Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Matemática Aplicada

Matemática por trás da arte computacional

Uma visão geral da matemática envolvida na criação de ferramentas de design e modelagem 3D

Rebeca Mariana Nascimento Yamaoka

Monografia Final

Trabalho de
Formatura Supervisionado

Supervisor: Eduardo Colli

São Paulo
2025

"A arte desafia a tecnologia, e a tecnologia inspira a arte".
- Andrew Stanton (Diretor de "Procurando Nemo")
"O processo de criação é, em si, a recompensa".
- Hayao Miyazaki (Co-fundador do Studio Ghibli)

Agradecimentos

"Então, o que você acha que vai fazer? Como vai gastar sua vida?"
— Jerry, Soul: Uma Aventura com Alma (2020, Disney/Pixar)

"Não sei ao certo... Mas sei que... Vou viver cada minuto dela".
— Joe, Soul: Uma Aventura com Alma (2020, Disney/Pixar)

Primeiramente, queria agradecer meus pais pelo apoio e amor incondicional. Seja na minha paixão por arte, nos meus estudos da matemática ou qualquer outro interesse que eu tivesse vocês sempre me apoiaram por completo. Eu amo vocês e muito obrigada por absolutamente tudo que vocês sempre me proporcionaram. Não há palavras para descrever o quanto vocês significam para mim.

Segundo, queria agradecer minha amiga Kakazita. Obrigada por sempre estar ao meu lado e por ser essa pessoa incrível. Eu te adoro e quero sempre compartilhar mais aventuras com você.

Por fim, quero agradecer meu orientador Prof. Colli. Agradeço por ter aceitado me mentorar nesse trabalho e por ter me apoiado na pesquisa de um tópico que junta duas áreas pelas quais eu tenho muito carinho.

Resumo

Rebeca Mariana Nascimento Yamaoka. **Matemática por trás da arte computacional: *Uma visão geral da matemática envolvida na criação de ferramentas de design e modelagem 3D***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2025.

A computação gráfica é uma área da informática em constante evolução, impulsionada pela necessidade de solucionar novos desafios em diversos campos, como entretenimento, medicina ou simulações científicas. Para acompanhar essas demandas, a matemática que fundamenta suas técnicas também se desenvolveu, tornando-se mais eficaz e sofisticada. Este trabalho busca estudar as ferramentas matemáticas essenciais utilizadas na área de forma acessível. Assim, este texto serve como um guia introdutório para estudantes e profissionais que desejam entender a base teórica por trás da computação gráfica moderna.

Palavras-chave: Computação gráfica. Curvas de Bézier. Superfícies de Bézier. B-spline. NURBS. Subdivisão de superfícies. Modelagem geométrica.

Abstract

Rebeca Mariana Nascimento Yamaoka. **The Mathematics Behind Computational Art: *An overview of the mathematics behind 3D modeling and design tools***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2025.

Computer graphics is a continuously evolving field of computing, driven by the need to address new challenges in diverse areas such as entertainment, medicine, and scientific simulations. To keep up with these demands, the mathematical foundations underlying its techniques have also advanced, becoming more efficient and sophisticated. This work aims to study the essential mathematical tools used in the field in an accessible manner. Thus, this text serves as an introductory guide for students and professionals seeking to understand the theoretical foundations behind modern computer graphics.

Keywords: Computer graphics. Bézier curves. Bézier surfaces. NURBS. B-splines. Subdivision surfaces. Geometric modeling.

Lista de abreviaturas

IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo
NURBS	B-splines racionais não uniformes

Lista de figuras

Sumário

Introdução

A computação gráfica é a área da informática destinada à geração, à manipulação e à exibição de imagens ou vídeos digitais. Capaz de representar dados e informações, ou ser por si só uma forma de arte, essa parte da computação possui inúmeras aplicações em diferentes áreas. Na própria informática, utilizada ao produzir interfaces gráficas para sites da Internet, aplicativos e outros softwares. Além disso, ela pode ser aplicada na indústria de entretenimento (animações e jogos), arquitetura, *desgin*, publicidade, engenharia, pesquisas científicas (simulações e experimentos), medicina, entre outras.

O consenso entre pesquisadores é que a origem da computação gráfica foi dada em 1951, quando Jay Forrester e Robert Everett, ambos do MIT (Instituto de Tecnologia de Massachusetts), desenvolveram o *"Whirlwind I"(furacão)*, o primeiro computador com recursos gráficos para processar dados numéricos e projetar imagens em televisões ou monitores comuns. Alguns anos depois, o "descendente direto" do *Whirlwind I*, o sistema SAGE (*Semi-Automatic Ground Equipment*) do MIT foi inventado. Esse sistema de monitoramento e controle de voos usava gráficos vetoriais simples para exibir imagens de radares e se tornou parte essencial do sistema de defesa antimísseis dos Estados Unidos.

Em 1963, Dr. Ivan Sutherland, pioneiro da interação homem-computador (utilização acessível dos computadores pelos humanos), desenvolveu em sua tese o programa *Sketchpad* (ou *Robot Draftsman*). Ele foi o primeiro programa a utilizar interface gráfica do usuário (acrônimo GUI, do inglês *graphical user interface*). As imagens podiam ser desenhadas na tela do computador usando uma caneta óptica (dispositivo de entrada de computador no formato de um bastão sensível à luz). Ele é considerado o pioneiro dos sistemas CAD (*Computer-Aided Design* ou Desenho Assistido por Computador).

A partir dos anos 70, as curvas e superfícies de Bézier, B-spline, NURBS e subdivisão foram uma a uma adotadas em *softwares* CAD, Maya, Adobe Photoshop, entre outros. Essas ferramentas matemáticas serão o tema principal abordado nesse trabalho. O objetivo é apresentar uma visão geral dos conceitos e métodos matemáticos envolvidos na criação de ferramentas para computação gráfica 2D e 3D.

Para melhor compreensão da matemática desenvolvida, o trabalho está organizado em 12 capítulos. O primeiro deles é um resumo de conceitos e notações que serão utilizadas ao longo das explicações. Do segundo capítulo até o sexto, o trabalho abordará as curvas em geral e, em seguida, os casos mais utilizados: curvas de Bézier, B-splines, NURBS e curvas de subdivisão. Analogamente às curvas, os capítulos sete até dez abordarão os casos gerais de superfícies de Bézier, Bspline e NURBS. O capítulo onze explica alguns cuidados e convenções usadas na superfícies de subdivisão, que são exploradas no capítulo doze.

Capítulo 1

Conceitos e Notações

Os conceitos e notações usadas são dos livros *Curves and Surfaces for CAGD* (Farin, 2001), *Computational Geometry: Algorithms and Applications* (Berg *et al.*, 2008) e *Bézier and B-Spline Techniques* (Prautzsch *et al.*, 2002).

1.1 Pontos e vetores

Os **pontos** serão denotados como elementos do espaço euclidiano (espaço vetorial real de dimensão finita munido de produto interno) tridimensional, $\mathbb{E}^3$, e letras minúsculas, como

$$a = (a_1, a_2, a_3).$$

Os **vetores** serão denotados como elementos do espaço vetorial $\mathbb{R}^3$ e letras minúsculas. Para cada dois pontos a e b, há um único vetor v tal que

$$v = b - a;\ \ a, b \in \mathbb{E}^3, v \in \mathbb{R}^3.$$

Por outro lado, dado um vetor v, há um número infinito de pares a, b tais que $v = b - a$. Dado um vetor qualquer w, a mudança do ponto a para o ponto $a + w$ é chamada **translação**. Ou seja, vetores são invariantes por translação, enquanto pontos não são.

Elementos do espaço $\mathbb{E}^3$ podem ser subtraídos um dos outros, criando vetores, mas não somados (essa operação não está definida para pontos, mas sim para vetores). A operação mais próxima da soma é a **combinação baricêntrica** (ou combinação afim), uma combinação linear com pesos somando um:

$$b = \sum_{j=0}^{n} \alpha_j b_j;\ \ b_j \in \mathbb{E}^3,\ \ \alpha_1 + \cdots + \alpha_n = 1.$$

Reescrevendo como soma de ponto e vetor:

$$b = b_0 + \sum_{j=0}^{n} \alpha_j (b_j - b_0).$$

Um caso importante dessas combinações é o da **combinação convexa**.

1.2 Combinação convexa e fecho convexo

O **fecho convexo** de um conjunto finito de pontos é o menor conjunto convexo que contém todos os pontos. Equivalentemente, é a intersecção de todos os conjuntos convexos que contêm todos os pontos, onde um conjunto convexo é tal que, para quaisquer dois pontos dele, a reta que os conecta também está contida no conjunto.

No contexto do trabalho, que se restringe a dimensões 2 e 3, o fecho convexo de um conjunto finito de pontos corresponde a:

- Um **polígono convexo** em $\mathbb{R}^2$ (degenerado em um segmento de reta, se os pontos forem colineares).
- Um **poliedro convexo** em $\mathbb{R}^3$ (degenerado em um polígono plano, se os pontos forem coplanares).

A **combinação convexa** é uma combinação baricêntrica, mas além dos pesos usados somarem um, estes são não negativos. Toda combinação convexa de pontos está sempre contida no fecho convexo destes pontos.

1.3 Transformações afins

Baseando-se na noção de combinação baricêntrica, uma transformação ϕ, que mapeia $\mathbb{E}^3$ nele mesmo, é chamada de transformação afim (ou mapeamento afim) se é invariante em relação à combinação baricêntrica. Ou seja, se

$$x = \sum \alpha_j a_j;\ \sum \alpha_j = 1, x, a_j \in \mathbb{E}^3$$

e ϕ é uma transformação afim, $\phi : \mathbb{E}^3 \to \mathbb{E}^3$, então

$$\phi(x) = \sum \alpha_j \phi(a_j);\ \phi(x), \phi(a_j) \in \mathbb{E}^3.$$

A interpretação é que a expressão $x = \sum \alpha_j a_j$ especifica como os pesos devem ser distribuídos nos pontos a_j para que sua média seja x e essa relação ainda é válida após a aplicação da transformação afim.

Dado um sistema de coordenadas e um ponto $x = (x_1, x_2, x_3) \in \mathbb{E}^3$, uma transformação afim qualquer pode ser representada pela forma

$$\phi(x) = Ax + v,$$

onde A é uma matriz é 3×3 e o v é um vetor do $\mathbb{R}^3$.

1.4 Interpolação linear

Sejam a, b dois pontos distintos em $\mathbb{E}^3$. O conjunto de todos os ponto $x \in \mathbb{E}^3$ da forma

$$x = x(t) = (1-t)a + tb;\ t \in \mathbb{R}$$

é chamada de reta que passa por a e b. Quaisquer três ou mais pontos numa mesma reta são chamados de colineares. Para $t = 0$, a reta passa pelo ponto a e se $t = 1$, ela passa pelo ponto b. Para $0 \leq t \leq 1$, o ponto x passeia sobre reta conectando a a b.

A interpolação linear é invariante em relação às transformações afins: se ϕ é uma trasformação afim de $\mathbb{E}^3$ nele mesmo e x é dado como anteriormente, então

$$\phi(x) = \phi((1-t)a + tb) = (1-t)\phi(a) + t\phi(b).$$

Além disso, a interpolação linear é em si uma transformação afim dos reais na reta em $\mathbb{E}^3$: dado $t \in \mathbb{R}$,

$$t = \sum \alpha_j a_j;\ \sum \alpha_j = 1, a_j \in \mathbb{R},$$

então

$$\begin{cases} x(t) = (1-t)a + tb = (1 - \sum \alpha_j a_j)a + \sum \alpha_j a_j b \\ x(a_j) = (1-a_j)a + a_j b \end{cases} \Rightarrow$$

$$\Rightarrow \sum \alpha_j x(a_j) = (\sum \alpha_j - \sum \alpha_j a_j)a + \sum \alpha_j a_j b = x(t).$$

A interpolação linear também está relacionada com coordenadas baricêntricas. Dados a, x e b três pontos colineares em $\mathbb{E}^3$ tais que

$$x = \alpha a + \beta b;\ \alpha + \beta = 1,$$

então α e β são chamadas de **coordenadas baricêntricas** de x em relação a a e b.

A conexão entre coordenadas baricêntricas e combinações baricêntricas é dada ao usar $\alpha = 1 - t$ e $\beta = t$. Isso mostra que as coordenadas baricêntricas podem ser negativas, bastando tomar $t \notin [0, 1]$.

Para quaisquer três pontos colineares a, b, c, as coordenadas baricêntricas de b em relação a a e c são dadas por

$$\alpha = \frac{\text{vol}_1(b, c)}{\text{vol}_1(a, c)},$$
$$\beta = \frac{\text{vol}_1(a, b)}{\text{vol}_1(a, c)},$$

onde vol_1 denota o volume unidimensional, ou seja, a distância entre os dois pontos.

A razão simples de três pontos colineares a, b e c é definida por

$$ratio(a, b, c) = \frac{\text{vol}_1(a, b)}{\text{vol}_1(a, c)},$$

mas a razão que será usada é a razão simples de Farin do livro *Curves and Surfaces for CAGD* (Farin, 2001),

$$ratio_f(a, b, c) = \frac{\text{vol}_1(a, b)}{\text{vol}_1(b, c)}.$$

Se α e β forem as coordenadas baricêntricas de b em relação a a e c, então

$$ratio_f(a, b, c) = \frac{\beta}{\alpha}.$$

As coordenadas baricêntricas de um ponto não mudam em relação às transformações afins e o quociente também não. Assim, dada ϕ uma transformação afim,

$$ratio_f(\phi(a), \phi(b), \phi(c)) = \frac{\beta}{\alpha}.$$

Transformações afins também preservam razões simples.

1.4.1 Interpolação linear por partes

Dados $b_0, \dots, b_n \in \mathbb{E}^3$ formando um polígono B. Esse polígono consiste em uma sequência de segmentos de retas, cada uma interpolando os pares de pontos b_i, b_{i+1} e é chamado de interpolante linear por partes dos pontos b_i. A interpolação linear por partes é invariante em relação a transformações afins.

1.5 Polinômios de Bernstein

Os **polinômios de Bernstein** são polinômios da forma

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

onde os coeficientes binomiais são dados por

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!}, & \text{se } 0 \le i \le n \\ 0, & \text{caso contrário.} \end{cases}$$

Uma importante propriedade desses polinômios é que eles satisfazem a recursão

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

com

$$B_0^0(t) \equiv 1$$

e

$$B_j^n(t) \equiv 0 \text{ para } j \notin \{0, \cdots, n\}.$$

Basta observar que

$$\begin{aligned} B_i^n(t) &= \binom{n}{i} t^i (1-t)^{n-i} \\ &= \binom{n-1}{i} t^i (1-t)^{n-i} + \binom{n-1}{i-1} t^i (1-t)^{n-i} \\ &= (1-t) B_i^{n-1}(t) + t B_{i-1}^{n-1}(t). \end{aligned}$$

Outra propriedade da forma polinômios de Bernstein é a partição da unidade:

$$\sum_{j=0}^{n} B_j^n(t) \equiv 1.$$

Esse fato é provado com a ajuda do **Binômio de Newton**:

$$1 = [t + (1-t)]^n = \sum_{j=0}^{n} \binom{n}{j} t^j (1-t)^{n-j} = \sum_{j=0}^{n} B_j^n(t).$$

1.6 Splines

Uma função **spline** é uma função polinomial por partes, ou seja, o domínio da função pode ser particionado em intervalos cuja união é o domínio inteiro e em cada um desses intervalos a função é descrita por um polinômio. Os segmentos polinomiais podem ter graus diferentes, mas são incomuns em aplicações práticas e não serão abordados nesse trabalho.

Definindo matematicamente, dada uma curva polinomial s, com $s : [a, b] \to \mathbb{R}$, e m nós, $a = u_0, u_1, \cdots, u_{m-1}, u_m = b$, onde $u_i \leq u_{i+1}$ para todo i. Os **nós** são os valores de t tais que $s(t)$ corresponde a junção dos segmentos polinomiais. A curva s é chamada de **spline de grau n**, se $s(t)$ é $n - r$ vezes diferenciável em qualquer nó de multiplicidade r (um nó u_{i+1} tem multiplicidade r se $u_i < u_{i+1} = \cdots = u_{i+r} < u_{i+r+1}$) e $s(t)$ é um polinômio de grau $\leq n$ em cada intervalo $[u_i, u_{i+1}]$, para $i = 0, \cdots, m-1$.

É comum se referir ao *spline* de grau n como *spline* de ordem $n + 1$.

