1)M(\mathbf{u}_1|\mathbf{t}_1\mathbf{t}_2\mathbf{t}_3) = \frac{d(\mathbf{t}_1, \mathbf{t}_2, \mathbf{u}_1)}{d(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)d(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)}.$$

Let us now consider a point $t \in \partial S_0 \cap \partial S_1$ on the boundary between S_0 and S_1, and let

$$t = \sigma_1(\mathbf{t})t_1 + \sigma_3(\mathbf{t})t_3$$

be the representation of $\mathbf{t}$ in barycentric coordinates w.r.t. $\mathbf{t}_1$ and $\mathbf{t}_3$. Note that

$$\sigma_3(\mathbf{t}) = \frac{d(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t})}{d(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_3)} = \frac{d(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t})}{d(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)}$$

Our definition of the half-open convex hull implies, that $t \in [\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)$, but $t \notin [\mathbf{t}_0, \mathbf{t}_2, \mathbf{t}_3)$.

Therefore

$$M(\mathbf{t}|\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3) = \lambda_0(\mathbf{t})M(\mathbf{t}|\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3) = \frac{\sigma_3(\mathbf{t})}{d(\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2)}.$$

On the other hand, we get

$$\lim_{\mathbf{u}_0\to\mathbf{t}} M(\mathbf{u}_0|\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3) = \lim_{\mathbf{u}_1\to\mathbf{t}} M(\mathbf{u}_1|\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3) = \frac{\sigma_3(\mathbf{t})}{d(\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2)} = M(\mathbf{t}|\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3).$$

The example illustrates that a linear B-spline $M(\mathbf{u}|\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3)$ over the knot configuration in the above figure for this example as defined by (3.2), (3.3) is well-defined and continuous everywhere. □

Bivariate B-splines offer a variety of features that make them attractive for interactive surface design. Their main properties may be summarized as follows:

Theorem 3.1 (Bivariate B-Splines).
Let $V = \{\mathbf{t}_0,\ldots,\mathbf{t}_m\} \subset \mathbb{R}^2$, and let $\mathcal{T}_V$ be the triangulation of $\mathbb{R}^2$ that results after connecting every two knots in V with a line. The B-spline $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)$ satisfies the following properties:

- **Piecewise Polynomial:** $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)$ is a piecewise polynomial of degree $n = m-2$ over $\mathcal{T}_V$.
- **Local Support:** $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)$ is supported on the closed convex hull $[V] = [\mathbf{t}_0,\ldots,\mathbf{t}_m]$ of its knots.
- **Non-Negativity:** $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m) \geq 0$ for all $\mathbf{u} \in \mathbb{R}^2$.
- **Smoothness:** If at most μ knots are collinear, $2 \leq \mu \leq m$, then $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)$ is $C^{m-\mu-1}$ continuous everywhere. Hence, if the knots are in general position (i.e., no three of them are collinear), then $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)$ is C^{n-1} continuous everywhere.

Proof ▶ The fact that $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)$ is a locally supported piecewise polynomial of degree n over $\mathcal{T}_V$ follows directly from the definition.

Since every $t \in [\mathbf{t}_0,\ldots,\mathbf{t}_m]$ is a convex combination of at most 3 knots, the recursion (3.3) also shows that $M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_m)$ is non-negative.

The proof of $C^{m-\mu-1}$-continuity is omitted here and the reader is refered to [2]. ◀

3.2. Bivariate B-Splines Computation

In this subsection we introduce an algorithm, which computes values of bivariate B-splines.

input	set of points in $\mathbb{R}^2$ $V = \{t_i,\ \ i = \overline{0,m}\}$ vertex $\mathbf{u}$, $\mathbf{u} \in \mathbb{R}^2$
output	vertex v, where $v = M(\mathbf{u}\|V)$

Algorithm 3.1 (Bivariate B-Splines Computation).

```
 1  for i:=0 to m-2 do
 2    for j:=i+1 do m-1 do
 3      b[i,j,2 mod 2]:=det(ComputeBarycentricCoordinates([i,j,m],u));
 4  for i:=3 to m do
 5    for j:=0 to m-i do
 6      for k:=1 to m-i+1 do
 7        begin
 8          K:=ComputeBarycentricCoordinates([j,k,m+2-i],u);
 9          b[j,k,i mod 2]:=K.0*b[k,m+2-i,(i-1) mod 2]+
10                          K.1*b[j,m+2-i,(i-1) mod 2]+
11                          K.2*b[j,k    ,(i-1) mod 2];
12       end;
13  v:=b[0,1,m mod 2];
```

In lines $\overline{1,3}$ determinants of barycentric coordinates are computed according to formula (3.2). The main cycle (line 4) running $O(m)$ times represents the depth of the recursive computation according to formula (3.3).

Definition 3.3 (Sequence, Level).
Term $\mathcal{B} = M(\mathbf{u}|\mathbf{t}_0, \ldots, \mathbf{t}_m)$ consists of $(m+1)$ arguments $\mathbf{t}_0, \ldots, \mathbf{t}_m$. These form a *sequence* of arguments. Level of a sequence equals to the number of arguments t_* decreased by 1 in the term M corresponding to this sequence. Term $\mathcal{B}$ represents the $m-$th *level* of the recurrence. There exists one and only one sequence for level m. Note that there exist exactly 3^{m-2} sequences of length 3 for level 2 (bottom of the recurrence). $\square$

Considering the time and space complexity it is appropriate to decompose the recursion always by its first three arguments i.e. for $\mathcal{B}$ and W according to formula (3.3) we set $i_j = = j,\ \ j = \overline{0,2}$. For every level we define a set of corresponding *division sequences* of length 3. For every level, for every sequence there exists exactly one division sequence according to the rule $i_j = j,\ \ j = \overline{0,2}$. Note that the division sequence consists of first three arguments of its corresponding sequence. The division sequence denotes how the recursive computation subsume smaller results.

Lemma 3.1 (Sequences).
If we decompose the recursion always by its first three arguments, then for every level k, for every sequence for level k, subsequences of first $(k-1)$ right arguments are identical.

Proof ▶

1^{O}

$$\begin{array}{lcccl} & & \diagup & 0 & \overbrace{12}^{2}\,3\ldots m \\ level = m & 0\ldots m & - & 1 & 023\ldots m \\ & & \diagdown & 2 & 01\,\underbrace{3\ldots m}_{m-2} \end{array}$$

2^{O}

$$\begin{array}{lcccl} & & \diagup & a & \overbrace{bi}^{2}\,(i+1)\ldots m \\ level = m-i+2 & abi(i+1)\ldots m & - & b & ai(i+1)\ldots m \\ & & \diagdown & i & ab\,\underbrace{(i+1)\ldots m}_{m-i} \end{array}$$

◀

Corollary 3.1 (Time for *level* 2 is $O(m^2)$).
Sufficient amount of sequences to be computed (different sequences) for $level = 2$ is $O(m^2)$.

Proof ▶ $level = 2$-sequences are of length 3. Lemma 3.1 implies: last argument is m. ◀

Corollary 3.2 (Time for $\mathcal{B}$=$O(m^3)$).
Sufficient amount of sequences to be computed for $\mathcal{B}$ is $O(m^3)$.

Proof ▶ Is evident. ◀

Theorem 3.2 (Division Sequences).
Sufficient amount of division sequences to be computed for every level is $O(m^2)$.

Proof ▶ Lemma 3.1 implies: for every level k, third argument from left in every sequence for level k is the same. Definition of division sequences implies: for every level k last=third argument in every division sequence for level k is the same. ◀

Corollary 3.3.
Every division sequence in computation is used exactly once.

