UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA APLICADA

Inclusão entre nuvem de pontos e digitalização 3D: estratégias e implementação

por

Vinícius Fernandes Moretti

Dissertação submetida como requisito parcial
para a obtenção do grau de

Mestre em Matemática Aplicada

Prof. Dr. João Batista Da Paz Carvalho
Orientador

Porto Alegre, 4 de setembro de 2015.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Moretti, Vinícius Fernandes

Inclusão entre nuvem de pontos e digitalização 3D: estratégias e implementação / Vinícius Fernandes Moretti.— Porto Alegre: PPGMAp da UFRGS, 2015.

88 p.: il.

Dissertação (mestrado) —Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Matemática Aplicada, Porto Alegre, 2015.
Orientador: Carvalho, João Batista Da Paz

Dissertação: Matemática Aplicada,
Dissertação, Tese, Mestrado, Doutorado

Inclusão entre nuvem de pontos e digitalização 3D: estratégias e implementação

por

Vinícius Fernandes Moretti

Dissertação submetida ao Programa de Pós-Graduação em Matemática Aplicada do Instituto de Matemática da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de

Mestre em Matemática Aplicada

Linha de Pesquisa: Análise Numérica e Computação Científica

Orientador: Prof. Dr. João Batista Da Paz Carvalho

Banca examinadora:

Prof. Dr. Carlos Amaral Hölbig
PPGCA-UPF

Prof. Dr. Marcus Rolf Peter Ritt
PPGC-UFRGS

Prof. Dr. Rudnei Dias da Cunha
DMPA-UFRGS

Dissertação apresentada e aprovada em
4 de setembro de 2015.

Prof. Dr. Carlos Hoppen
Coordenador

It's the oldest story in the world.

One day you're 17 and planning for someday

and then quietly and without you even noticing,

someday is today.

And then, someday is yesterday.

And this is your life.

(One Tree Hill - S09E13)

AGRADECIMENTOS

Meus sinceros agradecimentos a todos que contribuíram de maneira direta ou indireta nesse projeto.

A toda minha família, pelo apoio nesse longo período entre graduação e mestrado, sem questionar as minhas decisões (somente me chamando de louco de vez em quando).

A minha mãe, ao meu pai (*in memoriam*) e ao meu irmão, por sempre acreditarem em mim.

A minha prima Carolina, pelos tantos desabafos e conversas.

Aos mais que amigos Éverton e Felipe, pelas conversas, conselhos e incentivos.

Ao professor João Batista, pelos ensinamentos e pela paciência.

À Yasmin, essa pessoa especial que me dá muitos puxões de orelha e carinho.

Ao Doctor, por ser meu companheiro nos momentos de solidão.

À FAPERGS, pela ajuda financeira.

À banca examinadora, pelas suas contribuições.