1.7 Forma polar do polinômio

Dado um polinômio $p(t)$ de grau n,

$$p(t) = \sum_{i=0}^{n} \alpha_i t^i,$$

sua forma polar $P(a_1, \dots, a_n)$ é a única função que:

- **é multiafim:** afim em cada variável a_j,
- **é simétrica:** a ordem das variáveis não importa ($P(u, v) = P(v, u)$),
- **recupera o polinômio na diagonal:** $P(t, \dots, t) = p(t)$,
- **encontra as derivadas de p:** a r-ésima derivada de p é dada por

$$p^{(r)}(t) = \frac{n!}{(n-r)!}[P(\underbrace{t, \dots, t}_{n-r}, \underbrace{1, \dots, 1}_{r}) - P(\underbrace{t, \dots, t}_{n-r}, \underbrace{0, \dots, 0}_{r})].$$

A fórmula nesse caso é dada por

$$P(a_1, \dots, a_n) = \sum_{i=0}^{n} \alpha_i \frac{S_i}{\binom{n}{i}}$$

onde S_i, soma simétrica dos produtos $a_1 \dots a_i$, é dada por

$$S_i = \sum_{1 \le k_1 < k_2 < \dots < k_i \le n} a_{k_1} a_{k_2} \dots a_{k_i}.$$

Outra propriedade importante da forma polar é que

$$\begin{aligned} P'(u_2, \dots, u_n) &= \frac{n}{b-a}[P(b, \underbrace{u_2, \dots, u_n}_{n-1}) - P(a, \underbrace{u_2, \dots, u_n}_{n-1})] \\ &= n[P(1, \underbrace{u_2, \dots, u_n}_{n-1}) - P(0, \underbrace{u_2, \dots, u_n}_{n-1})], \end{aligned}$$

onde P' é a forma polar do polinômio $p'(t)$ e de forma geral,

$$P^{(r)}(u_{r+1}, \dots, u_n) = \frac{n!}{(n-r)!}[P(\underbrace{1, \dots, 1}_{r}, \underbrace{u_{r+1}, \dots, u_n}_{n-r}) - P(\underbrace{0, \dots, 0}_{r}, \underbrace{u_{r+1}, \dots, u_n}_{n-r})],$$

com $P^{(r)}$ forma polar do polinômio $p^{(r)}(t)$.

1.8 Transformações projetivas

Dado um plano P (plano de imagem) e um ponto o (centro ou origem da projeção) em $\mathbb{E}^3$. Um ponto p pode ser projetado em P através de o, ao achar a intersecção $\hat{p}$ entre a reta que conecta o a p e o plano P. Para a projeção estar bem definida, é necessário que $o \notin P$. Qualquer ponto de $\mathbb{E}^3$ pode ser projetado em P dessa maneira.

Em particular, é possível projetar uma linha L no plano P. A projeção não é uma transformação afim, a razão entre os pontos correspondentes de L e sua projeção L' não são os mesmo. No entanto, a projeção preserva a razão cruzada (ou *cross ratio*), dados

quatro pontos colineares a, b, c e d, ela é dada por

$$(a, b; c, d) = \frac{vol_1(a, c)vol_1(b, d)}{vol_1(a, d)vol_1(b, c)}.$$

Será utilizada uma definição particular equivalente,

$$\text{cr}(a, b, c, d) = \frac{\text{ratio}_f(a, b, d)}{\text{ratio}_f(a, c, d)}.$$

Essa convenção tem a vantagem de ser simétrica: $\text{cr}(a, b, c, d) = \text{cr}(d, c, b, a)$.

A razão cruzada é invariante em relação a projeções, teorema da razão cruzada: $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ dependem somente dos ângulos α, β e γ (ver figura 1.1).

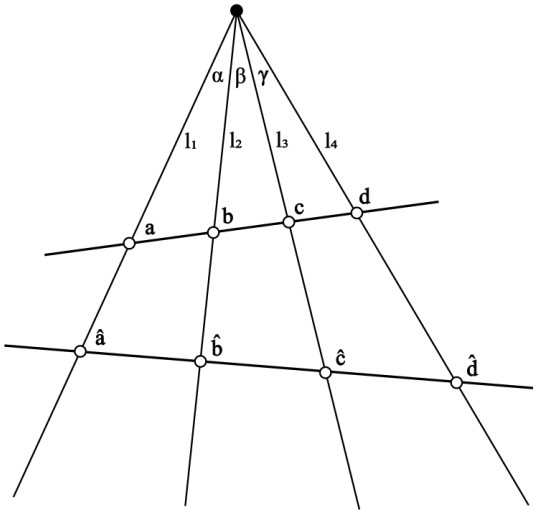

Figura 1.1: *Razão cruzada: a razão cruzada, de Farin, de a, b, c, d e de $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ dependem somente dos ângulos α, β e γ.*

Denotando a área do triângulo de vértices p, q, r por $\Delta(p, q, r)$, é possível demonstrar que

$$\text{ratio}_f(a, b, c) = \frac{\Delta(a, b, o)}{\Delta(b, c, o)}.$$

Então

$$\begin{aligned}\text{cr}(a, b, c, d) &= \frac{\Delta(a, b, o)/\Delta(b, d, o)}{\Delta(a, c, o)/\Delta(c, d, o)} \\ &= \frac{l_1 l_2 \text{sen}(\alpha)/l_1 l_4 \text{sen}(\beta + \gamma)}{l_1 l_3 \text{sen}(\alpha + \beta)/l_3 l_4 sen(\gamma)} \\ &= \frac{\text{sen}(\alpha)/\text{sen}(\beta + \gamma)}{\text{sen}(\alpha + \beta)/\text{sen}(\gamma)}.\end{aligned}$$

Logo, a razão só depende dos ângulos em o. Ou seja,

$$\mathrm{cr}(a, b, c, d) = \mathrm{cr}(\hat{a}, \hat{b}, \hat{c}, \hat{d}).$$

Voltando para as retas L e L', estas podem ser interpretadas como cópias da reta real. Então a projeção de L em L' pode ser vista como uma transformação que mapeia a reta real nela mesma. Com essa definição, a projeção define uma **transformação projetiva** da reta real nela mesma.

Uma importante observação sobre as tranformações projetivas da reta real nela mesma é que elas são definidas por três pontos de pré-imagem e três pontos de imagem. Dados três números a, b, c de pré-imagem e três números $\hat{a}, \hat{b}, \hat{c}$ de imagem, a imagem $\hat{t}$ do ponto t é dada por

$$\mathrm{cr}(a, b, t, c) = \mathrm{cr}(\hat{a}, \hat{b}, \hat{t}, \hat{c}).$$

Sendo $\rho = (b - a)/(c - b)$ e $\hat{\rho} = (\hat{b} - \hat{a})/(\hat{c} - \hat{b})$, isso é equivalente a

$$\frac{\rho}{(t - a)(c - t)} = \frac{\hat{\rho}}{(\hat{t} - \hat{a})(\hat{c} - \hat{t})}.$$

Resolvendo para $\hat{t}$:

$$\hat{t} = \frac{(t - a)\hat{\rho}\hat{c} + (c - t)\hat{a}\rho}{\rho(c - t) + \hat{\rho}(t - a)}.$$

Com uma escolha conveniente para os pontos de imagem e pré-imagem, $a = \hat{a} = 0, c = \hat{c} = 1$. A equação se torna

$$\hat{t} = \frac{t\hat{\rho}}{\rho(1 - t) + \hat{\rho}t}.$$

Curvas de forma livre

Diferentemente das construções geométricas usando segmentos de círculos, cônicas e retas, curvas de forma livre (*freeform curves*) suaves, modeladas a partir de um pequeno número de pontos de controle, são ferramentas mais recentes.

Curvas de Bézier estão entre as curvas de forma livre mais usadas. Este capítulo abordará sua construção, baseada no algoritmo de Casteljau, e algumas de suas propriedades. Para design de curvas mais complexas, há a necessidade de curvas que ofereçam controle da forma local e as curvas B-spline fazem esse papel. As curvas NURBS (do inglês *Non-Uniform Rational B-Splines*, ou B-splines racionais não uniformes) possuem mais uma possibilidade de ajustes finos ao atribuir pesos a cada ponto de controle. Elas são usadas para criar as curvas de forma livre planares ou espaciais mais complexas, assim como todos os tipos de cônicas.

As curvas Bézier, B-spline e NURBS são definidas a partir de um pequeno número de **pontos de controle** conectados em um **polígono de controle**. A partir desses pontos, a curva suave é criada automaticamente por um algoritmo geométrico e ao mudar o polígono de controle, a curva resultante também é modificada.

Há duas maneiras principais ao começar o design de curvas iterativas com pontos de controle:

- Interpolação: definir uma sequência ordenada de pontos (e possivelmente também suas direções) e criar uma curva suave que passe exatamente por eles.
- Aproximação: definir a curva preliminar (ou o rascunho da curva) com um polígono de controle e refiná-la até criar uma curva suave que siga a mesma forma. Esse caso será visto em curvas de subdivisão (Capítulo 5).

A abordagem utilizada para a compreensão das curvas é a dos livros *Curves and Surfaces for CAGD: A Practical Guide* (Farin, 2001) (curvas de Bézier, NURBS), *Bézier and B-Spline Technique* (Prautzsch *et al.*, 2002) (curvas B-spline) e *The NURBS Book* (Piegl e Tiller, 1997) (curvas NURBS). As motivações e problemas abordados são oferecidos pelo livro *Architectural geometry* (Pottmann *et al.*, 2015).

Capítulo 2

Curvas de Bézier

A base das curvas de Bézier foi criada em 1957, quando Paul de Casteljau, enquanto travalhava na Citroën, desenvolveu o *algoritmo de Casteljau* para o cálculo de uma família de curvas que seriam nomeadas e popularizadas por Pierre Étienne Bézier. Em 1962, o funcionário francês da Renault desenvolveu a notação do algortimo, usando nós com alças para controle da forma das curvas, e publicou o primeiro trabalho sobre as curvas de Bézier.

Essas curvas, podendo ser intuitivamente manipuladas por usuários por meio de softwares, foram adotadas como curvas padrão da linguagem *PostScript* e, mais tarde, por programas vetoriais como *Adobe Illustrator*, *CorelDRAW* e *Inkscape*. Grande parte das fontes de contorno são definidas com curvas de Bézier compostas (Seção 2.5).

A curva de Bézier é uma curva polinomial expressa como uma interpolação linear dos pontos de controle. Essas curvas serão utilizadas para a construção de superfícies de Bézier que serão abordadas no capítulo 6 (também desenvolvida por Pierre Bézier).

2.1 Parábolas

Começando pela construção da geração de uma parábola, sua generalização levará às curvas de Bézier. Dados os pontos quaisquer $b_0, b_1, b_2 \in \mathbb{E}^3$ e $t \in \mathbb{R}$, constroem-se as retas

$$b_0^1(t) = (1-t)b_0 + tb_1$$
$$b_1^1(t) = (1-t)b_1 + tb_2$$

e, a partir destas,

$$b_0^2(t) = (1-t)b_0^1(t) + tb_1^1(t).$$

Substituindo as duas primeiras equações na última, obtém-se

$$b_0^2(t) = (1-t)^2 b_0 + 2t(1-t)b_1 + t^2 b_2.$$

Essa é uma expresão quadrática em relação a t e $b_0^2(t) = b^2(t)$ traça uma parábola quando t varia em $(-\infty, +\infty)$,

$$b_0^2(t) = b^2(t) = (b_0 - 2b_1 + b_2)t^2 + (-2b_0 + 2b_1)t + b_0.$$

A construção consiste na repetição do método de interpolação linear (2.1). Para t entre 0 e 1, $b^2(t)$ está dentro do triângulo formado pelos três pontos b_0, b_1, b_2, em particular, $b^2(0) = b_0$ e $b^2(1) = b_2$.

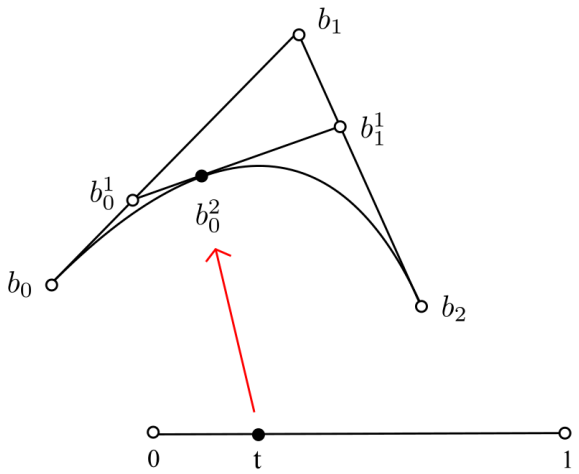

Figura 2.1: *Parábola: construção pela repetição de interpolações lineares.*

Inspecionando as razões entre os comprimentos dos segmentos,

$$\text{ratio}_f(b_0, b_0^1, b_1) = \text{ratio}_f(b_1, b_1^1, b_2) = \text{ratio}_f(b_0^1, b_0^2, b_1^1) = \frac{t}{1-t}.$$

Como consequência de ser uma interpolação linear por partes (seção 1.4), a construção da parábola é invariante em relação a transformações afins.

2.2 Algoritmo de Casteljau

A construção vista anteriormente para a parábola pode ser generalizada para gerar uma curva polinomial arbitrária de grau n.

Algoritmo de Casteljau: Dados $b_0, b_1, \cdots, b_n \in \mathbb{E}^3$ e $t \in \mathbb{R}$,

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + t b_{i+1}^{r-1}(t) \quad \begin{cases} r = 1, \cdots, n \\ i = 0, \cdots, n-r \end{cases}$$

e $b_i^0(t) = b_i$. Assim, $b_0^n(t)$ é o ponto com parâmetro t na **curva de Bézier** b^n, ou seja $b^n(t) = b_0^n(t)$.

O polígono P formado pelos pontos $b_0, b_1, \cdots, b_n$ é chamado de **polígono de controle** da curva b^n. Similarmente, os vértices do polígono b_i são chamados **pontos de controle**. A curva $b^n(t)$ é a *aproximação de Bernstein-Bézier* dos pontos de controle, uma terminologia da teoria de aproximação (Seção 1.5).

Os coeficientes $b_i^r(t)$ são escritos em forma de cascata (triângulo) no *Esquema de Casteljau.*

$$
\begin{array}{ccccccc}
b_0 & & & & & & \\
b_1 & b_0^1 & & & & & \\
b_2 & b_1^1 & b_0^2 & & & & \\
b_3 & b_2^1 & b_1^2 & b_0^3 & & & \\
\vdots & & & & & & \\
b_{n-1} & b_{n-2}^1 & b_{n-3}^2 & \cdots & b_0^{n-1} & & \\
b_n & b_{n-1}^1 & b_{n-2}^2 & \cdots & b_1^{n-1} & b_0^n &
\end{array}
$$

Obtendo a curva b^n de Bézier completa, dependendo somente dos pontos $b_0, b_1, \cdots, b_n$ e do parâmetro t: como para $r = 1, \ldots, n$ e $i = 0, \ldots, n-r$

$$
\begin{aligned}
b_i^r(t) &= (1-t)b_i^{r-1}(t) + t b_{i+1}^{r-1}(t) \\
&= (1-t)^r b_i + r(1-t)^{r-1} t b_{i+1} + \ldots + r(1-t)t^{r-1} b_{i+r-1} + t^j b_{i+r} = \\
&= \sum_{k=0}^{r} \binom{r}{k} t^k (1-t)^{r-k} b_{i+k}.
\end{aligned}
$$

Então,

$$
\begin{aligned}
b^n(t) = b_0^n(t) &= (1-t)b_0^{n-1}(t) + t b_1^{n-1}(t) \\
&= (1-t)\left[\sum_{k=0}^{n-1} \binom{n-1}{k} t^k (1-t)^{n-1-k} b_k\right] + t\left[\sum_{k=0}^{n-1} \binom{n-1}{k} t^k (1-t)^{n-1-k} b_{1+k}\right] = \\
&= \sum_{k=0}^{n-1} \binom{n-1}{k} t^k (1-t)^{n-k} b_k + \sum_{k=0}^{n-1} \binom{n-1}{k} t^{k+1} (1-t)^{n-1-k} b_{1+k} = \\
&= \sum_{k=0}^{n-1} \binom{n-1}{k} t^k (1-t)^{n-k} b_k + \sum_{k=1}^{n} \binom{n-1}{k-1} t^k (1-t)^{n-k} b_k = \\
&= (1-t)^n b_0 + \sum_{k=1}^{n-1} \left[\binom{n-1}{k} + \binom{n-1}{k-1}\right] t^k (1-t)^{n-k} b_k + t^n b_n = \\
&= \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} b_i
\end{aligned}
$$

Usando a notação dos polinômios de Bernstein,

$$
b^n(t) = \sum_{i=0}^{n} b_i B_i^n(t).
$$

2.3 Propriedades da curva de Bézier

O algoritmo de Casteljau possibilita a inferência de importantes propriedades das curvas de Bézier.

Invariância afim: transformações fins (Seção 1.3) são importantes ferramentas para sistemas CAD, usadas em reposicionamento ou redimensionamento de objetos, por exemplo. Curvas de Bézier são invariantes em relação a transformações afim, ou seja, ao computar os pontos $b_0^n(t)$ e aplicar uma transformação afim nesses pontos obtém-se o mesmo resultado que aplicar a essa transformação no polígono de controle e utilizar o algoritmo de Casteljau no polígono transformado. A invariância afim é uma consequência direta do algoritmo de Casteljau, uma vez que este é composto por uma sequência de interpolações lineares (sequência de transformações afins) e estas são invariantes em relação a transformações afins.

- Exemplo prático da propriedade: uma curva cúbica b^3 avaliada em 500 pontos, rotacionada e plotada. Uma possibilidade seria computar todos os 500 pontos, depois rotacionar cada um deles (500 aplicações de rotação) e plotar. Outra possibilidade seria rotacionar somente os 4 pontos de controle, avaliar os 500 pontos e plotá-los (somente 4 aplicações de rotação).