Proof ▶ Lemma 3.1 implies: for every two levels $i, j,\ i \neq j$, third argument in every division sequence for $level = i$ is different from third argument in every division sequence for $level = j$. ◀

Corollary 3.4 (Space For $\mathcal{B}$=$O(m^2)$).
Sufficient space complexity for computation of $\mathcal{B}$ is $O(m^2)$.

Proof ▶ Corrolary 3.3 and function *mod* 2 used in algorithm 3.1 enable space complexity $O(m^2)$. ◀

Note that Algorithm 3.1 computes sequences for $level = 2$ first. After this it completes computation of division sequences and sequences for $level = 3$ and continues this way up to $level = m$.

Remark 3.1 (Randomness).
If we change the rule for creating the division sequence and we take random three arguments instead of first three, we inevitably obtain $O(exp) \stackrel{def}{=} O(2^n)$ time and space computational complexity. To prove this asserion all we have to do is to construct a computation and show that it has an exponential complexity.

In the construction we change the rule for creating division sequences. For every division sequence between levels $(m-i, m-i-1),\ i > 0$ there exists previous division sequence between levels increased by one, which was used in the computation. For every division sequence there exist three division sequences in the next step. We use the following rule: Suppose a division sequence $A = [t_i, t_{i+1}, t_{i+2}]$. The new division sequences will be:

$$B_j = [t_{i+1+j}, t_{i+2+j}, t_{i+3+j}],\ \ j = \overline{0,2}. \tag{3.4}$$

Example 3.2.
Let $m = 50$. In a division sequence [012] between levels $m = 50$ and $(m-1) = 49$ the maximum argument is 2. Between levels $(m-1)$ and $(m-2)$ there are three division sequences [123], [234] and [345] according to (3.4), and the maximum argument is 5. It is evident that while we increase a level by one, the maximum argument is increased by 3. For a level 32 the maximum argument that is removed in a sequence is 50. Until level 32 we can use (3.4). If we step to level 31 we would need t_{51}, t_{52} and t_{53}. But $m = 50$. We have to use a different rule for creating division sequences (removing arguments). For our purpose there is an important fact: we made at least $n = \lfloor \frac{m}{3} \rfloor - 1$ steps and we produced at least 3^n sequences. Thus we have exponential complexity. Another fact that is relevant for our notions is $\binom{n}{\lfloor \frac{n}{3} \rfloor} = O(exp)$. $\square$

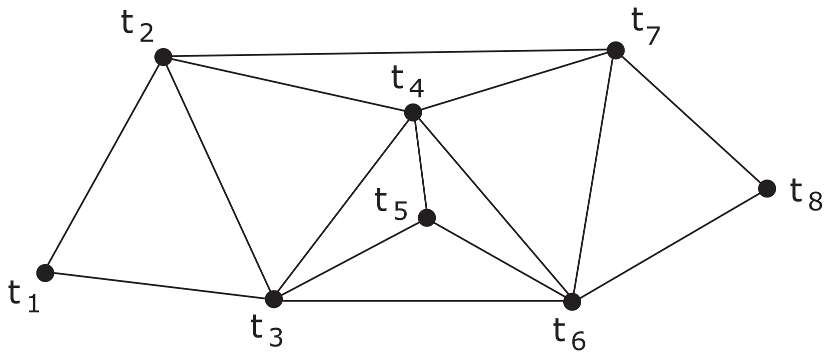
The complexity of Algorithm 3.1 compared to the de Casteljau algorithm is the same. The complexity is $O(n^3)$. The approach of Algorithm 3.1 is different. It is based on special rule for creating the division sequences.

Chapter 4

A New B-Spline Scheme

How can we use bivariate B-splines to define piecewise polynomial surfaces over arbitrary triangulations? The previous section demonstrates that lots and lots of bivariate B-splines exist. Which one should we choose and how should we normalize?

We shall proceed according to the following recipe: Suppose that a triangulation of $\mathbb{R}^2$ or of a bounded region $D \subset \mathbb{R}^2$ is given:



With each vertex we associate a *cloud* of n additional knots. We then need a rule for selecting $\binom{n+2}{2}$ subsets of $(n+3)$ knots from the 3 clouds of a given triangle. Each such subset then gives a bivariate B-spline of degree n. The linear span of all these B-splines is the spline space of interest. If the B-splines can be renormalized to form a partition of unity, we thus obtain an affinely invariant surface scheme.

Hence the problem to develop an affinely invariant scheme reduces to

- finding a rule for selecting $\binom{n+2}{2}$ subsets of $(n+3)$ knots from the 3 clouds of a give triangle, and
- coming up with the right normalization

4.1. A New Spline Space

Surprisingly enough, both the above requirements can be satisfied simultaneously by matching the B-spline recurrence (3.2), (3.3) with the recurrence for the normalized B-weights (2.1), (2.2).

Definition 4.1 (Knot Net).
Let $\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n} \in \mathbb{R}^2$ be given and assume that every triple of knots $(\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k}), 0 \leq i+j+k \leq n$ is affinely independent, i.e., that $\triangle_{ijk} = [\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k}]$ forms a proper triangle. Define

$$\Omega_{ijk} = \bigcap_{(i'j'k') \leq (ijk)} \triangle_{i'j'k'} \tag{4.1}$$

and

$$\Omega_n = \text{int}\ (\bigcap_{i+j+k=n} \Omega_{ijk}) \tag{4.2}$$

If the interior int Ω_n is non-empty, then $\mathcal{K} = \{\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n}\}$ is called a *knot net associated with the triangle* $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2] = [\mathbf{t}_{0,0}, \mathbf{t}_{1,0}, \mathbf{t}_{2,0}]$. The set of points $C_i = \{\mathbf{t}_{i,0}, \ldots, \mathbf{t}_{i,n}\}$ is called the *cloud of knots associated with the vertex* $\mathbf{t}_i = \mathbf{t}_{i,0}$. Test of Ω_n emptiness can be constructed using the notions in the B-patch checking section.
□

It is clear from the above definition that every knot net $\mathcal{K} = \{\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n}\}$ contains a knot arrangement $\mathcal{A} = \{\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n-1}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n-1}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n-1}\}$, according to definition (2.1), which is obtained from $\mathcal{K}$ by simply dropping the knots $\mathbf{t}_{0,n}, \mathbf{t}_{1,n}, \mathbf{t}_{2,n}$.

This phenomenon is completely analogous to the curve case, where a B-spline segment of degree n is defined by $2n$ many knots, while the $(n+1)$ B-spline blending functions that contribute to this segment require $(2n+2)$ knots for their definition.