Sumário

LISTA DE FIGURAS . . . . . **viii**

LISTA DE TABELAS . . . . . **x**

LISTA DE SíMBOLOS . . . . . **xi**

RESUMO . . . . . **xiii**

ABSTRACT . . . . . **xiv**

1 INTRODUÇÃO **1**

2 DESCRIÇÃO DOS MÉTODOS **5**

2.1 Definições e teoremas fundamentais 5

2.2 Teste de inclusão em polígonos de Feito-Torres 26

2.3 Teste de inclusão em poliedros de Feito-Torres 28

2.4 Algoritmo de Badouel 35

2.5 Algoritmo de Möller 40

2.6 Algoritmo de Segura 44

2.7 Novo Algoritmo de Segura 48

3 IMPLEMENTAÇÃO COMPUTACIONAL **58**

3.1 Discretizações e testes aleatórios 58

3.2 Estratégias algorítmicas 65

4 EXPERIMENTOS COMPUTACIONAIS **69**

4.1 Testes de eficiência e eficácia 69

4.2 Discussão dos resultados 78

5 CONCLUSÕES **85**

REFERÊNCIAS BIBLIOGRÁFICAS **86**

Lista de Figuras

2.1 Exemplos de polígonos que não estão bem definidos. 6

2.2 Segmento orientado. 6

2.3 Índice de um ponto interior: soma dos ângulos é 2π. 10

2.4 Teorema da Curva de Jordan. 11

2.5 Inclusão de um ponto em um triângulo pela origem. 12

2.6 Lema 2.3. 16

2.7 Conjuntos convexos e não-convexos. 16

2.8 Envelope convexo. 19

2.9 Exemplos de aplicação do algoritmo de Feito-Torres versão 2D. 27

2.10 Demonstração do teorema 2.3. 31

2.11 Exemplos de aplicação do algoritmo de Feito-Torres versão 3D. 34

2.12 Representação paramétrica do ponto P. 36

2.13 Translação e mudança de base do raio no algoritmo de Möller. . 41

2.14 Interpretação do algoritmo de Segura. 45

2.15 Volume com sinal do tetraedro $Q_1V_0V_1V_2$. 49

2.16 Interpretação geométrica do sinal de α. 51

3.1 Orientação de uma face, regra da mão direita. 59

3.2 Configurações de triângulos adjacentes em arquivo STL. 59

4.1 Sólidos testados. 70

4.2 Sólidos testados. 71

4.3 Sólidos testados. 72

4.4 Melhora no desempenho, pedras brutas (sólidos 1 a 18). 82

4.5 Melhora no desempenho, modelos regulares (sólidos I a VII). . . 82

4.6 Vértices vs. tempo para o teste 1 (modelos regulares). 83

4.7 Vértices vs. tempo para o teste 1 (pedras brutas). 84

Lista de Tabelas

2.1 Valores do sinal dos triângulos pela origem e de $INCLUSAO$. . 28

2.2 Valores de $INCLUSAO$. 34

2.3 Ponto em vértice ou aresta no algoritmo de Badouel. 38

2.4 Ponto em vértice ou aresta no algoritmo de Möller. 43

2.5 Ponto em vértice ou aresta no algoritmo de Segura. 46

4.1 Teste 1: tempo relativo, 5000 pontos gerados dentro. 73

4.2 Teste 2: tempo relativo, 5000 pontos gerados fora. 74

4.3 Teste 3: tempo relativo, 5000 pontos gerados dentro/fora. . . . 75

4.4 Teste 1 (modelo regular): tempo relativo, 5000 pontos dentro. . 76

4.5 Teste 2 (modelo regular): tempo relativo, 5000 pontos fora. . . 76

4.6 Teste 3 (modelo regular): tempo relativo, 5000 pontos dentro/fora. 77

4.7 Tempo absoluto (em segundos), algoritmo de Badouel (regulares). 77

4.8 Tempo absoluto (em segundos), algoritmo de Badouel (pedras brutas). 78

LISTA DE SÍMBOLOS

Lista de Símbolos

$P \cdot Q$	Produto escalar dos vetores P e Q.
$P \times Q$	Produto vetorial dos vetores P e Q.
$[ABC]$	Área (com sinal) do triângulo ABC.
$[ABCD]$	Volume (com sinal) do tetraedro $ABCD$.
$\|M\|$	Valor absoluto do número real M ou determinante da matriz M.
$\text{int}\,(P)$	Interior de P na topologia correspondente a dimensão de P.
∂P	Fronteira de P na topologia correspondente a dimensão de P.
$\#S$	Cardinalidade do conjunto S.

RESUMO

Neste trabalho são investigadas soluções eficientes para o problema de determinar se uma nuvem de pontos está contida (ou, alternativamente, invade) a digitalização tridimensional da superfície de um sólido não necessariamente convexo. Estratégias baseadas no Teorema da Curva de Jordan, generalizadas para o caso tridimensional, bem como estratégias baseadas no estudo de volumes com sinal de tetraedros, foram testadas e comparadas segundo sua eficácia e eficiência computacional. Os experimentos computacionais foram feitos com digitalizações de pedras brutas disponibilizadas pelo Centro Tecnológico de Pedras de Soledade, RS. Este trabalho estabelece importante contribuição para a solução de relevante e mais complexo problema em Geometria Computacional: determinar se há inclusão (ou, alternativamente, invasão) espacial entre dois sólidos com superfícies digitalizadas, e que conseqüentemente tem variadas aplicações.

