



UNIVERSIDADE ESTADUAL DE
CAMPINAS

Instituto de Matemática, Estatística e
Computação Científica

FAISTER CABRERA CARVALHO

Nova superfície paramétrica e algoritmo de renderização

Campinas

2016

UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística
e Computação Científica

FAISTER CABRERA CARVALHO

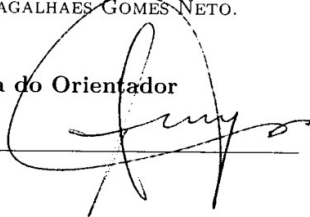
NOVA SUPERFÍCIE PARAMÉTRICA E ALGORITMO DE
RENDERIZAÇÃO

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em matemática aplicada.

Orientador: Francisco de Assis Magalhães Gomes Neto

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO FAISTER CABRERA CARVALHO, E ORIENTADA PELO PROF. DR. FRANCISCO DE ASSIS MAGALHÃES GOMES NETO.

Assinatura do Orientador

A handwritten signature in black ink, appearing to be 'F. Assis', written over a horizontal line.

CAMPINAS
2016

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

C253n Carvalho, Faister Cabrera, 1990-
Nova superfície paramétrica e algoritmo de renderização / Faister Cabrera
Carvalho. – Campinas, SP : [s.n.], 2016.

Orientador: Francisco de Assis Magalhães Gomes Neto.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de
Matemática, Estatística e Computação Científica.

1. Computação gráfica. 2. Curvas em superfícies. 3. Interpolação. I. Gomes
Neto, Francisco de Assis Magalhães, 1964-. II. Universidade Estadual de
Campinas. Instituto de Matemática, Estatística e Computação Científica. III.
Título.

Informações para Biblioteca Digital

Título em outro idioma: New parametric surface and rendering algorithm

Palavras-chave em inglês:

Computer graphics

Curves on surfaces

Interpolation

Área de concentração: Matemática Aplicada

Titulação: Mestre em Matemática Aplicada

Banca examinadora:

Francisco de Assis Magalhães Gomes Neto [Orientador]

Jorge Stolfi

Maicon Ribeiro Correa

Data de defesa: 29-02-2016

Programa de Pós-Graduação: Matemática Aplicada

Dissertação de Mestrado defendida em 29 de fevereiro de 2016 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Prof.(a). Dr(a). FRANCISCO DE ASSIS MAGALHÃES GOMES NETO

Prof.(a). Dr(a). JORGE STOLFI

Prof.(a). Dr(a). MAICON RIBEIRO CORREA

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no processo de vida acadêmica do aluno.

Agradecimentos

Agradeço ao Dr. Francisco de Assis Magalhães Gomes Neto, meu orientador, por ter me incentivado a desenvolver minha pesquisa em uma área de meu interesse.

Agradeço também à CAPES pelos dois anos de bolsa que recebi, que me permitiram me dedicar exclusivamente à pesquisa.

Agradeço à minha família pelo apoio financeiro e emocional. Um agradecimento especial vai para o meu irmão Micael Cabrera Carvalho, que me ajudou durante toda a duração da pesquisa.

Resumo

O principal foco da Computação Gráfica é o armazenamento e renderização de objetos tridimensionais computacionalmente. Objetos reais são modelados e apresentados visualmente para os usuários. Esse processo de geração da imagem a ser exibida para o usuário é chamado de renderização. Existem vários modelos que podem ser utilizados, com suas vantagens e desvantagens, e vários métodos para renderizar tais modelos.

No trabalho atual um novo modelo de superfícies curvas paramétricas é introduzido juntamente com um algoritmo de renderização que, diferentemente dos algoritmos para os modelos de superfícies curvas paramétricas atuais, não depende de métodos numéricos e aproximações, sendo capaz de identificar intersecções entre um raio e a superfície com um número constante de operações.

Durante o desenvolvimento do modelo proposto foram utilizadas a função de interpolação de Hermite e uma função de interpolação quadrática em partes muito pouco explorada na literatura, com comparações entre ambas. A função quadrática em partes possibilitou que o algoritmo proposto fosse executado em tempo constante ao reduzir a ordem das equações envolvidas no problema, o que não foi possível com a interpolação de Hermite.

Por fim, restrições do algoritmo proposto foram analisadas e possíveis novas linhas de pesquisa foram levantadas para tentar eliminá-las. Um programa que implementa o algoritmo proposto também foi codificado, e alguns objetos foram modelados usando a superfície proposta.

Palavras-chave: Computação gráfica, Curvas em superfícies, Interpolação.

Abstract

Computer graphic's main goal is to store and render tridimensional objects computationally. Real objects are modeled and presented visually to the user. This process of generating the image to be shown to the user is called "rendering". There are many models that can be used, with advantages and disadvantages, and many methods to render such models.

In the present work a novel model of curved parametric surfaces is introduced along with a rendering algorithm that, unlike current methods, doesn't depend on numerical methods, being able to identify ray/surface intersections with a constant number of operations.

During the research, the Hermite interpolation was used, as well as a partitioned quadratic interpolation with almost no presence in the literature. The quadratic function has allowed the proposed algorithm to run in constant time by reducing the order of equations involved in the problem, something that was not possible with the Hermite interpolation.

At last, the proposed algorithm's restrictions were analysed and possible new lines of research were suggested to try and remove such restrictions. A program that implements the proposed algorithm was also coded, and some objects were modeled using the proposed surface.

Keywords: Computer graphics, Curves on surfaces, Interpolation.

Lista de ilustrações

Figura 1 – Exemplo de objeto formado por uma malha de polígonos interligados.	11
Figura 2 – Exemplo de uma superfície curva de Bézier com seus pontos de controle.	11
Figura 3 – Ilustração de um triângulo sendo renderizado por rasterização.	12
Figura 4 – Ilustração do processo de renderização por ray-casting.	13
Figura 5 – Interpolação de Hermite com $f(0) = 0.1$, $f(1) = -0.2$, $f'(0) = 2$ e $f'(1) = 1$	16
Figura 6 – Representação visual das bases de Hermite.	17
Figura 7 – Representação visual das funções quadráticas por partes análogas às bases de Hermite.	19
Figura 8 – Interpolação quadrática por partes com $f(0) = 0.1$, $f(1) = -0.2$, $f'(0) = 2$ e $f'(1) = 1$	20
Figura 9 – Comparação entre a interpolação de Hermite e a quadrática por partes.	20
Figura 10 – Domínio das funções de interpolação bivariáveis propostas.	21
Figura 11 – Exemplo de interpolação bivariável com suas constantes destacadas.	22
Figura 12 – Exemplo de função f_u	23
Figura 13 – Exemplo de funções f_{u1} e f_{u2}	24
Figura 14 – Exemplo de função f_v	25
Figura 15 – Exemplo de interpolação bicúbica de Hermite.	28
Figura 16 – Exemplo de interpolação biquadrática em quadrantes.	30
Figura 17 – Curvas de nível dos exemplos anteriores.	30
Figura 18 – Objeto de exemplo: Bola (6 superfícies).	51
Figura 19 – Objeto de exemplo: Jarro (14 superfícies).	52
Figura 20 – Objeto de exemplo: Pião (30 superfícies).	52
Figura 21 – Objetos em escala de cinza.	52

Sumário

1	INTRODUÇÃO	10
1.1	Computação gráfica	10
1.2	Formatos de representação 3D	10
1.2.1	Poligonal	10
1.2.2	Superfície de Bézier	11
1.3	Renderização	12
1.3.1	Rasterização	12
1.3.2	Ray-casting	13
1.4	Objetivos	14
2	FORMATOS DE REPRESENTAÇÃO 3D PROPOSTOS	15
2.1	Interpolação univariável	15
2.1.1	Interpolação de Hermite	15
2.1.2	Interpolação quadrática por partes	17
2.2	Interpolação bivariável	21
2.2.1	Construção de uma função de interpolação bivariável	22
2.2.2	Interpolação bicúbica de Hermite	26
2.2.3	Interpolação biquadrática em quadrantes	28
3	RENDERIZAÇÃO	31
3.1	Decomposição em polígonos e rasterização	31
3.2	Ray-casting	31
3.2.1	Renderização direta	32
3.2.2	Espaço de coordenadas transformadas	33
3.2.2.1	Espaço original	33
3.2.2.2	Transformação linear	34
3.2.2.3	Transformação linear em coordenadas homogêneas	35
3.2.2.4	Domínio na variável t	41
3.2.2.5	Constantes de altura	41
3.2.2.6	Constantes de derivadas	43
3.2.2.7	Algoritmo	46
4	PROTÓTIPO	51
5	CONCLUSÕES	54
	REFERÊNCIAS	56

1 INTRODUÇÃO

1.1 Computação gráfica

A Computação Gráfica (CG) é responsável por gerar computacionalmente efeitos visuais utilizados em várias áreas, como na indústria cinematográfica, na indústria de jogos digitais, na engenharia, dentre outras.

O principal foco da CG é o armazenamento e renderização de objetos tridimensionais computacionalmente. Como exemplo, consideremos um jogo eletrônico de corrida. Nele é preciso armazenar as representações de todos os objetos do mundo virtual do jogo, incluindo os carros dos participantes, a pista da corrida, o cenário, etc. Além de ter esses objetos armazenados no computador, é preciso “desenhá-los” na tela, para que o jogador veja o que está acontecendo. Esse processo de “desenhar” os objetos é chamado de renderização e, no caso de jogos digitais, precisa ser realizado a uma frequência mínima de 30 vezes por segundo para que o jogador tenha a sensação de que o mundo do jogo está em constante movimento, o que adiciona uma restrição de tempo máximo ao processo.

Existem alguns formatos diferentes empregados no armazenamento e representação de objetos tridimensionais, cada um com suas vantagens e desvantagens, e existem diferentes processos de renderização, também com suas vantagens e desvantagens. Seguem as descrições de alguns desses formatos e processos.

1.2 Formatos de representação 3D

1.2.1 Poligonal

Polígonos são regiões planas definidas a partir de três ou mais vértices coplanares, que são ligados por arestas. Normalmente os polígonos usados em computação gráfica têm apenas três vértices, ou seja, são polígonos triangulares. Objetos formados por polígonos são compostos por uma malha de polígonos (normalmente unidos em suas arestas), de forma que representem os objetos modelados (diz-se que um objeto 3D é modelado quando um artista 3D o constrói).

A representação poligonal de objetos é a mais utilizada pela indústria da CG, principalmente pela possibilidade de se usar o processo de renderização chamado “rasterização”, discutido posteriormente, que utiliza aceleração de hardware para executar as operações matriciais necessárias com muita velocidade, graças ao paralelismo. Outra vantagem é que a modelagem de objetos em polígonos é razoavelmente intuitiva, facilitando

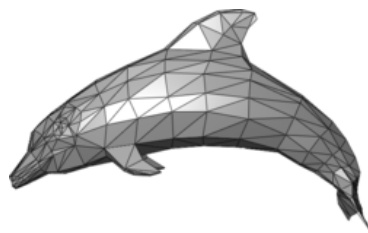


Figura 1 – Exemplo de objeto formado por uma malha de polígonos interligados.

Fonte: https://commons.wikimedia.org/wiki/File:Dolphin_triangle_mesh.png

o trabalho dos artistas.

A principal desvantagem de representar um objeto por polígonos é a impossibilidade de representar superfícies curvas. Polígonos são faces planas. Para representar um objeto curvo de forma aceitável é preciso aproximar a superfície curva por uma malha com muitos polígonos menores interligados, o que acaba aumentando consideravelmente a quantidade de polígonos utilizada.

1.2.2 Superfície de Bézier

A superfície de Bézier é uma representação de superfície curva através de pontos de controle. Segundo (FARIN; HOSCHEK; KIM, 2002), essa superfície foi desenvolvida simultaneamente por Pierre Bézier e Paul de Casteljau para a representação e desenho de corpos automobilísticos.

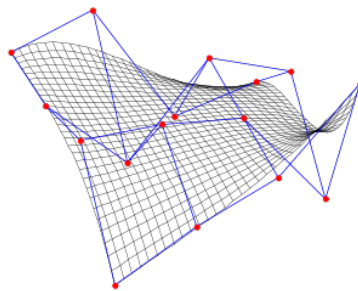


Figura 2 – Exemplo de uma superfície curva de Bézier com seus pontos de controle.

Fonte: https://commons.wikimedia.org/wiki/File:Bézier_surface_example.svg

Os pontos de controle definem a superfície curva, mas não são necessariamente intersectados pela mesma. Mais detalhes podem ser encontrados em (BÉZIER, 1986).

Na CG, superfícies de Bézier costumam ser preferíveis a polígonos para representar superfícies curvas, por aproximarem corretamente tais curvas com uma quantidade relativamente pequena de vértices se comparada com uma aproximação poligonal semelhante.

A principal desvantagem do uso de superfícies de Bézier em CG é a dificuldade em renderizar uma superfície de Bézier diretamente. Para renderizar tais superfícies é

possível utilizar métodos numéricos (GEIMER; ABERT, 2005) ou decompor a superfície em vários polígonos que aproximam a superfície, renderizando esses polígonos em vez da mesma (SHENG; HIRSCH, 1992). Isso, em princípio, permite que a superfície seja renderizada com o grau de detalhamento que quisermos (aumentando o número de iterações do método numérico ou decompondo a superfície em polígonos menores para obter uma melhor aproximação), mas gera inconveniências, como a possível baixa convergência do método numérico (exigindo muitas iterações) ou o processo adicional de decomposição da superfície em polígonos, que afetam o tempo de renderização.

1.3 Renderização

A renderização consiste no processo computacional responsável por construir uma imagem bidimensional a partir de um conjunto de objetos tridimensionais. Normalmente, é gerada uma representação da visão que uma “câmera” teria do espaço tridimensional com os objetos a serem renderizados. Tais objetos são "desenhados" na imagem resultante de acordo com a visão em perspectiva dessa câmera. Atualmente, existem dois métodos principais para fazer a renderização: a Rasterização e o Ray-casting.

1.3.1 Rasterização

No processo de rasterização, os objetos a serem renderizados são compostos por polígonos (normalmente triangulares). Calculamos as posições dos vértices de cada polígono na imagem resultante (com relação à câmera) e iteramos entre os pixels internos a essas posições, colorindo-os adequadamente a partir de texturas e de simulação de iluminação. Esse processo é repetido para todos os polígonos dos objetos a serem exibidos, de modo que a quantidade de operações depende diretamente da quantidade de polígonos a serem exibidos.

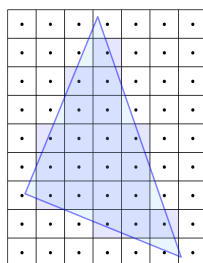


Figura 3 – Ilustração de um triângulo sendo renderizado por rasterização.

Fonte: https://commons.wikimedia.org/wiki/File:Pixels_covered_by_a_triangle.png

Esse processo é muito usado, principalmente na indústria de jogos digitais, por ser facilmente acelerado via hardware específico e paralelismo. Computadores e consoles

de vídeo-game atuais são equipados com hardwares específicos para acelerar a rasterização, e tais hardwares estão cada vez mais potentes, permitindo a renderização de um número cada vez maior de polígonos em tempo real.

Por outro lado, o único formato 3D usado na rasterização é o poligonal, o que dificulta a representação de objetos curvos, como vimos anteriormente. Outra desvantagem da rasterização é a de que o tempo necessário para o processo de renderização depende diretamente do número de polígonos a serem renderizados, o que limita a quantidade de polígonos da cena a ser renderizada e, conseqüentemente, limita o nível de detalhamento dos objetos, já que, para ter mais detalhes geométricos, é preciso ter mais polígonos.

1.3.2 Ray-casting

No processo de Ray-casting, o método é diferente daquele adotado no processo de rasterização: em vez de iterar pelos objetos a serem renderizados, itera-se pelos pixels da imagem resultante. A cada pixel associamos um raio partindo da câmera, e realizamos uma busca espacial nos objetos a serem renderizados para encontrar a intersecção que seja a mais próxima da câmera entre esse raio e um objeto. O pixel é, então, colorido de acordo com essa intersecção, utilizando informações de textura e iluminação do ponto de intersecção. O processo está ilustrado na Figura 4.

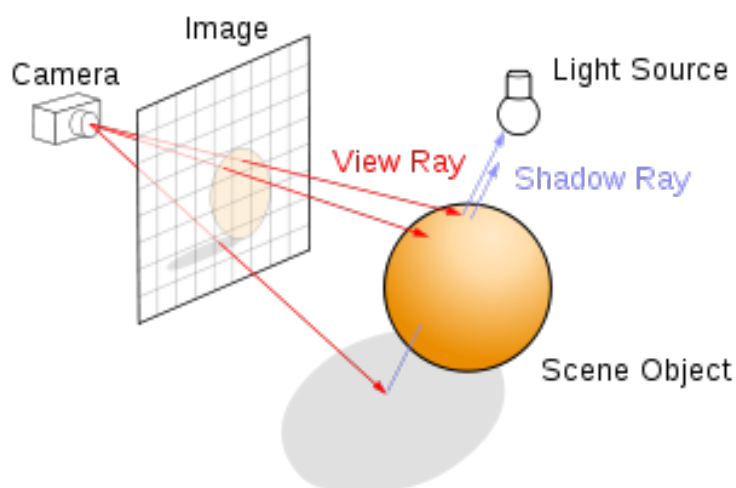


Figura 4 – Ilustração do processo de renderização por ray-casting.