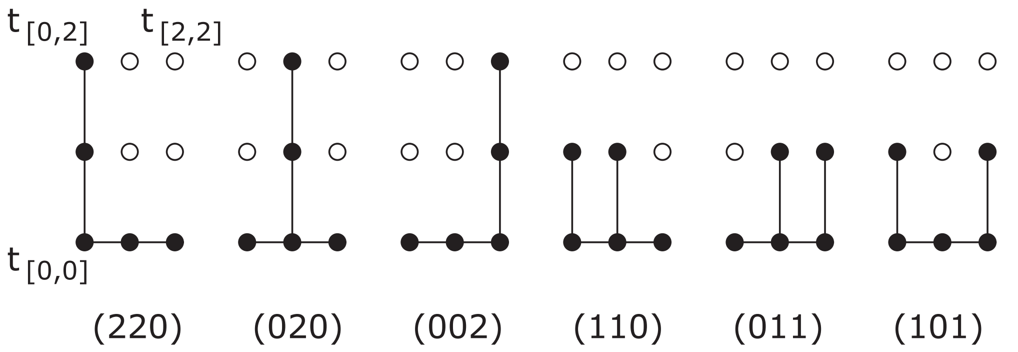
Definition 4.2 (Normalized B-Splines).
Let $\mathcal{K} = \{\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n}\}$ be a knot net associated with the triangle $\triangle = [\mathbf{t}_{0,0}, \mathbf{t}_{1,0}, \mathbf{t}_{2,0}]$. We define

$$V^{\mathcal{K}}_{ijk} = \{\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,i}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,j}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,k}\}, \ \ i+j+k=n, \tag{4.3}$$

and consider the functions

$$N^{\mathcal{K}}_{ijk}(\mathbf{u}) = |d(\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k})|.M(\mathbf{u}|V^{\mathcal{K}}_{ijk}). \tag{4.4}$$

The functions $N_{ijk}^{\mathcal{K}}(\mathbf{u})$ are called *normalized B-splines over* $\mathcal{K}$. Note that (4.3) selects exactly $\binom{n+2}{2}$ subsets $V_{ijk}^{\mathcal{K}}$, $i+j+k=n$, of $n+3$ knots from the 3 clouds of a given triangle $\triangle$. For the quadratic case, this selection rule is illustrated by the following figure: $\square$



The motivation for our selection rule (4.3) and for the normalization (4.4) comes from the following theorem:

Theorem 4.1 (B-Weights and Normalized B-Splines).
Let $\mathcal{K} = \{\mathbf{t}_{0,0}, \ldots, \mathbf{t}_{0,n}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n}\}$ be a knot net, and let $\mathcal{A} = \{\mathbf{t}_{0,0}, \ldots \ldots, \mathbf{t}_{0,n-1}, \mathbf{t}_{1,0}, \ldots, \mathbf{t}_{1,n-1}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,n-1}\}$ be the corresponding knot arrangement which is obtained from $\mathcal{K}$ by dropping the knots $\mathbf{t}_{0,n}, \mathbf{t}_{1,n}, \mathbf{t}_{2,n}$. Let $N_{ijk}^{\mathcal{K}}(\mathbf{u})$ be the normalized B-splines over $\mathcal{K}$, and let $B_{ijk}^{\mathcal{A}}$ be the normalized B-weights over $\mathcal{A}$. Then

$$N_{ijk}^{\mathcal{K}}(\mathbf{u}) = B_{ijk}^{\mathcal{A}}(\mathbf{u}) \quad \forall \mathbf{u} \in \Omega_n, \;\; i+j+k=n, \tag{4.5}$$

where Ω_n is defined according to (4.2).

Proof ▶ We will process by induction on n. Noting that $\Omega_0 = \text{int } \triangle_{000}$ the assertion for $n=0$ follows from (2.1) and (3.2).

Suppose it holds for $(n-1)$. Due to the condition (4.2) on the knot net $\mathcal{K}$, the B-splines $M(\mathbf{u}|V_{i'j'k'}^{\mathcal{K}})$, $i'+j'+k'=(n-1)$, of degree $(n-1)$ vanish for $\mathbf{u} \in \Omega_n$ whenever any of the indices i', j', k' is negative. The recursion formula (2.2) for the normalized B-weights thus gives

$$\begin{aligned}
B_{ijk}^{\mathcal{A}} &= \frac{d(\mathbf{u}, \mathbf{t}_{1j}, \mathbf{t}_{2k})}{d(\mathbf{t}_{0,i-1}, \mathbf{t}_{1j}, \mathbf{t}_{2k})} N_{i-1,jk}^{\mathcal{A}}(\mathbf{u}) + \frac{d(\mathbf{t}_{0i}, \mathbf{u}, \mathbf{t}_{2k})}{d(\mathbf{t}_{0,i}, \mathbf{t}_{1,j-1}, \mathbf{t}_{2k})} N_{i,j-1,k}^{\mathcal{A}}(\mathbf{u}) + \frac{d(\mathbf{t}_{0i}, \mathbf{t}_{1j}, \mathbf{u})}{d(\mathbf{t}_{0,i}, \mathbf{t}_{1j}, \mathbf{t}_{2,k-1})} N_{ij,k-1}^{\mathcal{A}}(\mathbf{u}) \\
&= \sigma_{i-1,j,k} \;\; d(\mathbf{u}, \mathbf{t}_{1j}, \mathbf{t}_{2k}) M(\mathbf{u}|V_{i-1,j,k}^{\mathcal{K}}) + \\
&+ \sigma_{i,j-1,k} \;\; d(\mathbf{t}_{0i}, \mathbf{u}, \mathbf{t}_{2k}) M(\mathbf{u}|V_{i,j-1,k}^{\mathcal{K}}) + \\
&+ \sigma_{i,j,k-1} \;\; d(\mathbf{t}_{0i}, \mathbf{t}_{1j}, \mathbf{u}) M(\mathbf{u}|V_{i,j,k-1}^{\mathcal{K}})
\end{aligned}$$

where $\sigma_{ijk} = \text{sgn } d(\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k})$. Since all the triangles $\triangle_{ijk} = [\mathbf{t}_{0,i}, \mathbf{t}_{1,j}, \mathbf{t}_{2,k}]$ have the same orientation, we get $\sigma_{ijk} = \sigma_{i-1,j,k} = \sigma_{i,j-1,k} = \sigma_{i,j,k-1}$, whenever $i, j, k > 0$, and therefore

$$\begin{aligned}
B^{\mathcal{A}}_{ijk}(\mathbf{u}) &= \sigma_{i,j,k}\ d(\mathbf{t}_{0i},\mathbf{t}_{1j},\mathbf{t}_{2k})\{\frac{d(\mathbf{u},\mathbf{t}_{1j},\mathbf{t}_{2k})}{d(\mathbf{t}_{0i},\mathbf{t}_{1j},\mathbf{t}_{2k})}M(\mathbf{u}|V^{\mathcal{K}}_{i-1,j,k})+\\
&\qquad + \frac{d(\mathbf{t}_{0i},\mathbf{u},\mathbf{t}_{2k})}{d(\mathbf{t}_{0i},\mathbf{t}_{1j},\mathbf{t}_{2k})}M(\mathbf{u}|V^{\mathcal{K}}_{i,j-1,k}) + \frac{d(\mathbf{t}_{0i},\mathbf{t}_{1j},\mathbf{u})}{d(\mathbf{t}_{0i},\mathbf{t}_{1j},\mathbf{t}_{2k})}M(\mathbf{u}|V^{\mathcal{K}}_{i,j,k-1})\}\\
&= |d(\mathbf{t}_{0i},\mathbf{t}_{1j},\mathbf{t}_{2k})|\{\lambda_{ijk,0}(\mathbf{u})M(\mathbf{u}|V^{\mathcal{K}}_{i-1,j,k}) + \lambda_{ijk,1}(\mathbf{u})M(\mathbf{u}|V^{\mathcal{K}}_{i,j-1,k})+\\
&\qquad + \lambda_{ijk,2}(\mathbf{u})M(\mathbf{u}|V^{\mathcal{K}}_{i,j,k-1})\}\\
&= |d(\mathbf{t}_{0i},\mathbf{t}_{1j},\mathbf{t}_{2k})|M(\mathbf{u}|V^{K}_{ijk})\\
&= N^{\mathcal{K}}_{ijk}(\mathbf{u}),
\end{aligned}$$

where we have used (3.3). ◀

Note, that Theorem (4.1) implies in particular that the normalized B-splines $N^{\mathcal{K}}_{ijk}(\mathbf{u})$ are linearly independent. The Theorem (4.1) also implies that the normalized B-splines $N^{\mathcal{K}}_{ijk}(\mathbf{u})$ form a partition of unity on Ω_n, i.e.,

$$\sum_{i+j+k=n} N^{\mathcal{K}}_{ijk}(\mathbf{u}) = \sum_{i+j+k=n} B^{\mathcal{A}}_{ijk}(\mathbf{u}) = 1 \quad \forall \mathbf{u}\in\Omega_n, \tag{4.6}$$