ABSTRACT

This work investigates efficient solutions to the problem of determining whether or not a cloud of points is contained (or alternatively, invades) the spatial digitization of the surface of a not-necessarily convex solid. Strategies based on the well-known Jordan Curve Theorem, once generalized to the 3D case, as well as those based on the analysis of signed volumes of tetrahedra, were tested and compared according to their robustness and efficiency. The numerical experiments used digitization of raw stones made available by the the Technological Center of Stones, Gems and Jewelry of the city of Soledade, in this state. The present work makes an important contribution to the solution to a relevant further complex problem in Computational Geometry: to determine whether or not there is spatial inclusion (or, alternatively, invasion) between two solids with digitized surfaces, which have several further applications.

1 INTRODUÇÃO

Estratégias para otimização do aproveitamento de gemas coradas digitalizadas tridimensionalmente [4] visam estabelecer, matematicamente, os melhores valores para os planos de lapidação dessas gemas, segundo variados objetivos, mas dentre os quais o mais comum é obter o maior volume possível para cada pedra bruta e modelo de lapidação escolhidos. O modelo de lapidação, que algumas vezes tem uma representação matemática conhecida, mesmo que apenas aproximada, trata-se de um volume sólido que pode ser posicionado no espaço através de seis parâmetros: três coordenadas de centro geométrico e três ângulos de giro. Um sétimo e importante parâmetro do modelo de lapidação é sua escala em relação a um protótipo de tamanho padronizado.

Dessa forma, uma metodologia matematicamente simples para resolver o problema de otimizar o aproveitamento do corte (lapidação) de pedras consiste em definir um problema de otimização naqueles seis parâmetros, com o objetivo de maximizar a escala obedecendo a restrição de que o modelo reposicionado e reescalado deve permanecer sempre interior a pedra bruta a ser trabalhada.

Por outro lado, existem peculiaridades que trazem alguma complexidade a solução desse problema. Primeiramente, deve ser considerado que a pedra a ser trabalhada, aqui referida como pedra bruta, é conhecida somente através da digitalização tridimensional da sua superfície externa. Em segundo lugar, deve ser considerado que o próprio modelo de lapidação padronizado (protótipo) também pode ser conhecido somente através da digitalização tridimensional da sua superfície externa. Isso acontece, sobretudo, em situações onde modelos matemáticos não conseguem aproximar satisfatoriamente a geometria pretendida. Assim sendo, podemos considerar duas situações:

(i) A restrição de que o modelo em escala deve ser interior à pedra bruta implica que é necessário que uma nuvem de pontos do espaço, obtidos a partir da digitalização do modelo, mas conforme posicionamento e escala pretendidos, deve ser interior à digitalização da pedra bruta;

(ii) Uma vez que sobre o modelo de lapidação temos mais hipóteses de regularidade do que a pedra bruta, do ponto de vista geométrico, é natural estabelecer o teste dual que valida se algum dos vértices da digitalização da pedra bruta invade o sólido obtido pelo posicionamento e escala do modelo de lapidação digitalizado.

Devemos observar, ainda, que a presença da hipótese de convexidade normalmente guia as metodologias de solução de problemas geométricos, e sobretudo pode garantir a existência e mesmo a unicidade de soluções do respectivo problema matemático.

O presente trabalho tem por objetivo investigar soluções eficientes para o problema de determinar se uma nuvem de pontos está inclusa (ou, alternativamente, invade) a digitalização tridimensional da superfície de um sólido não necessariamente convexo. Conforme a descrição do parágrafo anterior, esse problema básico é parte essencial para a validação sobre inclusão ou invasão espacial entre dois sólidos com superfícies digitalizadas, e aqui não é feita a hipótese de convexidade para permitir atender a qualquer das duas situações mencionadas.