Fonte: https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.svg

Como o processo é repetido para cada pixel da imagem resultante, a quantidade de operações depende diretamente da resolução de tal imagem. A quantidade de objetos a serem renderizados também impacta o desempenho. Porém, é possível usar técnicas de organização e busca espacial para amenizar esse impacto, iterando apenas entre os objetos mais próximos do raio em questão.

Uma das principais vantagens do Ray-casting é a possibilidade de renderizar objetos 3D não poligonais, desde que exista um método adequado para calcular intersecções entre uma linha e o objeto em questão. Isso permite a renderização de estruturas curvas, como esferas e cilindros, com facilidade, o que não é possível na rasterização, que explora a linearidade dos polígonos.

O Ray-casting permite também um nível de realismo adicional, se comparado com o processo de rasterização, pois é fácil computar reflexões e refrações durante o processo. Além disso, também é possível rastrear os raios de luz a partir das fontes de luz e calcular uma iluminação realista e interativa entre os objetos da cena (apesar desse processo aumentar consideravelmente o tempo de processamento necessário para obter uma imagem satisfatória), o que é impraticável na rasterização pura.

O Ray-casting, porém, não é muito utilizado em jogos eletrônicos, pela falta de hardware específico de suporte ao processo, sendo muito utilizado pela indústria cinematográfica, devido ao nível superior de realismo e à ausência da necessidade de renderização em tempo real (presente nos jogos digitais).

1.4 Objetivos

Tendo em vista o estado atual da CG, é de interesse investigar o desenvolvimento de formatos de superfícies curvas desprovidos das inconveniências das superfícies paramétricas atuais, porém abrangentes o bastante para modelar objetos arbitrários.

Ao renderizar uma superfície paramétrica como a de Bézier por Ray-casting, a principal dificuldade encontrada está na detecção das intersecções entre o raio partindo da câmera e a superfície. A abordagem mais comum, de se definir uma função de distância entre o raio e a superfície, não pode ser aplicada com facilidade a tal superfície. As demais abordagens exploradas pela literatura, como a renderização por métodos numéricos (GEIMER; ABERT, 2005) e a decomposição poligonal da superfície (SHENG; HIRSCH, 1992), introduzem processos adicionais à renderização que podem ser proibitivos para algumas aplicações.

Nosso objetivo é, portanto, investigar uma nova superfície paramétrica que facilite a definição de uma função de distância para ser utilizada em métodos de renderização direta por Ray-casting, bem como o desenvolvimento de um algoritmo capaz de renderizar tal superfície.

Além disso, para evitar a necessidade de métodos numéricos para a detecção das intersecções, é interessante que o grau da função de distância desenvolvida não seja maior que quatro no parâmetro do raio. Dessa forma, suas raízes (que representam as intersecções desejadas) poderão ser obtidas por métodos analíticos de complexidade constante.

2 FORMATOS DE REPRESENTAÇÃO 3D PROPOSTOS

Os formatos descritos abaixo foram desenvolvidos com o objetivo de modelar uma superfície curva entre quatro vértices.

2.1 Interpolação univariável

Podemos modelar uma superfície curva com uma interpolação bivariável entre seus vértices. Com funções de interpolação diferentes podemos formar superfícies diferentes, de modo que um estudo das funções de interpolação disponíveis é interessante.

Nessa seção, veremos como definir uma função de interpolação univariável que respeite os gradientes estabelecidos nos extremos (para permitir um controle sobre a curva interpolada).

Para simplificar a formulação da função de interpolação, foi considerada a interpolação f com parâmetro x entre 0 e 1, respeitando os valores $f(0)$ e $f(1)$ e os gradientes $f'(0)$ e $f'(1)$ pré-estabelecidos. Essa função univariável será utilizada para modelar a superfície em cada uma das duas variáveis de interpolação. Detalhes dessa utilização encontram-se na Seção 2.2.

2.1.1 Interpolação de Hermite

Considerando a função de interpolação como um polinômio univariável, sabendo que temos quatro restrições, só poderemos cumprir todas as restrições de forma única com um polinômio de terceiro grau.

Definimos, então, a função de interpolação no formato

$$f(x) = ax^3 + bx^2 + cx + d,$$

e sua derivada como

$$f'(x) = 3ax^2 + 2bx + c.$$

As restrições que temos são:

$$\begin{aligned}f(0) &= d \\f(1) &= a + b + c + d \\f'(0) &= c \\f'(1) &= 3a + 2b + c\end{aligned}$$

Resolvendo o sistema de equações para os coeficientes a , b , c e d , obtemos:

$$\begin{aligned}a &= f'(1) + f'(0) + 2(f(0) - f(1)) \\b &= -f'(1) - 2f'(0) + 3(f(1) - f(0)) \\c &= f'(0) \\d &= f(0)\end{aligned}$$

Portanto, a função de interpolação se torna:

$$\begin{aligned}f(x) &= \left(f'(1) + f'(0) + 2(f(0) - f(1))\right)x^3 \\&\quad - \left(f'(1) + 2f'(0) + 3(f(0) - f(1))\right)x^2 + f'(0)x + f(0)\end{aligned}\tag{2.1}$$

Essa função de interpolação é conhecida como função de interpolação de Hermite, e é muito usada na computação gráfica e em várias outras áreas. Um exemplo pode ser visto na Figura 5.

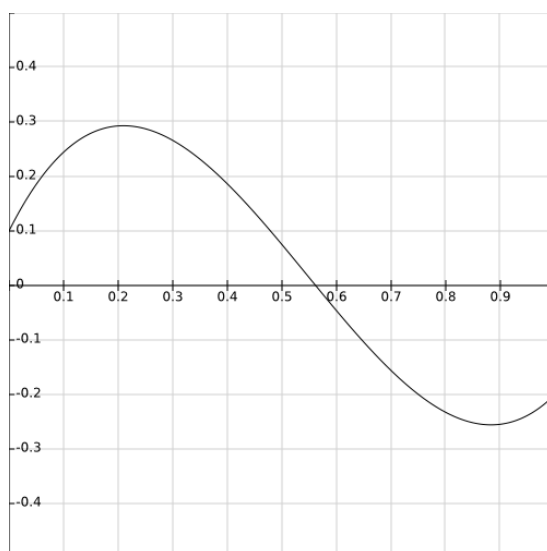


Figura 5 – Interpolação de Hermite com $f(0) = 0.1$, $f(1) = -0.2$, $f'(0) = 2$ e $f'(1) = 1$.

$$\text{Função: } f(x) = 3.6x^3 - 5.9x^2 + 2x + 0.1.$$

Outra forma de representar a interpolação de Hermite é como a combinação linear das funções conhecidas como bases de Hermite, sendo elas:

$$H_{00}(x) = 2x^3 - 3x^2 + 1$$

$$H_{01}(x) = -2x^3 + 3x^2$$

$$H_{10}(x) = x^3 - 2x^2 + x$$

$$H_{11}(x) = x^3 - x^2$$

As bases de Hermite estão representadas visualmente na Figura 6.

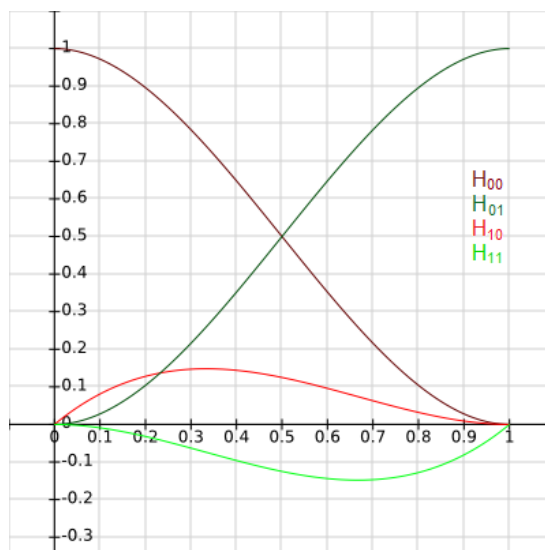


Figura 6 – Representação visual das bases de Hermite.

A função de interpolação de Hermite é equivalente a

$$f(x) = f(0) \cdot H_{00}(x) + f'(0) \cdot H_{10}(x) + f(1) \cdot H_{01}(x) + f'(1) \cdot H_{11}(x).$$

Podemos representar a função associada à curva da Figura 5 como

$$f(x) = 0.1 \cdot H_{00}(x) + 2 \cdot H_{10}(x) - 0.2 \cdot H_{01}(x) + H_{11}(x).$$

2.1.2 Interpolação quadrática por partes

A função de interpolação de Hermite, apesar de simples e funcional, é cúbica. Para as aplicações exploradas neste trabalho, uma função de interpolação cúbica gera inconveniências, conforme será explicado na Seção 2.2. Portanto, foi formulada uma interpolação quadrática que cumprisse as mesmas quatro restrições.

O problema, entretanto, é que não é possível cumprir as quatro restrições com uma função quadrática para quaisquer valores arbitrários de $f(0)$, $f(1)$, $f'(0)$ e $f'(1)$. Para contornar este problema, o domínio de interpolação $[0, 1]$ foi dividido em duas partes iguais: $[0, 0.5]$ e $[0.5, 1]$. Em cada uma das partes, foi definida uma função quadrática que cumpra

as duas restrições presentes em seu domínio, adicionando duas novas restrições: as duas funções devem possuir o mesmo valor e mesma direção gradiente no ponto $x = 0.5$.

Definindo as duas funções de interpolação quadráticas como

$$\begin{aligned} f_1(x) &= ax^2 + bx + c, & 0 \leq x \leq 0.5, \\ f_2(x) &= dx^2 + ex + g, & 0.5 \leq x \leq 1, \end{aligned}$$

temos as seguintes restrições:

$$\begin{aligned} f(0) &= c \\ f(1) &= d + e + g \\ f'(0) &= b \\ f'(1) &= 2d + e \\ \frac{a}{4} + \frac{b}{2} + c &= \frac{d}{4} + \frac{e}{2} + g \\ a + b &= d + e \end{aligned}$$

Resolvendo esse sistema de equações para os coeficientes a , b , c , d , e e g , obtemos:

$$\begin{aligned} a &= 2(f(1) - f(0)) - \frac{3f'(0) + f'(1)}{2} \\ b &= f'(0) \\ c &= f(0) \\ d &= 2(f(0) - f(1)) + \frac{f'(0) + 3f'(1)}{2} \\ e &= 2(2(f(1) - f(0)) - f'(1)) - f'(0) \\ g &= 2f(0) - f(1) + \frac{f'(0) + f'(1)}{2} \end{aligned}$$

As duas funções de interpolação tornam-se, então:

$$f_1(x) = \left(2(f(1) - f(0)) - \frac{3f'(0) + f'(1)}{2} \right) x^2 + f'(0)x + f(0), \quad 0 \leq x \leq 0.5 \quad (2.2)$$

$$\begin{aligned} f_2(x) &= \left(2(f(0) - f(1)) + \frac{f'(0) + 3f'(1)}{2} \right) x^2 \\ &+ \left(2(2(f(1) - f(0)) - f'(1)) - f'(0) \right) x \\ &+ 2f(0) - f(1) + \frac{f'(0) + f'(1)}{2}, \quad 0.5 \leq x \leq 1 \end{aligned} \quad (2.3)$$

É fácil verificar que $f_1(1-x)$ é equivalente a $f_2(x)$ com $f(0)$ substituída por $f(1)$, $f'(0)$ substituída por $-f'(1)$ e vice-versa. Como o cálculo de $f_1(x)$ tem custo computacional menor, apenas $f_1(x)$ foi utilizada na formulação das superfícies deste trabalho, sendo as fórmulas propostas aplicadas a ambos os domínios, ajustando as constantes pré-definidas adequadamente.

Também podemos representar a interpolação quadrática proposta como a combinação linear de funções análogas às bases de Hermite. Neste caso, definimos as seguintes funções:

$$Q_{00}(x) = \begin{cases} -2x^2 + 1 & 0 \leq x \leq 0.5 \\ 2x^2 - 4x + 2 & 0.5 \leq x \leq 1 \end{cases}$$

$$Q_{01}(x) = \begin{cases} 2x^2 & 0 \leq x \leq 0.5 \\ -2x^2 + 4x - 1 & 0.5 \leq x \leq 1 \end{cases}$$

$$Q_{10}(x) = \begin{cases} -\frac{3}{2}x^2 + x & 0 \leq x \leq 0.5 \\ \frac{1}{2}x^2 - x + \frac{1}{2} & 0.5 \leq x \leq 1 \end{cases}$$

$$Q_{11}(x) = \begin{cases} -\frac{1}{2}x^2 & 0 \leq x \leq 0.5 \\ \frac{3}{2}x^2 - 2x + \frac{1}{2} & 0.5 \leq x \leq 1 \end{cases}$$

Tais funções são apresentadas visualmente na Figura 7.

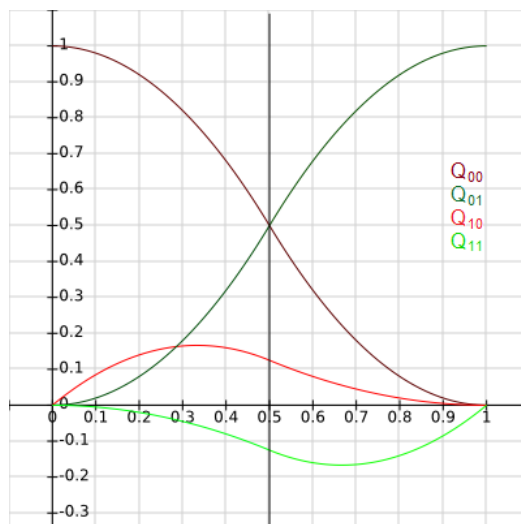


Figura 7 – Representação visual das funções quadráticas por partes análogas às bases de Hermite.

A função de interpolação quadrática por partes proposta é equivalente a

$$f(x) = f(0) \cdot Q_{00}(x) + f'(0) \cdot Q_{10}(x) + f(1) \cdot Q_{01}(x) + f'(1) \cdot Q_{11}(x).$$

A interpolação quadrática equivalente ao exemplo apresentado na Figura 5, com suas duas funções f_1 e f_2 , pode ser vista na Figura 8.

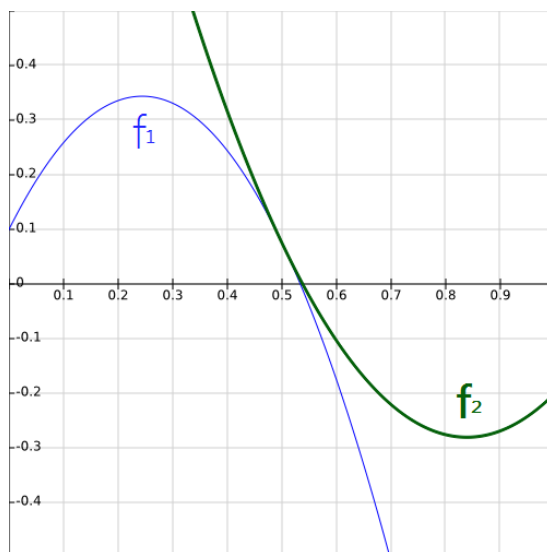


Figura 8 – Interpolação quadrática por partes com $f(0) = 0,1$, $f(1) = -0,2$, $f'(0) = 2$ e $f'(1) = 1$.

Funções: $f_1(x) = -4,1x^2 + 2x + 0,1$ e $f_2(x) = 3,1x^2 - 5,2x + 1,9$.

Verificou-se que a função de interpolação proposta é um caso específico das funções de interpolação apresentadas no Capítulo 3.9 de (SPÄTH, 1995). A falta de outras citações na literatura indica que essa função de interpolação não foi explorada o bastante e pode ter aplicações úteis em diversas áreas.

Uma comparação entre os exemplos apresentados anteriormente utilizando a interpolação de Hermite e a interpolação quadrática proposta pode ser vista na Figura 10.

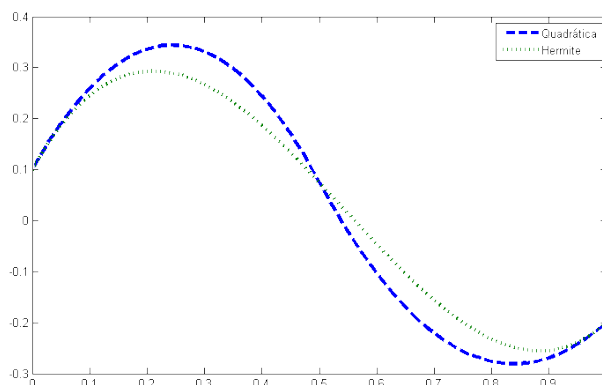


Figura 9 – Comparação entre a interpolação de Hermite e a quadrática por partes.