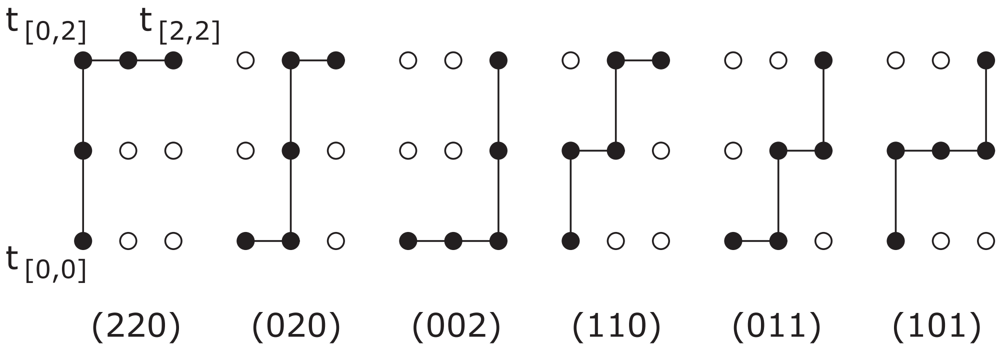
In [2] we can see that this property does not hold just locally, but globally as well.

Remark 4.1 (Multiple Knots).
Suppose that all knots $\mathbf{t}_{0,0} = \cdots = \mathbf{t}_{0,n} = \mathbf{t}_0$, $\mathbf{t}_{1,0} = \cdots = \mathbf{t}_{1,n} = \mathbf{t}_1$, $\mathbf{t}_{2,0} = \cdots = \mathbf{t}_{2,n} = \mathbf{t}_2$ in a knot net $\mathcal{K}$ have multiplicity $(n+1)$. Theorem (4.1) shows that the normalized B-splines $N^{\mathcal{K}}_{ijk}$ then specialize to the Bernstein polynomials $B^{\triangle}_{ijk}$ w.r.t. $\triangle = [\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2]$. □

Remark 4.2 (Knot Selection).
It should be pointed out that the rule (4.3) for selecting $\binom{n+2}{2}$ subsets of $(n+3)$ knots from the 3 clouds of a given triangle is completely different from another rule as developed earlier in [3], and subsequently used, e.g., in [4].

While the rule (4.3) is based on matching the B-spline recurrence with the recurrence for the normalized B-weights, the previous rule is based on the purely combinatorial recipe of associating a different knot set with every non descending path from $(0,0)$ to $(2,n)$ on the lattice points in $[0,2]\times[0,n]$. Note that there are exactly $\binom{n+2}{n}$ such different paths.

This different knot selection rule for the quadratic case is illustrated in the following figure:



One of the main differences between the two rules is, e.g., the fact, that every knot set according to the rule (4.3) contains all three vertices $\mathbf{t}_{0,0}, \mathbf{t}_{1,0}, \mathbf{t}_{2,0}$ of the triangle $\triangle$, while this is not the case for the combinatorial recipe. □

Using definition (4.2) above, the **construction of the new spline space** may then be completed as follows: suppose that

$$\mathcal{T} = \{\triangle(I) = [\mathbf{t}_{i_0,0}, \mathbf{t}_{i_1,0}, \mathbf{t}_{i_2,0}] | I = (i_0, i_1, i_2) \in \mathcal{I} \subset \mathbb{Z}^3_+\} \tag{4.7}$$

defines a triangulation of $\mathbb{R}^2$ or of a bounded domain $D \subset \mathbb{R}^2$. We associate with every vertex t_i a cloud $C_i = \{\mathbf{t}_{i,0}, \ldots, \mathbf{t}_{i,n}\}$ of knots such that $\mathcal{K}_I = \{\mathbf{t}_{i_0,0}, \ldots, \mathbf{t}_{i_0,n}, \mathbf{t}_{i_1,0}, \ldots, \mathbf{t}_{i_1,n}, \mathbf{t}_{i_2,0}, , \ldots, \mathbf{t}_{i_2,n}\}$ is a knot net for every $I \in \mathcal{I}$. We then define the sets

$$V^I_{ijk} = \{\mathbf{t}_{i_0,0}, \ldots, \mathbf{t}_{i_0,i}, \mathbf{t}_{i_1,0}, \ldots, \mathbf{t}_{i_1,j}, \mathbf{t}_{i_2,0}, \ldots, \mathbf{t}_{i_2,k}\} \tag{4.8}$$

according to (4.3) and define the normalized B-splines

$$N^I_{ijk}(\mathbf{u}) = |d(\mathbf{t}_{i_0,i}, \mathbf{t}_{i_1,j}, \mathbf{t}_{i_2,k})| M(\mathbf{u}|V^I_{ijk}) \tag{4.9}$$

according to (4.4). The linear space

$$S^n_V = \text{span } \{N^I_{ijk}(\mathbf{u}) | I \in \mathcal{I}\} \tag{4.10}$$

of all these normalized B-splines is the new spline space. Since the normalized B-splines are linearly independent, every surface in this place can then be uniquely represented as a linear combination

$$F(\mathbf{u}) = \sum_{I \in \mathcal{I}} \sum_{i+j+k=n} N^I_{ijk}(\mathbf{u}) \mathbf{c}^I_{ijk}, \;\; \mathbf{c}_{ijk} \in \mathbb{R}^t. \tag{4.11}$$

The coefficients $\mathbf{c}^I_{ijk} \in \mathbb{R}^t$, are called *control points*. They form the *control net.* □

4.2. The B-Patch Continuity

In the article [2] we can find an useful theorem about the B-patch continuity. Following theorem gives us an instruction how to modify specific B-Batch parameters to satisfy desired continuity conditions as well.

Theorem 4.2 (Piecewise Polynomial Representation).
Let $\mathcal{F}$ be a piecewise polynomial of degree n over a given triangulation $\mathcal{T}$, and let $\mathcal{F}_l$ be the restriction of $\mathcal{F}$ to the triangle $\triangle(l)$, $l \in \mathcal{T}$. Let f_l be the polar form of $\mathcal{F}_l$. Suppose that $\mathcal{F}$ is C^{υ}-continuous everywhere. Let $\mu = n - \upsilon$, and assume that every vertex $\mathbf{t}_i$ has multiplicity μ, i.e., $\mathbf{t}_i = \mathbf{t}_{i,0} = \cdots = \mathbf{t}_{i,\mu-1}$. The $\mathcal{F}$ can be represented as a linear combination of the normalized B-splines $N^l_{ijk}(\mathbf{u})$, and we get $\mathcal{F}(\mathbf{u}) = \sum_{l\in\mathcal{T}} \sum_{i+j+k=n} N^l_{ijk}(\mathbf{u})\mathbf{c}^l_{ijk}$ with

$$\mathbf{c}^l_{ijk} = f_l(\mathbf{t}_{l_0,0}, \ldots, \mathbf{t}_{l_0,i-1}, \mathbf{t}_{l_0,0}, \ldots, \mathbf{t}_{l_0,j-1}, \mathbf{t}_{l_0,0}, \ldots, \mathbf{t}_{l_0,k-1}). \tag{4.12}$$