O problema de determinar se um ponto P é interior a um sólido espacial é bastante conhecido na literatura de geometria computacional. Por causa das tarefas cada vez mais complexas em muitas aplicações (planejamento urbano, cadastro 3D de objetos, controle de trajetórias de veículos teleguiados, realidade virtual, entre outras) aplicativos computacionais vêm sendo cada vez mais usados para modelar, analisar e visualizar dados tridimensionais de uma maneira eficiente. Uma das tarefas mais desafiadoras é a analise em 3D do posicionamento de objetos

de diferentes tipos, isto é, geometrias [6]; nesse contexto, o teste de inclusão, que objetiva determinar se um ponto é interior a um poliedro dado, é uma das operações mais básicas que áreas como Geometria Computacional e Computação Gráfica necessitam. O teste de inclusão não é um problema complexo, mas o maior desafio é obter soluções que sejam robustas e eficientes, uma vez que tais testes são aplicados repetidamente um número muito grande de vezes [6, 17].

A estratégia mais conhecida é o Método do Raio (*Ray Crossing method*), que emprega o conhecido Teorema da Curva de Jordan ao contabilizar a paridade do número de intersecções de uma semirreta (raio) qualquer, originada no ponto P, com a fronteira do sólido em questão. No caso particular de um sólido cuja fronteira é uma estrutura (malha) de faces poligonais adjacentes, oriunda do processo de digitalização, encontros singulares do raio com arestas ou vértices da discretização geram intersecções simultâneas com um número muitas vezes incerto de faces adjacentes, o que normalmente dificulta tal contagem. Além disso, situações onde o raio é paralelo ou coplanar ao plano de alguma(s) face(s), ou ainda quando existem faces quasedegeneradas, podem acarretar sérias dificuldades numéricas [17]. Para aumentar a eficiência, o método do Raio pode ser aplicado repetidamente, ou com direção aleatória [21]. Normalmente, melhor performance é obtida com uma combinação do Método do Raio com estruturas de dados hierárquicas do tipo *octree* ou do tipo *grid* [20], mas a possivelmente grandes custos de armazenamento em memória [17]. Um outro método muito conhecido é o baseado em estruturas espaciais chamadas de BSP (*Binary Space Partitioning*), que têm como desvantagem um custo de processamento muito grande quando a geometria/topologia dos modelos é irregular. Para a versão bidimensional do problema de inclusão, quando é necessário determinar se um ponto é interior a uma curva plana, inúmeras soluções têm sido propostas para o caso de polígonos com número qualquer de arestas, convexos ou não [2, 7, 11, 19].

Alternativamente ao método do Raio, tem sido proposto em vários artigos [7, 8, 15, 20], contabilizar sinais volume de tetraedros formados por faces do

poliedro e pelo ponto a ser testado, o que tem se mostrado simples e seguro, mas que é ineficiente para discretizações com número muito grande de faces [17]. Em [17], é proposto um novo método de solução que usa pré-processamento e determinação de triângulos espaciais no interior de um poliedro, e que generaliza o método de Horn [14]. Existem também as estratégias que resolvem o problema de inclusão bidimensional calculando o número de voltas (*winding number* ou índice do ponto) de cada triângulo em relação a interseccção entre o raio e o plano desse triângulo, conforme [11, 21]; essas estratégias têm sido preteridas por requererem avaliações de funções trigonométricas, que sabidamente trazem ineficiência por implicarem em um maior número de operações básicas de aritmética de máquina e por trazerem embutidos erros de ponto flutuante, que podem se acumular catastroficamente.

No capítulo 2 descrevemos os métodos a serem utilizados nesse trabalho, bem como as principais definições matemáticas que os suportam. No capítulo 3 descrevemos nossas estratégias computacionais e de comparação envolvendo um pequeno conjunto de métodos de solução e apresentamos noções sobre tipos de arquivos de discretizações de malhas. O capítulo 4 expõe os resultados dos testes realizados com um conjunto seleto de sólidos e também discute conclusões diante dos resultados obtidos.

2 DESCRIÇÃO DOS MÉTODOS

Aqui são dadas algumas definições fundamentais sobre geometria (grande parte delas pode ser encontrada em [3, 7, 8, 9, 13, 21]) bem como são descritos os métodos e algoritmos utilizados ao longo do trabalho.