2.2 Interpolação bivariável

Podemos modelar superfícies utilizando funções de interpolação com dois parâmetros (u e v) que interpolam entre os valores de quatro vértices. Podemos utilizar tais funções de interpolação, por exemplo, para interpolar cada coordenada em um espaço tridimensional, de modo que cada par de parâmetros (u, v) produziria um ponto tridimensional. Existem, ainda, outras formas de modelar superfícies a partir de tais funções de interpolação, como exploraremos no Capítulo 3.

Para simplificar as formulações desta seção, definimos o espaço de coordenadas u e v , tais que $0 \leq u \leq 1$ e $0 \leq v \leq 1$, como domínio das funções, e nomeamos os vértices A , B , C e D . As funções de interpolação desejadas devem ter o valor do vértice A quando $[u, v] = [0, 0]$, o do vértice B quando $[u, v] = [1, 0]$, o do vértice C quando $[u, v] = [1, 1]$ e o do vértice D quando $[u, v] = [0, 1]$, como apresentado na Figura 10.

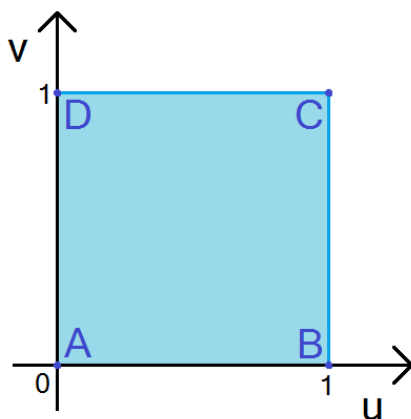


Figura 10 – Domínio das funções de interpolação bivariáveis propostas.

Seja $f_{uv}(u, v)$ a função de interpolação desejada, definimos as seguintes constantes:

$$f^A = f_{uv}(0, 0)$$

$$f^B = f_{uv}(1, 0)$$

$$f^C = f_{uv}(1, 1)$$

$$f^D = f_{uv}(0, 1)$$

As funções de interpolação também devem respeitar as direções gradientes estabelecidas nos quatro vértices. Assim, nomeamos as seguintes constantes pré-definidas:

$$\partial f^{AB} = \frac{\partial f_{uv}}{\partial u}(0, 0)$$

$$\partial f^{BA} = -\frac{\partial f_{uv}}{\partial u}(1, 0)$$

$$\begin{aligned}\partial f^{BC} &= \frac{\partial f_{uv}}{\partial v}(1, 0) \\ \partial f^{CB} &= -\frac{\partial f_{uv}}{\partial v}(1, 1) \\ \partial f^{CD} &= -\frac{\partial f_{uv}}{\partial u}(1, 1) \\ \partial f^{DC} &= \frac{\partial f_{uv}}{\partial u}(0, 1) \\ \partial f^{DA} &= -\frac{\partial f_{uv}}{\partial v}(0, 1) \\ \partial f^{AD} &= \frac{\partial f_{uv}}{\partial v}(0, 0)\end{aligned}$$

Um exemplo da interpolação bivariável desejada com suas constantes destacadas pode ser observado na Figura 11.

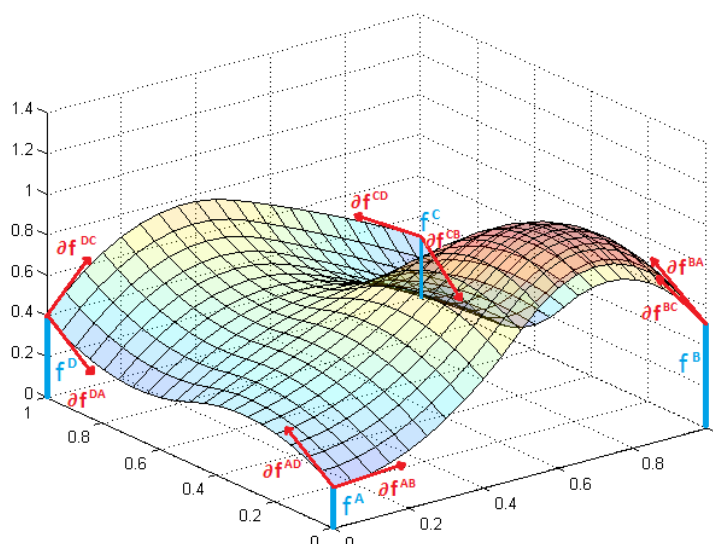


Figura 11 – Exemplo de interpolação bivariável com suas constantes destacadas.

Além disso, duas superfícies adjacentes, ou seja, que possuem dois vértices em comum, devem possuir valores e gradientes iguais em toda sua interface. Isso é necessário para que a superfície resultante tenha derivada contínua ou, em outras palavras, seja suave e não tenha pontas ou cantos.

2.2.1 Construção de uma função de interpolação bivariável

Podemos utilizar uma função de interpolação univariável (como as apresentadas na Seção 2.1) para construir uma função de interpolação bivariável, com o processo descrito a seguir.

Nossa função de interpolação bivariável $f_{uv}(u, v)$ será formada a partir da soma de duas outras funções:

$$f_{uv}(u, v) = f_u(u, v) + f_v(u, v).$$

A função $f_u(u, v)$ deverá respeitar as seguintes restrições:

$$\begin{aligned} f_u(0, 0) &= f^A & \frac{\partial f_u}{\partial u}(0, 0) &= \partial f^{AB} \\ f_u(1, 0) &= f^B & \frac{\partial f_u}{\partial u}(1, 0) &= -\partial f^{BA} \\ f_u(1, 1) &= f^C & \frac{\partial f_u}{\partial u}(0, 1) &= \partial f^{DC} \\ f_u(0, 1) &= f^D & \frac{\partial f_u}{\partial u}(1, 1) &= -\partial f^{CD} \\ \frac{\partial f_u}{\partial v}(u, 0) &= 0, \forall u & \frac{\partial f_u}{\partial v}(u, 1) &= 0, \forall u \end{aligned}$$

Já a função $f_v(u, v)$ respeitará as seguintes restrições:

$$\begin{aligned} f_v(0, 0) &= 0 & \frac{\partial f_v}{\partial v}(0, 0) &= \partial f^{AD} \\ f_v(1, 0) &= 0 & \frac{\partial f_v}{\partial v}(0, 1) &= -\partial f^{DA} \\ f_v(1, 1) &= 0 & \frac{\partial f_v}{\partial v}(1, 0) &= \partial f^{BC} \\ f_v(0, 1) &= 0 & \frac{\partial f_v}{\partial v}(1, 1) &= -\partial f^{CB} \\ \frac{\partial f_v}{\partial u}(0, v) &= 0, \forall v & \frac{\partial f_v}{\partial u}(1, v) &= 0, \forall v \end{aligned}$$

Desta forma, a função $f_{uv}(u, v)$ respeitará todas as restrições definidas anteriormente para a interpolação bivariável desejada.

A função $f_u(u, v)$ pode ser construída da seguinte forma:

$$f_u(u, v) = f_i(v) \left(f_{u1}(u) - f_{u2}(u) \right) + f_{u2}(u).$$

Um exemplo visual de função f_u é apresentado na Figura 12.

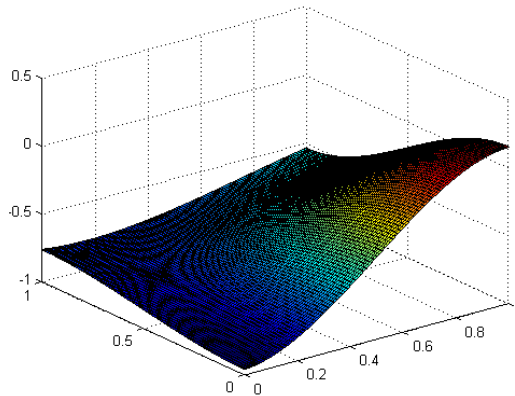


Figura 12 – Exemplo de função f_u .

A função de interpolação f_i é construída utilizando uma interpolação univariável, tal que:

$$\begin{aligned} f_i(0) &= 0 \\ f'_i(0) &= 0 \\ f_i(1) &= 1 \\ f'_i(1) &= 0 \end{aligned} \tag{2.4}$$

A função $f_{u1}(u)$ deve respeitar as seguintes restrições:

$$\begin{aligned} f_{u1}(0) &= f^A \\ f'_{u1}(0) &= \partial f^{AB} \\ f_{u1}(1) &= f^B \\ f'_{u1}(1) &= -\partial f^{BA} \end{aligned} \tag{2.5}$$

Construímos, então, a função $f_{u2}(u)$ tal que:

$$\begin{aligned} f_{u2}(0) &= f^D \\ f'_{u2}(0) &= \partial f^{DC} \\ f_{u2}(1) &= f^C \\ f'_{u2}(1) &= -\partial f^{CD} \end{aligned} \tag{2.6}$$

A Figura 13 apresenta exemplos de funções f_{u1} e f_{u2} em suas respectivas bordas do domínio.

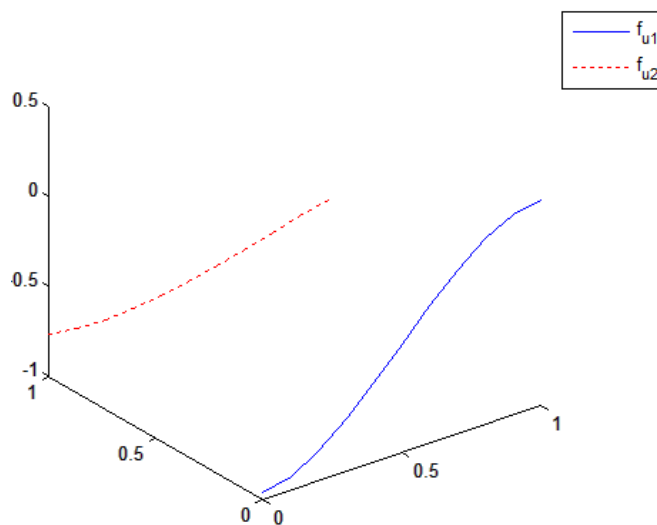


Figura 13 – Exemplo de funções f_{u1} e f_{u2} .

Já a função $f_v(u, v)$ é construída da seguinte forma:

$$f_v(u, v) = f_i(u) \left(f_{v1}(u) - f_{v2}(u) \right) + f_{v2}(v).$$

Sendo f_i a mesma função de interpolação definida anteriormente.

Na Figura 14 podemos ver um exemplo de função f_v .

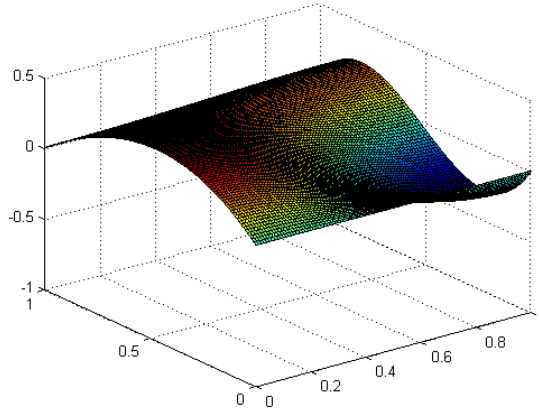


Figura 14 – Exemplo de função f_v .

De forma similar à utilizada para construir $f_u(u, v)$, construímos a função $f_{v1}(v)$, tal que:

$$\begin{aligned} f_{v1}(0) &= 0 \\ f'_{v1}(0) &= \partial f^{AD} \\ f_{v1}(1) &= 0 \\ f'_{v1}(1) &= -\partial f^{DA} \end{aligned} \tag{2.7}$$

Construímos, também, a função $f_{v2}(v)$ que respeite as seguintes restrições:

$$\begin{aligned} f_{v2}(0) &= 0 \\ f'_{v2}(0) &= \partial f^{BC} \\ f_{v2}(1) &= 0 \\ f'_{v2}(1) &= -\partial f^{CB} \end{aligned} \tag{2.8}$$

A função de interpolação bivariável $f_{uv}(u, v)$ obtida será diferente dependendo da função de interpolação univariável utilizada para construí-la, porém respeitará todas as restrições estabelecidas anteriormente.

Uma abordagem similar foi utilizada em (COONS, 1967) para definir a superfície conhecida como "Coons Patch".

Para as imagens de exemplo apresentadas nesta seção, os valores das constantes definidas anteriormente são:

$$\begin{aligned} f^A &= -1 & \partial f^{AB} &= 0.1 & \partial f^{CD} &= -0.3 \\ f^B &= 0.1 & \partial f^{BA} &= 0 & \partial f^{DC} &= -0.3 \\ f^C &= -0.5 & \partial f^{BC} &= -2.8 & \partial f^{DA} &= 1.2 \\ f^D &= -0.8 & \partial f^{CB} &= 2.4 & \partial f^{AD} &= 1.6 \end{aligned}$$

2.2.2 Interpolação bicúbica de Hermite

Construiremos, então, uma função de interpolação bicúbica utilizando a interpolação cúbica de Hermite no processo descrito anteriormente. Todas as funções de interpolação univariável construídas a seguir serão baseadas em (2.1).

Com as restrições estabelecidas em (2.5), obtemos:

$$\begin{aligned} f_{u1}(u) &= \left(\partial f^{AB} - \partial f^{BA} + 2(f^A - f^B) \right) u^3 \\ &\quad - \left(2\partial f^{AB} - \partial f^{BA} + 3(f^A - f^B) \right) u^2 + \partial f^{AB} u + f^A. \end{aligned} \quad (2.9)$$

A partir das restrições estabelecidas em (2.6), temos:

$$\begin{aligned} f_{u2}(u) &= \left(\partial f^{DC} - \partial f^{CD} + 2(f^D - f^C) \right) u^3 \\ &\quad - \left(2\partial f^{DC} - \partial f^{CD} + 3(f^D - f^C) \right) u^2 + \partial f^{DC} u + f^D. \end{aligned} \quad (2.10)$$

Definimos, então, $f_i(v)$ com as restrições estabelecidas em (2.4), obtendo:

$$f_i(v) = -2v^3 + 3v^2. \quad (2.11)$$

A seguir, construímos:

$$\begin{aligned} f_u(u, v) &= f_i(v) \left(f_{u2}(u) - f_{u1}(u) \right) + f_{u1}(u) \\ &= (-2v^3 + 3v^2) \left(\left(\partial f^{DC} - \partial f^{CD} - \partial f^{AB} + \partial f^{BA} + 2(f^D - f^C - f^A + f^B) \right) u^3 \right. \\ &\quad - \left(2(\partial f^{DC} - \partial f^{AB}) - \partial f^{CD} + \partial f^{BA} + 3(f^D - f^C - f^A + f^B) \right) u^2 \\ &\quad \left. + (\partial f^{DC} - \partial f^{AB}) u + f^D - f^A \right) + \left(\partial f^{AB} - \partial f^{BA} + 2(f^A - f^B) \right) u^3 \\ &\quad - \left(2\partial f^{AB} - \partial f^{BA} + 3(f^A - f^B) \right) u^2 + \partial f^{AB} u + f^A. \end{aligned} \quad (2.12)$$

Pelas restrições estabelecidas em (2.7), temos:

$$f_{v1}(v) = (\partial f^{AD} - \partial f^{DA})v^3 - (2\partial f^{AD} - \partial f^{DA})v^2 + \partial f^{AD}v. \quad (2.13)$$

Respeitando as restrições estabelecidas em (3.2), definimos:

$$f_{v2}(v) = (\partial f^{BC} - \partial f^{CB})v^3 - (2\partial f^{BC} - \partial f^{CB})v^2 + \partial f^{BC}v. \quad (2.14)$$

Construímos, então:

$$\begin{aligned} f_v(u, v) &= f_i(u)(f_{v2}(v) - f_{v1}(v)) + f_{v1}(v) \\ &= (-2u^3 + 3u^2) \left((\partial f^{BC} - \partial f^{CB} - \partial f^{AD} + \partial f^{DA})v^3 \right. \\ &\quad \left. - (2(\partial f^{BC} - \partial f^{AD}) - \partial f^{CB} + \partial f^{DA})v^2 + (\partial f^{BC} - \partial f^{AD})v \right) \\ &\quad + (\partial f^{AD} - \partial f^{DA})v^3 - (2\partial f^{AD} - \partial f^{DA})v^2 + \partial f^{AD}v. \end{aligned} \quad (2.15)$$

Por fim, somamos (2.12) a (2.15), obtendo a função de interpolação:

$$\begin{aligned} f_{uv}(u, v) &= f_u(u, v) + f_v(u, v) \\ &= 2 \left(\partial f^{CD} - \partial f^{DC} + \partial f^{AB} - \partial f^{BA} - 2(f^D - f^C - f^A + f^B) + \partial f^{CB} - \partial f^{BC} \right. \\ &\quad \left. + \partial f^{AD} - \partial f^{DA} \right) u^3 v^3 + \left(3 \left(\partial f^{DC} - \partial f^{CD} - \partial f^{AB} + \partial f^{BA} + 2(f^D - f^C - f^A \right. \right. \\ &\quad \left. \left. + f^B) \right) + 2 \left(2(\partial f^{BC} - \partial f^{AD}) - \partial f^{CB} + \partial f^{DA} \right) \right) u^3 v^2 - 2(\partial f^{BC} - \partial f^{AD})u^3 v \\ &\quad + \left(\partial f^{AB} - \partial f^{BA} + 2(f^A - f^B) \right) u^3 + \left(2 \left(2(\partial f^{DC} - \partial f^{AB}) - \partial f^{CD} + \partial f^{BA} \right. \right. \\ &\quad \left. \left. + 3(f^D - f^C - f^A + f^B) \right) + 3(\partial f^{BC} - \partial f^{CB} - \partial f^{AD} + \partial f^{DA}) \right) u^2 v^3 \\ &\quad - 3 \left(2(\partial f^{DC} - \partial f^{AB} + \partial f^{BC} - \partial f^{AD}) - \partial f^{CD} + \partial f^{BA} - \partial f^{CB} + \partial f^{DA} + 3(f^D \right. \\ &\quad \left. - f^C - f^A + f^B) \right) u^2 v^2 + 3(\partial f^{BC} - \partial f^{AD})u^2 v - \left(2\partial f^{AB} - \partial f^{BA} + 3(f^B \right. \\ &\quad \left. - f^A) \right) u^2 - 2(\partial f^{DC} - \partial f^{AB})uv^3 + 3(\partial f^{DC} - \partial f^{AB})uw^2 + \partial f^{AB}u + (\partial f^{AD} \\ &\quad - \partial f^{DA} - 2(f^D - f^A))v^3 + (3(f^D - f^A) - 2\partial f^{AD} + \partial f^{DA})v^2 + \partial f^{AD}v + f^A. \end{aligned} \quad (2.16)$$

Um exemplo de função de interpolação bicúbica de Hermite pode ser visto na Figura 15.