Invariância em relação a transformação afim de parâmetros: a definição das curvas de Bézier no intervalo $[0, 1]$ é uma questão de conveniência: o algoritmo não depende do intervalo em si e sim das razões envolvidas. É possível pensar na curva definida num intervalo $[a, b] \subset \mathbb{R}$, $t = (u-a)/(b-a)$ e o algoritmo segue da mesma forma. O algoritmo generalizado de Casteljau é dado pela forma

$$b_i^r(u) = \frac{b-u}{b-a} b_i^{r-1}(u) + \frac{u-a}{b-a} b_{i+1}^{r-1}(u).$$

No caso da forma de polinômios de Bernstein:

$$\sum_{i=0}^{n} b_i B_i^n(t) = \sum_{i=0}^{n} b_i B_i^n\left(\frac{u-a}{b-a}\right).$$

A passagem do intervalo $[0, 1]$ para $[a, b]$ é um mapeamento afim e, assim, as curvas de Bézier são invariantes em relação à transformação afim de seu parâmetro.

Propriedade do fecho convexo: para $t \in [0, 1]$, a curva $b^n(t)$ está contida no fecho convexo (Seção 1.2) dos pontos de controle (do polígono de controle). Isso ocorre pois cada ponto intermediário b_i^r é obtido por uma combinação convexa dos anteriores, b_i^{r-1} e b_{i+1}^{r-1}. Em nenhum passo do algoritmo de Casteljau são produzidos pontos fora do fecho convexo de b_i.

Pontos extremos da interpolação: as curvas de Bézier passam pelos pontos $b^n(0) = b_0$ e $b^n(1) = b_n$.

Simetria: as curvas de Bézier geradas a partir dos pontos $b_0, b_1, \cdots, b_n$ e $b_n, b_{n-1}, \cdots, b_1$

são iguais. Como

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

Então,

$$\sum_{i=0}^{n} b_i B_i^n(t) = \sum_{i=0}^{n} b_{n-i} B_i^n(1-t).$$

Os polinômios de Bernstein são simétricos em relação a t e $1-t$.

Derivada primeira da curva de Bézier: Dada uma curva de Bézier b^n de grau n,

$$\begin{aligned} b^n(t) &= \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} b_i = \\ &= (1-t)^n b_0 + \sum_{i=1}^{n-1} \binom{n}{i} t^i (1-t)^{n-i} b_i + t^n b_n, \end{aligned}$$

sua primeira derivada é dada por

$$\dot{b}(t) = n(1-t)^{n-1} b_0 + \sum_{i=1}^{n-1} \binom{n}{i} [i t^{i-1}(1-t)^{n-i} - (n-i) t^i (1-t)^{n-i-1}] b_i + n t^{n-1} b_n.$$

Observando que

$$i\binom{n}{i} = \frac{n!}{(i-1)!(n-i)!} = n\frac{(n-1)!}{(i-1)!(n-i)!} = n\binom{n-1}{i-1}$$

e

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} = \frac{n}{(n-i)}\frac{(n-1)!}{i!(n-i-1)!} = \frac{n}{(n-i)}\binom{n-1}{i},$$

então

$$\begin{aligned} \dot{b}^n(t) &= n(1-t)^{n-1} b_0 + \sum_{i=1}^{n-1} \binom{n}{i} i t^{i-1}(1-t)^{n-i} b_i - \sum_{i=1}^{n-1} \binom{n}{i} (n-i) t^i (1-t)^{n-i-1} b_i + n t^{n-1} b_n = \\ &= n(1-t)^{n-1} b_0 + n \sum_{i=1}^{n-1} \binom{n-1}{i-1} t^{i-1}(1-t)^{n-i} b_i - n \sum_{i=1}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-i-1} b_i + \\ &+ n t^{n-1} b_n = \\ &= n(1-t)^{n-1} b_0 + n \sum_{i=0}^{n-2} \binom{n-1}{i} t^i (1-t)^{n-i-1} b_{i+1} - n \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-i-1} b_i = \\ &= n \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-i-1} (b_{i+1} - b_i) \end{aligned}$$

ou, usando polinômios de Bernstein,

$$\dot{b}(t) = n \sum_{i=0}^{n-1} (b_{i+1} - b_i) B_i^{n-1}(t).$$

Usando-se $\Delta b_i = b_{i+1} - b_i$, essa expressão pode ser reescrita como

$$\dot{b}(t) = n \sum_{i=0}^{n-1} \Delta b_i B_i^{n-1}(t). \tag{2.1}$$

Ou seja, a derivada da curva de Bézier também é uma curva de Bézier.

Por outro lado, o algoritmo de Casteljau também constrói os pontos $b_0^{n-1}(t)$ e $b_1^{n-1}(t)$. Dado $t \in \mathbb{R}$, é possível calcular o vetor $b_1^{n-1}(t) - b_0^{n-1}(t)$,

$$\begin{aligned} b_1^{n-1}(t) - b_0^{n-1}(t) &= \sum_{k=0}^{n-1} \binom{n-1}{k} t^k (1-t)^{n-1-k} b_{k+1} - \sum_{k=0}^{n-1} \binom{n-1}{k} t^k (1-t)^{n-1-k} b_k = \\ &= \sum_{k=0}^{n-1} \binom{n-1}{k} t^k (1-t)^{n-k-1} (b_{k+1} - b_k) \\ &\Rightarrow \dot{b}^n(t) = n(b_1^{n-1}(t) - b_0^{n-1}(t)). \end{aligned}$$

Por último, é possível observar pela construção dessas curvas que os pontos b_1 e b_{n-1} controlam as tangentes nos pontos b_0 e b_n, respectivamente.

Subdivisão das curvas de Bézier: dada uma curva de Bézier b^n com polígono de controle $b_0, \cdots, b_n$, o algoritmo de Casteljau a subdivide em duas outras curvas de Bézier de grau n com subpolígonos de controle $c_0, \cdots, c_n$ e $d_0, \cdots, d_n$, respectivamente. Dado $s \in [0, 1]$, os novos pontos de controle são dados por

$$\begin{gathered} c_j = b_0^j(s), \ \ j \in 0, \cdots, n, \\ d_j = b_{n-j}^j(s), \ \ j \in 0, \cdots, n, \\ c_n = d_n = b^n(s). \end{gathered}$$

2.4 Limitações de curvas de Bézier

As curvas de Bézier possuem limitações que motivarão o estudo das próximas curvas de forma livre.

Número de pontos de controle: para curvas mais complexas, existe a necessidade do aumento do número de pontos de controle e, portanto, do grau da curva de Bézier. No entando, isso pode acabar afastando a forma da curva de Bézier do seu polígono de controle, efeito mais conhecido como o **fenômeno de Runge**. Esse fato torna essas curvas impráticas para design, uma vez que há pouco controle da forma geral final.

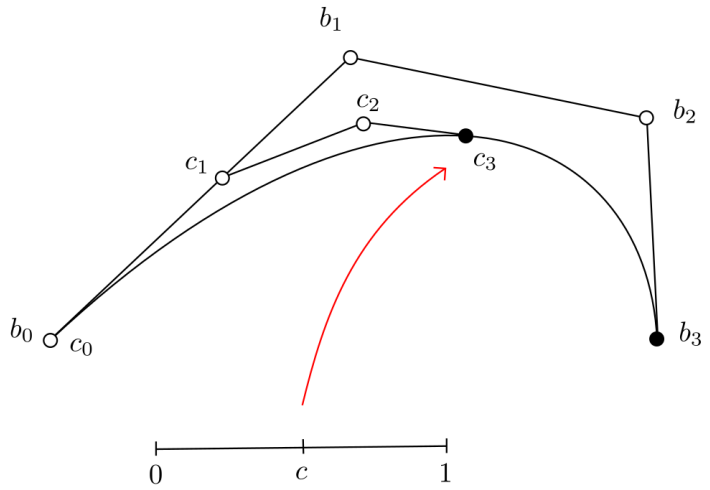

Figura 2.2: *Exemplo: subdivisão da curva de Bézier com* $n = 3$.

Controle global da curva: os pontos de controle das curvas de Bézier possuem controle global da curva. Isso significa que ao adicionar um novo ponto de controle ou ao modificar a posição de um deles, a curva toda se modifica. No design, em geral, há a busca por controle local dos pontos de controle.

2.5 Composição de curvas de Bézier

Curvas de Bézier de grau baixo, $n = 2$ ou $n = 3$, podem ser compostas para gerar formas que são muito complexas para uma única curva de Bézier. As curvas são chamadas de **curvas de Bézier compostas** ou **splines de Bézier**. Essa técnica é utilizada na criação de design de fontes, ou nas próprias ferramentas como a caneta do *Adobe Photoshop* e a curva de Bézier do *Maya*.

Ao juntar as curvas é necessário controlar a suavidade da sua junção. Sejam $b_0, \cdots, b_3$ e $b_3, \cdots, b_6$ os pontos de dois segmentos de curvas cúbicas b^3_- e b^3_+, respectivamente. Como ambas compartilham do ponto b_3, elas duas formam uma curva contínua, C^0. Porém, se essa for a única exigência, sua junção pode formar uma quina (ou um ponto angular).

Usando a propriedade das derivadas dos pontos de fronteira (ou extremidades) das curvas de Bézier, para que haja suavidade no ponto b_3, os pontos b_2, b_3 e b_4 devem ser colineares. Isso garantirá que a reta tangente em b_3 seja igual em ambas as curvas.

Uma condição mais forte é requerida para os dois segmentos de curva formarem um curva de classe C^1 (derivada contínua). Como a derivada da curva (mais precisamente, o comprimento do vetor tangente) depende do domínio da curva, seja b^3_- definida no intervalo $[a, b]$ e b^3_+ no inetrvalo $[b, c]$. As derivadas em ambos os segmentos no parâmetro b são obtidas usando:

$$\frac{3}{b-a}[b_3 - b_2] = \frac{3}{c-b}[b_4 - b_3].$$

A interpretação geométrica é que a razão simples de Farin entre os três pontos b_2, b_3 e b_4 é a mesma que a dos parâmetros a, b e c.

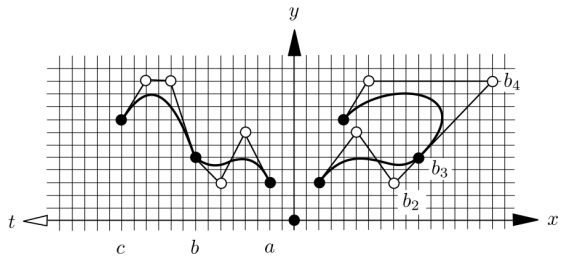

Figura 2.3: *Composição das curvas de Bézier: exemplo C^0, mas não C^1.*

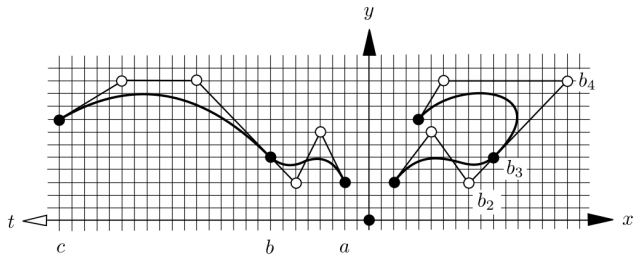

Figura 2.4: *Composição das curvas de Bézier: exemplo C^1.*

No caso dos softwares como *Adobe Photoshop* e *Maya*, as curvas compostas de Bézier usadas são cúbicas. Essas são intuitivamente criadas e editadas com pontos de ancoragem, correspondendo aos pontos de controle extremos, e alças desses, fazendo o papel dos outros dois pontos de controle e, portanto, das tangentes dos pontos extremos. As alças (ou controladores) podem ser integradas, para a junção das curvas ser suave como anteriormente, ou separadas, para a criação de cantos ou quinas.

Capítulo 3

Curvas B-Spline

O conceito das curvas B-spline remonta ao século 19, quando Nikolai Lobachevsky explorou uma ideia similar na Universidade Estatal de Cazã, mas o termo "B-spline" foi criado por Isaac Jacob Schoenberg em 1978, refletindo a função de ser uma base para splines (*basis spline*).

A curva B-spline é uma curva polinomial por partes, ou seja, é a curva resultante da junção de uma sequência de segmentos de curvas polinomiais de mesmo grau. As conexões entre os segmentos (**nós**) são feitas com o maior grau possível de suavidade (mesma tangente, curvatura, etc). Por essas características, a curva B-spline é mais versátil que as curvas de Bézier. Essas curvas serão utilizadas para a construção de superfícies B-spline que serão abordadas no capítulo 7.

3.1 Motivação pelos Splines

Assim como as curvas de Bézier podem representar curvas polinomiais com combinações lineares de polinômios de Bernstein, é desejável que os **splines** (Seção 1.6) $s(t)$ possam ser escritos como combinações afins de pontos de controle, como

$$s(t) = \sum_i c_i N_i^n(t), \tag{3.1}$$

onde $N_i^n(u)$ é uma base funções spline com suporte (subconjunto do domínio da função onde esta é não nula) mínimo e certas propriedades de continuidade. Essas funções são as chamadas funções B-splines (do inglês *basis spline*) e quando o spline é escrito da forma (3.1), ele é chamado de **curva B-spline**.

3.2 Definição recursiva das funções B-spline

Primeiro, é necessário definir as **funções B-spline**. Seja (u_i), por simplicidade, uma sequência bi-infinita e estritamente crescente de nós, $u_i < u_{i+1}$ para todo i. As funções

B-spline N_i^n são definidas com esses nós pela fórmula recursiva

$$N_i^0(t) = \begin{cases} 1 & \text{se } t \in [u_i, u_{i+1}) \\ 0 & \text{caso contrário} \end{cases}$$

e

$$N_i^n(t) = \alpha_i^{n-1} N_i^{n-1}(t) + (1 - \alpha_{i+1}^{n-1}) N_{i+1}^{n-1}(t),$$

onde

$$\alpha_i^{n-1} = \frac{(t - u_i)}{(u_{i+n} - u_i)}$$

é o parâmetro local com respeito ao suporte de N_i^{n-1}. Substituindo-o na fórmula,

$$N_i^n(t) = \frac{(t - u_i)}{(u_{i+n} - u_i)} N_i^{n-1}(t) + \frac{(u_{i+n+1} - t)}{(u_{i+n+1} - u_{i+1})} N_{i+1}^{n-1}(t).$$

Em caso de nós múltiplos, as funções B-splines $N_i^n(t)$ são definidas pela mesma fórmula recursiva e a convenção

$$N_i^{r-1} = N_i^{r-1}/(u_{i+r} - u_i) = 0 \quad se\ u_i = u_{i+r}$$

onde r é a multiplicidade do nó.

Pela definição acima, as propriedades das funções B-spline são evidentes:

- $N_i^n(t)$ é polinomial por partes de grau n,
- $N_i^n(t)$ é positiva em (u_i, u_{i+n+1}),
- $N_i^n(t)$ é zero fora do intervalo$[u_i, u_{i+n+1}]$,
- $N_i^n(t)$ é contínua à direita.

Uma observação interessante é o caso particular onde $u_1 = \cdots = u_n = 0$ e $u_{n+1} = \cdots = u_{2n} = 1$. Pela fórmula recursiva para $N_0^n, \cdots, N_n^n$ e $t \in [0, 1)$ coincide com a fórmula para polinômios de Bernstein, ou seja,

$$N_i^n(t) = B_i^n(t) \quad \text{para } i = 0, \cdots, n \text{ e } t \in [0, 1).$$

3.3 O algoritmo de de Boor

O **algortimo de de Boor** é um uma algoritmo numericamente estável para avaliação de curvas spline na forma B-spline. Considere a combinação linear

$$s(t) = \sum_i c_i^0 N_i^n(t)$$

de funções B-spline de grau n em uma sequência de nós (u_i). Como qualquer soma pode ser convertida para uma soma bi-infinita com auxílio de termos nulos, é assumido, sem perda de generalidade, que a sequência de nós e a soma acima é bi-infinita. Como $N_i^n(t)$ tem suporte local, essa soma é finita para qualquer t. Em particular, se $t \in [u_n, u_{n+1})$,

então $s(t)$ pode ser escrito como

$$s(t) = \sum_{i=0}^{n} c_i^0 N_i^n(t).$$

Usando a recursão das funções B-spline repetidamente e coletando seus termos, é obtido que

$$\begin{aligned} s(t) &= \sum_{i=0}^{n} c_i^0 N_i^n(t) \\ &= \sum_{i=1}^{n} c_i^1 N_i^{n-1}(t) \\ &\vdots \\ &= \sum_{i=n}^{n} c_i^n N_i^0(t) = c_n^n, \end{aligned}$$

onde c_i^j é dado pela combinação afim

$$c_i^j = (1-\alpha)c_{i-1}^{j-1} + \alpha c_i^{j-1}, \ \alpha = \alpha_i^{n-j} = \frac{t-u_i}{u_{i+n+1-j} - u_i}.$$

É possível notar que $\alpha \in [0,1]$ pois $t \in [u_n, u_{n+1})$, isto é, as combinações afins são convexas.

Uma importante consequência do algoritmo de Boor é que a curva B-spline $s(t)$ em cada intervalo nodal (intervalo entre nós) é uma combinação afim convexa dos $n+1$ coeficientes consecutivos $c_i = c_i^0$. Portanto, se c_i representa pontos de um espaço afim, então $s(t)$ também é um ponto. Por essa razão, os c_i são chamados de pontos de controle de $s(t)$.