The complete proof of Theorem (4.2) can be found in [2]. The question is how to create continuous surface using B-patch scheme. The idea is as follows. The highest level structure is a control vertices network. Using control vertices network we create piecewise polynomial surface which represents next level structure. We can use for example C^2-continuous cubic B-spline surface (C^2-continuous cubic B-spline curve is discussed in Appendix *B-spline Curves*). Using Theorem (4.2) we get control vertices for triangular B-spline surface (B-patch representation of piecewise polynomial surface) representing next level. Finally we get the B-patch representation. This idea is summed up in the following diagram:

control vertices → continuous surface →
→ control vertices (B-patch) → B-patch representation

Now we describe the recipe more precisely. Level one means we have control vertices network V. Level two means we have C^2-continuous cubic piecewise polynomial B-spline surface $\mathcal{F}$. To get to level three i.e. to get control vertices for B-patch representation of $\mathcal{F}$, we will use a Theorem (4.2) and the following formula:

$$f(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \sum_{\Gamma_n=\{I,J,K\}} \frac{i!}{n!} \frac{\partial^{j+k} F(0)}{\partial^j_u \partial^k_v} \prod_{j\in J} u_j \prod_{k\in K} v_k. \tag{4.13}$$

The formula (4.13) can be derived from formulae (1.2), (1.4) and (1.6). The final level four represents the B-patch representation of $\mathcal{F}$.

Following this recipe we can create continuous surfaces using triangular B-spline surfaces.

In the section bivariate B-splines we have presented another aproach to triangular B-spline surfaces. The main purpose is to explore enough theoretical backround and the outcome is that bivariate B-splines and triangular B-spline surfaces can share their properties.

Using enough knowledge about new B-spline scheme we can now create smooth surfaces via the formula (4.13).

Closing

In the B-patch chapter we have seen the B-patch definition, the complexity of B-patch point computation algorithm, the complexity is $O(n^3)$, we have analyzed certain implementation and design problems, we have seen properties of triangular B-spline surfaces.

In the next chapter we have discussed another approach via Bivariate B-splines, we have introduced an algorithm of bivariate B-splines computation with the complexity of $O(n^3)$, we have discussed properties of Bivariate B-splines.

New B-spline scheme represents method how to combine results from previous chapters into new surface.

There exist several examples in practice and it has been shown for few years of development of 3D modeling applications, that the less complicated the surface scheme is, the easier and faster the implementation, the computation and the manipulation is.

From this perspective, triangular B-spline surfaces are more complicated than Bézier Triangles and it is easier to use Bézier Triangles instead of triangular B-spline surfaces, in almost any case it is sufficient. Several 3D designers and 3D modeling experts recommend to use as most simple surface as possible due to 3D accelerators limited time and space computability.

NURBS as well as triangular B-spline surfaces come under more complicated surfaces category and therefore are used to say at least in special applications and research in geometric modeling.

As we have seen, the main purpose of this thesis is to explore both theoretical and practical aspects of triangular B-spline surfaces. There are many aspects that could be studied and the main theoretical aspect of this thesis is to look at the triangular B-spline surfaces from more positions. We have used mostly the general approach than the specific one. We have compared properties of B-patches, bivariate B-splines, new B-spline scheme and triangular Bézier surfaces (Appendix B). The second aspect is the practical one. After we have prepared enough theoretical basis, what is more important than to start with implementation questions? In the B-patch and bivariate B-splines section we have seen development of implementation notions starting with substantive data structures and algorithms.

FUTURE WORK

There exist whole space of possible future work concerning Triangular B-spline Surfaces e.g. higher degrees of surface continuity, using Triangular B-spline Surfaces for covering holes in objects, designing a graphic accelerator card, more detailed comparison of algorithms complexities among various surfaces schemes, and so on. Triangular B-spline surfaces represent an important surface in geometric modeling and it is constantly researched.

Appendix A

B-Spline curves

This appendix presents an overview of a theory for arbitrary-degree B-spline curves.

A.1. Non-parametric curves

Now we shall restrict ourselves to *functional curves* of the form $y = f(x)$, f denotes a polynomial. These planar curves can be writen in parametric form

$$\mathbf{b}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} t \\ f(t) \end{bmatrix}$$

We are interested in functions f that are expressed in terms of the Bernstein basis:

$$f(t) = b_0 B_0^n(t) + \cdots + b_n B_n^n(t)$$

Note that $b_j \in \mathbb{R}$. Functional curves are a subset of parametric curves. We can also find a control polygon and in order to find it, we recall linear precision property for Bézier curves

$$\mathbf{b}(t) = \sum_{i=0}^{n} \begin{bmatrix} i/n \\ b_i \end{bmatrix} B_i^n(t). \tag{A.1}$$

Thus the control polygon of the function $f(t) = \sum_{i=0}^{n} b_i B_i^n$ is given by the points $(j/n, b_j)$, $j = \overline{0, n}$, b_j are called *Bézier ordinates.*

Bézier curves are invariant under affine reparametrizations. We may consider any interval $[a, b]$ instead of $[0, 1]$, then the abscissae values are $a + i(b - a)/n$, $i = \overline{0, n}$.

A.2. Spline Curves in Bézier Form

A spline curve is the continuous map of a collection of intervals $u_0 < \cdots < u_L$ into $\mathbb{E}^3$, where each interval $[u_i, u_{i+1}]$ is mapped onto a polynomial curve segment. Each real number

u_i is called a *breakpoint* or a *knot*. Set $\{u_i,\ i = \overline{0,n}\}$ is called a *knot sequence*. We have a curve $\mathbf{s}$. For every u thus there exists a corresponding point $\mathbf{s}(u)$. u is a *global parameter* (or global coordinate). For the interval $[u_i, u_{i+1}]$ we define a *local parameter* t by setting

$$t = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{\Delta_i} \tag{A.2}$$

Note that $t \in [0, \ldots, 1]$ while $u \in [u_i, \ldots, u_{i+1}]$.

By $\mathbf{s}_i$ we denote the i-th segment of the curve $\mathbf{s}$ and we write $\mathbf{s}(u) = \mathbf{s}_i(t)$. Concerning the use of derivatives we have

$$\frac{\mathrm{d}\mathbf{s}(u)}{\mathrm{d}u} = \frac{\mathrm{d}\mathbf{s}_i(u)}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}u} = \frac{1}{\Delta_i}\frac{\mathrm{d}\mathbf{s}_i(t)}{\mathrm{d}t} \tag{A.3}$$

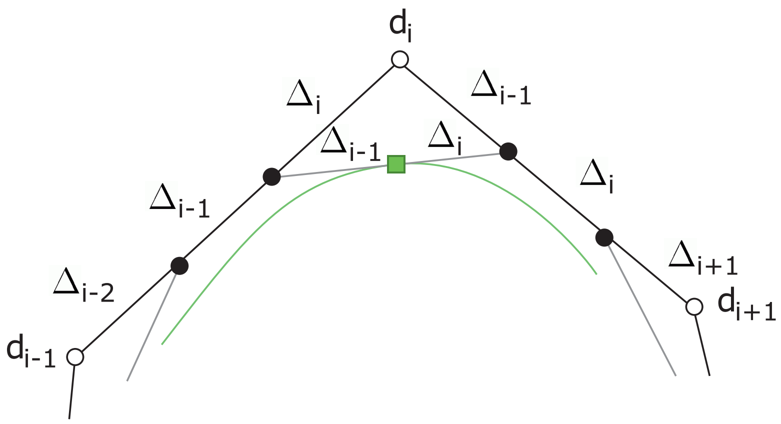
Points $\mathbf{s}(u_i) = \mathbf{s}_i(0) = \mathbf{s}_{i-1}(1)$ are called *junction points*. The sequence of the Bézier polygons for all curve segments itself forms a polygon. It is called the *piecewise Bézier polygon* of $\mathbf{s}$.