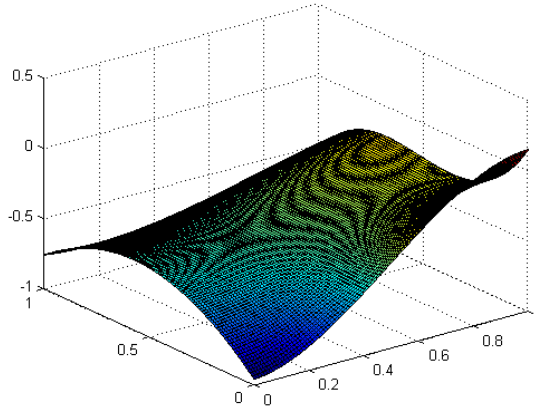


Figura 15 – Exemplo de interpolação bicúbica de Hermite.

$$\text{Função: } 8.2u^3v^3 - 15.5u^3v^2 + 8.8u^3v - 2.1u^3 - 13u^2v^3 + 24.3u^2v^2 - 13.2u^2v + 3.1u^2 + 0.8uv^3 - 1.2uv^2 + 0.1u - 1.4v^2 + 1.6v - 1.$$

2.2.3 Interpolação biquadrática em quadrantes

Todo o processo desenvolvido com a interpolação de Hermite na seção anterior pode ser repetido utilizando a interpolação quadrática da equação (2.2), obtendo como resultado uma função de interpolação biquadrática em partes.

Como a função de interpolação univariável utilizada está restrita a uma parte do domínio, o mesmo se aplicará às funções desenvolvidas nessa seção. Portanto, como agora temos duas variáveis, o domínio acabará dividido em quatro partes. Passarei a chamar tais partes de quadrantes.

Como sabemos que podemos reordenar as constantes para obter outra parte do domínio utilizando a mesma função, podemos desenvolver a função de interpolação somente para o quadrante com $0 \leq u \leq 0.5$ e $0 \leq v \leq 0.5$. Para utilizar a função obtida nos demais quadrantes basta rotacionar as constantes adequadamente.

Construímos $f_{u1}(u)$ respeitando as restrições estabelecidas em (2.5), obtendo:

$$f_{u1}(u) = \left(2(f^B - f^A) - \frac{3\partial f^{AB} - \partial f^{BA}}{2} \right) u^2 + \partial f^{AB}u + f^A, \quad 0 \leq u \leq 0.5 \quad (2.17)$$

Com as restrições estabelecidas em (2.6), definimos:

$$f_{u2}(u) = \left(2(f^C - f^D) - \frac{3\partial f^{DC} - \partial f^{CD}}{2} \right) u^2 + \partial f^{DC}u + f^D, \quad 0 \leq u \leq 0.5 \quad (2.18)$$

A partir das restrições estabelecidas em (2.4), obtemos:

$$f_i(x) = 2x^2, \quad 0 \leq x \leq 0.5 \quad (2.19)$$

Construindo $f_u(u, v)$, temos:

$$\begin{aligned}
f_u(u, v) &= (1 - f_i(v))f_{u1}(u) + f_i(v)f_{u2}(u) \\
&= f_i(v)(f_{u2}(u) - f_{u1}(u)) + f_{u1}(u) \\
&= (2v^2) \left(\left(2(f^C - f^D - f^B + f^A) - \frac{3(\partial f^{DC} - \partial f^{AB}) - \partial f^{CD} + \partial f^{BA}}{2} \right) u^2 \right. \\
&\quad \left. + (\partial f^{DC} - \partial f^{AB})u + f^D - f^A \right) + \left(2(f^B - f^A) - \frac{3\partial f^{AB} - \partial f^{BA}}{2} \right) u^2 \\
&\quad + \partial f^{AB}u + f^A, \quad 0 \leq u \leq 0.5, \quad 0 \leq v \leq 0.5
\end{aligned} \tag{2.20}$$

De acordo com as restrições estabelecidas em (2.7), construímos:

$$f_{v1}(v) = \frac{\partial f^{DA} - 3\partial f^{AD}}{2}v^2 + \partial f^{AD}v, \quad 0 \leq v \leq 0.5 \tag{2.21}$$

Pelas restrições estabelecidas em (3.2), temos:

$$f_{v2}(v) = \frac{\partial f^{CB} - 3\partial f^{BC}}{2}v^2 + \partial f^{BC}v, \quad 0 \leq v \leq 0.5 \tag{2.22}$$

Ao construir $f_v(u, v)$, obtemos:

$$\begin{aligned}
f_v(u, v) &= (1 - f_i(u))f_{v1}(v) + f_i(u)f_{v2}(v) \\
&= f_i(u)(f_{v2}(v) - f_{v1}(v)) + f_{v1}(v) \\
&= (2u^2) \left(\frac{\partial f^{CB} - \partial f^{DA} + 3(\partial f^{AD} - \partial f^{BC})}{2}v^2 + (\partial f^{BC} - \partial f^{AD})v \right) \\
&\quad + \frac{\partial f^{DA} - 3\partial f^{AD}}{2}v^2 + \partial f^{AD}v, \quad 0 \leq u \leq 0.5, \quad 0 \leq v \leq 0.5
\end{aligned} \tag{2.23}$$

Por fim, somamos (2.20) a (2.23), obtendo a função de interpolação de um quadrante:

$$\begin{aligned}
f_{uv}(u, v) &= f_u(u, v) + f_v(u, v) \\
&= \left(4(f^C - f^D - f^B + f^A) - 3(\partial f^{DC} - \partial f^{AB} - \partial f^{AD} + \partial f^{BC}) \right. \\
&\quad \left. + \partial f^{CD} - \partial f^{BA} + \partial f^{CB} - \partial f^{DA} \right) u^2v^2 + 2(\partial f^{BC} - \partial f^{AD})u^2v \\
&\quad + \left(2(f^B - f^A) - \frac{3\partial f^{AB} - \partial f^{BA}}{2} \right) u^2 + 2(\partial f^{DC} - \partial f^{AB})uv^2 + \partial f^{AB}u \\
&\quad + \left(2(f^D - f^A) + \frac{\partial f^{DA} - 3\partial f^{AD}}{2} \right) v^2 + \partial f^{AD}v + f^A, \quad 0 \leq u \leq 0.5, \quad 0 \leq v \leq 0.5
\end{aligned} \tag{2.24}$$

A função (2.24) deve ser, então, ajustada para cada quadrante, reordenando os vértices e as derivadas adequadamente.

Para calcular o valor de um ponto qualquer com a função (2.16) são necessárias 9 somas e 9 multiplicações. Já com a função (2.24), são necessárias 2 comparações (para determinar o quadrante), 6 somas e 6 multiplicações. Em qualquer sistema no qual desvios condicionais não sejam muito mais dispendiosos que operações numéricas, a interpolação biquadrática deve obter desempenho superior.

Um exemplo da interpolação biquadrática em quadrantes proposta pode ser visto na Figura 16. Os mesmos parâmetros do exemplo apresentado na Figura 15 foram usados para possibilitar a comparação visual entre as duas funções, cujas curvas de nível foram colocadas lado a lado na Figura 17.

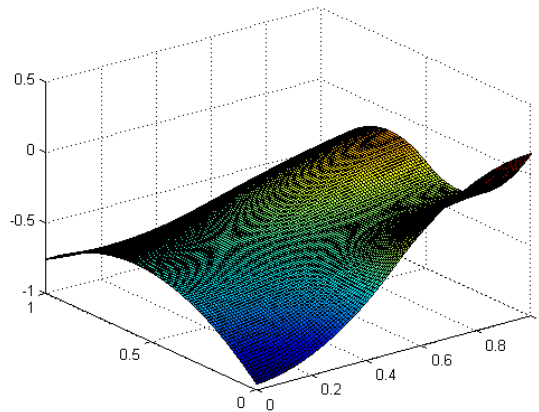


Figura 16 – Exemplo de interpolação biquadrática em quadrantes.

$$\text{Função: } f_{uv}(u, v) = \begin{cases} 12.1u^2v^2 - 8.8u^2v + 2.05u^2 - 0.8uv^2 + 0.1u - 1.4v^2 + 1.6v - 1, & \text{se } 0 \leq u \leq 0.5, 0 \leq v \leq 0.5 \\ -10.7u^2v^2 + 8.8u^2v - 2.15u^2 + 22uv^2 - 17.6uv + 4.3u - 7.1v^2 + 6v - 2.05, & \text{se } 0.5 \leq u \leq 1, 0 \leq v \leq 0.5 \\ 4.3u^2v^2 - 6.2u^2v + 1.6u^2 - 9.2uv^2 + 13.6uv - 3.5u + 1.1v^2 - 2.2v, & \text{se } 0.5 \leq u \leq 1, 0.5 \leq v \leq 1 \\ -5.7u^2v^2 + 9u^2v - 2.4u^2 + 0.8uv^2 - 1.6uv + 0.5u - 1.4v^2 + 1.6v - 1, & \text{se } 0 \leq u \leq 0.5, 0.5 \leq v \leq 1 \end{cases}$$

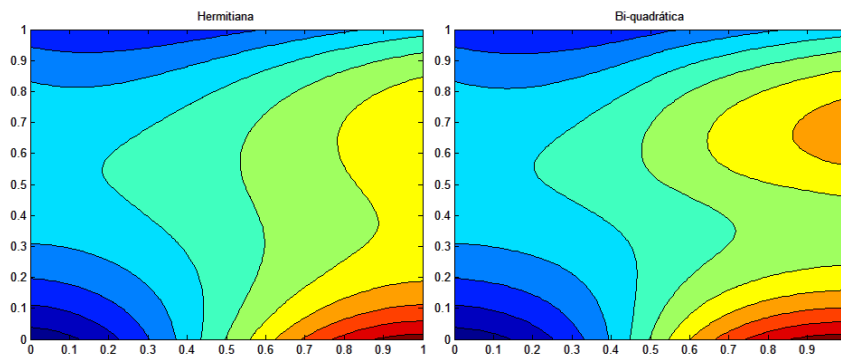


Figura 17 – Curvas de nível dos exemplos anteriores.

3 RENDERIZAÇÃO

Para que as superfícies modeladas no Capítulo 2 sejam utilizadas na computação gráfica, são necessários algoritmos eficientes para renderizá-las. Neste capítulo, estudaremos possíveis algoritmos de renderização para os modelos propostos.

3.1 Decomposição em polígonos e rasterização

Um possível algoritmo de renderização para as superfícies propostas consiste em calcular as coordenadas de uma grade de pontos da superfície, formar polígonos com esses pontos e renderizar os polígonos gerados por meio de rasterização.

Esse algoritmo já é utilizado para renderizar superfícies bicúbicas de Bézier e, apesar de permitir ajustar a precisão e a velocidade de execução pelo número de pontos e polígonos computados, adiciona a etapa de geração dos polígonos à renderização, além de ter custo diretamente proporcional ao número de polígonos gerados quando efetuada a rasterização.

No presente trabalho, esse algoritmo foi deixado de lado em favor do desenvolvimento de algoritmos de renderização baseados em Ray-tracing.

3.2 Ray-casting

Como mencionado no Capítulo 1, no processo de renderização por Ray-casting, a cada pixel da imagem, associamos um raio partindo da câmera e, então, iteramos entre os objetos a serem renderizados verificando se o raio intersecta algum objeto. Caso intersecte, detectamos o ponto de intersecção mais próximo da câmera e utilizamos esse ponto para determinar a cor do pixel.

Podemos renderizar as superfícies propostas com a técnica de Ray-casting desde que sejamos capazes de detectar intersecções entre um raio e a superfície. Analisemos, então, como fazer tal detecção.

Para simplificar as formulações a seguir, definimos E como o vetor tridimensional da posição da câmera, R como o vetor tridimensional da direção do raio e, por fim, os pontos do raio partindo da câmera pela seguinte função:

$$p(t) = E + Rt. \quad (3.1)$$

Um procedimento utilizado para a detecção de intersecções em Ray-casting de

superfícies implícitas envolve a formulação de uma função univariável de distância entre o raio e a superfície, seguida da detecção das raízes dessa função de distância. Cada raiz representa distância nula do raio até a superfície ou, em outras palavras, uma intersecção. Mais detalhes podem ser obtidos em (HART, 1996).

Nossa superfície, entretanto, é paramétrica em u e v . Por esse motivo, formular uma função de distância univariável é uma tarefa complexa.

3.2.1 Renderização direta

Como mencionado no início da Seção 2.2, é possível utilizar as funções de interpolação bivariáveis propostas para interpolar entre as coordenadas dos vértices de uma superfície separadamente. Dessa forma, teremos uma função diferente para cada coordenada.

Definamos as coordenadas do espaço tridimensional euclidiano onde estão as superfícies como (X, Y, Z) . Teremos, então, uma função $f_{uv}^x(u, v)$ que, dados u e v , retorna a coordenada X do ponto da superfície referente a esses parâmetros. O mesmo vale para as coordenadas Y e Z .

Se tentarmos, então, montar uma função de distância euclidiana entre os pontos da superfície e os pontos do raio, obteremos uma função trivariável extremamente complexa, e encontrar suas raízes, se for possível, será proibitivo.

Uma outra abordagem seria, em vez de construir uma única função de distância, construir três funções, uma para cada coordenada:

$$\begin{aligned} D_x(u, v, t) &= f_{uv}^x(u, v) - p_x(t) \\ D_y(u, v, t) &= f_{uv}^y(u, v) - p_y(t) \\ D_z(u, v, t) &= f_{uv}^z(u, v) - p_z(t) \end{aligned} \tag{3.2}$$

Os valores de u , v e t que são raízes para os três polinômios simultaneamente representarão intersecções do raio com a superfície.

Para simplificar a tarefa, podemos tentar usar a independência entre as três equações para eliminar variáveis e tentar obter uma equação univariável, cujas raízes seriam mais fáceis e baratas de serem encontradas. Para fazer isso, as três equações foram modeladas no software Mathematica, e a função Eliminate foi utilizada para eliminar as variáveis t e v das equações. O Mathematica não foi capaz de fazer a segunda eliminação com a superfície bicúbica, porém teve sucesso com a superfície biquadrática proposta.

Após a eliminação das variáveis t e v das equações geradas a partir da superfície biquadrática, foi obtida uma equação univariável de oitavo grau em u . As raízes dessa equação representam raízes da função de distância.

Apesar de termos conseguido simplificar a tarefa consideravelmente, agora temos novos problemas. Os coeficientes da equação univariável são extremamente custosos de serem calculados, exigindo mais de 2000 somas, 10000 multiplicações e 1000 cálculos de potências. Se a superfície não for animada, é possível fazer o pré-cálculo de várias operações com cerca de 200 variáveis auxiliares, diminuindo consideravelmente o número de operações. Ainda assim, calcular os coeficientes é inviável para aplicações que exigem alto desempenho, como a renderização de jogos digitais. Além disso, vale lembrar que o processo deve ser repetido para cada um dos quadrantes da superfície, exigindo quatro vezes esse número de operações para cada superfície. Por fim, outro problema está no fato de que equações de oitavo grau não possuem solução analítica, exigindo a aplicação de um método numérico que pode ser por si só bastante custoso e impreciso.

Para aplicações que não exigem um alto desempenho, esta abordagem de renderização direta pode ser interessante, porém existem outras formas de encontrar intersecções que não sofrem dos problemas listados. Estudaremos essas abordagens a seguir.

3.2.2 Espaço de coordenadas transformadas

As funções de interpolação definidas no Capítulo 2 foram usadas, até o momento, para interpolar cada coordenada separadamente. Entretanto, podemos utilizá-las de outra forma.