Pela construção, também é possível concluir que

$$\sum_{i=0}^{n} N_i^n(t) = 1, \text{ para } t \in [u_n, u_{n+1}),$$

ou seja, as funções B-spline formam uma partição da unidade.

Observação: para um dado $t \in \mathbb{R}$, o algortimo de Boor aplicado, como já descrito, para $c_0^0, \ldots, c_n^0$ não computa $s(t)$ em geral, mas uma $s_n(t)$ polinomial que coincide com $s(t)$ no intervalo $[u_n, u_{n+1})$.

Generalizando para $t \in [u_j, u_{j+1})$, $j \geq n$, aplicando o mesmo raciocínio, as funções B-splines $N_i^n(t)$ são não nulas para $u_i \leq t \leq u_{i+n+1}$. Assim,

$$u_i \leq u_j \ \text{ e } \ u_{j+1} \leq u_{i+n+1} \Rightarrow j - n \leq i \leq j$$

e portanto,

$$s(t) = \sum_{i=j-n}^{j} c_i^0 N_i^n(t).$$

Então todo polinômio de grau n pode ser escrito, em $[u_j, u_{j+1})$, como combinação linear das funções B-spline $N^n_{j-n}(t), \dots, N^n_j(t)$. Além disso, como as funções são linearmente independentes no intervalo e o número de coeficientes não nulos c_i é $n+1$, igual à dimensão do espaço de polinômios nesse intervalo, a combinação linear é única.

Seja $S_i(a_1 \dots a_n)$ a forma polar (seção 1.7) de s_i (polinômio de grau n no intervalo $[u_i, u_{i+1})$) que coincide com $s(t)$ no intervalo $[u_i, u_{i+1})$. Então as formas gerais dos pontos de controle de s são dadas por

$$c_i = S_j(u_{i+1}, \dots, u_{i+n}),\ i = j-n, \dots, j.$$

Para a prova, seja

$$p^r_i = S_j(u_{i+1}, \dots, u_{i+n-r}, \underbrace{t, \dots, t}_{r})$$

e

$$t = (1-\alpha)u_i + \alpha u_{i+n-r+1}.$$

Como S_j é multiafim e simétrica, segue

$$p^r_i = (1-\alpha)p^{r-1}_{i-1} + \alpha p^{r-1}_i,\ \alpha = \alpha^{n-r}_i = \frac{t-u_i}{u_{i+n-r+1}-u_i}$$

e, em particular,

$$p^0_i = S_j(u_{i+1}, \dots, u_{i+n}) \quad \text{e} \quad p^n_j = s_j(t).$$

Observação 2: no intervalo $[u_n, u_{n+1}]$, as funções B-spline $N^n_0(t), \dots, N^n_n(t)$ formam uma base para o espaço de todos os polinômios de grau até n.

Observação 3: os segmentos da curva B-spline s_i (polinômio de grau n no intervalo $[u_i, u_{i+1})$) determinam os pontos de controle $c_{i-n}, \dots, c_i$. Por outro lado, cada ponto c_k é determinado por qualquer segmento $s_k, \dots, s_{k+n}$, isto é,

$$c_i = S_i(u_{i+1}, \dots, u_{i+n}) = \cdots = S_{i+n}(u_{i+1}, \dots, u_{i+n}),$$

onde $S_i(u_{i+1}, \dots, u_{i+n})$ é a forma polar (ou *blossom*) da $s_i(t)$.

Observação 4: a prova acima mostra que o polinômio simétrico $S_n(a_1, \dots, a_n)$ pode ser computado pela generalização do algoritmo de Boor. Para isso, basta substituir $\alpha = \alpha(a)$ na fórmula recursiva

$$\alpha(a_r) = \frac{a_r - u_i}{u_{i+n-r+1} - u_i}.$$

Se k das n variáveis $a_1, \dots, a_n$ forem nós, então somente $n-k$ passos de recursão serão necessários para computar $S_j(a_1, \dots, a_n)$.

3.3.1 Derivadas e suavidade

A derivada dos segmentos polinomiais s_n das curvas B-spline podem ser escritos como

$$s_n'(t) = \sum_{i=0}^{n} d_i N_i^{n-1}(t), \;\; t \in [u_n, u_{n+1}),$$

onde os d_i podem ser expressos em termos de c_i. Seja $S_n'(a_2, \dots, a_n)$ a forma polar de $s_n'(t)$ e seja a direção $\Delta = u_{i+n} - u_i$ dada pelo suporte da função B-spline $N_i^{n-1}(t)$. Então segue que

$$\begin{aligned} d_i &= S_n'(u_{i+1}, \dots, u_{i+n-1}) \\ &= \frac{n}{\Delta} S_n'(\Delta, u_{i+1}, \dots, u_{i+n-1}) \\ &= \frac{n}{u_{i+n} - u_i}(c_i - c_{i-1}). \end{aligned}$$

Como d_i não depende do intervalo nodal $[u_n, u_{n+1})$, a derivada da curva B-spline s pode ser escrita para qualquer $t \in \mathbb{R}$ como

$$s'(t) = \sum_i \frac{n}{u_{i+n} - u_i} \nabla c_i N_i^{n-1}(t),$$

onde $\nabla c_i = c_i - c_{i-1}$ denota a primeira diferença regressiva (calcula a variação em relação ao ponto anterior).

Uma curva B-spline de grau n, s, é contínua em todo nó de multiplicidade n. Dado $u_0 < a_1 = \cdots = a_n < a_n + 1$, então segue da observação 3 que

$$\begin{aligned} s_0(u_1) &= S_0(u_1, \dots, u_n) = c_0 \\ &= S_n(u_1, \dots, u_n) \\ &= s_n(u_n). \end{aligned}$$

Então, se u_i é um nó de multiplicidade r, a $(n-r)$-ésima derivada de s é contínua em u_i. Em outras palavras, a curva B-spline satisfaz o critério de suavidade dos splines.

3.4 Definição das curvas B-spline

Uma curva B-spline de ordem $n+1$ (ou grau n) é definida

- pelo grau n de cada segmento (sempre o mesmo) da curva polinomial,
- pela sequência de $m+1$ nós (vetor nó) $u_0, u_1, \cdots, u_m$ e $u_i \leq u_{i+1}$,
- pelo polígono de controle $c_0, \dots, c_L$, com $L = m - n - 1$, ou seja, $m - n$ pontos de controle (calculados usando a forma polar se estiver procurando a forma B-spline de um spline).

A curva B-spline é então definida como

$$s(t) = \sum_{i=0}^{m-n-1} c_i N_i^n(t).$$

3.5 Forma de Bézier das curvas B-spline

Como essas curvas consistem de uma junção de segmentos polinomiais, cada um destes pode ser escrito na forma de curva de Bézier. Dada uma curva B-spline $s(t)$, $s_i(t)$ é a restrição da curva para o intervalo $I = [u_i, u_{i+1}]$ e S_i a sua forma polar. Então os pontos de Bézier da curva $s_i(t)$ com $t \in [u_i, u_{i+1}]$ são dados por

$$b_k^I = S_i(\underbrace{u_i, \dots, u_I}_{n-k}, \underbrace{u_{I+1}, \dots, u_{I+1}}_{k}), \ k = 0, \dots, n.$$

3.6 Propriedades da curva B-spline

Pela construção vista anteriormente, as propriedades básicas das funções B-spline e das curvas B-splines podem ser inferidas.

- Funções B-spline de grau n com sequência de nós que não se anulam em nenhum intervalo nodal são linearmente independentes nesse intervalo.
- As funções B-spline $N_0^n, \dots, N_k^n$ com os nós $u_0, \dots, u_{k+n+1}$ formam uma base para todos os splines de grau n com suporte em $[u_0, u_{k+n+1}]$ e os mesmos nós.
- Similarmente, as funções B-spline $N_0^n, \dots, N_k^n$ nos nós $u_0, \dots, u_{k+n+1}$ restritos ao intervalo $[u_n, u_{k+1})$ formam uma base para todos os splines de grau n restritos ao mesmo intervalo.
- As funções B-spline de grau n formam uma partição da unidade, isto é,

$$\sum_{i=0}^{k} N_i^n(t) = 1, \text{ para } t \in [u_n, u_{k+1}).$$

- Uma curva B-spline $s(t)$, $t \in [u_n, u_{k+1}]$ de grau n com nós extremos de multiplicidade n,

$$(u_0 =)u_1 = \cdots = u_n \quad \text{e} \quad u_{k+1} = \cdots = u_{k+n}(= u_{k+n+1})$$

  tem os mesmos pontos extremos e tangentes extremas dos polígono de controle.
- Os nós extremos u_0 e a_{k+n+1} não têm influência sobre N_o^n nem N_k^n no intervalo $[u_n, u_{k+1}]$.
- As função B-spline são positivas no interior de seu suporte

$$N_i^n(t) > 0 \quad \text{para} \quad t \in (u_i, u_{i+1}).$$

- As funções B-spline satisfazem a fórmula recursiva de Boor, Masnfield e Cox

$$N_i^n(t) = \alpha_i^{n-1} N_i^{n-1}(t) + (1 - \alpha_{i+1}^{n-1}) N_{i+1}^{n-1}(t),$$

  onde $\alpha_i^{n-1} = (t - u_i)/(u_{i+n} - u_i)$ representa o parâmetro local no suporte de N_i^{n-1}.

- A derivada de uma única função B-spline é dada por

$$\frac{d}{dt} N_i^n(t) = \frac{n}{a_{i+n} - a_i} N_i^{n-1}(t) - \frac{n}{a_{i+n+1} - a_{i+1}} N_{i+1}^{n-1}(t).$$

- Curvas de Bézier são casos especiais de curvas B-spline.

As curvas B-spline também mantêm propriedades importantes já vistas nas curvas de Bézier.

- **Invariância afim:** A representação de splines por B-splines (curvas B-spline) é invariante em relação a transformações afins.
- **Propriedade do fecho convexo:** Qualquer segmento s_j no intervalo $[u_j, u_{j+1})$ de grau n de uma curva B-spline está no fecho convexo dos $n + 1$ pontos de controle $c_{j-n}, \ldots, c_j$.

No entanto, a conclusão mais importante é aquela do **controle local da curva**, ou seja, a mudança da posição ou adição de pontos de controle à curva só altera uma parte limitada dela.

3.7 Limitações da curva B-spline

A motivação para a continuação dos estudo de outras curvas vem de uma limitação das curvas B-spline. Mesmo com a capacidade de criação de curvas de forma livre complexas, as curvas B-spline não são capazes de representar curvas mais simples como círculos, elipses ou hipérboles com exatidão. Mais especificamente, das cônicas, somente a parábola é um caso especial de curva B-spline (na verdade, é uma curva de Bézier). Essa limitação vem do fato de as curvas B-spline serem polinomiais e as cônicas citadas serem equações racionais (usam divisão),

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \text{ (elipse, círculo)},$$
$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \text{ (hipérbole)}.$$

Todas as curvas cônicas podem ser representadas usando funções racionais, que são definidas como razões de polinômios. Na verdade, elas são representadas por funções racionais da forma

$$x(t) = \frac{X(t)}{W(t)}, \; y(t) = \frac{Y(t)}{W(t)}, \tag{3.2}$$

onde $X(t)$, $Y(t)$ e $W(t)$ são polinômios e cada função das coordenadas tem o mesmo denominador.

Capítulo 4

Curvas NURBS

No início, NURBS eram utilizadas somente em pacotes de CAD para empresas automotivas e, posteriormente, tornaram-se parte de programas gráficos. A interpretação das curvas e superfícies NURBS (Capítulo 8) só estiveram disponíveis em workstations em 1989. Em 1993, a primeira NURBS interativa para PCs foi desenvolvida por CAS Berlim, uma pequena empresa iniciante, em cooperação com a Universidade Técnica de Berlim. Hoje em dia, a tecnologia das NURBS está presente na maioria dos programas gráficos.

Como comentado anteriormente, o acrônimo NURBS é a abreviação de B-splines racionais não uniformes. No entanto, o uso da expressão "não uniforme" é um pouco enganadora, uma vez que essas curvas podem ter um vetor nó uniforme (distância entre nós consecutivos é igual). O termo novo mais importante é o "racional", ele surge da característica dessas curvas adicionarem um parâmetro de peso racional w_i aos pontos de controle. Assim, as curvas NURBS são generalizações das curvas B-spline e das curvas de Bézier. B-splines são curvas NURBS com pesos iguais ou pesos "1", também chamadas de curvas B-splines puras, simples ou não racionais.

As cônicas podem ser expressadas por curvas NURBS. Ao desenvolver sua matemática, será visto que as curvas NURBS em um espaço de dimensão $d = 2$ ou 3 não são nada mais que projeções centrais (Seção 1.8) de curvas B-spline não racionais que estão num espaço de dimensão $d + 1$.

4.1 Motivação: Cônicas como quadráticas racionais

Usamos a seguinte definição de cônicas: uma cônica em $\mathbb{E}^2$ é a projeção de uma parábola de $\mathbb{E}^3$ em um plano. Seja $\{z = 1\}$ esse plano. Como o estudo na seção será sobre curvas planares, é possível pensar nesse plano como uma cópia de $\mathbb{E}^2$. Assim, os pontos $(x, y, 1)$ são identificados como (x, y). A projeção é caracterizada como

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}.$$

É possível notar que o ponto (x, y) é a projeção de uma família interia de pontos: todo ponto na linha (wx, wy, w) é projetada em (x, y). Usamos a notação $(w\mathbf{x}, w)$ com $\mathbf{x} \in \mathbb{E}^2$ para (wx, wy, w).

Seja $c(t) \in \mathbb{E}^2$ um ponto de uma cônica. Então existem números reais w_0, w_1, w_2 e pontos b_0, b_1, b_2 tais que

$$c(t) = \frac{w_0 b_0 B_0^2(t) + w_1 b_1 B_1^2(t) + w_2 b_2 B_2^2(t)}{w_0 B_0^2(t) + w_1 B_1^2(t) + w_2 B_2^2(t)}. \tag{4.1}$$

Para provar a afirmação, seja $c(t) \in \mathbb{E}^2$ com $(c(t), 1)) \in \mathbb{E}^3$. Esse ponto é uma projeção do ponto $(w(t)c(t), w(t))$, que está na parábola 3D. A componente $w(t)$ desse ponto 3D deve ser uma função quadrática em t e ela pode ser expressa na forma de Bernstein:

$$w(t) = w_0 B_0^2(t) + w_1 B_1^2(t) + w_2 B_2^2(t).$$

Determinado $w(t)$, é possível escrever

$$w(t) \begin{bmatrix} c(t) \\ 1 \end{bmatrix} = \begin{bmatrix} c(t) \sum w_i B_i^2(t) \\ \sum w_i B_i^2(t). \end{bmatrix}$$

O lado esquerdo da equação denota uma parábola e portanto pode ser reescrito como

$$\sum_{i=0}^{2} \begin{bmatrix} p_i \\ w_i \end{bmatrix} B_i^2(t) = \begin{bmatrix} c(t) \sum w_i B_i^2(t) \\ \sum w_i B_i^2(t). \end{bmatrix}$$

com pontos $p_i \in \mathbb{E}^2$. Logo

$$\sum_{i=0}^{2} p_i B_i^2(t) = c(t) \sum_{i=0}^{2} w_i B_i^2(t),$$

e

$$c(t) = \frac{p_0 B_0^2(t) + p_1 B_1^2(t) + p_2 B_2^2(t)}{w_0 B_0^2(t) + w_1 B_1^2(t) + w_2 B_2^2(t)}.$$

Com $p_i = w_i b_i$ a prova está dada.

Os pontos b_i são o polígono de controle da cônica c, os valores w_i são os pesos dos correspondentes vértices do polígono. O polígono de controle da cônica é o polígono de controle da parábola 3D projetada em c.

A forma (4.1) é chamada de forma quadrática racional de uma cônica. Se todas os pesos forem iguais, a quadrática não-racional é recuperada, ou seja, a parábola.

No espaço projetivo, todas as cônicas são equivalentes: todas as tranformações projetivas levam cônicas em cônicas. Na geometria afim, as cônicas são classificadas em três classes: hipérboles, parábolas e elipses.

Dado uma cônica da forma padrão, o segmento complementar dela é obtido ao inverter o sinal de w_1. Se $c(t)$ é um ponto da cônica original e $\hat{c}(t)$ é um ponto do segmento complementar da cônica, é fácil verificar que b_1, $c(t)$ e $\hat{c}(t)$ são colineares. Assumindo que

$w_1 > 0$, então o comportamento de $\hat{c}(t)$ determina o tipo de cônica que ela é. Se $\hat{c}(t)$ não possui singularidades em $[0, 1]$, então ela é uma elipse. Se ela possui uma singularidade, ela é uma parábola. Por fim, se ela possui duas singularidades, ela é uma hipérbole.