A.3. C^2 Cubic B-Spline Curves

We shall be interested in C^2 piecewise cubic spline curves. Consider L intervals over the knot sequence $u_0 < \cdots < u_L$ and consider two adjacent curve segments $\mathbf{s}_{i-1}$ and $\mathbf{s}_i$. In order to be C^1 at u_i, the relevant Bézier points must satisfy

$$b_{3i} = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i} b_{3i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i} b_{3i+1}. \tag{A.4}$$

In order to be C^2, an auxiliary point d_i must exist such that the points b_{3i-2}, b_{3i-1}, d_i and d_i, b_{3i+1}, b_{3i+2} are in the same ratio $\Delta_{i-1} : \Delta_i$. Following figure demonstrates this point:



A C^2 cubic spline curve defines the auxiliary points d_i. They form a polygon P. Conversely, a polygon P and a knot sequence $\{u_i\}$ also define C^2 cubic spline curve. Set

$$b_{3i-2} = \frac{\Delta_{i-1} + \Delta_i}{\Delta} d_{i-1} + \frac{\Delta_{i-2}}{\Delta} d_i, \;\; b_{3i-1} = \frac{\Delta_i}{\Delta} d_{i-1} + \frac{\Delta_{i-2} + \Delta_{i-1}}{\Delta} d_i, \tag{A.5}$$

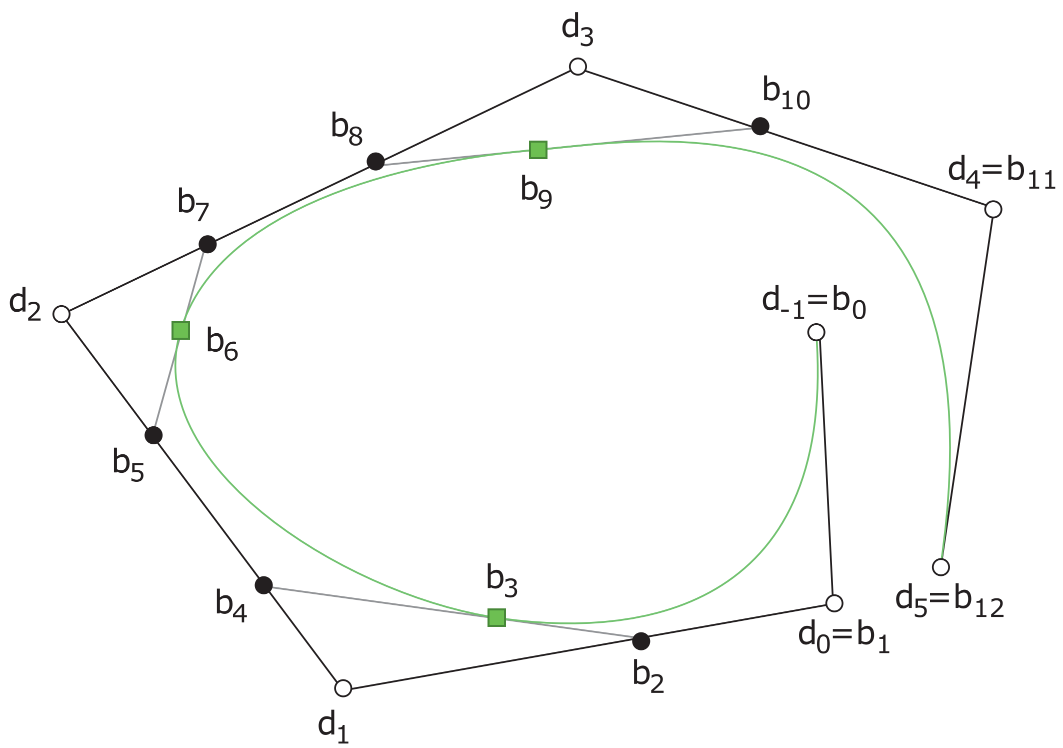
where $\Delta = \Delta_{i-2} + \Delta_{i-1} + \Delta_i$. With the junction points b_{3i} the piecewise Bézier curve defined by the d_i meets C^2 conditions at every knot u_i, $i = \overline{2, L-2}$.

To complete the definition of the cubic piecewise Bézier polygon we set

$$b_0 = d_{-1}, \;\; b_1 = d_0, \;\; , b_2 = \frac{\Delta_1}{\Delta_0 + \Delta_1} d_0 + \frac{\Delta_0}{\Delta_0 + \Delta_1} d_1, \tag{A.6}$$

$$b_{3L} = d_{L+1}, \;\; b_{3L-1} = d_L, \;\; , b_{3L-2} = \frac{\Delta_{L-1}}{\Delta_{L-2} + \Delta_{L-1}} d_{L-1} + \frac{\Delta_{L-2}}{\Delta_{L-2} + \Delta_{L-1}} d_L, \tag{A.7}$$

Now we have C^2 spline curve. If a cubic spline curve is expressed in terms of the *B-spline polygon*, it is usually called a C^2 *cubic B-spline curve*. Following figure clarifies previous concepts:



Cubic B-spline curves satisfy following properties:

- **convex hull property**
- **linear precision**
- **affine invariance**
- **symmetry**
- **endpoint interpolation**
- **variation diminishing property**
- **local control**

A.4. B-Splines

The B-spline theory is explained more naturally in the setting, where we restrict ourselves to non-parametric curves.

Consider a C^2 cubic B-spline non-parametric curve with its B-spline polygon. We find the abscissae value ξ_i for d_i to be

$$\xi_i = d_i^{(u)} = \frac{1}{3}(u_{i-1} + u_i + u_{i+1}), \;\; i = \overline{1,3}. \tag{A.8}$$

We can now give an algorithm for the *design* of a cubic B-spline funcion:

1. **Given** knots u_i
2. **Find** abscissae ξ_i
3. **Define** real numbers d_i to obtain a B-spline control polygon with vertices (ξ_i, d_i)
4. **Evaluate** Bézier control polygon

A.4.1. Knot Insertion

We will now define an algorithm to *refine* a piecewise linear function, which will be interpreted as a B-spline polygon.

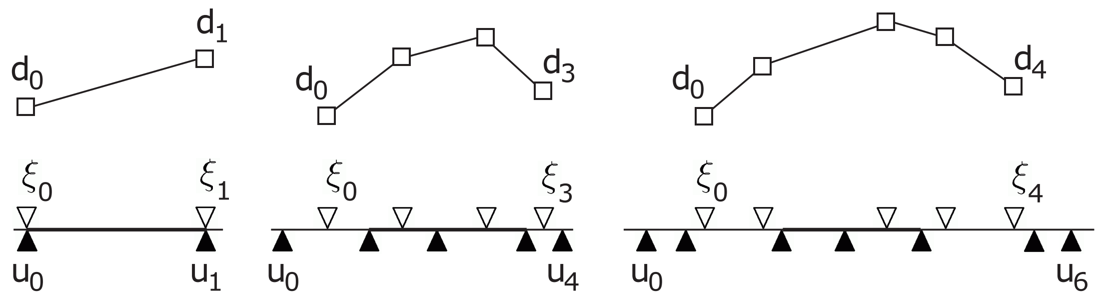
Suppose we have a number n (degree of the B-spline curve), a number L (number of polynomial segments of the B-spline curve) and a non-decreasing knot sequence $u_0, \ldots, u_{L+2n-2}$. If $u_i = \cdots = u_{i+r-1}$, we say that u_i has *multiplicity* r. When we define B-spline curves later, we will use only the interval $[u_{n-1}, \ldots, u_{L+n-1}]$ (*domain knots*) as their domain. If all domain knots are *simple* (multiplicity one), L denotes the number of domain intervals. For each domain knot multiplicity, the number of domain intervals drops by one. The following table illustrates this interplay:

example	n	L	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	domain intervals
(1)	n=3	L=3	0	1	2	3	4	5	6	7	3
(2)	n=3	L=3	0	1	2	3	3	5	6	7	2
(3)	n=3	L=3	0	0	0	3	4	5	5	5	3