3.2.2.1 Espaço original

Suponhamos que a superfície a ser modelada tenha vértices $(0, 0, 0)$, $(1, 0, 0)$, $(1, 1, 0)$ e $(0, 1, 0)$. Podemos utilizar as funções de interpolação propostas para interpolar a coordenada Z assumindo $u = X$ e $v = Y$. Dessa forma, teremos apenas uma função de interpolação, que poderemos subtrair da coordenada Z dos pontos do raio para obter uma única função de distância: $D(X, Y, Z) = I(X, Y) - Z$, sendo I a função de interpolação utilizada.

Além disso, podemos expressar X , Y e Z linearmente em função do parâmetro do raio (t):

$$\begin{aligned} X &= p_x(t) \\ &= E_x + R_x t \\ Y &= p_y(t) \\ &= E_y + R_y t \\ Z &= p_z(t) \\ &= E_z + R_z t \end{aligned}$$

Nossa função de distância se torna, então,

$$D(t) = I(E_x + R_x t, E_y + R_y t) - E_z - R_z t.$$

Nesse caso, as coordenadas de interpolação u e v são lineares em t , e temos uma função de distância univariável em t cujo grau dependerá da função de interpolação adotada. Adotando a interpolação bicúbica proposta teremos uma função de sexto grau, pois o termo $u^3 v^3$ se torna um polinômio de sexto grau em t , e adotando a interpolação biquadrática proposta teremos uma função de quarto grau, pois o termo $u^2 v^2$ se torna um polinômio de quarto grau em t .

Nos deparamos, então, com a principal vantagem da interpolação biquadrática proposta: equações de quarto grau possuem solução analítica, que pode ser calculada a um custo baixo. Já a equação de sexto grau obtida a partir da interpolação bicúbica precisaria ser resolvida com um método numérico, que pode ser impreciso ou custoso.

Apesar desse detalhe interessante, nos restringimos a uma situação muito limitada, pois para conseguirmos a função de distância quártica determinamos que nossa superfície estivesse entre os vértices do quadrado unitário no plano (X, Y) . Podemos relaxar um pouco essa restrição, mantendo a função de distância ainda quártica.

3.2.2.2 Transformação linear

Assumamos que o espaço original possa ser convertido por uma transformação linear para o espaço de interpolação. Dessa forma, temos:

$$\begin{bmatrix} U_x & U_y & U_z \\ V_x & V_y & V_z \\ Z_x & Z_y & Z_z \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} u \\ v \\ z \end{bmatrix} \quad (3.3)$$

Sendo z o contradomínio da nossa função de interpolação (diferente da coordenada Z do espaço original), temos:

$$\begin{aligned} u &= U_x p_x(t) + U_y p_y(t) + U_z p_z(t) \\ &= U_x E_x + U_y E_y + U_z E_z + (U_x R_x + U_y R_y + U_z R_z)t \\ v &= V_x p_x(t) + V_y p_y(t) + V_z p_z(t) \\ &= V_x E_x + V_y E_y + V_z E_z + (V_x R_x + V_y R_y + V_z R_z)t \\ z &= Z_x p_x(t) + Z_y p_y(t) + Z_z p_z(t) \\ &= Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t \end{aligned}$$

Podemos, então, definir a função de distância como:

$$D(t) = I(u, v) - z$$

Os parâmetros de interpolação u e v continuam sendo lineares em t e, como consequência, o grau da função de distância permanece o mesmo de antes. Porém, ainda estamos com a superfície restrita demais, pois a transformação linear proposta só poderia modelar transformações simples, como o escalonamento e a rotação.

A transformação proposta pode ser modelada como uma matrix 3×3 que, multiplicada pelo vetor de um ponto, resulta no ponto transformado. Essa modelagem matricial é tão comum na computação gráfica que os hardwares disponíveis para aceleração gráfica atualmente são otimizados para realizar operações matriciais.

3.2.2.3 Transformação linear em coordenadas homogêneas

Não é possível modelar transformações como a translação no espaço tridimensional Euclidiano com uma matriz 3×3 . Uma alternativa para modelar essa e outras transformações é utilizar o espaço de coordenadas homogêneas (RIESENFELD, 1981). Nesse espaço uma nova coordenada w é adicionada a cada ponto, obtendo, então, um espaço de quatro dimensões. Essa coordenada adicional é considerada como um divisor para as demais coordenadas (u , v e z). Para converter um ponto (X, Y, Z) do espaço tridimensional Euclidiano para coordenadas homogêneas basta definir $u = X$, $v = Y$, $z = Z$ e $w = 1$, e para converter um ponto com coordenadas homogêneas para coordenadas tridimensionais Euclidianas basta definir $X = \frac{u}{w}$, $Y = \frac{v}{w}$, e $Z = \frac{z}{w}$.

No espaço de coordenadas homogêneas, podemos definir matrizes de transformação 4×4 que são muito mais abrangentes. Tais matrizes de transformação podem ser construídas levando quaisquer quatro pontos não-coplanares do espaço original para outros quatro pontos também não-coplanares do espaço transformado. Além disso, se $w = 0$, o ponto é considerado como "ponto no infinito", e toda reta que em coordenadas euclidianas é paralela ao vetor direção definido pelas coordenadas do ponto se cruza naquele ponto. Por exemplo, todas as retas que em coordenadas euclidianas são paralelas ao eixo z , quando levadas ao infinito negativo, se intersectam no ponto $u = 0$, $v = 0$, $z = -1$, $w = 0$ em coordenadas homogêneas. Em outras palavras, ao levar um ponto do espaço original para o infinito em coordenadas homogêneas, ele passa a ser um "ponto de fuga" de retas paralelas no espaço transformado, tornando paralelas todas as retas que o cruzam no espaço original.

Essa propriedade das transformações em coordenadas homogêneas é muito utilizada, por exemplo, para modelar visões em perspectiva: o "ponto de fuga" passa a ser a câmera, e os objetos a serem renderizados são transformados tais que suas coordenadas se tornam as coordenadas da imagem resultante e a profundidade. Utilizaremos a transformação, entretanto, de outra forma.

Utilizando o conceito de pontos de fuga, definimos uma transformação que leve três pontos quaisquer para os pontos $(0, 0, 0)$, $(1, 0, 0)$ e $(0, 1, 0)$, e um quarto "ponto de fuga" para o infinito na direção $(0, 0, -1)$. A superfície modelada ficará, no espaço

original, entre os vértices de um paralelogramo formado a partir dos três pontos escolhidos, que no espaço transformado representam os cantos do quadrado unitário do plano (u, v) . Além disso, toda reta vertical no espaço transformado corresponde a uma reta partindo do ponto de fuga no espaço original. Isso é importante, pois precisamos garantir que os planos verticais das bordas da superfície sejam iguais entre superfícies conectadas, e agora teremos essa garantia se ambas as superfícies utilizarem o mesmo ponto de fuga.

Com isso devemos conseguir modelar objetos compostos por várias superfícies conectadas em volta de um mesmo ponto de fuga.

Para simplificar a transformação, definamo-la em forma matricial. Dados os pontos A, B, D e o ponto de fuga F , temos:

$$\begin{bmatrix} U_x & U_y & U_z & U_w \\ V_x & V_y & V_z & V_w \\ Z_x & Z_y & Z_z & Z_w \\ W_x & W_y & W_z & w_w \end{bmatrix} \times \begin{bmatrix} A_x & B_x & D_x & F_x \\ A_y & B_y & D_y & F_y \\ A_z & B_z & D_z & F_z \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad (3.4)$$

Com essa transformação, a reflexão de A em torno da reta que cruza B e D será transformada para o vértice $(1, 1, 0)$, que é o vértice C das funções de interpolação propostas. Na prática, isso nos limita a modelar superfícies cujos vértices formem um paralelogramo.

Vale notar que, em espaços de coordenadas homogêneas, todo múltiplo não nulo de um ponto é equivalente quando convertido para coordenadas euclidianas $((\frac{\lambda u}{\lambda w}, \frac{\lambda v}{\lambda w}, \frac{\lambda z}{\lambda w}) = (\frac{u}{w}, \frac{v}{w}, \frac{z}{w}))$. Portanto, podemos multiplicar a matriz resultante por qualquer valor não nulo e a transformação representada continuará a mesma em coordenadas euclidianas.

Ao resolver a equação matricial proposta, é fácil perceber que todos os termos encontrados estão sendo divididos por uma expressão comum, que chamaremos de λ . Podemos, então, multiplicar a matriz por λ para reduzir a quantidade de operações necessárias, sem alterar a transformação representada pela mesma em coordenadas euclidianas. A transformação obtida (já feita a simplificação) tem as constantes:

$$\begin{aligned} U_x &= (D_y - F_y)(A_z - F_z) - (D_z - F_z)(A_y - F_y) \\ U_y &= (D_z - F_z)(A_x - F_x) - (D_x - F_x)(A_z - F_z) \\ U_z &= (D_x - F_x)(A_y - F_y) - (D_y - F_y)(A_x - F_x) \\ U_w &= -A_x U_x - A_y U_y - A_z U_z \\ V_x &= (A_y - F_y)(B_z - F_z) - (A_z - F_z)(B_y - F_y) \\ V_y &= (A_z - F_z)(B_x - F_x) - (A_x - F_x)(B_z - F_z) \\ V_z &= (A_x - F_x)(B_y - F_y) - (A_y - F_y)(B_x - F_x) \end{aligned}$$

$$\begin{aligned}
V_w &= -A_x V_x - A_y V_y - A_z V_z \\
Z_x &= (B_y - A_y)(D_z - A_z) - (B_z - A_z)(D_y - A_y) \\
Z_y &= (B_z - A_z)(D_x - A_x) - (B_x - A_x)(D_z - A_z) \\
Z_z &= (B_x - A_x)(D_y - A_y) - (B_y - A_y)(D_x - A_x) \\
Z_w &= -A_x Z_x - A_y Z_y - A_z Z_z \\
W_x &= (B_y - A_y)(D_z - A_z) - (B_z - A_z)(D_y - A_y) \\
&= Z_x \\
W_y &= (B_z - A_z)(D_x - A_x) - (B_x - A_x)(D_z - A_z) \\
&= Z_y \\
W_z &= (B_x - A_x)(D_y - A_y) - (B_y - A_y)(D_x - A_x) \\
&= Z_z \\
w_w &= -F_x W_x - F_y W_y - F_z W_z \\
&= -F_x Z_x - F_y Z_y - F_z Z_z
\end{aligned}$$

É fácil observar que algumas expressões envolvidas nos cálculos são repetidas. Portanto, é possível fazer o pré-cálculo de tais expressões para economizar operações. Além disso, as constantes W_x , W_y e W_z podem ser substituídas por Z_x , Z_y e Z_z .

Com as constantes calculadas, é possível recuperar o valor de λ ($\lambda = B_x U_x + B_y U_y + B_z U_z + U_w$, por exemplo). Assim, podemos obter a matriz original ao dividir todas as constantes por λ . Seria interessante comparar a estabilidade do método com e sem essa etapa, já que os resultados analíticos são os mesmos.

Aplicamos, então, a transformação obtida à função do raio e convertemos o resultado para coordenadas euclidianas, obtendo:

$$\begin{aligned}
u &= \frac{U_w + U_x p_x(t) + U_y p_y(t) + U_z p_z(t)}{w_w + W_x p_x(t) + W_y p_y(t) + W_z p_z(t)} \\
&= \frac{U_w + U_x E_x + U_y E_y + U_z E_z + (U_x R_x + U_y R_y + U_z R_z)t}{w_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t} \\
v &= \frac{V_w + V_x p_x(t) + V_y p_y(t) + V_z p_z(t)}{w_w + W_x p_x(t) + W_y p_y(t) + W_z p_z(t)} \\
&= \frac{V_w + V_x E_x + V_y E_y + V_z E_z + (V_x R_x + V_y R_y + V_z R_z)t}{w_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t} \\
z &= \frac{Z_w + Z_x p_x(t) + Z_y p_y(t) + Z_z p_z(t)}{w_w + W_x p_x(t) + W_y p_y(t) + W_z p_z(t)} \\
&= \frac{Z_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t}{w_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t}
\end{aligned}$$

Para simplificar as formulações, definamos E^* como o centro da câmera no espaço transformado com coordenadas homogêneas e R^* como a direção do raio nesse

mesmo espaço. As coordenadas euclidianas do raio no espaço transformado se tornam, então:

$$\begin{aligned}
u &= \frac{U_w + U_x E_x + U_y E_y + U_z E_z + (U_x R_x + U_y R_y + U_z R_z)t}{w_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t} \\
&= \frac{E_u^* + R_u^* t}{E_w^* + R_z^* t} \\
v &= \frac{V_w + V_x E_x + V_y E_y + V_z E_z + (V_x R_x + V_y R_y + V_z R_z)t}{w_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t} \\
&= \frac{E_v^* + R_v^* t}{E_w^* + R_z^* t} \\
z &= \frac{Z_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t}{w_w + Z_x E_x + Z_y E_y + Z_z E_z + (Z_x R_x + Z_y R_y + Z_z R_z)t} \\
&= \frac{E_z^* + R_z^* t}{E_w^* + R_z^* t}
\end{aligned}$$

Nesse caso, nossa função de distância se torna:

$$\begin{aligned}
D(t) &= I(u, v) - z \\
&= I\left(\frac{E_u^* + R_u^* t}{E_w^* + R_z^* t}, \frac{E_v^* + R_v^* t}{E_w^* + R_z^* t}\right) - \frac{E_z^* + R_z^* t}{E_w^* + R_z^* t}.
\end{aligned}$$

Considerando como função de interpolação I a função biquadrática proposta em (2.24), temos a função de distância:

$$\begin{aligned}
D(t) &= f\left(\frac{E_u^* + R_u^* t}{E_w^* + R_z^* t}, \frac{E_v^* + R_v^* t}{E_w^* + R_z^* t}\right) - \frac{E_z^* + R_z^* t}{E_w^* + R_z^* t} \\
&= \left(4(f^C - f^D - f^B + f^A) - 3(\partial f^{DC} - \partial f^{AB} - \partial f^{AD} + \partial f^{BC}) + \partial f^{CD} - \partial f^{BA} + \partial f^{CB} \right. \\
&\quad \left. - \partial f^{DA}\right) \frac{(E_u^* + R_u^* t)^2 (E_v^* + R_v^* t)^2}{(E_w^* + R_z^* t)^4} + 2(\partial f^{BC} - \partial f^{AD}) \frac{(E_u^* + R_u^* t)^2 (E_v^* + R_v^* t)}{(E_w^* + R_z^* t)^3} \\
&\quad + \left(2(f^B - f^A) - \frac{3\partial f^{AB} - \partial f^{BA}}{2}\right) \frac{(E_u^* + R_u^* t)^2}{(E_w^* + R_z^* t)^2} \\
&\quad + 2(\partial f^{DC} - \partial f^{AB}) \frac{(E_u^* + R_u^* t)(E_v^* + R_v^* t)^2}{(E_w^* + R_z^* t)^3} + \partial f^{AB} \frac{E_u^* + R_u^* t}{E_w^* + R_z^* t} \\
&\quad + \left(2(f^D - f^A) + \frac{\partial f^{DA} - 3\partial f^{AD}}{2}\right) \frac{(E_v^* + R_v^* t)^2}{(E_w^* + R_z^* t)^2} + \partial f^{AD} \frac{E_v^* + R_v^* t}{E_w^* + R_z^* t} + f^A - \frac{E_z^* + R_z^* t}{E_w^* + R_z^* t}.
\end{aligned}$$

Nomeemos os seguintes coeficientes para simplificar as formulações:

$$\begin{aligned}
e_1 &= \left(4(f^C - f^D - f^B + f^A) - 3(\partial f^{DC} - \partial f^{AB} - \partial f^{AD} + \partial f^{BC})\right. \\
&\quad \left. + \partial f^{CD} - \partial f^{BA} + \partial f^{CB} - \partial f^{DA}\right) \\
e_2 &= 2(\partial f^{BC} - \partial f^{AD}) \\
e_3 &= \left(2(f^B - f^A) - \frac{3\partial f^{AB} - \partial f^{BA}}{2}\right) \\
e_4 &= 2(\partial f^{DC} - \partial f^{AB}) \\
e_5 &= \left(2(f^D - f^A) + \frac{\partial f^{DA} - 3\partial f^{AD}}{2}\right)
\end{aligned}$$

Assim, obtemos:

$$\begin{aligned}
D(t) &= e_1 \frac{(E_u^* + R_u^*t)^2 (E_v^* + R_v^*t)^2}{(E_w^* + R_z^*t)^4} + e_2 \frac{(E_u^* + R_u^*t)^2 (E_v^* + R_v^*t)}{(E_w^* + R_z^*t)^3} \\
&\quad + e_3 \frac{(E_u^* + R_u^*t)^2}{(E_w^* + R_z^*t)^2} + e_4 \frac{(E_u^* + R_u^*t)(E_v^* + R_v^*t)^2}{(E_w^* + R_z^*t)^3} + \partial f^{AB} \frac{E_u^* + R_u^*t}{E_w^* + R_z^*t} \\
&\quad + e_5 \frac{(E_v^* + R_v^*t)^2}{(E_w^* + R_z^*t)^2} + \partial f^{AD} \frac{E_v^* + R_v^*t}{E_w^* + R_z^*t} + f^A - \frac{E_z^* + R_z^*t}{E_w^* + R_z^*t}.
\end{aligned}$$