As singularidades, correspondendo aos pontos no infinito de $\hat{c}(t)$, são determinados pela raízes reais do denominador $\hat{w}(t)$ de $\hat{c}(t)$. Elas são dadas por

$$t_{1,2} = \frac{1 + w_1 \pm \sqrt{w_1^2 - 1}}{2 + 2w_1}.$$

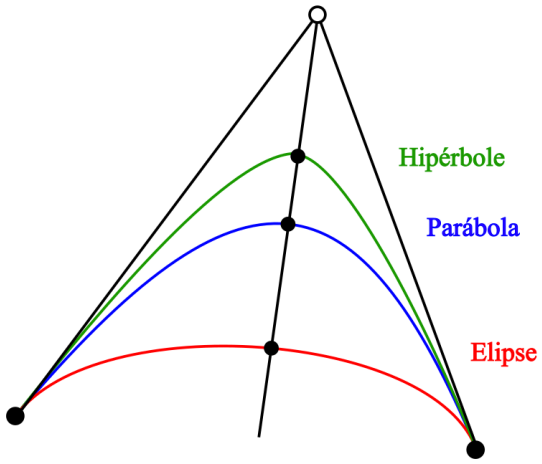

Figura 4.1: *Cônicas: dependem do peso* w_1.

Assim, uma cônica é uma elipse se $w_1 < 1$, uma parábola se $w_1 = 1$ e uma hipérbole se $w_1 > 1$. Para o caso especial do círculo, deixe o quadrático racional (com $w_1 < 1$) descrever o arco de um círculo. Pelas propriedades de simetria do círculo, o polígono de controle forma um triângulo isósceles. Sabendo o ângulo $\phi = \angle(b_0, b_1, b_2)$, o peso é dado por

$$w_1 = \text{sen}(\phi/2).$$

A justificativa é encontrada ao observar o ponto médio da curva (ou o *Shoulder point*). Dada uma curva Bézier racional de grau 2 de pontos de controle b_0, b_1, b_2 e pesos correspondentes $w_0 = 1, w_1, w_2 = 1$, ela pode ser escrita como

$$c(t) = \frac{w_0 b_0 B_0^2(t) + w_1 b_1 B_1^2(t) + w_2 b_2 B_2^2(t)}{w_0 B_0^2(t) + w_1 B_1^2(t) + w_2 B_2^2(t)} = \frac{b_0 B_0^2(t) + w_1 b_1 B_1^2(t) + b_2 B_2^2(t)}{B_0^2(t) + w_1 B_1^2(t) + B_2^2(t)}. \tag{4.2}$$

O ponto médio da curva (onde $t = 0.5$) é chamado de *shoulder point* S. Ele é dado então por

$$S = \frac{b_0 \frac{1}{4} + w_1 b_1 \frac{1}{2} + b_2 \frac{1}{4}}{\frac{1}{4} + w_1 \frac{1}{2} + \frac{1}{4}} = \frac{b_0 + b_2 + 2w_1 b_1}{2 + 2w_1}.$$

Então, com S posicionado corretamente na cônica, ele deve satisfazer

$$\frac{||S - d_1||}{||S - M||} = \frac{1}{w_1},$$

onde $M = \frac{d_0+d_2}{2}$, ponto médio de d_0 e d_2.

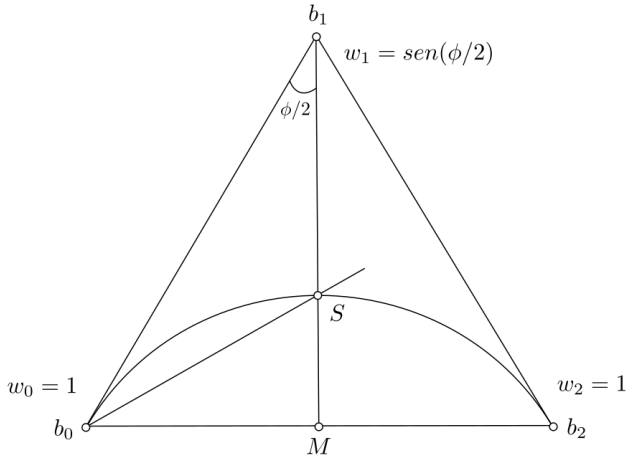

Figura 4.2: *Cálculo do peso w_1: S é o shoulder point, M é o ponto médio de d_0 e d_2.*

Sendo $\beta = \angle(b_1, b_0, M)$, pela simetria dos arcos b_0S e Sb_2, b_0S é a bissetriz de β. Pelo teorema da bissetriz

$$w_1 = \frac{||S - M||}{||S - d_1||} = \frac{||M - b_0||}{||b_1 - b_0||} = \cos(\theta)$$

com $\theta = \frac{\pi-\phi}{2}$,

$$w_1 = \cos(\frac{\pi - \phi}{2}) = \text{sen}(\phi/2).$$

O círculo todo pode ser representado pela junção dos arcos. Por exemplo, pela junção de três arcos iguais, os ângulos α iguais a 60 graus e, portanto, os pesos dos pontos de Bézier são $1/2$.

4.2 Motivação: Curvas de Bézier racionais

Cônicas podem ser expressas como curvas (de Bézier) quadráticas racionais e sua generalização para curvas racionais de grau maior é direta. Uma curva racional de Bézier de grau n em $\mathbb{E}^3$ é a projeção de uma curva de Bézier de grau n de $\mathbb{E}^4$ em um hiperplano $w = 1$. Pensando no hiperplano 4D como uma cópia de $\mathbb{E}^3$, um ponto de $\mathbb{E}^4$ é dado pelas coordenadas (x, y, z, w). Procedendo da mesma forma que foi feito para as cônicas, uma

curva de Bézier racional de grau n é dada por

$$c(t) = \frac{w_0 b_0 B_0^n(t) + \cdots + w_n b_n B_n^n(t)}{w_0 B_0^n(t) + \cdots + w_n B_n^n(t)}, \ c(t), b_j \in \mathbb{E}^3.$$

Os w_i são os pesos e b_i o polígono de controle. É a projeção do polígono controle 4D $(w_i c_i, w_i)$ da pré-imagem não racional de $c(t)$.

Com todos os pesos iguais a um, é obtida uma curva de Bézier não racional, caso em que o denominador é identicamente igual a um. Se w_i for negativo há singularidades, então só serão usados pesos não negativos. As curvas de Bézier racionais compartilham todas as propriedades que as não racionais possuem, como, por exemplo, invariância afim. Elas podem ser reescritas como

$$c(t) = \sum_{i=0}^{n} b_i \frac{w_i B_i^n(t)}{\sum_{i=0}^{N} w_i B_i^n(t)},$$

com a base de funções racionais

$$\frac{w_i B_i^n(t)}{\sum_{i=0}^{N} w_i B_i^n(t)}$$

somando um.

Os pesos w_i são normalmente usados como parâmetros de forma. Com o aumento do valor de um certo w_i, a curva é puxada na direção do correspondente ponto de controle c_i, mas o efeito da mudança de peso não é o mesmo que a mudança de posição de um ponto de controle.

Duas propriedades que são diferentes no caso racional em relação ao caso não racional. Primeiro, a invariância projetiva, ou seja, qualquer transformada projetiva de curva de Bézier racional pode ser obtida pela transformação projetiva de seu polígono de controle. A segunda é a propriedade de precisão linear: se os pontos de controle são colineares e os pesos forem ajustados corretamente então a curva será um segmento de reta perfeito. Dados c_i pontos de controle de Bézier distribuídos em uma reta

$$b_i = (1 - \alpha_i) b_0 + \alpha_i b_n, \ i = 0, \ldots, n$$

com α_i valores reais arbitrários, para que a curva trace o segmento $\overline{b_0 b_n}$ de reta de uma forma linear, $w_0 = 1$ e

$$w_i = \frac{i}{n+1-i} \frac{1 - \alpha_{i-1}}{\alpha_i}, \ i = 1, \ldots, n.$$

Definindo os pontos de peso q_i como

$$q_i = \frac{w_i b_i + w_{i+1} b_{i+1}}{w_i + w_{i+1}}.$$

Esses pontos são definidos pelos pesos e podem ser usados como parâmetros da forma

para a curva.

4.3 Definição da curva NURBS

Uma curva NURBS (B-spline racional não uniforme) 3D é a projeção pela origem de um curva B-spline não racional 4D no plano $w = 1$. O polígono de controle é dado pelos vértices $d_0, \dots, d_L$ (já visto que $L = m - n - 1$), com $d_i \in \mathbb{E}^3$, $w_0, \dots, w_L$ são os pesos associados a cada um dos pontos, respectivamente, e $N_i^n(t)$ são as funções B-splines vistas anteriormente de nós $u_0, \dots, u_m$. Um ponto $C(t)$ na curva NURBS de ordem n é dado por

$$C(t) = \sum_{i=0}^{L} d_i \frac{w_i N_i^n(t)}{\sum_{i=0}^{L} w_i N_i^n(t)}, \ u_0 \leq t \leq u_m.$$

Outra forma de escrever essa fórmula e mais usada é

$$C(t) = \sum_{i=0}^{L} d_i R_i^n(t),$$

onde

$$R_i^n(t) = \frac{w_i N_i^n(t)}{\sum_{i=0}^{L} w_i N_i^n(t)}$$

são as funções de base racionais. Pela construção é fácil ver que essas funções, assim como os polinômios de Bernstein e as funções B-spline, são uma partição da unidade,

$$\sum_{i=0}^{L} R_i^n(t) = 1.$$

4.4 Cônicas como caso especial de curvas NURBS

Para a representação de cônicas com curvas NURBS, são utilizados 3 pontos de controle, d_0, d_1 e d_2, e seus respectivos pesos, w_0, w_1 e w_2. Dependendo da mudança desses pesos, parábolas, hipérboles, elipses e círculos são criados.

- Parábola: pesos $w_0 = 1$, $w_1 = 1$ e $w_2 = 1$
- Hipérbole: pesos $w_0 = 1$, $w_1 > 1$ e $w_2 = 1$
- Elipse: pesos $w_0 = 1$, $w_1 < 1$ e $w_2 = 1$
- Círculo: pesos $w_0 = 1$, $w_1 = \text{sen}(\phi/2)$ e $w_2 = 1$, onde $\phi = \angle(d_0, d_1, d_2)$.

4.5 Propriedades da curva NURBS

As curvas NURBS mantêm propriedades importantes já vistas nas curvas de Bézier, em específico, aquelas vistas para as funções B-spline.

- **Invariância afim:** A curvas NURBS são invariantes em relação a tranformações afins.
- **Propriedade do fecho convexo:** Qualquer segmento s_j no intervalo $[u_j, u_{j+1})$ de grau n de uma curva NURBS está no fecho convexo dos $n + 1$ pontos de controle $d_{j-n}, \ldots, d_j$.
- **Propriedade de controle local:** se o ponto de controle d_i é movido ou há a mudança do valor de seu peso w_i, isso afeta somente uma parte da curva no intervalo $[u_i, u_{i+n+1})$. Ao aumentar o peso w_i de um ponto de controle d_i a curva se aproxima mais do ponto de controle, enquanto diminuir o peso faz com que a curva se afaste mais do ponto de controle. O comportamento é intuitivo e, pela derivação geométrica, segue imediatamente que a mudança de peso de um certo ponto de controle só afeta uma certa região de influência.

A curva NURBS adiciona a propriedade importante de invariância projetiva e capacidade de representação das cônicas com exatidão.

Capítulo 5

Curvas de Subdivisão

Curvas de subdivisão são geradas ao refinar iterativamente um polígono de controle preliminar até elas se tornarem curvas suaves no limite. A subdivisão já foi exemplificada no algoritmo de Casteljau para a geração de curvas de Bézier (Seção da subdivisão das curvas de Bézier). Pela propriedade de subdivisão é possível refinar iterativamente um polígono de controle para encontrar uma sequência de polígonos que convergem rapidamente para a curva de Bézier. Isso pode ser visto como o pocesso de *"corner cutting"*, a cada iteração cantos são cortados do polígono.

Essas curvas são polígonos definidos por seus pontos originais e o nível da subdivisão (ou de refinamento) k.

Nessa seção, os algoritmos de subdivisão abordados são: o algoritmo de Chaikin para geração de B-splines quadráticos, o algoritmo de Lane-Riesenfeld como a generalização do algoritmo de Chaikin que produz, no limite, uma curva B-spline uniforme de grau n, e o esquema de quatro pontos para produção de curvas de subdivisão interpoladas.

5.1 Algoritmo de Chaikin

Em 1974, George Chaikin, introduziu o primeiro algoritmo recursivo de subdivisão para geração de curvas suaves em *An algorithm for high-speed curve generation* (Chaikin, 1974).

Considerando quatro pontos P_0, P_1, P_2 e P_3 em $\mathbb{E}^2$ ou $\mathbb{E}^3$ formando o polígono original aberto. Sendo

$$\begin{gathered} R_0 = \tfrac{1}{4}P_0 + \tfrac{3}{4}P_1, \\ Q_1 = \tfrac{3}{4}P_1 + \tfrac{1}{4}P_2 \quad \text{e} \quad R_1 = \tfrac{1}{4}P_1 + \tfrac{3}{4}P_2 \\ Q_2 = \tfrac{3}{4}P_2 + \tfrac{1}{4}P_3, \end{gathered}$$

os pontos $P_0, R_0, Q_1, R_1, Q_2, P_3$ formam o novo polígono (cortando fora os triângulos R_0, P_1, Q_1 e R_1, P_2, Q_2) de seis pontos. Para a próxima iteração, o procedimento é o mesmo,

renomeando os pontos para $P_0^* = P_0, P_1^* = R_0, P_2^* = Q_1, P_3^* = R_1, P_4^* = Q_2, P_5^* = P_3$ e então

$$\begin{gathered} R_0 = \frac{1}{4}P_0^* + \frac{3}{4}P_1^*, \\ Q_i = \frac{3}{4}P_i^* + \frac{1}{4}P_{i+1}^* \quad \text{e} \quad R_i = \frac{1}{4}P_i^* + \frac{3}{4}P_{i+1}^*, i \in 1,2,3,4 \\ Q_4 = \frac{3}{4}P_4^* + \frac{1}{4}P_5^*. \end{gathered}$$

A curva gerada começa no ponto P_0, tangente ao segmento de reta $\overline{P_0P_1}$ nesse ponto, intersecta o ponto médio de $\overline{P_1P_2}$ (tangente ao segmento de reta $\overline{P_1P_2}$ no ponto médio) e termina no ponto P_3, tangente ao segmento de reta $\overline{P_2P_3}$ nesse ponto.

O algoritmo de Chaikin para a geração de curvas B-splines quadráticos uniformes é feito a partir da generalização na construção acima. Dado $N+1$ pontos $P_0, \ldots, P_N$ em $\mathbb{E}^2$ ou $\mathbb{E}^3$, cada iteração do algoritmo de Chaikin para a geração de curva B-spline quadrática aberta uniforme no seu limite é dada por

$$\begin{gathered} R_0 = \frac{1}{4}P_0 + \frac{3}{4}P_1, \\ Q_i = \frac{3}{4}P_i + \frac{1}{4}P_{i+1} \quad \text{e} \quad R_i = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}, i \in 1, \ldots, N-1 \\ Q_{N-1} = \frac{3}{4}P_{N-1} + \frac{1}{4}P_N, \end{gathered}$$

com a renomeação dos pontos ao final.

Para a curva B-spline quadrática fechada uniforme o mesmo é válido, mas não é necessário de preocupar com segmento inicial e final:

$$Q_i = \frac{3}{4}P_i + \frac{1}{4}P_{i+1} \quad \text{e} \quad R_i = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}, i \in 0, \ldots, N.$$

Observe que, havia $N+1$ pontos no início da iteração ao final há $2N+2$ pontos e o novo N deve ser $2N+1$.

5.2 Algoritmo de Lane-Riesenfeld

Como uma generalização do algoritmo de Chaikin, por Judson Lane e Richard Riesenfeld (1980) em *A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces* (Lane e Riesenfeld, 1980), o algoritmo de Lane-Riesenfeld produz no limite curvas B-spline uniformes de grau n. O algoritmo de Chaikin pode ser visto como um procedimento de "divisão e média"(*split and average*): a cada iteração, o polígono é dividido no (acrescentado um) ponto médio de cada um dos seus lados, obtendo um novo polígono intermediário. Então achando os pontos médios de cada um dos lados desse polígono intermediário e os conectando, é formado o polígono final da iteração (com o cuidado dos segmentos inicial e final se for uma curva aberta).

Lane-Riesenfeld realizou a generalização do processo de "divisão e média", para "divisão e $(n-1)$-média"(dividir os lado do polígono somente uma vez e $n-1$ vezes o processo de achar o ponto médio e conectar eles) produzindo no limite curvas B-spline uniformes de grau n.

Dados o grau n e $P_0, P_1, \ldots, P_{N-1}$, N pontos originais do polígono ($n-1 \leq N$), se o polígono for fechado gerará uma curva B-spline de grau n uniforme fechada. O processo

de "divisão e $(n-1)$-média"pode ser reescrito como

- Refinamento ("divisão"): cada ponto P_i é duplicado,

$$\text{Pontos originais: } P_0, P_1, \dots, P_{N-1} \Rightarrow \text{Pontos refinados: } P_0, P_0, P_1, P_1, \dots, P_{N-1}, P_{N-1}.$$

- Média: a média é aplicada n vezes aos $2N$ pontos refinados,

$$\begin{gathered}\text{Primeira iteração: } P_0, \tfrac{P_0+P_1}{2}, P_1, , \dots, \tfrac{P_{N-2}+P_{N-1}}{2}, P_{N-1}, \tfrac{P_{N-1}+P_0}{2}, \\ \text{Segunda iteração (até aqui Chaikin): } \tfrac{P_0+\frac{P_0+P_1}{2}}{2}, \tfrac{\frac{P_0+P_1}{2}+P_1}{2}, \dots, \tfrac{P_{N-1}+\frac{P_{N-1}+P_0}{2}}{2}, \tfrac{\frac{P_{N-1}+P_0}{2}+P_0}{2}, \\ \vdots \\ \text{n-ésima iteração: } Q_0, \dots, Q_{2N-1}\end{gathered}$$

Q_i, com $i \in 0, \dots, 2N-1$ formando o novo polígono. Observe que depois de duplicar para $2N$ o algoritmo não gera mais pontos, mas os reposiciona mais perto do B-spline esperado.