We define $n + L$ *Greville abscissae* ξ_i by

$$\xi_i = \frac{1}{n}(\underbrace{u_i + \cdots + u_{i+n-1}}_{n}), \quad i = \overline{0, L+n-1}. \tag{A.9}$$

We have give ordinates d_i and the control polygon $P = \{(\xi_i, d_i), \; i = \overline{0, L+n-1}\}$. Note that the polygon is defined over Greville abscissae instead of over the original knot sequence. This is illustrated by the following figure:



$n = 1, L = 1$ $\qquad$ $n = 2, L = 2$ $\qquad$ $n = 3, L = 2$

We will now define basic technique over control polygon, the knot insertion algorithm. Suppose a real number $u \in [u_{n-1}, \ldots, u_{L+n-1}]$ is given and we want to *insert* it into the knot sequence. We call the new knot sequence a *refined knot sequence*. It defines new set of Greville abscissae, called ξ_i^u, which are the abscissae for vertices (ξ_i^u, d_i^u) of the new polygon P^u.

A.4.2. Knot Insertion Algorithm

Given: $u \in [u_{n-1}, \ldots, u_{L+n-1}]$.
Find: refined polygon P^u, defined over the new knot sequence that includes u.

1. Find the largest I with $u_I \le u < u_{I+1}$. If $u = u_I$ and *multiplicity*$(u_I) = n$ then **stop**. Else:

2. for $i = \overline{0, I-n+1}$ set $\xi_i^u = \xi_i$.

3. for $i = \overline{I-n+2, I+1}$ set

$$\xi_i^u = \frac{1}{n}(\underbrace{u_i + \cdots + u_{i+n-2}}_{n-1}) + \frac{1}{n}u.$$

4. for $i = \overline{I+2, L+n}$ set $\xi_i^u = \xi_{i-1}$
5. for $i = \overline{0, L+n}$ set $d_i^u = P(\xi_i^u)$.
6. Renumber the knot sequence to include u as u_{I+1} and replace L by $(L+1)$.

We must verify that knot insertion is a *consistent* process. If we insert $k,\ k > 1$ knots, the final result must be independent of the order in which the knots were inserted, more precisely, we have to prove that $P^{uv} = P^{vu}$. The proof is omitted and can be found in [1].

A.4.3. The de Boor Algorithm

Upwards described operation of knot insertion we shall use for the definition of B-spline curves.

In general, for degree n, when the knot u has the multiplicity n, after next insertion no longer changes of the polygon are reached. We use this fact in the algorithmic definition of a special function, called B-spline curve. The algorithm used in this definition is called the de Boor algorithm, which informally says, that to evaluate an n^{th} degree B-spline curve (given by its de Boor ordinates and knot sequence) at a parameter value u, insert u into the knot sequence until it has multiplicity n. The corresponding polygon vertex is the desired function value.

Formally: Let us denote a B-spline curve of degree n with control polygon P by B_nP, and its value at parameter value u by $[B_nP](u)$. We will only define the curve for values of between u_{n-1} and u_{L+n-1}.

de Boor algorithm: Let $u \in [u_I, u_{I+1}) \subset [u_{n-1}, u_{L+n-1}]$. Define

$$d_i^k(u) = \frac{u_{i+n-k} - u}{u_{i+n-k} - u_{i-1}} d_{i-1}^{k-1}(u) + \frac{u - u_{i-1}}{u_{i+n-k} - u_{i-1}} d_i^{k-1}(u), \tag{A.10}$$

for $k = \overline{1, n-r},\ i = \overline{I-n+k+1, I+1}$. Then

$$\mathbf{s}(u) = [B_nP](u) = d_{I+1}^{n-r}(u) \tag{A.11}$$

Note that for $0 = u_0 = \cdots = u_{n-1} < u_n = \cdots = u_{2n-1} = 1$ and for $u \in [0,1]$ the formula for $d_i^k(u)$ is specialized to $d_i^k(u) = (1-u)d_{i-1}^{k-1} + ud_i^{k-1},\ k = \overline{1,n}$ and the de Boor algorithm is specialized to the de Casteljau algorithm.

A.4.4. The B-Spline Basis

In this section we shall investigate smoothness of B-spline curves. How often is a B-spline curve diferentiable at a point u? We have to consider only the breakpoints. The curve is infinitely often differentiable at all other points.

Consider existing knot u of multiplicity r and knot sequence $0 = u_0 = \dots = u_{n-1} <$ $< u_n = \dots = u_{n+r-1} < u_{n+r} = \dots = u_{2n+r-1} = 1$. We have to reinsert the knot $u = u_n$. The de Boor algorithm only consists of $(n-r)$ levels. Taking into account the multiplicities of the end knots, we have $d_i^k(u) = (1-u)d_{i-1}^{k-1} + ud_i^{k-1}$, $k = \overline{1, n-r}$. These are the $(n-r)$ levels in a de Casteljau algorithm. Two polynomial curve segments meeting at u are at least $(n-r)$ times defferentiable there.

Now consider a knot sequence $u_0, \dots, u_k$ and the set of piecewise polynomials of degree n defined over it. Each function in that set is $(n - r_i)$ times continuously defferentiable at knot u_i. All these piecewise polynomials form a linear space. Its dimension is

$$dim = (n+1) + \sum_{i=1}^{k-1} r_i. \tag{A.12}$$

For the proof we shall construct an element of our piecewise polynomial lienar space. We may start by specifying the first segment defined over $[u_0, u_1]$, what we can do in $(n+1)$ ways, which is the number of coefficients that we can specify for a polynomial of degree n. The next segment must agree with the first segment in position and $(n - r_i)$ derivatives at u_1, thus leaving only r_1 coefficients to be chosen for the second segment. Continuing further we obtain $(A.12)$.

Now we look at B-spline curves that are piecewise polynomials over knot sequence $[u_{n-1}, u_{L+n-1}]$. The dimension of the linear space that they form is $L + n$. We shall define $L + n$ linearly independent piecewise polynomials (basis) in our linear function space. Define functions $N_i^n(u)$ called *B-splines* by defining their de Boor ordinates to satisfy $d_i =$ $= 1$ and $d_j = 0$, $\forall j \neq i$. The $N_i^n(u)$ are elements of the linear space formed by all piecewise polynomials over u_{n-1}, u_{L+n-1}. They have local support: $N_i^n(u) \neq 0$ only if $u \in [u_{i-1}, u_{i+n}]$. This follows from knot insertion and the de Boor algorithm, which is a local operation.

Important property of the N_i^n is their *linear independence*. In order to show this, we must verify that $\sum_{j=0}^{L+n-1} c_j N_j^n(u) = 0$ implies $c_j = 0$, $\forall j$. Because of the local support it is sufficient to concentrate on one interval $[u_i, u_{i+1}]$, $u_i < u_{i+1}$. B-splines also have *minimal support*: if a piecewise polynomial with the same smoothness properties over the same knot vector has less support than N_i^n, it must be the zero function. We will complete our proof, if we show that the linear space of piecewise polynomials defined over $[u_{i-n}, u_{i+n+1}]$ does not contain a nonzero elements that vanishes over $[u_i, u_{i+1}]$. Such a piecewise polynomial cannot exist. It would have to be a nonzero local support function over $[u_{i+1}, u_{i+n+1}]$. The existence of such a function would contradict the fact that B-splines are of minimal local support.