Nosso objetivo é encontrar as raízes da função de distância, ou seja, as raízes de $D(t) = 0$. Multiplicando esta equação por $(E_w^* + R_z^*t)^4$ podemos eliminar as divisões da equação mantendo as mesmas raízes:

$$\begin{aligned}
(E_w^* + R_z^*t)^4 D(t) &= e_1(E_u^* + R_u^*t)^2(E_v^* + R_v^*t)^2 + e_2(E_u^* + R_u^*t)^2(E_v^* + R_v^*t)(E_w^* + R_z^*t) \\
&\quad + e_3(E_u^* + R_u^*t)^2(E_w^* + R_z^*t)^2 + e_4(E_u^* + R_u^*t)(E_v^* + R_v^*t)^2(E_w^* + R_z^*t) \\
&\quad + \partial f^{AB}(E_u^* + R_u^*t)(E_w^* + R_z^*t)^3 + e_5(E_v^* + R_v^*t)^2(E_w^* + R_z^*t)^2 \\
&\quad + \partial f^{AD}(E_v^* + R_v^*t)(E_w^* + R_z^*t)^3 + f^A(E_w^* + R_z^*t)^4 - (E_z^* + R_z^*t)(E_w^* + R_z^*t)^3 \\
&= \left(e_1 R_u^{*2} R_v^{*2} + \left((e_5 R_z^* + R_u^* e_4) R_v^{*2} + (e_3 R_z^* + e_2 R_v^*) R_u^{*2} \right. \right. \\
&\quad \left. \left. + (\partial f^{AB} R_u^* - R_z^* + \partial f^{AD} R_v^* + f^A R_z^*) R_z^{*2} \right) R_z^* \right) t^4 \\
&\quad + \left(\left((3(\partial f^{AD} R_v^* - R_z^* + \partial f^{AB} R_u^*) E_w^* + (\partial f^{AB} E_u^* - E_z^* + \partial f^{AD} E_v^*) R_z^* \right. \right. \\
&\quad \left. \left. + 2(E_u^* R_u^* e_3 + E_v^* R_v^* e_5 + 2f^A E_w^* R_z^*) \right) R_z^* + ((E_v^* e_2 + 2E_w^* e_3) R_u^* \right. \\
&\quad \left. + 2(E_u^* e_2 + E_v^* e_4) R_v^*) R_u^* + (E_u^* e_4 + 2E_w^* e_5) R_v^{*2} \right) R_z^* \\
&\quad \left. + (2(E_u^* R_v^* + E_v^* R_u^*) e_1 + (R_u^* e_2 + R_v^* e_4) E_w^*) R_u^* R_v^* \right) t^3 \\
&\quad + \left(\left((3((\partial f^{AB} E_u^* - E_z^* + \partial f^{AD} E_v^*) R_z^* + (\partial f^{AB} R_u^* - R_z^* + \partial f^{AD} R_v^*) E_w^* \right. \right. \\
&\quad \left. \left. + 2f^A E_w^* R_z^*) + 4(E_u^* R_u^* e_3 + E_v^* R_v^* e_5) \right) R_z^* + ((E_v^* e_2 + E_w^* e_3) R_u^* \right. \\
&\quad \left. + 2(E_u^* e_2 + E_v^* e_4) R_v^*) R_u^* + (E_u^* e_4 + E_w^* e_5) R_v^{*2} \right) E_w^* \\
&\quad + \left(\left((R_z^* e_3 + R_v^* e_2) E_u^* + 2(R_u^* e_2 + R_v^* e_4) E_v^* \right) E_u^* \right. \\
&\quad \left. + (R_z^* e_5 + R_u^* e_4) E_v^{*2} \right) R_z^* + (E_u^{*2} R_v^{*2} + (E_v^* R_u^* + 4E_u^* R_v^*) E_v^* R_u^*) e_1 \Big) t^2 \\
&\quad + \left(\left(2(E_v^* R_u^* + E_u^* R_v^*) e_1 + (E_u^* e_2 + E_v^* e_4) R_z^* \right) E_u^* E_v^* \right. \\
&\quad + \left(\left((R_v^* e_2 + 2R_z^* e_3) E_u^* + 2(R_u^* e_2 + R_v^* e_4) E_v^* \right) E_u^* + (R_u^* e_4 + 2R_z^* e_5) E_v^{*2} \right. \\
&\quad \left. + ((\partial f^{AB} R_u^* - R_z^* + \partial f^{AD} R_v^*) E_w^* + 3(\partial f^{AB} E_u^* - E_z^* + \partial f^{AD} E_v^*) R_z^*) E_w^* \right. \\
&\quad \left. + 2(R_u^* E_u^* e_3 + R_v^* E_v^* e_5 + 2f^A E_w^* R_z^*) E_w^* \right) E_w^* \Big) t \\
&\quad + e_1 E_u^{*2} E_v^{*2} + \left((e_2 E_u^* + e_4 E_v^*) E_u^* E_v^* + (e_3 E_u^{*2} + e_5 E_v^{*2} \right. \\
&\quad \left. + (\partial f^{AB} E_u^* - E_z^* + \partial f^{AD} E_v^* + f^A E_w^*) E_w^* \right) E_w^* \\
&= 0.
\end{aligned} \tag{3.6}$$

Verificamos que a transformação linear em coordenadas homogêneas proposta não aumenta o grau da equação, que permanece sendo de quarto grau em t .

3.2.2.4 Domínio na variável t

Sabemos que, no espaço transformado, o domínio da função é o quadrante $0 \leq u \leq 0.5$ e $0 \leq v \leq 0.5$ com $w > 0$. Porém, agora é interessante expressarmos essas desigualdades em função de t . Temos, então, as seguintes desigualdades:

$$\begin{aligned} 0 &\leq \frac{E_u^* + R_u^* t}{E_w^* + R_z^* t} \leq 0.5 \\ 0 &\leq \frac{E_v^* + R_v^* t}{E_w^* + R_z^* t} \leq 0.5 \\ E_w^* + R_z^* t &> 0 \end{aligned}$$

As restrições em t dependerão dos sinais de alguns termos. Sabendo, pela última restrição, que $E_w^* + R_z^* t > 0$, obtemos as seguintes restrições sobre t :

$$\begin{cases} t \geq -\frac{E_u^*}{R_u^*}, & \text{se } R_u^* > 0 \\ t \leq -\frac{E_u^*}{R_u^*}, & \text{caso contrário} \end{cases}$$

$$\begin{cases} t \leq \frac{E_u^* - 2E_w^*}{2R_u^* - R_z^*}, & \text{se } 2R_u^* - R_z^* > 0 \\ t \geq \frac{E_u^* - 2E_w^*}{2R_u^* - R_z^*}, & \text{caso contrário} \end{cases}$$

$$\begin{cases} t \geq -\frac{E_v^*}{R_v^*}, & \text{se } R_v^* > 0 \\ t \leq -\frac{E_v^*}{R_v^*}, & \text{caso contrário} \end{cases}$$

$$\begin{cases} t \leq \frac{E_v^* - 2E_w^*}{2R_v^* - R_z^*}, & \text{se } 2R_v^* - R_z^* > 0 \\ t \geq \frac{E_v^* - 2E_w^*}{2R_v^* - R_z^*}, & \text{caso contrário} \end{cases}$$

$$\begin{cases} t > -\frac{E_w^*}{R_z^*}, & \text{se } R_z^* > 0 \\ t < -\frac{E_w^*}{R_z^*}, & \text{caso contrário} \end{cases}$$

A partir destas restrições é possível determinar o intervalo de valores de t que corresponde ao domínio da função de interpolação. Valores de t fora deste intervalo não devem ser considerados intersecções com a superfície.

3.2.2.5 Constantes de altura

As constantes f^A , f^B , f^C e f^D definem a altura (coordenada z) da superfície nos cantos do quadrado unitário no espaço transformado, o que representa, no espaço original, a distância dos mesmos ao ponto de fuga. Porém, como essas alturas estão no espaço transformado, é desejável que possamos determinar as mesmas a partir do espaço original.

Para encontrar a altura do canto no espaço transformado basta encontrarmos o ponto equivalente no espaço original, o transformarmos com a mesma transformação anterior, e então isolarmos sua coordenada z .

Analisemos o cálculo de f^A a partir do espaço original.

Sabemos, pela equação 3.4, que ao transformar o ponto A obtemos as coordenadas $u = 0$, $v = 0$, $z = 0$ e $w = \lambda$ no espaço transformado com coordenadas homogêneas (sendo λ a constante não nula usada para simplificar a transformação). Portanto, podemos formar as seguintes igualdades:

$$\begin{aligned} Z_x A_x + Z_y A_y + Z_z A_z &= -Z_w \\ W_x A_x + W_y A_y + W_z A_z &= \lambda - w_w \end{aligned}$$

Sabemos, também pela equação 3.4, que ao transformar o ponto de fuga F , obtemos as coordenadas $u = 0$, $v = 0$, $z = -\lambda$ e $w = 0$ no espaço transformado. Portanto, podemos formar as seguintes igualdades:

$$\begin{aligned} Z_x F_x + Z_y F_y + Z_z F_z &= -Z_w - \lambda \\ W_x F_x + W_y F_y + W_z F_z &= -w_w \end{aligned}$$

Podemos definir a coordenada z do canto A da superfície pelo espaço original a partir de um multiplicador $m_A > 0$, tal que a distância entre o canto da superfície e o ponto de fuga no espaço original seja m_A vezes a distância entre o ponto de fuga e o ponto A . Nesse caso, se $m_A = 1$, o ponto da superfície com $u = 0$ e $v = 0$ coincide com o ponto A . As coordenadas do canto da superfície no espaço original são, então:

$$\begin{aligned} X &= F_x + m_A(A_x - F_x) \\ Y &= F_y + m_A(A_y - F_y) \\ Z &= F_z + m_A(A_z - F_z) \end{aligned}$$

Calculamos a coordenada da altura (z) aplicando a transformação da equação (3.5) a esse ponto e simplificamos a expressão obtida utilizando as igualdades listadas anteriormente. Obtemos assim:

$$\begin{aligned}
f^A &= \frac{Z_w + Z_x(F_x + m_A(A_x - F_x)) + Z_y(F_y + m_A(A_y - F_y)) + Z_z(F_z + m_A(A_z - F_z))}{w_w + W_x(F_x + m_A(A_x - F_x)) + W_y(F_y + m_A(A_y - F_y)) + W_z(F_z + m_A(A_z - F_z))} \\
&= \frac{Z_w + m_A(Z_x A_x + Z_y A_y + Z_z A_z) + (1 - m_A)(Z_x F_x + Z_y F_y + Z_z F_z)}{w_w + m_A(W_x A_x + W_y A_y + W_z A_z) + (1 - m_A)(W_x F_x + W_y F_y + W_z F_z)} \\
&= \frac{Z_w - m_A Z_w - (1 - m_A)(\lambda + Z_w)}{w_w + m_A(\lambda - w_w) - (1 - m_A)w_w} \\
&= \frac{\lambda m_A - \lambda}{\lambda m_A} \\
&= \frac{m_A - 1}{m_A}
\end{aligned}$$

O mesmo processo pode ser aplicado aos demais cantos da superfície para calcular f^B , f^C e f^D a partir dos multiplicadores m_B , m_C e m_D .

3.2.2.6 Constantes de derivadas

Além das coordenadas z dos cantos, também é desejável que as derivadas sejam definidas a partir do espaço original. A forma mais intuitiva de fazer isso é definir um plano tangente à superfície em cada canto e, então, calcular as derivadas de forma que a superfície seja tangente a esses planos.

Seja N^A a direção normal ao plano tangente ao canto da superfície com $u = 0$ e $v = 0$ no espaço original.

Sabemos que o produto vetorial entre duas direções resulta em uma nova direção perpendicular a ambas. Uma forma simples de calcular a derivada representada por ∂f^{AB} consiste em calcular o produto vetorial de N^A com a direção normal à face que conecta o ponto de fuga aos pontos A e B (que representa a borda $v = 0$ no espaço transformado). O vetor resultante será tangente ao canto da superfície com $u = 0$ e $v = 0$ no espaço original e estará contido no plano que representa a borda $v = 0$.

Como N^A já está pré-definido, o primeiro passo é calcular a direção normal da face conectando o ponto de fuga F aos pontos A e B . Para isso calculamos o produto vetorial entre os vetores $A - F$ e $B - F$:

$$\begin{aligned}
N_x^{AB} &= (A_y - F_y)(B_z - F_z) - (A_z - F_z)(B_y - F_y) \\
N_y^{AB} &= (A_z - F_z)(B_x - F_x) - (A_x - F_x)(B_z - F_z) \\
N_z^{AB} &= (A_x - F_x)(B_y - F_y) - (A_y - F_y)(B_x - F_x)
\end{aligned}$$

Verificamos, então, que essa direção normal é equivalente às constantes V_x , V_y e V_z da equação (3.5), de modo que podemos usar esses valores, evitando cálculos desnecessários.

Finalmente, calculamos o produto vetorial V^{AB} de N^A com N^{AB} :

$$\begin{aligned} V_x^{AB} &= N_y^A N_z^{AB} - N_z^A N_y^{AB} \\ &= N_y^A V_z - N_z^A V_y \\ V_y^{AB} &= N_z^A N_x^{AB} - N_x^A N_z^{AB} \\ &= N_z^A V_x - N_x^A V_z \\ V_z^{AB} &= N_x^A N_y^{AB} - N_y^A N_x^{AB} \\ &= N_x^A V_y - N_y^A V_x \end{aligned}$$

Conhecendo o vetor V^{AB} , que é tangente à fronteira da superfície que conecta os vértices A e B no ponto em que $u = 0$ e $v = 0$, modelamos os pontos partindo desse canto, nessa direção. Tais pontos no espaço original são da forma:

$$q(x) = F + m_A(A - F) + V^{AB}x \quad (3.7)$$

Para facilitar os próximos passos, definamos alguns vetores no espaço transformado com coordenadas homogêneas.

Primeiro, o canto A^* da superfície no qual $u = 0$ e $v = 0$:

$$\begin{aligned} A_u^* &= U_x(F_x + m_A(A_x - F_x)) + U_y(F_y + m_A(A_y - F_y)) + U_z(F_z + m_A(A_z - F_z)) + U_w \\ A_v^* &= V_x(F_x + m_A(A_x - F_x)) + V_y(F_y + m_A(A_y - F_y)) + V_z(F_z + m_A(A_z - F_z)) + V_w \\ A_z^* &= Z_x(F_x + m_A(A_x - F_x)) + Z_y(F_y + m_A(A_y - F_y)) + Z_z(F_z + m_A(A_z - F_z)) + Z_w \\ A_w^* &= Z_x(F_x + m_A(A_x - F_x)) + Z_y(F_y + m_A(A_y - F_y)) + Z_z(F_z + m_A(A_z - F_z)) + w_w \end{aligned}$$

Em seguida, o vetor tangente à fronteira nesse canto:

$$\begin{aligned} V_u^{AB*} &= U_x V_x^{AB} + U_y V_y^{AB} + U_z V_z^{AB} \\ V_v^{AB*} &= V_x V_x^{AB} + V_y V_y^{AB} + V_z V_z^{AB} \\ V_z^{AB*} &= Z_x V_x^{AB} + Z_y V_y^{AB} + Z_z V_z^{AB} \\ V_w^{AB*} &= Z_x V_x^{AB} + Z_y V_y^{AB} + Z_z V_z^{AB} \\ &= V_z^{AB*} \end{aligned}$$

De posse desses vetores, aplicamos a transformação aos pontos formados por (3.7), obtendo:

$$q^*(x) = A^* + V^{AB*}t.$$

Analisemos agora, no espaço transformado com coordenadas euclidianas, o coeficiente angular ∂f^{AB} da reta que passa por tais pontos e pelo canto A^* , na borda com $v = 0$:

$$\begin{aligned}
\partial f^{AB} &= \frac{\frac{q^*(x)_z}{q^*(x)_w} - \frac{A_z^*}{A_w^*}}{\frac{q^*(x)_u}{q^*(x)_w} - \frac{A_u^*}{A_w^*}} \\
&= \frac{\frac{A_z^* + V_z^{AB*}t}{A_w^* + V_z^{AB*}t} - \frac{A_z^*}{A_w^*}}{\frac{A_u^* + V_u^{AB*}t}{A_w^* + V_z^{AB*}t} - \frac{A_u^*}{A_w^*}} \\
&= \frac{\frac{A_z^* + V_z^{AB*}t}{A_w^* (1 + \frac{V_z^{AB*}}{A_w^*}t)} - \frac{A_z^* (1 + \frac{V_z^{AB*}}{A_w^*}t)}{A_w^* (1 + \frac{V_z^{AB*}}{A_w^*}t)}}{\frac{A_u^* + V_u^{AB*}t}{A_w^* (1 + \frac{V_z^{AB*}}{A_w^*}t)} - \frac{A_u^* (1 + \frac{V_z^{AB*}}{A_w^*}t)}{A_w^* (1 + \frac{V_z^{AB*}}{A_w^*}t)}} \\
&= \frac{A_z^* + V_z^{AB*}t - A_z^* (1 + \frac{V_z^{AB*}}{A_w^*}t)}{A_u^* + V_u^{AB*}t - A_u^* (1 + \frac{V_z^{AB*}}{A_w^*}t)} \\
&= \frac{V_z^{AB*}t - \frac{A_z^* V_z^{AB*}}{A_w^*}t}{V_u^{AB*}t - \frac{A_u^* V_z^{AB*}}{A_w^*}t} \\
&= \frac{V_z^{AB*} - \frac{A_z^*}{A_w^*} V_z^{AB*}}{V_u^{AB*} - \frac{A_u^*}{A_w^*} V_z^{AB*}}.
\end{aligned}$$

Sabendo que $\frac{A_z^*}{A_w^*} = f^A$ e $\frac{A_u^*}{A_w^*} = 0$, temos:

$$\begin{aligned}
\partial f^{AB} &= \frac{V_z^{AB*} - \frac{A_z^*}{A_w^*} V_z^{AB*}}{V_u^{AB*} - \frac{A_u^*}{A_w^*} V_z^{AB*}} \\
&= \frac{V_z^{AB*} - f^A V_z^{AB*}}{V_u^{AB*}} \\
&= \frac{V_z^{AB*}}{V_u^{AB*}} (1 - f^A).
\end{aligned}$$

Nota-se que o ponto A^* é desnecessário para o cálculo da derivada, de modo que não é preciso calculá-lo, e a coordenada $V_v^{AB*} = 0$ também é desnecessária, sendo sempre nula.