Já se o polígono for aberto gerará uma curva B-spline de grau n uniforme aberta e é necessário cuidados adicionais, assim como no algoritmo de Chaikin. Para os pontos internos do polígono a regra continua a mesma, mas para os extremos são mantidos:

- Refinamento ("divisão"): cada ponto P_i, com $i = 1, \dots, N-2$, é duplicado, já P_0 e P_{N-1} se mantém

$$\text{Pontos originais: } P_0, P_1, \dots, P_{N-1} \Rightarrow \text{Pontos refinados: } P_0, P_1, P_1, \dots, P_{N-2}, P_{N-2}, P_{N-1}.$$

- Média: a média é aplicada n vezes aos pontos internos refinados, as extremidades sempre mantidas,

$$\begin{gathered}\text{Primeira iteração: } P_0, \tfrac{P_0+P_1}{2}, P_1, \dots, \tfrac{P_{N-2}+P_{N-1}}{2}, P_{N-1}, \\ \text{Segunda iteração: } P_0, \tfrac{P_0+\frac{P_0+P_1}{2}}{2}, \tfrac{\frac{P_0+P_1}{2}+P_1}{2}, \dots, \tfrac{\frac{P_{N-2}+P_{N-1}}{2}+P_{N-1}}{2}, P_{N-1}, \\ \vdots \\ \text{n-ésima iteração: } P_0, Q_1, \dots, Q_{2N-4+n}, P_{N-1}\end{gathered}$$

$P_0, Q_1, \dots, Q_{2N-4-n}, P_{N-1}$ formando o novo polígono. Observar que, depois de duplicar para $2N$, cada iteração gera um ponto a mais.

5.3 Esquema de quatro pontos

O esquema de quatro pontos foi desenvolvido pelos pesquisadores Nira Dyn, John A. Gregory e David Levin no artigo *A 4-point interpolatory subdivision scheme for curve design* (Dyn *et al.*, 1987) em 1983.

Novamente, um conjunto de pontos gerará uma sequência de polígonos que no limite produz uma curva suave, mas agora exigi-se que a curva interpole os pontos dados. Primeiro,

cria-se um ponto novo P_i^* a partir dos quatro pontos originais P_{i-1}, P_i, P_{i+1} e P_{i+2}:

$$P_i^* = -\frac{1}{16}P_{i-1} + \frac{9}{16}P_i + \frac{9}{16}P_{i+1} - \frac{1}{16}P_{i+2}$$

Note que os coeficientes somarem 1 é uma propriedade importante para o esquema ser geométrico. Os coeficientes são derivados a partir da interpolação cúbica de quatro pontos.

$$P_i^* = -wP_{i-1} + (\frac{1}{2} + w)P_i + (\frac{1}{2} + w)P_{i+1} - wP_{i+2}$$

Com $w = 1/16$, o esquema é o visto acima, mas nem todos os valores de w funcionam. O algoritmo dos quatro pontos é dado pela forma geral acima com $0 < w < 1/8$.

Assim, esse esquema mantém os pontos originais e adiciona um, em cada iteração, entre cada (P_i, P_{i+1}). Dados $P_0, \dots, P_{N-1}$,

- Pontos originais mantidos:
$$Q_{2i} = P_i,$$
- Pontos adicionados a cada $\overline{P_iP_{i+1}}$:
$$Q_{2i+1} = -\frac{1}{16}P_{i-1} + \frac{9}{16}P_i + \frac{9}{16}P_{i+1} - \frac{1}{16}P_{i+2}.$$

Para o caso fechado a curva vai estar bem definida a cada iteração (usando $P_{N+i} = P_i$ e $P_{-i} = P_{N-i}$), mantendo os pontos iniciais e gerando N novos. Se a o polígono original $P_0, \dots, P_{N-1}$ for aberto, novamente é necessária uma adaptação. Um caso comum é a extrapolação linear (assumir que os pontos que faltam estão alinhados com os pontos extremos conhecidos),

$$P_{-1} = 2P_0 - P_1, \; P_N = 2P_{N-1} - P_{N-2}.$$

As demonstrações das construções de curvas suaves no limite dos algoritmos podem ser vistas no livro *Subdivision Methods for Geometric Design: A Constructive Approach* (Warren e Weimer, 2001).

Superfícies de forma livre

A flexibilidade necessária para design 3D é dificilmente alcançada com superfícies como cilindros, cones, esferas, etc. Superfícies de forma livre (*freeform surfaces*) oferecem bem mais flexibilidade.

As *superfícies de Bézier e B-spline* serão desenvolvidas como extensões das suas curvas de forma livre correspondentes. Enquanto os métodos usados para a construção dessas duas superfícies são severamente restritivos em tipos topológicos, as superfícies de subdivisão superam essas limitações de uma forma simples e elegante. Seu uso foi o que iniciou a indústria de animação digital.

A abordagem utilizada para a compreensão das curvas é a do livro *Curves and Surfaces for CAGD: A Practical Guide* (Farin, 2001). As motivações e problemas abordados são oferecidos pelo livro *Architectural geometry* (Pottmann *et al.*, 2015).

Capítulo 6

Superfícies de Bézier

6.1 Interpolação bilinear

Na seção 1.4 e no capítulo 2, a interpolação linear em $\mathbb{E}^3$ foi estudada e propridades derivadas dela foram usadas para o desenvolvimento de curvas de Bézier. Analogamente, é possível basear a teoria de **superfícies de Bézier por produto tensorial** no conceito de **interpolação bilinear**.

Sejam $b_{0,0}, b_{0,1}, b_{1,0}, b_{1,1}$ quatro pontos distintos de $\mathbb{E}^3$. O conjunto de todos os pontos $\mathbf{x} \in \mathbb{E}^3$ da forma

$$\mathbf{x}(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1} b_{i,j} B_i^1(u) B_j^1(v), \tag{6.1}$$

onde B_i^1 e B_j^1 são polinômios de Bernstein de grau um, é chamado de **paraboloide hiperbólico** pelos quatro pontos $b_{i,j}$.

Na forma matricial:

$$\mathbf{x}(u,v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}.$$

A forma acima abusa um pouco da notação ao colocar os pontos dentro da matriz, mas é uma boa visualização do que está acontecendo na interpolação. Como (6.1) é linear em ambos u e v e interpola os pontos originais, a superfície $\mathbf{x}$ é chamada de superfície de interpolação bilinear.

Uma superfície de interpolação bilinear pode ser vista como um mapeamento do quadrado da unidade $0 \leq u, v \leq 1$ no plano u, v. O quadrado da unidade é chamado de domínio do interpolante, enquanto a superfície $\mathbf{x}$ é a sua imagem. Um segmento de reta paralelo a uma das arestas do domínio corresponde a uma curva na sua imagem: ela é chamada de curva isoparamétrica. Toda curva isoparamétrica no paraboloide hiperbólico é uma reta e, portanto, paraboloides hiperbólicos são **superfícies regradas** (superfície gerada pelo movimento de uma reta). Em particular, a reta isoparamética $u = 0$ é mapeada

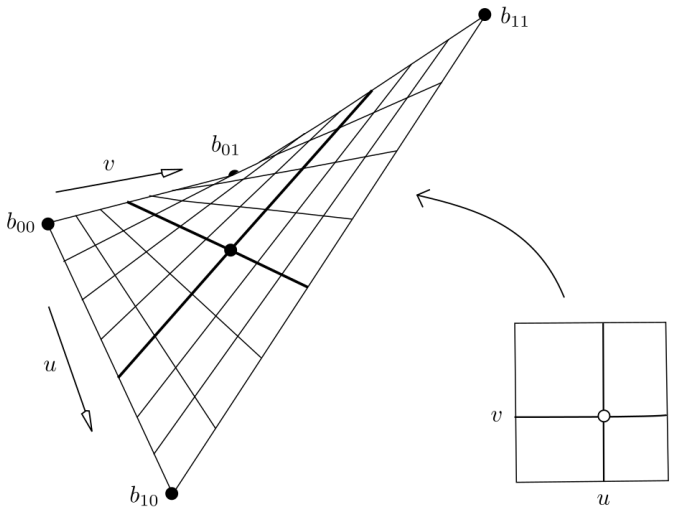

Figura 6.1: *Interpolação bilinear: paraboloide hiperbólico é definido por quatro pontos $b_{i,j}$.*

na reta que passa pelos ponto $b_{0,0}$ e $b_{0,1}$ (o argumento análogo vale para as outras curvas de fronteira).

Ao invés de avaliar o interpolante bilinear diretamente, é possível aplicar um processo de duas etapas, computando dois pontos intermediários

$$b_{0,0}^{0,1} = (1-v)b_{0,0} + vb_{0,1},$$
$$b_{1,0}^{0,1} = (1-v)b_{1,0} + vb_{1,1},$$

e o resultado final é obtido por

$$\mathbf{x}(u,v) = b_{0,0}^{1,1}(u,v) = (1-u)b_{0,0}^{0,1} + ub_{0,1}^{0,1}.$$

Isso equivale a computar os coeficientes da reta isomética v = constante primeiro e então avaliar essa reta isoparamética em u. Computando a reta isoparamética u = constante primeiro e depois avaliando esta em v, gera o mesmo resultado.

Como interpolação linear é uma trasformação afim e como na sua construção é utilizada interpolação linear nas direções u e v, é normal ver o termo **aplicação (transformação ou mapeamento) biafim**.

O termo paraboloide hiperbólico vem da geometria analítica. Considerando a superfície (não parametrizada) $z = xy$, ele pode ser interpretado como o interpolante bilinear dos quatro pontos

$$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}.$$

Se a superfície for intersectada com um plano paralelo ao xOy, a curva resultante é uma hipérbole. Já se ela for intersectada com um plano contendo o eixo z, a curva

resultante é uma parábola.

6.2 Algortimo de Casteljau

Assim como curvas de Bézier podem ser encontradas pela repetida aplicação da interpolação linear, a superfície de Bézier será encontrada a partir da repetição da interpolação bilinear.

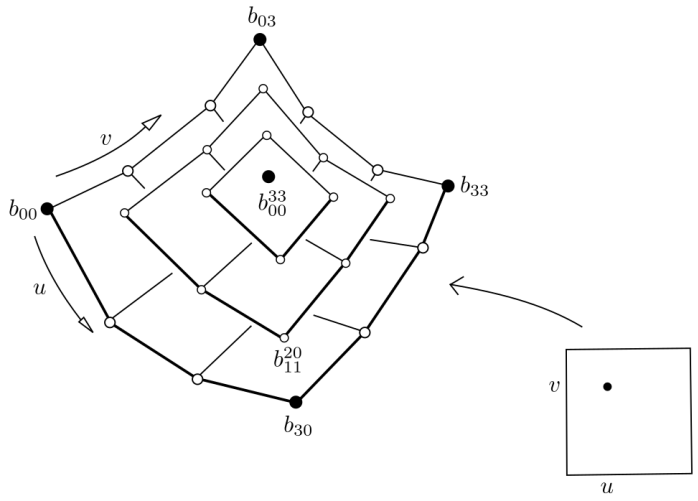

Figura 6.2: *Derivação do algoritmo de Casteljau para superfícies: o ponto na superfície é encontrado pela repetição da interpolação linear.*

Seja uma matriz retangular de pontos $b_{i,j}$ onde $0 \leq i, j \leq n$ e os parâmetros (u, v). O algoritmo seguinte gera um ponto na superfície determinada pela matriz de $b_{i,j}$.

Dados $\{b_{i,j}\}_{i,j=0}^{n}$ e $(u, v) \in \mathbb{R}^2$, seja

$$b_{i,j}^{r,r} = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} b_{i,j}^{r-1,r-1} & b_{i,j+1}^{r-1,r-1} \\ b_{i+1,j}^{r-1,r-1} & b_{i+1,j+1}^{r-1,r-1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix},$$
$$r = 1, \ldots, n$$
$$i, j = 0, \ldots, n-r$$

e $b_{i,j}^{0,0} = b_{i,j}$. Então $b_{0,0}^{n,n}(u, v)$ é o ponto com parâmetros (u, v) na superfície de Bézier $b^{n,n}$. A rede de pontos $b_{i,j}$ é chamada de **rede de Bézier** ou **rede de controle** da superfície $b^{n,n}$. Os pontos $b_{i,j}$ são chamados de pontos de controle ou pontos de Bézier, assim como no caso das curvas.

Na próxima seção, será possível ver superfícies com graus diferentes para u e v. Tais superfícies possuem redes de controle $\{b_{i,j}\}$ com $i = 0, \ldots, m$ e $j = 0, \ldots, n$. O algoritmo de Casteljau para estas superfícies existe, mas precisa de uma distinção de casos: pelo algoritmo proposto até agora, este não pode ser executado até encontrar o ponto da superfície. Depois de $k = min(m, n)$, a $b_{i,k}^{k,k}$ intermediária forma um polígono de controle

de curva. Para obter um ponto na superfície, é necessário seguir com o algoritmo de Casteljau univariado.

6.3 Produto tensorial

Uma definição intuitiva de uma superfície é: "uma superfície é o trajeto percorrido por uma curva que se desloca no espaço, modificando seu formato."

Para formalizar o conceito intuitivo, assumimos que a curva em movimento é de Bézier de grau constante m. A todo momento, a curva em movimento é determinada por um conjunto de pontos de controle. Cada ponto de controle original move-se pelo espaço ao longo de uma curva. Outra suposição feita é que esta curva é de Bézier e que todas as curvas por onde os pontos de controle atravessam têm o mesmo grau.

Formalizando, seja a curva inicial de Bézier de grau m:

$$b^m(u) = \sum_{i=0}^{m} b_i B_i^m(u).$$

Cada b_i move-se ao longo de uma curva de Bézier de grau n:

$$b_i = b_i(v) = \sum_{j=0}^{n} b_{i,j} B_j^n(v).$$

Combinando as duas equações, é possível obter o ponto $b^{m,n}(u,v)$ na superfície $b^{m,n}$ como

$$b^{m,n}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{i,j} B_i^m(u) B_j^n(v). \tag{6.2}$$

Com esta notação, a curva original $b^m(u)$ tem os pontos de Bézier $b_{i,0}$ com $i = 0, \ldots, m$. A definição de superfície de Bézier dada por (6.2) é equivalente à dada pelo algoritmo de Casteljau. Essa equação é obtida ao mover a curva isoparamétrica correspondente a $v = 0$, mas as outras curvas de fronteira também podiam ter sido usadas como a curva original.

Uma curva isoparamétrica arbitrária $\hat{v}$ = constante de uma superfície de Bézier $b^{m,n}$ é uma curva de Bézier de grau m em u e seus $m+1$ pontos de Bézier são obtidos ao avaliar todas as colunas da rede de controle em v = constante. A fórmula é dada por

$$b_{i,0}^{0,n}(\hat{v}) = \sum_{j=0}^{n} b_{j,j} B_j^n(\hat{v}), \;\; i = 0 \ldots, m.$$

Os coeficientes da reta isoparamétrica podem ser obtidos ao aplicar $m+1$ vezes o algortimo de Casteljau. Um ponto na superfície é então obtido ao aplicar-se mais uma vez o algoritmo de Casteljau.

Curvas isoparaméticas u = constante são tratadas de forma análoga. Note, no entanto, que outras retas no domínio são mapeadas em curvas de grau mais alto na superfície: elas são em geral de grau $n + m$.

6.4 Derivadas das Superfícies de Bézier

No caso das curvas, as derivadas foram encontradas ao diferenciar os pontos de controle. O mesmo será feito para as superfícies: as derivadas que serão consideradas serão as derivadas parciais $\partial/\partial u$ e $\partial/\partial v$. A derivada parcial é o vetor tangente de uma curva isoparamética e pode ser encontrada calculando-se:

$$\frac{\partial}{\partial u} b^{m,n}(u,v) = \sum_{j=0}^{n} \left[\frac{\partial}{\partial u} \sum_{i=0}^{m} b_{i,j} B_i^m(u) \right] B_j^n(v).$$

Os termos entre colchetes dependem somente de u e é possível aplicar a fórmula para a derivada da curva de Bézier já vista (2.1):

$$\frac{\partial}{\partial u} b^{m,n}(u,v) = m \sum_{j=0}^{n} \sum_{i=0}^{m-1} \Delta^{1,0} b_{i,j} B_i^{m-1}(u) B_j^n(v).$$

O operador de diferenças padrão é generalizado para dois pontos, o índice superior $(1,0)$ significa que a diferenciação é só feita no primeiro índice inferior: $\Delta^{1,0} b_{i,j} = b_{i+1,j} - b_{i,j}$. Calculando a derivada parcial em relação a v, o operador de diferenças que atua somente no segundo índice é: $\Delta^{0,1} b_{i,j} = b_{i,j+1} - b_{i,j}$. Assim,

$$\frac{\partial}{\partial v} b^{m,n}(u,v) = n \sum_{i=0}^{m} \sum_{j=0}^{n-1} \Delta^{0,1} b_{i,j} B_j^{n-1}(v) B_i^m(u).$$

Novamente, um problema de superfície pode ser quebrado em vários problemas univariados: para computar a derivada parcial em relação a u, é possível interpretar todas as colunas da rede de controle como curvas de Bézier de grau m e computar suas derivadas (avaliá-las no valor desejado u). Então, interpretar essas derivadas como os coeficientes de outra curva de Bézier de grau n e comutar seu valor em v.