B-Splines N_i^n are linearly independent. Every polynomial $\mathbf{s}$ over $[u_{n-1}, u_{L+n-1}]$ may be written uniquely in the form $\mathbf{s}(u) = \sum_{i=0}^{L+n-1} d_i N_i^n(u)$. The B-splines thus form a basis for this space. This reveals the origin of their name, the basis splines.

Appendix B

Triangular Bézier Patches

In this appendix we review the properites of triangular Bézier patches. Let $F(\mathbf{u}) = = \sum B^{\triangle}_{ijk}(\mathbf{u})\mathbf{b}_{ijk}$ be a triangular Bézier patch w.r.t. $\triangle = [\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2]$. Then the shape of the surface is related to the shape of its control net as follows:

- **Convex Hull Property:** if $\mathbf{u} \in \triangle$, the $F(\mathbf{u})$ lies inside the closed convex hull of the control net $\{\mathbf{b}_{ijk} \mid i + j + k = n\}$.
- **Corner Vertex Interpolation:** $F(\mathbf{t}_0) = \mathbf{b}_{n00}$, $F(\mathbf{t}_1) = \mathbf{b}_{0n0}$, $F(\mathbf{t}_2) = \mathbf{b}_{00n}$,i.e., F interpolates the corner vertices of its control net.
- **Tangent Plane at Corner Vertices:** At the corner vertices $F(\mathbf{t}_0)$, $F(\mathbf{t}_1)$ and $F(\mathbf{t}_2)$, the surface F is tangent to its control net.
- **Boundary Curves:** The restriction of F to the edge $[\mathbf{t}_1, \mathbf{t}_2]$ is a Bézier curve with control points $\mathbf{b}_{ojk}$. Analogous statements hold for the other boundary curves.
- **Affine Invariance:** The relationship between the surface F and its control net is affinely invariant: If $\phi : \mathbb{R}^t \to \mathbb{R}^t$ is an affine map, then the Bézier points of the image surface $\bar{F} = F \circ \phi$ are the images $\phi(\mathbf{b}_{ijk})$ of the Bézier points $\mathbf{b}_{ijk}$.

You can find more information about triangular Bézier patches in references [1], [2].

Appendix C

Bivariate B-Splines Computation

Demonstration of fast bivariate B-splines recurrence computation:

														4	567
														5	467
														6	457
											3	4567	/	3	$\underline{567}$
											4	3567	–	5	367
											5	3467	\	6	357
														3	$\underline{467}$
														4	$\underline{367}$
														6	347
														2	$\underline{567}$
														5	267
							2	34567	/	2	$\underline{4567}$			6	257
							3	24567	–	4	2567	/			
							4	23567	\	5	2467	\		2	$\underline{467}$
														4	$\underline{267}$
														6	247
										2	$\underline{3567}$			2	$\underline{367}$
										3	$\underline{2567}$			3	$\underline{267}$
										5	2367	–		6	237
														1	$\underline{567}$
														5	167
										1	$\underline{4567}$			6	157
										4	1567	/			
										5	1467	\		1	$\underline{467}$
					1	234567	/	1	$\underline{34567}$					4	$\underline{167}$
					2	134567	–	3	14567	/				6	147
					3	124567	\	4	13567	\					
										1	$\underline{3567}$			1	$\underline{367}$
										3	$\underline{1567}$			3	$\underline{167}$
										5	1367	–		6	137
								1	$\underline{24567}$		1	$\underline{2567}$		1	$\underline{267}$
								2	$\underline{14567}$		2	$\underline{1567}$		2	$\underline{167}$
								4	12567	–	5	1267	–	6	127
														0	$\underline{567}$
														5	067
											0	$\underline{4567}$		6	057
											4	0567	/		
											5	0467	\	0	$\underline{467}$
								0	$\underline{34567}$					4	$\underline{067}$
								3	04567	/				6	047
								4	03567	\					
	/	0	1234567	/	0	$\underline{234567}$					0	$\underline{3567}$		0	$\underline{367}$
01234567	–	1	0234567	–	2	034567	/				3	$\underline{0567}$		3	$\underline{067}$
	\	2	0134567	\	3	024567	\				5	0367	–	6	037
								0	$\underline{24567}$		0	$\underline{2567}$		0	$\underline{267}$
								2	$\underline{04567}$		2	$\underline{0567}$		2	$\underline{067}$
								4	02567	–	5	0267	–	6	027
					0	$\underline{134567}$		0	$\underline{14567}$		0	$\underline{1567}$		0	$\underline{167}$
					1	$\underline{034567}$		1	$\underline{04567}$		1	$\underline{0567}$		1	$\underline{067}$
					3	014567	–	4	01567	–	5	0167	–	6	017

Demonstration of slow bivariate B-splines recurrence computation:

											3	4567	
											4	3567	...
											5	3467	
								2	34567	/	4	2567	
								3	24567	–	5	2467	
								4	23567	\	6	2457	
											5	2367	
											6	2357	
											7	2356	
											4	1567	
											5	1467	
											6	1457	
					1	234567	/	3	14567	/	5	1367	
					2	134567	–	4	13567	–	6	1357	
					3	124567	\	5	13467	\	7	1356	
											4	$\underline{1367}$	
											6	1347	
											7	1346	
											5	1267	
											6	1257	
											7	1256	
								4	12567	/	**4**	$\underline{1267}$	
								5	12467	–	**6**	1247	
								6	12457	\	**7**	1246	
											4	$\underline{1257}$	
											5	$\underline{1247}$	
											7	1245	
											4		
											5	...	
											6		
								3	04567	/	.		
								4	03567	–	.	...	
								5	03467	\	.		
	/	0	1234567	/	2	034567	/	4	02567	/			
01234567	–	1	0234567	–	3	024567	–	5	02467	–			
	\	2	0134567	\	4	023567	\	6	02457	\			
								5	02367	/			
								6	02357	–			
								7	02356	\			
								4	01567	/			
								5	01467	–			
								6	01457	\			
					3	014567	/	5	01367	/			
					4	013567	–	6	01357	–			
					5	013467	\	7	01356	\			
								4	$\underline{01367}$				
								6	01347	/			
								7	01346	\			

Bibliography

[1] Farin, G.: *Curves and Surfaces for Computer Aided Geometric Design*. New York, Academic Press 1990.

[2] Seidel, H.P.: *Polar Forms and Triangular B-Spline Surfaces*. http://citeseer.nj.nec.com

[3] Dahmen, W. and Micchelli, C.A.: *On the linear independence of multivariate B-Splines*. SIAM J.Numer. Anal., 19:993-1012, 1982

[4] Auerbach, S., Gmelig Meyling, R.H.J., Neamtu, M. and Schaeben, H.: *Approximation and geometric modeling with simplex B-Splines associated with irregular triangles*. Computer Aided Geometric Design, 8:67-87, 1991

[5] Stanek, M.: *Seminars on computational complexity*. Bratislava, MFF UK, 2000

[6] Chalmovianský, P., Ferko, A., Galbavý, R., Niepel Ľ.: *Geometric algorithms complexity*. Bratislava, MFF UK, 2001

[7] Murdock, K.L.: *3DS Max 4 Bible*. New York, Hungry Minds, 2001