O mesmo processo pode ser aplicado para o cálculo das demais derivadas a partir das direções normais N^A , N^B , N^C e N^D .

É possível concluir também que

$$\begin{aligned}
 N^{BC} &= \begin{bmatrix} Z_x - U_x \\ Z_y - U_y \\ Z_z - U_z \end{bmatrix}, \\
 N^{CD} &= \begin{bmatrix} Z_x - V_x \\ Z_y - V_y \\ Z_z - V_z \end{bmatrix} \text{ e} \\
 N^{DA} &= \begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix},
 \end{aligned}$$

reduzindo o número de operações necessárias.

Essas constantes de derivadas só precisam ser calculadas em todos os quadros de uma animação se a superfície tiver direções normais animadas ou se os pontos forem animados. Caso contrário, basta realizar o cálculo uma vez e armazenar as constantes obtidas para que sejam usadas na renderização de cada novo quadro.

3.2.2.7 Algoritmo

Agora que todas as constantes estão bem resolvidas, podemos montar o algoritmo de renderização, que está dividido em duas etapas.

O Algoritmo 3.1 faz o pré-processamento, calculando a matriz de transformação e as constantes utilizadas nos coeficientes da equação. Se os pontos da superfície e suas direções normais forem estáticos, essa etapa só precisa ser executada uma vez, desde que as constantes $e_1, e_2, e_3, e_4, e_5, ab, ad, a, U, V, Z$ e w_w sejam armazenadas para posterior uso.

Já o Algoritmo 3.2 calcula os coeficientes da equação (3.6) e encontra a intersecção mais próxima entre o raio partindo da câmera e os quatro quadrantes da superfície.

Os algoritmos foram desenvolvidos com operações vetoriais (para utilizar os recursos de aceleração das placas de vídeo), sendo que $cross(V_1, V_2)$ representa o produto vetorial de V_1 com V_2 , $min(V_1, V_2)$ equivale ao vetor cujos componentes são os menores componentes em cada coordenada de V_1 e V_2 , $max(V_1, V_2)$ equivale ao vetor cujos componentes são os maiores componentes em cada coordenada de V_1 e V_2 , $dot(V_1, V_2)$ representa o produto interno de V_1 com V_2 , e $V_1 * V_2$ é a multiplicação elemento-a-elemento entre os vetores V_1 e V_2 .

O valor das constantes foi colocado em colunas diferentes para quadrantes diferentes. Portanto, é possível encontrar as raízes da equação de quarto grau em todos os quadrantes simultaneamente através de operações matriciais aceleradas pela placa de

vídeo. Essas raízes podem ser encontradas por qualquer método (analítico ou numérico), de modo que o mesmo não foi definido no algoritmo.

Algoritmo 3.1 Pré-processamento

Require: $A, B, D, F, m_A, m_B, m_C, m_D, N_A, N_B, N_C$ e N_D

- 1: $AF \leftarrow A - F$
 - 2: $BF \leftarrow B - F$
 - 3: $DF \leftarrow D - F$
 - 4: $BA \leftarrow B - A$
 - 5: $DA \leftarrow D - A$
 - 6: $U \leftarrow \text{cross}(DF, AF)$
 - 7: $u_w \leftarrow -\text{dot}(A, U)$
 - 8: $V \leftarrow \text{cross}(AF, BF)$
 - 9: $v_w \leftarrow -\text{dot}(A, V)$
 - 10: $Z \leftarrow \text{cross}(BA, DA)$
 - 11: $z_w \leftarrow -\text{dot}(A, Z)$
 - 12: $w_w \leftarrow -\text{dot}(F, Z)$
 - 13: $aux_a \leftarrow (m_A - 1)/m_A$
 - 14: $aux_b \leftarrow (m_B - 1)/m_B$
 - 15: $aux_c \leftarrow (m_C - 1)/m_C$
 - 16: $aux_d \leftarrow (m_D - 1)/m_D$
 - 17: $a \leftarrow \begin{bmatrix} aux_a & aux_b & aux_c & aux_d \end{bmatrix}$
 - 18: $b \leftarrow \begin{bmatrix} aux_b & aux_c & aux_d & aux_a \end{bmatrix}$
 - 19: $c \leftarrow \begin{bmatrix} aux_c & aux_d & aux_a & aux_b \end{bmatrix}$
 - 20: $d \leftarrow \begin{bmatrix} aux_d & aux_a & aux_b & aux_c \end{bmatrix}$
 - 21: $N^{BC} \leftarrow Z - U$
 - 22: $N^{CD} \leftarrow Z - V$
 - 23: $V^{AB} \leftarrow \text{cross}(N^A, V)$
 - 24: $V^{BA} \leftarrow \text{cross}(N^B, V)$
 - 25: $V^{BC} \leftarrow \text{cross}(N^B, N^{BC})$
 - 26: $V^{CB} \leftarrow \text{cross}(N^C, N^{BC})$
 - 27: $V^{CD} \leftarrow \text{cross}(N^C, N^{CD})$
 - 28: $V^{DC} \leftarrow \text{cross}(N^D, N^{DC})$
 - 29: $V^{DA} \leftarrow \text{cross}(N^D, U)$
 - 30: $V^{AD} \leftarrow \text{cross}(N^A, U)$
-

$$\begin{aligned}
31: & \text{aux}_a \leftarrow 1 - \text{aux}_a \\
32: & \text{aux}_b \leftarrow 1 - \text{aux}_b \\
33: & \text{aux}_c \leftarrow 1 - \text{aux}_c \\
34: & \text{aux}_d \leftarrow 1 - \text{aux}_d \\
35: & \text{aux}_{ab} \leftarrow \frac{\text{dot}(Z, V^{AB})}{\text{dot}(U, V^{AB})} * \text{aux}_a \\
36: & \text{aux}_{ba} \leftarrow \frac{\text{dot}(Z, V^{BA})}{\text{dot}(N^{BC}, V^{BA})} * \text{aux}_b \\
37: & \text{aux}_{bc} \leftarrow \frac{\text{dot}(Z, V^{BC})}{\text{dot}(V, V^{BC})} * \text{aux}_b \\
38: & \text{aux}_{cb} \leftarrow \frac{\text{dot}(Z, V^{CB})}{\text{dot}(N^{CD}, V^{CB})} * \text{aux}_c \\
39: & \text{aux}_{cd} \leftarrow \frac{\text{dot}(Z, V^{CD})}{\text{dot}(N^{BC}, V^{CD})} * \text{aux}_c \\
40: & \text{aux}_{dc} \leftarrow \frac{\text{dot}(Z, V^{DC})}{\text{dot}(U, V^{DC})} * \text{aux}_d \\
41: & \text{aux}_{da} \leftarrow \frac{\text{dot}(Z, V^{DA})}{\text{dot}(N^{CD}, V^{DA})} * \text{aux}_d \\
42: & \text{aux}_{ad} \leftarrow \frac{\text{dot}(Z, V^{AD})}{\text{dot}(V, V^{AD})} * \text{aux}_a \\
43: & ab \leftarrow \left[\begin{array}{cccc} \text{aux}_{ab} & \text{aux}_{bc} & \text{aux}_{cd} & \text{aux}_{da} \end{array} \right] \\
44: & ba \leftarrow \left[\begin{array}{cccc} \text{aux}_{ba} & \text{aux}_{cb} & \text{aux}_{dc} & \text{aux}_{ad} \end{array} \right] \\
45: & bc \leftarrow \left[\begin{array}{cccc} \text{aux}_{bc} & \text{aux}_{cd} & \text{aux}_{da} & \text{aux}_{ab} \end{array} \right] \\
46: & cb \leftarrow \left[\begin{array}{cccc} \text{aux}_{cb} & \text{aux}_{dc} & \text{aux}_{ad} & \text{aux}_{ba} \end{array} \right] \\
47: & cd \leftarrow \left[\begin{array}{cccc} \text{aux}_{cd} & \text{aux}_{da} & \text{aux}_{ab} & \text{aux}_{bc} \end{array} \right] \\
48: & dc \leftarrow \left[\begin{array}{cccc} \text{aux}_{dc} & \text{aux}_{ad} & \text{aux}_{ba} & \text{aux}_{cb} \end{array} \right] \\
49: & da \leftarrow \left[\begin{array}{cccc} \text{aux}_{da} & \text{aux}_{ab} & \text{aux}_{bc} & \text{aux}_{cd} \end{array} \right] \\
50: & ad \leftarrow \left[\begin{array}{cccc} \text{aux}_{ad} & \text{aux}_{ba} & \text{aux}_{cb} & \text{aux}_{dc} \end{array} \right] \\
51: & e_3 \leftarrow 2 * (b - a) - (3 * ab - ba) / 2 \\
52: & e_5 \leftarrow 2 * (d - a) - (3 * ad - da) / 2 \\
53: & e_1 \leftarrow cd - ba + cb - da - 3 * (dc + bc) - 2 * (e_3 + e_5 + 4 * a - 2 * c) \\
54: & e_2 \leftarrow 2 * (bc - ad) \\
55: & e_4 \leftarrow 2 * (dc - ab)
\end{aligned}$$

Algoritmo 3.2 Encontrar intersecção mais próxima

Require: $e_1, e_2, e_3, e_4, e_5, ab, ad, a, U, u_w, V, v_w, Z, z_w, w_w, E$ e R

$$\begin{aligned}
1: & R^* \leftarrow \left[\begin{array}{ccc} \text{dot}(U, R) & \text{dot}(V, R) & \text{dot}(Z, R) \end{array} \right] \\
2: & R^* \leftarrow \left[\begin{array}{cccc} R_u^* & R_v^* & R_z^* - R_u^* & R_z^* - R_v^* \\ R_v^* & R_z^* - R_u^* & R_z^* - R_v^* & R_u^* \\ R_z^* & R_z^* & R_z^* & R_z^* \end{array} \right]
\end{aligned}$$

-
- 3: $E^* \leftarrow \left[\begin{array}{ccc} \text{dot}(U, E) + u_w & \text{dot}(V, E) + v_w & \text{dot}(Z, E) + z_w \end{array} \right]$
 - 4: $E^* \leftarrow \left[\begin{array}{c} E^* \quad w_w - z_w + E_z^* \end{array} \right]$
 - 5: $E^* \leftarrow \left[\begin{array}{cccc} E_u^* & E_v^* & E_w^* - E_u^* & E_w^* - E_v^* \\ E_v^* & E_w^* - E_u^* & E_w^* - E_v^* & E_u^* \\ E_z^* & E_z^* & E_z^* & E_z^* \\ E_w^* & E_w^* & E_w^* & E_w^* \end{array} \right]$
 - 6: $\text{aux}_1 \leftarrow ab * E_u^* - E_z^* + ad * E_v^*$
 - 7: $\text{aux}_2 \leftarrow \text{aux}_1 * R_z^*$
 - 8: $\text{aux}_3 \leftarrow ab * R_u^* - R_z^* + ad * R_v^*$
 - 9: $\text{aux}_4 \leftarrow \text{aux}_3 * E_w^*$
 - 10: $\text{aux}_5 \leftarrow 2 * (E_u^* * R_u^* * e_3 + E_v^* * R_v^* * e_5)$
 - 11: $\text{aux}_6 \leftarrow 2 * (E_u^* * R_v^* + E_v^* * R_u^*) * e_1$
 - 12: $\text{aux}_7 \leftarrow e_5 * R_z^* + R_u^* * e_4$
 - 13: $\text{aux}_8 \leftarrow E_v^* * e_2 + E_w^* * e_3$
 - 14: $\text{aux}_9 \leftarrow E_u^* * e_2 + E_v^* * e_4$
 - 15: $\text{aux}_{10} \leftarrow 2 * \text{aux}_9$
 - 16: $\text{aux}_{11} \leftarrow e_3 * R_z^* + e_2 * R_v^*$
 - 17: $\text{aux}_{12} \leftarrow \text{aux}_{11} * E_u^*$
 - 18: $\text{aux}_{13} \leftarrow (E_u^* * e_4 + E_w^* * e_5) * R_v^{*2}$
 - 19: $\text{aux}_{14} \leftarrow R_u^* * e_2 + R_v^* * e_4$
 - 20: $\text{aux}_{15} \leftarrow (\text{aux}_8 * R_u^* + \text{aux}_{10} * R_v^*) * R_u^*$
 - 21: $\text{aux}_{16} \leftarrow 2 * \text{aux}_{14} * E_v^*$
 - 22: $\text{aux}_{17} \leftarrow E_u^* * E_v^*$
 - 23: $\text{aux}_{18} \leftarrow R_u^* * R_v^*$
 - 24: $\text{aux}_{19} \leftarrow 3 * \text{aux}_4 + \text{aux}_2$
 - 25: $c_4 \leftarrow e_1 * \text{aux}_{18}^2 + \left(\text{aux}_{11} * R_u^{*2} + \text{aux}_7 * R_v^{*2} + \text{aux}_3 * R_z^{*2} \right) * R_z^*$
 - 26: $c_3 \leftarrow \left((\text{aux}_{19} + \text{aux}_5) * R_z^* + \text{aux}_{15} + E_w^* * e_3 * R_u^{*2} + \text{aux}_{13} + E_w^* * e_5 * R_v^{*2} \right) * R_z^* + \left(\text{aux}_6 + \text{aux}_{14} * E_w^* \right) * \text{aux}_{18}$
 - 27: $c_2 \leftarrow \left((2 * (\text{aux}_2 + \text{aux}_5) + \text{aux}_{19}) * R_z^* + \text{aux}_{15} + \text{aux}_{13} \right) * E_w^* + \left((\text{aux}_{11} + \text{aux}_{16}) * E_u^* + \text{aux}_7 * E_v^{*2} \right) * R_z^* + \left(E_u^{*2} * R_v^{*2} + E_v^{*2} * R_u^{*2} + 4 * \text{aux}_{17} * \text{aux}_{18} \right) * e_1$
 - 28: $c_1 \leftarrow \left(\text{aux}_6 + \text{aux}_9 * R_z^* \right) * \text{aux}_{17} + \left((\text{aux}_{16} + \text{aux}_{12} + R_z^* * e_3 * E_u^*) * E_u^* + (\text{aux}_7 + R_z^* * e_5) * E_v^{*2} + (\text{aux}_4 + 3 * \text{aux}_2 + \text{aux}_5) * E_w^* \right) * E_w^*$
 - 29: $c_0 \leftarrow e_1 * \text{aux}_{17}^2 + \left(\text{aux}_9 * \text{aux}_{17} + (e_3 * E_u^{*2} + e_5 * E_v^{*2} + \text{aux}_1 * E_w^*) * E_w^* \right) * E_w^* + a$
 - 30: $[t_{1,1}, \dots, t_{4,4}] \leftarrow \text{soluções de } c_4 * t^4 + c_3 * t^3 + c_2 * t^2 + c_1 * t + c_0 = 0 \text{ (índices representam quadrante e solução, respectivamente)}$
-

```

31:  $t^{min} = [ 0 \ 0 \ 0 \ 0 ]$ 
32:  $t^{max} = [ \infty \ \infty \ \infty \ \infty ]$ 
33: for all  $i$  in  $[1, 2, 3, 4]$  do // Calculamos o intervalo válido para  $t$  em cada quadrante
34:   if  $R_{u,i}^* \geq 0$  then
35:      $t_i^{min} \leftarrow \max(t_i^{min}, -E_{u,i}^*/R_{u,i}^*)$ 
36:   else
37:      $t_i^{max} \leftarrow \min(t_i^{max}, -E_{u,i}^*/R_{u,i}^*)$ 
38:   end if
39:   if  $R_{z,i}^* \geq -2 * R_{u,i}^*$  then
40:      $t_i^{max} \leftarrow \min(t_i^{max}, (E_{w,i}^* - 2 * E_{u,i}^*)/(R_{z,i}^* + 2 * R_{u,i}^*))$ 
41:   else
42:      $t_i^{min} \leftarrow \max(t_i^{min}, (E_{w,i}^* - 2 * E_{u,i}^*)/(R_{z,i}^* + 2 * R_{u,i}^*))$ 
43:   end if
44:   if  $R_{v,i}^* \geq 0$  then
45:      $t_i^{min} \leftarrow \max(t_i^{min}, -E_{v,i}^*/R_{v,i}^*)$ 
46:   else
47:      $t_i^{max} \leftarrow \min(t_i^{max}, -E_{v,i}^*/R_{v,i}^*)$ 
48:   end if
49:   if  $R_{z,i}^* \geq -2 * R_{v,i}^*$  then
50:      $t_i^{max} \leftarrow \min(t_i^{max}, (E_{w,i}^* - 2 * E_{v,i}^*)/(R_{z,i}^* + 2 * R_{v,i}^*))$ 
51:   else
52:      $t_i^{min} \leftarrow \max(t_i^{min}, (E_{w,i}^* - 2 * E_{v,i}^*)/(R_{z,i}^* + 2 * R_{v,i}^*))$ 
53:   end if
54:   if  $R_{z,i}^* \geq 0$  then
55:      $t_i^{min} \leftarrow \max(t_i^{min}, -E_{w,i}^*/R_{z,i}^*)$ 
56:   else
57:      $t_i^{max} \leftarrow \min(t_i^{max}, -E_{w,i}^*/R_{z,i}^*)$ 
58:   end if
59: end for
60:  $t \leftarrow -1$ 
61: for all  $t_{i,j}$  in  $[t_{1,1}, \dots, t_{4,4}]$  do // Filtramos as intersecções obtidas com os intervalos
    calculados
62:   if  $t_{i,j} \geq t_i^{min}$  and  $t_{i,j} \leq t_i^{max}$  and  $(t_{i,j} < t$  or  $t = -1)$  then
63:      $t \leftarrow t_{i,j}$ 
64:   end if
65: end for
66: if  $t \neq -1$  then
67:   return  $t$ 
68: else
69:   return false
70: end if