É possível escrever as fórmulas para derivadas parciais de maior ordem:

$$\frac{\partial^r}{\partial u^r} b^{m,n}(u,v) = \frac{m!}{(m-r)!} \sum_{j=0}^{n} \sum_{i=0}^{m-r} \Delta^{r,0} b_{i,j} B_i^{m-r}(u) B_j^n(v) \tag{6.3}$$

e

$$\frac{\partial^s}{\partial v^s} b^{m,n}(u,v) = \frac{n!}{(n-s)!} \sum_{i=0}^{m} \sum_{j=0}^{n-s} \Delta^{0,s} b_{i,j} B_j^{n-s}(v) B_i^m(u).$$

Nessas equações, os operadores de diferenças são definidos por

$$\Delta^{r,0} b_{i,j} = \Delta^{r-1,0} b_{i+1,j} - \Delta^{r-1,0} b_{i,j}$$

e

$$\Delta^{0,s} b_{i,j} = \Delta^{0,s-1} b_{i,j+1} - \Delta^{0,s-1} b_{i,j}.$$

Com essas fórmulas é mais simples escrever o caso mais geral. As derivadas parciais mistas

para uma ordem arbitrária são:

$$\frac{\partial^{r+s}}{\partial u^r \partial v^s} b^{m,n}(u,v) = \frac{m!n!}{(m-r)!(n-s!)} \sum_{j=0}^{n-s} \sum_{i=0}^{m-r} \Delta^{r,s} b_{i,j} B_i^{m-r}(u) B_j^{n-s}(v),$$

com coeficientes $\Delta^{r,s} b_{i,j} \in \mathbb{E}^3$ (vetores).

A derivada parcial de uma superfície com valores em pontos é, por sua vez, uma superfície com valores vetoriais. É possível avaliá-la ao longo das retas isoparamétricas, das quais as quatro curvas da fronteira são as mais interessantes. Tal derivada, como por exemplo $\partial/\partial u|_{u=0}$, é chamada de derivada transversal à fronteira. Restringindo a equação (6.3) a $u = 0$, ficamos com

$$\frac{\partial^r}{\partial u^r} b^{m,n}(0,v) = \frac{m!}{(m-r)!} \sum_{j=0}^{n} \Delta^{r,0} b_{i,j} B_j^n(v). \tag{6.4}$$

Fórmulas similares são encontradas para os outros três lados. Portanto, as derivadas transversais à fronteira de ordem r, avaliadas ao longo da fronteira, dependem somente das $r+1$ linhas (ou coluna) dos pontos de Bézier perto da fronteira.

6.5 Propriedades das Superfícies de Bézier

A maioria das propriedades das superfícies de Bézier seguem diretamente das curvas de Bézier.

- **Invariância afim:** o algortimo de Casteljau consiste na repetição da interpolação bilinear e, possivelmente, de uma subsequente linear. Todas essas operações são invariantes em relação a transformações afins e, portanto, sua combinação também é. Além disso,

  $$\sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v) \equiv 1. \tag{6.5}$$

  Obs: no entanto, assim como no caso das curvas, as superfícies de Bézier não são invariantes em relação a transformações projetivas!

- **Fecho convexo:** para $0 \leq u, v \leq 1$, os termos $B_i^m(u) B_j^n(v)$ são não negativos. Assim, levando em consideração (6.5), então (6.2) é uma combinação convexa.

- **Curvas de fronteira:** as curvas de fronteira da superfície $b^{m,n}$ são curvas polinomiais. Seus polígonos de Bézier são dados pelos polígonos de fronteira da rede de controle. Em particular, todos os quatro cantos da rede de controle estão na malha.

6.6 Composição de superfícies de Bézier

Sejam $\mathbf{x}(u,v)$ e $\mathbf{y}(u,v)$ duas *patches* de superfícies (partes de superfícies maiores), definidas em $[u_{I-1}, u_I] \times [v_J, v_{J+1}]$ e $[u_I, u_{I+1}] \times [v_J, v_{J+1}]$, respectivamente. Elas são de classe C^r na curva de fronteira comum $\mathbf{x}(u_I, v) = \mathbf{y}(u_I, v)$ se todas as derivadas parciais de u

até a ordem r forem iguais, ou seja,

$$\left.\frac{\partial^r}{\partial u^r}\mathbf{x}(u,v)\right|_{u=u_I} = \left.\frac{\partial^r}{\partial u^r}\mathbf{y}(u,v)\right|_{u=u_I}.$$

Agora, supomos que as duas *patches* sejam dadas na forma de Bézier, que a rede de controle da *patch* "esquerda", $\mathbf{x}$, seja $\{b_{ij}\}$ com $0 \leq i \leq m, 0 \leq j \leq n$ e que a rede de controle da *patch* "direita", $\mathbf{y}$, seja $\{b_{ij}\}$ com $m \leq i \leq 2m, 0 \leq j \leq n$. Para fazer a transição de coordenadas locais da equação (6.4) para globais (u,v) é necessária a regra da cadeia:

$$\left(\frac{1}{\Delta_{I-1}}\right)^r \sum_{j=0}^{n} \Delta^{r,0} b_{m-r,j} B_j^n(v) = \left(\frac{1}{\Delta_I}\right)^r \sum_{j=0}^{n} \Delta^{r,0} b_{m,j} B_j^n(v),$$

onde $\Delta_I = u_{I+1} - u_I$. Como os $B_j^n(v)$ são linearmente independentes, é possível só comparar coeficientes:

$$\left(\frac{1}{\Delta_{I-1}}\right)^r \sum_{j=0}^{n} \Delta^{r,0} b_{m-r,j} = \left(\frac{1}{\Delta_I}\right)^r \sum_{j=0}^{n} \Delta^{r,0} b_{m,j}, \;\; j = 0, \ldots, n.$$

Essa é a condição C^r para curvas de Bézier, aplicada para todas as $n+1$ linhas da rede de Bézier composta. A condição C^r para superfícies de Bézier é: duas *patches* adjacentes são C^r ao longo da sua fronteira comum se, e somente se, todas as linhas de vértices da rede de controle podem ser interpretadas como polígonos de curvas por partes de Bézier C^r. A condição de suavidade é aplicada analogamente para a direção v.

A condição C^1 diz que para todo j, o polígono formado por $b_{0,j}, \ldots, b_{2m,j}$ é o polígono de controle de uma curva por partes de Bézier C^1. Para esse ser o caso, os três pontos $b_{m-1,j}$, $b_{m,j}$, $b_{m+1,j}$ devem ser colineares e manter a razão entre as distâncias para todos os j. A simples colinearidade não é suficiente, como já visto: superfícies compostas que possuem $b_{m-1,j}$, $b_{m,j}$, $b_{m+1,j}$ colineares para todos os j, mas não mantêm a razão entre as distâncias, não são C^1. Além disso, elas nem terão um plano tangente contínuo.

Suponha que um domínio retangular de uma *patch* de Bézier é subdividida em dois subretângulos por uma reta $u = \hat{u}$. A reta é mapeada em uma curva isoparamétrica no *patch*, que é então subdividido em duas *subpatches*. Essas duas *patches* sendo parte de uma superfície global, se juntam com continuidade C^n. Portanto, todas as linhas de pontos de controle são polígonos de controle de curvas de grau n, C^n por partes. As curvas estão relacionadas pela subdivisão univariável já vista (Seção 2.3).

No algoritmo de subdivisão encontrado é possível interpretar todas as linhas da rede de controle como polígonos de controle de curvas de Bézier. Subdividir cada uma das curvas em $u = \hat{u}$. Os pontos de controle resultantes formam duas redes de controle desejadas.

A subdivisão ao longo da reta isoparamétrica $v = \hat{v}$ é tratada analogamente. Para subdividir um *patch* em quatro *subpatches* geradas por duas retas isoparamétricas $u = \hat{u}$ e $v = \hat{v}$, o procedimento de subdivisão é aplicado duas vezes. Não importa em qual direção a subdivisão é feita primeiro.

Capítulo 7

Superfícies B-spline

As superfícies de Bézier herdam os problemas da curvas de Bézier: quando o grau delas é muito alto, a representação final fica longe da esperada pela malha de controle e, novamente, um ponto de controle tem efeito global. Para evitar esse problema, pode-se usar supefícies B-spline e NURBS.

As superfícies B-spline não racionais e NURBS têm papéis importantes em métodos de design de superfícies atualmente. Uma superfície B-spline paramétrica por produto tensorial pode ser escrita como

$$\mathbf{x}(u,v) = \sum_i \sum_j d_{ij} N_i^m(u) N_j^n(v), \tag{7.1}$$

onde é assumido que há uma sequência de nós na direção u e outra na direção v.

No caso das curvas B-spline, se os pontos extremos tivessem multiplicidade igual ao grau da curva, esses pontos de controle extremos B-spline (ou NURBS) também seriam pontos de controle Bézier. O mesmo é válido para o caso de superfícies: se os pontos de controle B-spline d_{ij}, para i ou j iguais a 0 ou 1, tiverem multiplicidade igual a de seus *patches*, então também são vértices de controle da rede da superfície por partes de Bézier.

Reescrevendo a equação (7.1) como

$$\mathbf{x}(u,v) = \sum_i N_j^n(v) \left[\sum_j d_{ij} N_i^m(u) \right],$$

é possível ver que para cada i, a soma entre colchetes descreve uma curva B-spline na variável u. Essa curva pode ser convertida para uma curva de Bézier pelo método visto na seção 3.5. Isso corresponde a interpretar as linhas da rede de controle de B-splines como polígonos B-spline univariados e, então, convertê-los para uma forma de Bézier por partes. Os pontos de Bézier encontrados podem ser interpretados como, coluna a coluna, polígonos B-spline, que podem novamente ser transformados em forma de Bézier um por um. A família final de polígonos de Bézier constitui a rede da superfície por partes de Bézier. O processo poderia ter sido feito primeiramente com as colunas e depois com as

linhas da rede de controle B-spline e o resultado seria o mesmo.

Assim como nas curvas, as superfícies B-spline podem ser fechadas ou abertas. No entanto, as superfícies B-splines podem ser fechadas de duas maneiras: é possível formar superfícies com a conectividade de um cilindro ou de um toro. Esse fenômeno é consequência das superfícies B-spline possuírem duas famílias de curvas, na direção u e na direção v, que podem ser fechadas ou abertas. Os três jeitos diferentes que as superfícies podem ser criadas são:

- Modo aberto para ambas: a superfície é quadrilateral,
- Modo fechado para uma e aberto para outra: a superfície parece um pedaço deformado de um cano (cilindro),
- Modo fechado para ambas: a superfície parece um toro deformado.

As três possuem diferentes topologias com as redes de controle onde um ou mais polígonos da fronteira se degeneram para um único ponto, mas pode-se modelar pedaços de superfícies com menos de quatro lados curvados. Nenhuma superfície de produto tensorial pode ter conectividade de um bitoro ou uma superfície mais complicada. Na verdade, nenhuma superfície com topologia de uma esfera é representável por uma superfície de produto tensorial, pelo menos não sem degenerações.

Capítulo 8

Superfícies NURBS

As superfícies de Bézier e B-splines podem ser generalizadas para suas contrapartes racionais de forma análoga a usada para o caso das curvas. Em outras palavras, defini-se uma superfície de Bézier racional ou NURBS como a projeção de um produto tensorial 4D de superfície Bézier ou B-spline. Assim, um *patch* de superfície de Bézier racional é dado pela forma

$$\mathbf{x}(u,v) = \frac{\sum_i \sum_j w_{i,j} b_{i,j} B_i^m(u) B_j^n(v)}{\sum_i \sum_j w_{i,j} B_i^m(u) B_j^n(v)},$$

e uma superfície NURBS é escrita como

$$\mathbf{s}(u,v) = \frac{\sum_i \sum_j w_{i,j} d_{i,j} N_i^m(u) N_j^n(v)}{\sum_i \sum_j w_{i,j} N_i^m(u) N_j^n(v)}. \tag{8.1}$$

Superfícies racionais são obtidas como projeções de *patches* de produtos tensoriais, mas não são em si *patches* de produtos tensoriais. Lembramos que superfícies por produto tensorial são da forma $\mathbf{x}(u,v) = \sum_i \sum_j c_{i,j} F_{i,j}(u,v)$, onde a base de funções $F_{i,j}$ pode ser expressa como produtos $F_{i,j}(u,v) = A_i(u)B_j(v)$. A base de funções para (8.1) é da forma

$$F_{i,j}(u,v) = \frac{w_{i,j} N_i^m(u) N_j^n(v)}{\sum_i \sum_j w_{i,j} N_i^m(u) N_j^n(v)}.$$

Pela estrutura do denominador, isso não pode, em geral, ser fatorado na forma requerida $F_{i,j}(u,v) = A_i(u)B_j(v)$.

Mesmo sem possuir a estrutura de produto tensorial, as superfícies racionais podem usar algoritmos de produtos tensoriais para sua manipulação. Por exemplo, o problema de achar a forma de uma superfície de Bézier racional por partes a partir de uma superfície B-spline bicúbica racional. Só é necessário converter cada linha da rede de controle B-spline para uma curva de Bézier cúbica racional por partes e então repetir o processo para cada coluna da rede resultante.

Outro exemplo, consideramos o problema de extrair as curvas isoparamétricas de

uma superfície de Bézier racional. Supondo que a curva corresponde a $v = \hat{v}$, é possível interpretar todas as colunas da rede de controle como polígonos de controle e avaliar cada um em $\hat{v}$, usando o algortimo de Casteljau racional, por exemplo. Para computar os pesos, todos os pontos obtidos e seus pesos podem ser interpretados como um polígono de controle de Bézier da curva isoparamétrica desejada.

Topologias mais complexas, como superfícies fechadas, não podem ser modeladas com uma única superfície NURBS. Juntar superfícies NURBS para obter uma superfície mais complexa é difícil e imprático. Esse dilema topológico é resolvido com superfícies de subdivisão.

Capítulo 9

Malhas poligonais (ou Meshes)

Até agora, foram discutidos alguns tipos de superfícies suaves. Em casos onde as formas desejadas não são simples, como cones ou cilindros, há o frequente uso de **malhas poligonais** (ou *polygon meshes*).

Uma malha poligonal é uma coleção de vértices (pontos), arestas e faces (formada por um conjunto de vértices) que definem um objeto tridimensional. As faces são polígonos fechados, normalmente triângulos, quadriláteros (*quads*) ou polígonos convexos. Elas se conectam ao longo das arestas e descrevem aproximadamente a forma de uma superfície suave.

Em animações 3D, jogos e modelagens 3D em geral o que se vê são, na maioria das vezes, malhas poligonais renderizadas de forma suave. Elas são onipresentes em gráficos, mas também são usadas em peso em simulações de engenharia.

9.1 Geometria e conectividade

Ao lidar com malhas poligonais, é necessário observar o conceito de conectividade ou a topologia da malha poligonal. A ideia deste conceito é a nomeação dos vértices da malha poligonal e a informação da forma como esses se conectam para criar as faces e as arestas.

Malhas poligonais com a mesma conectividade podem ter formatos diferentes. Só é necessário mudar as coordenadaas dos vértices (com certa limitação) e manter toda a informação da conectividade.

O maior número de faces gera maior liberdade no design. No entanto, isso pode ser um fardo e estratégias são necessárias para a geração dessas malhas poligonais. A estética tem um papel crucial na busca dessas estratégias.

A representação de uma mesma geometria por uma malha poligonal mais balanceada (melhor distribuição e qualidade dos polígonos), pode necessitar a mudança de conectividade. Algoritmos em *design* de malhas poligonais são basicamente formas inteligentes de *design*/mudança de conectividade e geometria.

Especificando um pouco mais sobre geometria e conectividade, entre os formatos de arquivos, o formato OBJ comumente usado em programas de modelagem 3D possui:

- uma lista de coordenadas (x, y, z) de cada vértice do objeto. A sua ordenação define a numeração/nomeação dos vértices,
- comandos de face. Estes especificam quais e em que ordem os vértices representam as faces da malha.

Algumas limitações serão importantes na construção das malhas poligonais. Entre elas, a exclusão das "junções em T", onde duas faces encontram a mesma aresta em outra face. Outra é evitar a geração de triângulos degenerados (vértices colineares): eles não geram faces e, portanto, são potenciais causas nas falhas dos programas que tenham esse tipo de malha poligonal como entrada.

Malhas poligonais são representações discretas de superfícies. Elas generalizam o fato de um polígono ser usado como uma representação discreta de uma curva suave (como já visto em curvas de subdivisão).