2.1 Definições e teoremas fundamentais

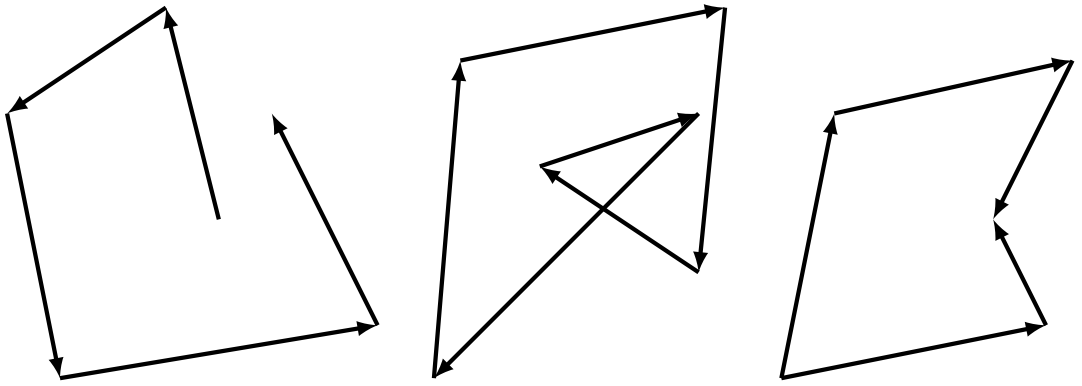
Diversas aplicações gráficas trabalham com figuras planares e muitos dos algoritmos que realizam operações com polígonos necessitam de certas propriedades que garantem que uma seqüência de segmentos forma de fato um polígono. Tais propriedades são [13]:

1. **Fechamento:** Cada segmento deve ser delimitado por exatamente dois vértices, e cada vértice deve estar na intersecção de exatamente dois segmentos.

2. **Auto-intersecções:** Quaisquer dois segmentos podem se interceptar somente se eles são adjacentes. Neste caso, o ponto de intersecção é o vértice comum aos segmentos.

3. **Orientação:** Cada segmento deve possuir uma direção e as direções de todos os segmentos devem ser coerentes entre si.

Na literatura, uma variedade de algoritmos foi proposta para testar essas condições, sendo que a maioria deles emprega a resolução de sistemas de equações lineares algébricas ou então utiliza funções trigonométricas em seus testes. Isso implica em problemas de estabilidade como, por exemplo, o crescimento de erros de ponto flutuante, o que tem como conseqüência uma perda de eficiência [7, 22].



(a) Não é fechado. (b) Auto-intersecções. (c) Orientação incoerente.

Figura 2.1: Exemplos de polígonos que não estão bem definidos.

Definição 2.1 (Segmento orientado)**.** *Um segmento orientado é um segmento de reta ao qual foi imposta uma escolha para os seus pontos inicial e final, chamados de extremidades. Tal segmento orientado pode ser designado pela notação AB, onde A é o seu ponto inicial e B o seu ponto final.*

Definição 2.2 (Medida de um segmento orientado)**.** *A medida ou comprimento de um segmento orientado AB, denotada com abuso de notação também por AB, é dada por:*

$$AB = \begin{cases} +|AB|, & \text{se } AB \text{ tem a mesma orientação da reta } r \\ -|AB|, & \text{caso contrário,} \end{cases} \tag{2.1}$$

onde A e B são dois pontos de uma reta orientada r e $|AB|$ representa a medida no sentido euclidiano do segmento de reta com extremidades A e B.

Sejam a e b as coordenadas de dois pontos A e B marcados sobre uma reta orientada r, tal como na figura 2.2.

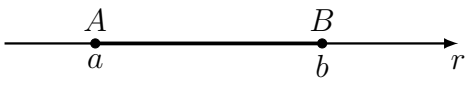


Figura 2.2: Segmento orientado.

A medida no sentido euclidiano do segmento AB é dada pelo módulo, ou seja, $AB = |b-a|$. A medida do segmento orientado é idêntica à medida euclidiana.

Se r e AB têm a mesma orientação, temos

$$AB = +|AB| = |b-a| = b-a. \tag{2.2}$$

Se r e AB têm a orientações opostas, temos

$$AB = -|AB| = -|b-a| = -(a-b) = b-a. \tag{2.3}$$

Assim, a medida de um segmento orientado é função das coordenadas dos pontos e independe da sua orientação.

Proposição 2.1. *Sejam A, B e C três pontos colineares. Então:*

(1) $AB + BA = 0$. *Em particular,* $AB = -BA$.