```

4 PROTÓTIPO

Foi desenvolvido um protótipo que implementa o algoritmo do capítulo anterior. A linguagem de programação *C* foi utilizada para implementar o Algoritmo 3.1, enquanto o Algoritmo 3.2 foi implementado em linguagem *GLSL*, para utilizar aceleração pela placa de vídeo.

Para obter as raízes da equação quártica, foi utilizada uma versão desenvolvida em *GLSL* do algoritmo de Descartes-Euler-Cardano (STRONG, 1859).

No estado atual da superfície proposta não é possível modelar alguns objetos clássicos da literatura, como o bule de Utah ("Utah teapot") (TORRENCE, 2006). Por este motivo, novos objetos foram modelados utilizando a superfície proposta, com níveis diferentes de detalhes.

Uma imagem com a versão atual do programa renderizando, por dois ângulos diferentes, uma simples bola, composta por 6 superfícies conectadas, é apresentada na Figura 18.

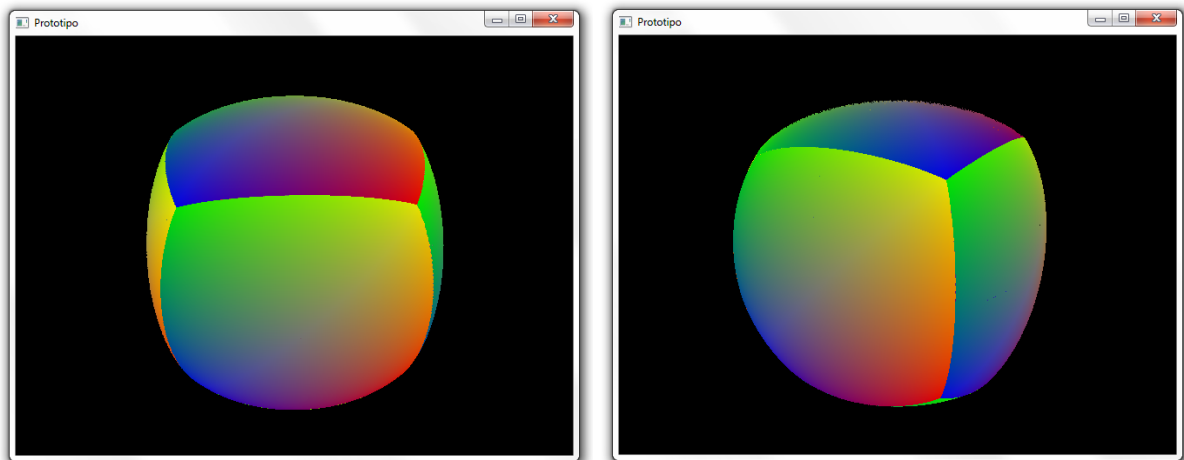


Figura 18 – Objeto de exemplo: Bola (6 superfícies).

Um jarro, composto por 14 superfícies conectadas, pode ser visto de dois ângulos diferentes na Figura 19.

Por fim, temos um pião, composto por 30 superfícies conectadas, renderizado por dois ângulos diferentes na Figura 20.

As cores foram utilizadas para indicar os pontos da superfície, sendo que azul representa o ponto *A*, vermelho representa o ponto *B*, amarelo representa o ponto *C* e verde representa o ponto *D*.

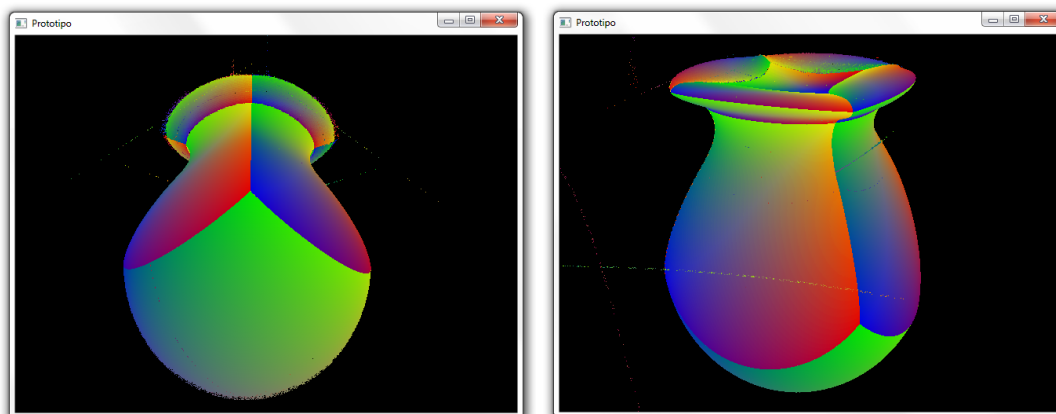


Figura 19 – Objeto de exemplo: Jarro (14 superfícies).

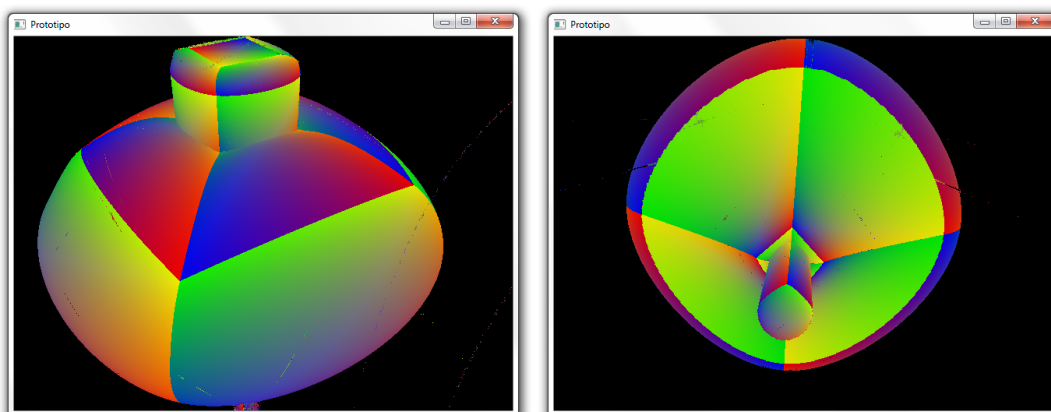


Figura 20 – Objeto de exemplo: Pião (30 superfícies).

Os três objetos, renderizados em escala de cinza, podem ser vistos na Figura 21.

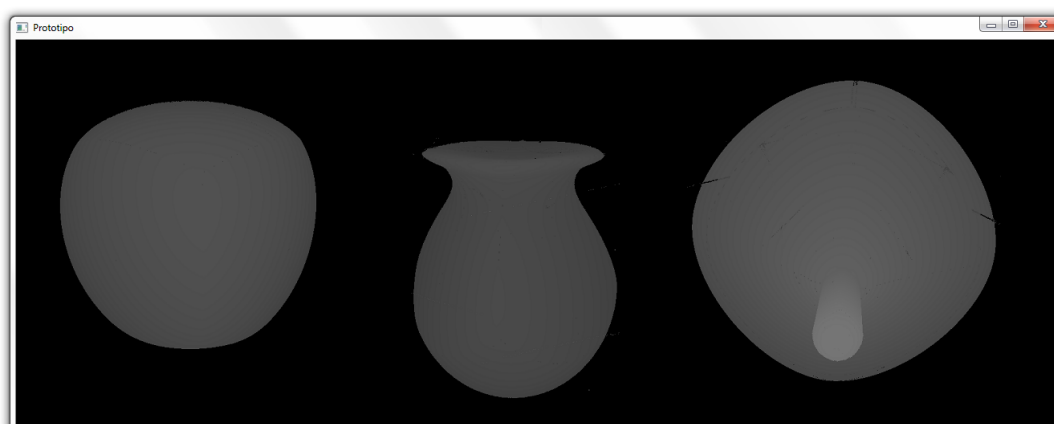


Figura 21 – Objetos em escala de cinza.

Algumas falhas visíveis na renderização se devem a erros de operações de ponto

flutuante no algoritmo de resolução da equação quártica, e só ocorrem em casos extremos. Evitar tais erros exigiria uma análise mais aprofundada da estabilidade de tal algoritmo.

O desempenho obtido foi satisfatório, atingindo taxas de quadros por segundo altas na renderização de poucas superfícies. A renderização de uma única superfície estática, em processador Intel Core i5 3330 e placa de vídeo nVidia Geforce GTX 760, produziu em torno de 1100 quadros de resolução 640x480 por segundo.

A versão atual do protótipo apresenta desempenho inversamente proporcional ao número de superfícies renderizadas, porém é possível aumentar o desempenho para múltiplas superfícies com a implementação de uma estrutura de aceleração da busca espacial, como a KD-tree (FOLEY; SUGERMAN, 2005) ou a BIH (WÄCHTER; KELLER, 2006), assim como a otimização do código atual e a experimentação com diferentes algoritmos de resolução de equações quárticas, analíticos e numéricos.

5 CONCLUSÕES

Apesar de o modelo de superfície proposto não ser tão flexível quanto alguns modelos clássicos (como a superfície de Bézier), o mesmo possui vantagens, como a possibilidade de renderização analítica, investigada no texto. Além disso, são necessárias menos operações para calcular os pontos que compõem a superfície proposta do que os que compõem as superfícies bicúbicas tradicionais.

Utilizando a técnica de transformação linear em coordenadas homogêneas para renderização, a superfície ainda fica sujeita às seguintes restrições:

1. Como consequência de levarmos um ponto finito do espaço original ao infinito do espaço transformado, o contrário acontece acima da superfície: todos os pontos daquele lado (ilimitado) da superfície no espaço original foram transformados para o intervalo limitado $0 < z < 1$ no espaço transformado. Isso significa que, quanto mais próxima a superfície chegar de $z = 1$ no espaço transformado, mais distante do ponto de fuga estará no espaço original, podendo chegar ao infinito se atingir $z = 1$. Na prática isso limita o cone de direções normais aceitáveis (N^A , N^B , N^C e N^D) para que a superfície esteja contida no espaço original. Esse limite não foi investigado ou determinado no presente texto, podendo ser identificado em um estudo futuro.
2. A superfície nunca terá plano tangente que contenha o ponto de fuga (ou esteja muito próximo do mesmo). Conseqüentemente, qualquer direção partindo do ponto de fuga intersectará a superfície em no máximo um ponto, o que impossibilita a modelagem de alguns objetos.
3. Os pontos conectados pela superfície devem formar um paralelogramo, ou serem pontos de um paralelogramo deslocados pelos multiplicadores explorados anteriormente (m_A , m_B , m_C e m_D), e superfícies conectadas devem usar os mesmos pontos e multiplicadores na conexão, o que pode dificultar a modelagem de objetos.
4. No presente trabalho não foi estudada uma forma de estimar uma caixa delimitadora ("bounding box") da superfície, que seria necessária para a utilização de algumas estruturas de aceleração de Ray-tracing.

Tais restrições podem ser contornados em trabalhos futuros. Algumas linhas de pesquisa com essa finalidade são propostas a seguir.

Para tentar evitar a primeira restrição, deve ser possível modificar a função de interpolação com o objetivo de controlar o valor máximo da superfície no espaço

transformado ao custo de uma divisão adicional do domínio de interpolação. Porém, assim teremos nove regiões diferentes para testar as intersecções (em lugar dos quatro quadrantes atuais), o que pode aumentar a quantidade de operações necessárias para encontrar todas as intersecções, prejudicando o desempenho do algoritmo. Seria interessante estudar essa possibilidade, limitando o algoritmo a testar as intersecções somente nas regiões que são percorridas pelo raio (utilizando testes preliminares simples). No pior dos casos, será necessário testar cinco regiões, embora normalmente isso não deva ser necessário, tornando o custo similar ao atual.

Também seria interessante estudar novas transformações espaciais que mantenham a função de distância com o quarto grau e que possam contornar as restrições 2 e 3.

Por fim, deve ser possível fazer uma estimativa simples de caixa delimitadora estimando as alturas máxima e mínima da superfície no espaço transformado. A superfície estará contida no bloco com base no quadrado unitário do plano (u, v) e entre essas duas alturas estimadas. Ao transformar esse bloco para o espaço original deve ser possível estimar uma caixa delimitadora a partir de seus vértices.

Referências

- BÉZIER, P. *The Mathematical Basis of the UNISURF CAD System*. Butterworth-Heinemann, 1986. ISBN 9780408221757. Disponível em: <<https://books.google.com.br/books?id=ao1RAAAAMAAJ>>.
- COONS, S. A. Surfaces for computer-aided design of space forms. DTIC Document, 1967.
- FARIN, G. E.; HOSCHEK, J.; KIM, M. S. *Handbook of Computer Aided Geometric Design*. Elsevier, 2002. ISBN 9780444511041. Disponível em: <<http://books.google.com.br/books?id=0SV5G8fgxLoC>>.
- FOLEY, T.; SUGERMAN, J. Kd-tree acceleration structures for a gpu raytracer. *In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, p. 15–22, 2005. ACM.
- GEIMER, M.; ABERT, O. Interactive ray tracing of trimmed bicubic bézier surfaces without triangulation. *In: Proceedings of WSCG*, p. 71–78, 2005.
- HART, J. C. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, Springer, v. 12, n. 10, p. 527–545, 1996.
- RIESENFELD, R. F. Homogeneous coordinates and projective planes in computer graphics. *IEEE Computer Graphics and Applications*, IEEE Computer Society Press, v. 1, n. 1, p. 50–55, 1981.
- SHENG, X.; HIRSCH, B. E. Triangulation of trimmed surfaces in parametric space. *Computer-Aided Design*, Elsevier, v. 24, n. 8, p. 437–444, 1992.
- SPÄTH, H. *One Dimensional Spline Interpolation Algorithms*. Taylor & Francis, 1995. (Ak Peters Series). ISBN 9781568810164. Disponível em: <<http://books.google.com.br/books?id=YtVQAAAAMAAJ>>.
- STRONG, T. *A Treatise on Elementary and Higher Algebra*. Pratt, Oakley & Company, 1859. Disponível em: <<https://books.google.com.br/books?id=45oKAAAAYAAJ>>.
- TORRENCE, A. Martin newell’s original teapot. *In: ACM SIGGRAPH 2006 Teapot Copyright Restrictions Prevent ACM from Providing the Full Text for the Teapot Exhibits*. New York, NY, USA: ACM, 2006. (SIGGRAPH '06). ISBN 1-59593-364-6. Disponível em: <<http://doi.acm.org/10.1145/1180098.1180128>>.
- WÄCHTER, C.; KELLER, A. Instant ray tracing: The bounding interval hierarchy. *Rendering Techniques*, v. 2006, p. 139–149, 2006.