9.2 Malhas quadrilaterais (quads)

Para obter malhas poligonais que representem superfícies, é primeiro necessário pensar nas malhas representando um plano. A forma mais simples de tesselar um plano com quadriláteros é usando quadrados (ou retângulos) arranjados de forma regular, cada quatro deles se conectando em um vértice. Uma malha quadrilateral bem formada terá a mesma conectividade que esta malha planar especial. Num vértice interior (um vértice que não está na fronteira da malha), exatas quatro faces, quatro arestas se encontram e a valência do vértice é quatro. Em geral, a valência de um vértice interior é o número de arestas que se encontram nele (a mesma quantidade de faces que passam pelo vértice).

Em uma malha quadrilateral, um vértice interior de valência quatro se chama vértice regular. Se a valência de um vértice interior for diferente de quatro, é chamado de vértice irregular. No exemplo do cubo, todos os seus vértices são irregulares de valência três, mas ele não satisfará a aproximação de uma superfície suave.

É importante ressaltar que os quadriláteros em uma malha quadrilateral em geral não são planares.

9.3 Malhas triangulares

Malhas triangulares consistem exclusivamente de triângulos. Assim como no caso quadrilateral, para tesselar um plano é possível juntar seis triângulos ao redor de cada vértice. Outra maneira é usar a topologia dos quadriláteros e cortá-los em suas diagonais (todas iguais), formando triângulos.

Os vértices interiores são regulares se sua valência for seis. Todas as faces são planares, mas é claro que a quantidade de triângulos necessária é o dobro do caso quadrilateral. Esse

aumento de processamento é um motivo para que, em geral, modelos finais 3D tenham malha quadrilateral.

9.4 Refinamento das malhas

Em *design*, assim como no caso da aproximação de curvas, é às vezes desejável começar a modelagem com uma malha rascunhada e refiná-la por um procedimento adequado. As superfícies de subdivisão trabalham dessa maneira. Antes de analisar essas superfícies, é importante estudar um pouco os princípios de refinamento. Pensamos nele como um procedimento de dois passos: mudança da conectividade (quantidade de vértices e como eles se conectam) e mudança da geometria (a posição dos vértices).

Examinamos primeiramente o caso da malha triangular: ao inserir pontos médios em cada aresta dos triângulos e conectá-los, é criada uma malha triangular mais refinada com quatro vezes mais faces que a original. Cada face é cortada para criar quatro faces e a geometria até aqui é a mesma. Com o objetivo de seguir o *design* mais atentamente, é possível mudar os vértices de lugar. A forma como isso ocorre será visto nas superfícies de subdivisão, mas é claro que a malha ganhou flexibilidade e aproximou mais a malha desejada.

A inserção de pontos médios não é a única forma de refinamento dessas malhas, mesmo tendo a vantagem de não introduzir vértices irregulares. Se o refinamento fosse dado ao inserir o baricentro de cada triângulo e conectá-lo aos vértices, a regularidade seria destruída em toda a malha.

No caso das malhas quadrilaterais, o refinamento também pode ser dado pela inserção de ponto médios das arestas. Neste caso, os pontos médios dos lados opostos se conectam formando uma cruz. Onde eles se conectam (o meio da cruz) é o baricentro da face. A face foi cortada em quatro subfaces, todas no mesmo paraboloide hiperbólico. Essa construção mantém a regularidade dos vértices.

Há outra forma de refinamento de malhas quadrilaterais usando os pontos médios das arestas. Ao invés de conectar numa forma de "cruz", é possível conectar os quatro pontos médios criando losangos. Ao redor dos vértices antigos restam triângulos e os juntando, polígonos novos são formados. Se os vértices antigos no interior eram regulares então o polígono é um quadrilátero, mas, em geral, não planar. Com exceção dos triângulos gerados na fronteira, esse tipo de refinamento dobra aproximadamente o número de faces. As faces foram escalonadas por um fator de $1/\sqrt{2}$ e rotacionadas 45°. A aplicação desse refinamento repetidas vezes é um dos algoritmos de subdivisão mais simples e gera no limite uma superfície suave.

9.5 Redução das malhas

O processo contrário ao de refinamento de malhas é chamado de redução de malhas, ele remove vértices "apropriados", selecionados por um algoritmo, e conecta os restantes de forma consistente.

Em modelagem 3D, é comum o processo de retopologia, onde basicamente essa redução da malha pode ser feita de forma manual. Artistas podem modelar sem restrições em programas como o Z-brush (uma modelagem livre, parecida com esculpir ou modelar uma massa) e depois importar essa base para um aplicativo como o Maya. Nele, há ferramentas como o *Quad Draw* que permitem a criação de quadriláteros cobrindo a sua base e assim mantendo a geometria em geral. O interessante dessa forma de redução de malha é que ela permite que o artista tenha a liberdade total com seu modelo (e faça escolhas sobre o detalhamento dele) e ainda o deixe mais leve (menor processamento com menos faces).

Obviamente, a capacidade de processamento não dita as escolhas artísticas e, portanto, o detalhamento não é perdido no processo. O processo de retopologia facilita o manuseio dos modelos 3D nos programas, mas quando estes são renderizados a subdivisão é aplicada e os detalhes "voltam". Tomando por exemplo o programa do Maya, artistas podem ter uma imagem prévia de seus modelos subdivididos com o *smooth* prévio.

9.6 Estética e relaxamento das malhas

Ainda que exista o aspecto subjetivo quando se discute estética, é possível falar sobre o balanceamento da distribuição das malhas.

Uma ideia básica é a de ajustar os vértices para que a malha final consista praticamente de polígonos regulares (precisamente regulares, normalmente não é possível). Uma ideia é aplicar os princípios físicos de sistemas massa-mola. Na implementação, os vértices são os pontos de massa e as arestas são as molas. Fixam-se alguns pontos (os pontos da fronteira) e deixam-se os outros vértices moverem-se livremente até o sistema atingir o equilíbrio. A técnica usada se chama relaxamento (*Relax tool*, no Maya).

Capítulo 10

Superfícies de subdivisão

Na área de computação gráfica 3D, as superfícies de subdivisão (*SubD surface* ou *Subsurf*) são superfícies geradas a partir da aplicação de algum algoritmo de subdivisão em uma malha inicial. O processo é análogo ao processo de curvas de subdivisão.

A primeira técnica vista nesse capítulo é a do algoritmo de Chaikin para superfícies. A segunda é uma extensão do processo de *corner-cutting* de Chaikin para faces irregulares desenvolvida por Daniel Doo e Malcolm Sabin, em 1978. O último esquema de subdivisão também foi criado no mesmo ano por Jim Clark e Edwin Catmull, generalizando a inserção de nós em superfícies B-spline bicúbicas.

A abordagem utilizada para a compreensão das curvas é a dos livros *Subdivision Methods for Geometric Design* (Warren e Weimer, 2001), *Recursively generated B-spline surfaces on arbitrary topological meshes* (Catmull e Clark, 1998), e as motivações e problemas abordados são oferecidos pelo livro *Architectural geometry* (Pottmann *et al.*, 2015).

10.1 Motivação

Algumas restrições topológicas foram descritas no estudo de superfícies B-spline e malhas quadrilaterais contendo somente vértices regulares. Existe uma correlação entre os dois problemas, pois as redes de controle de superfícies B-splines são malhas quadrilaterais.

As superfícies B-spline podem ser vistas como resultados de um processo de refinamento, que mantém a regularidade das malhas quadrilaterais, mas as refina até, no limite, obter superfícies suaves. Isso não muda a topologia e, portanto, para modelar superfícies com uma topologia mais geral é necessário o uso de redes de controle com vértices irregulares e técnica de como refiná-las (superfícies de subdivisão farão essa papel).

10.2 Superfície B-spline quadrática por subdivisão

O processo utilizado para a geração de superfícies B-spline quadráticas seguirá o esquema de supla aplicação do algoritmo de Chaikin de subdivisão para geração de curvas B-splines quadráticas. Dada uma rede de controle inicial, o algoritmo de Chaikin é aplicado

para cada coluna (polígono) dela e todos os pontos são conectados. Uma nova rede é criada e o algortimo de Chaikin é agora aplicado às linhas da rede nova. Sua conexão gerará a rede do final deste passo do refinamento e a repetição levará, no limite, a uma superfície B-spline quadrática. Lembramos que a aplicação do algoritmo de Chaikin pode ser feita nas linhas primeiro e depois nas colunas, e o resultado será o mesmo.

Juntando o processo em um único passo, para cada quadrilátero da rede de controle de vértices $P_{i,j}$, $P_{i+1,j}$, $P_{i,j+1}$ e $P_{i+1,j+1}$ é necessário aplicar o algoritmo de Chaikin duas vezes:

$$\begin{aligned} P'_{i,j} &= \tfrac{3}{4}P_{i,j} + \tfrac{1}{4}P_{i+1,j}, \\ P'_{i+1,j} &= \tfrac{1}{4}P_{i,j} + \tfrac{3}{4}P_{i+1,j}, \\ P'_{i,j+1} &= \tfrac{3}{4}P_{i,j+1} + \tfrac{1}{4}P_{i+1,j+1}, \\ P'_{i+1,j+1} &= \tfrac{1}{4}P_{i,j+1} + \tfrac{3}{4}P_{i+1,j+1} \end{aligned}$$

e

$$\begin{aligned} P^{\text{novo}}_{i,j} &= \tfrac{3}{4}P'_{i,j} + \tfrac{1}{4}P'_{i,j+1}, \\ P^{\text{novo}}_{i,j+1} &= \tfrac{1}{4}P'_{i,j} + \tfrac{3}{4}P'_{i,j+1}, \\ P^{\text{novo}}_{i+1,j} &= \tfrac{3}{4}P'_{i+1,j} + \tfrac{1}{4}P'_{i+1,j+1}, \\ P^{\text{novo}}_{i+1,j+1} &= \tfrac{1}{4}P'_{i+1,j} + \tfrac{3}{4}P'_{i+1,j+1}. \end{aligned}$$

Portanto, o passo pode se resumir a

$$\begin{aligned} P^{\text{novo}}_{i,j} &= \tfrac{9}{16}P_{i,j} + \tfrac{3}{16}P_{i+1,j} + \tfrac{3}{16}P_{i,j+1} + \tfrac{1}{4}P_{i+1,j+1}, \\ P^{\text{novo}}_{i+1,j} &= \tfrac{9}{16}P_{i+1,j} + \tfrac{3}{16}P_{i,j} + \tfrac{3}{16}P_{i+1,j+1} + \tfrac{1}{4}P_{i,j+1}, \\ P^{\text{novo}}_{i,j+1} &= \tfrac{9}{16}P_{i,j+1} + \tfrac{3}{16}P_{i,j} + \tfrac{3}{16}P_{i+1,j+1} + \tfrac{1}{4}P_{i+1,j}, \\ P^{\text{novo}}_{i+1,j+1} &= \tfrac{9}{16}P_{i+1,j+1} + \tfrac{3}{16}P_{i+1,j} + \tfrac{3}{16}P_{i,j+1} + \tfrac{1}{4}P_{i,j}. \end{aligned}$$

O algoritmo mostrado é para redes de controle fechadas, mas para as abertas os únicos cuidados são os mesmos do caso das curvas (nas arestas da fronteira não há somente uma divisão).

Essa construção só vale para topologias especiais, em que todas as faces são quadrilaterais e todos os vértices internos têm valência 4.

O algoritmo de subdivisão de Doo-Sabin é uma extensão do algoritmo visto acima e consegue lidar com vértices irregulares (valência diferente de 4).

10.3 Esquema de subdivisão de Doo-Sabin e Catmull-Clark

O esquema de subdivisão de Doo-Sabin generaliza o algoritmo de subdivisão de geração de superfícies B-spline quadráticas para malhas de controle arbitrárias (incluindo não quadrilaterais). A regras de subdivisão envolvidas são:

- **Geração de pontos de face:** para cada face j com n vértices $v_1, \ldots, v_n$, um ponto novo é criado como a média dos vértices da face,

$$f_j = \frac{1}{n}\sum_{i=1}^{n} v_i.$$

- **Geração de pontos de aresta:** para cada aresta, um novo ponto é criado como a média do ponto médio da aresta com a média dos pontos de face adjacentes (compartilhando a aresta),

$$e_k = \frac{\frac{v_i+v_{i+1}}{2} + \frac{f_j+f_{j+1}}{2}}{2} = \frac{v_i + v_{i+1} + f_j + f_{j+1}}{4}.$$

- **Geração de novos vértices:** cada vértice antigo v é substituído por um vértice novo v', calculado por

$$v' = \frac{n+3}{4n}v + \sum_{i=1}^{n} \frac{3 + 2\cos(\frac{2\pi i}{n})}{4n^2} v_i,$$

onde os v_i são os vértices vizinhos e n é a valência do vértice. Os valores dos pesos escolhidos por Doo e Sabin são consequência da análise de Fourier local (observando autovalores da matriz de subdivisão), para garantir suavidade e continuidade.

Já o esquema de subdivisão de Catmull-Clark generaliza a subdivisão para geração de superfícies B-spline bicúbicas para malhas de controle quadrilaterais. A regras de subdivisão envolvidas são conservadas para a geração de pontos de face e de aresta,

- **Geração de novos vértices:** cada vértice antigo v é substituído por um vértice novo v', calculado por

$$v' = \frac{Q}{n} + \frac{2R}{n} + \frac{S(n-3)}{n},$$

onde Q é a média de todos os pontos de faces adjacentes ao vértice v, R é a média dos pontos médios de todas as arestas antigas que incidiam no vértice v e S é o v.

As demonstrações das construções de superfícies suaves no limite dos algoritmos podem ser vistas no livro *Subdivision Methods for Geometric Design: A Constructive Approach* (Warren e Weimer, 2001).

Considerações finais

O objetivo deste trabalho foi desenvolver os fundamentos matemáticos de ferramentas gráficas essenciais: curvas e superfícies de Bézier, B-spline, NURBS e de subdivisão. O resultado constitui um guia conciso para estudos futuros que busquem aprofundar-se na construção de ferramentas de design gráfico, especialmente em sua fundamentação matemática.

Embora o trabalho tenha abordado as principais estruturas de representação geométrica e algumas de suas aplicações, há diversas áreas adajcentes que merecem exploração. Em representações de malhas, destacam-se as malhas implícitas, utilizadas para simulações de fluidos. No campo das representações não-uniformes, existem extensões das superfícies NURBS para topologias mais flexívies (as "junções em T"comentadas). Além disso, é possível explorar a implementação computacional dos algoritmos estudados.

Espera-se que este trabalho não sirva apenas como base teórica para futuros estudos, mas também inspire novas reflexões na área da matemática da computação gráfica, incentivando o desenvolvimento de abordagens inovadoras nesse campo multidisciplinar.

Referências

[Berg *et al.* 2008] Mark de Berg, Otfried Cheong, Marc van Kreveld e Mark Overmars. *Computational Geometry: Algorithms and Applications*. 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008. isbn: 3540779736 (citado na pg. 3).

[Catmull e Clark 1998] E. Catmull e J. Clark. "Recursively generated b-spline surfaces on arbitrary topological meshes". In: *Seminal Graphics: Pioneering Efforts That Shaped the Field, Volume 1*. New York, NY, USA: Association for Computing Machinery, 1998, pp. 183–188. isbn: 158113052X. url: https://doi.org/10.1145/280811.280992 (citado na pg. 59).

[Chaikin 1974] George Merrill Chaikin. "An algorithm for high-speed curve generation". *Comput. Graph. Image Process.* 3 (1974), pp. 346–349. url: https://api.semanticscholar.org/CorpusID:2650430 (citado na pg. 37).

[Dyn *et al.* 1987] Nira Dyn, David Levin e John A. Gregory. "A 4-point interpolatory subdivision scheme for curve design". *Computer Aided Geometric Design* 4.4 (1987), pp. 257–268. issn: 0167-8396. doi: https://doi.org/10.1016/0167-8396(87)90001-X. url: https://www.sciencedirect.com/science/article/pii/016783968790001X (citado na pg. 39).

[Farin 2001] Gerald Farin. *Curves and surfaces for CAGD: a practical guide*. 5th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. isbn: 1558607374 (citado nas pgs. 3, 6, 11, 41).

[Lane e Riesenfeld 1980] Jeffrey M. Lane e Richard F. Riesenfeld. "A theoretical development for the computer generation and display of piecewise polynomial surfaces". *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-2.1 (1980), pp. 35–46. doi: 10.1109/TPAMI.1980.4766968 (citado na pg. 38).

[Piegl e Tiller 1997] Les Piegl e Wayne Tiller. *The NURBS book (2nd ed.)* Berlin, Heidelberg: Springer-Verlag, 1997. isbn: 3540615458 (citado na pg. 11).

[Pottmann *et al.* 2015] Helmut Pottmann, Michael Eigensatz, Amir Vaxman e Johannes Wallner. "Architectural geometry". *Comput. Graph.* 47.C (abr. de 2015), pp. 145–164. issn: 0097-8493. doi: 10.1016/j.cag.2014.11.002. url: https://doi.org/10.1016/j.cag.2014.11.002 (citado nas pgs. 11, 41, 59).

[Prautzsch *et al.* 2002] Hartmut Prautzsch, Wolfgang Boehm e Marco Paluszny. *Bezier and B-Spline Techniques*. Berlin, Heidelberg: Springer-Verlag, 2002. isbn: 3540437614 (citado nas pgs. 3, 11).

[Warren e Weimer 2001] Joe Warren e Henrik Weimer. *Subdivision Methods for Geometric Design: A Constructive Approach*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. isbn: 1558604464 (citado nas pgs. 40, 59, 61).