(2) $AB = 0$ *se, e somente se,* $A = B$.

(3) $AC + CB = AB$.

Demonstração. Sejam a, b e c as coordenadas dos pontos A, B e C, respectivamente.

(1) $AB + BA = (b-a) + (a-b) = 0$. Assim, $AB = -BA$.

(2) $AB = 0 \Leftrightarrow b - a = 0 \Leftrightarrow b = a \Leftrightarrow A = B$.

(3) $AC + CB = (c-a) + (b-c) = b - a = AB$.

■

Notemos que o item (3) da proposição 2.1 é válido independentemente da posição do ponto C (na reta) em relação aos pontos A e B, diferentemente de quando se considera a medida no sentido euclidiano, onde seria necessário tomar os casos onde C está à esquerda de A, à direita de B ou então entre A e B.

Definição 2.3 (Poligonal). *Uma poligonal é uma seqüência de pontos A, B, C, D, ..., L, M, ou ainda, uma seqüência de segmentos orientados AB, BC, CD, ..., LM, MA. A poligonal pode ser identificada pela sua seqüência de pontos $ABCD \dots LM$. Os pontos A e M são chamados de extremidades da poligonal.*

Definição 2.4 (Polígono). *Um polígono é uma poligonal onde as duas extremidades coincidem. O polígono pode ser denotado pela sua seqüência de pontos $ABCD \dots LM$. Os segmentos AB, BC, CD, ..., LM, MA são chamados de arestas ou lados do polígono e os pontos A, B, C, D, ..., M são chamados de vértices do polígono.*

Definição 2.5 (Polígono simples). *Um polígono $P_0P_1 \dots P_{n-1}$ é dito simples se cumpre as seguintes propriedades:*

(1) Todos os seus vértices são diferentes: $\forall i \neq j, P_i \neq P_j$.

(2) Todo vértice é incidente a uma aresta.

(3) Não existem intersecções entre duas arestas.

Definição 2.6 (Divisão do plano). *Um segmento AB divide o plano em duas regiões, uma a sua esquerda e outra a sua direita.*

Definição 2.7 (Orientação de um triângulo). *Um triângulo ABC tem orientação positiva se os seus pontos interiores estão todos à esquerda de suas arestas, caso contrário o triângulo possui orientação negativa.*

Definição 2.8 (Área de um triângulo). *A área de qualquer triângulo ABC, denotada por $[ABC]$, é metade do produto da distância entre qualquer aresta e o seu vértice oposto.*

Definição 2.9 (Função sinal). *Seja x um número real, a função sinal é definida como*

$$\text{sign } (x) = \begin{cases} 1, & \textit{se } x > 0 \\ 0, & \textit{se } x = 0 \\ -1, & \textit{se } x < 0 \end{cases} \tag{2.4}$$

Definição 2.10 (Área com sinal de um triângulo)**.** *Sejam A, B e C três pontos do $\mathbb{R}^2$. A área com sinal do triângulo que tem A, B e C como vértices, denotada por $[ABC]$, é dada por*

$$[ABC] = \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} \tag{2.5}$$

onde $A = (x_A, y_A)$, $B = (x_B, y_B)$ e $C = (x_C, y_C)$. A área com sinal de um triângulo também pode ser tomada como sua própria área $[ABC]$ se ele tiver orientação positiva ou então como $-[ABC]$ se ele for negativamente orientado.

Um triângulo tem orientação positiva se o sinal da sua área é positivo e isso significa que seus vértices estão listados no sentido anti-horário. Quando os seus vértices estiverem listados no sentido horário, o sinal da sua área será negativo, assim como sua orientação.

Para qualquer triângulo ABC e qualquer ponto Q vale

$$[ABC] = [QAB] + [QBC] + [QCA]. \tag{2.6}$$

Ainda, é claro que $[ACB] = -[ABC]$.

Alternativamente, a orientação de um polígono pode ser matematicamente determinada pelo índice de algum ponto interior. O índice de um ponto Q interior a um polígono P com respeito a esse polígono é definido usando o ângulo entre os segmentos de reta formados pelo ponto Q e cada par de vértices consecutivos de P, detonados por $P_i\widehat{Q}P_{i+1}$.

Definição 2.11 (Índice de um ponto)**.** *Para qualquer polígono $P = P_0P_1 \dots P_{n-1}$ e qualquer ponto Q, o índice de Q com respeito ao polígono P é dado por*

$$w(P, Q) = \sum_{i=0}^{n-1} P_i\widehat{Q}P_{(i+1 \bmod n)}. \tag{2.7}$$

Teorema 2.1. *Todo polígono P está orientado no sentido horário ou negativo se, para qualquer ponto interior Q, $w(P, Q) = -2\pi$; o polígono está orientado no sentido anti-horário ou positivo se $w(P, Q) = 2\pi$, para qualquer ponto Q interior. Além do mais, para todo ponto exterior Q vale que $w(P, Q) = 0$.*

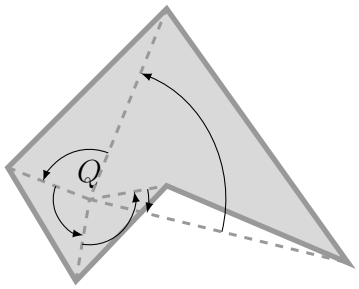


Figura 2.3: Índice de um ponto interior: soma dos ângulos é 2π.

O teste de inclusão de um ponto em um polígono calculando o seu índice, se implementado somente com o somatório dos ângulos, é o mais lento de todos [11]. Nesse teste, o índice é dado pelo múltiplo de 2π mais próximo. O principal problema com esse procedimento é que ele envolve funções trigonométricas e muitas divisões para cada ângulo testado, operações onerosas que carregam muitos erros de arredondamento [7].

Teorema 2.2 (Teorema da Curva de Jordan)**.** *Uma curva fechada simples em um plano o separa em duas regiões, uma limitada e outra ilimitada, sendo a curva a fronteira comum das duas regiões.*

Várias demonstrações já foram dadas para este teorema e elas fogem ao escopo desse trabalho. Para maiores detalhes, veja [5, 30]. A generalização para o caso multi-dimensional é conhecida como Teorema de Separação de Jordan-Brouwer, e a sua demonstração é também omitida. Entretanto, a generalização para três dimensões foi usada nos testes realizados nesse trabalho, conforme descrito na seção 3.2.

Esse teorema permite determinar se um ponto está dentro ou fora de um polígono. O processo consiste em definir um raio a partir do ponto a ser testado e contar o número de intersecções entre este raio e o polígono: se o número de cruzamentos for ímpar o ponto é interior; se for par, o ponto é exterior [10].

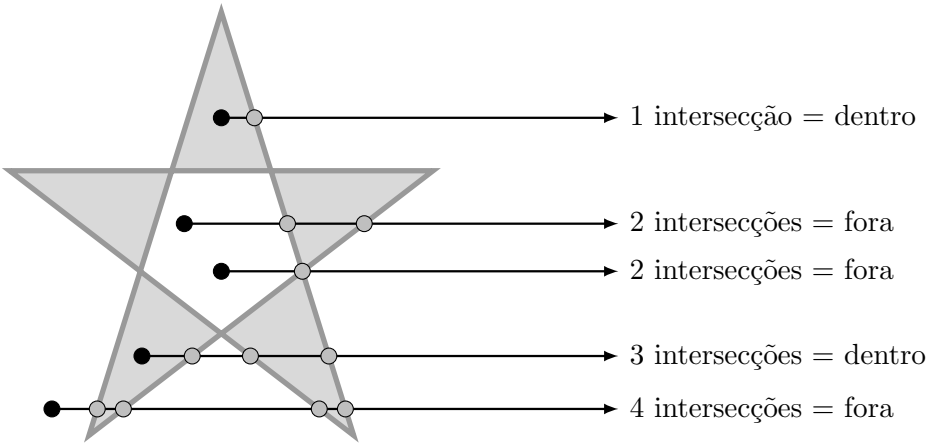


Figura 2.4: Teorema da Curva de Jordan.

Definição 2.12 (Área de um polígono)**.** *A área de qualquer polígono simples P, denotada por $[P]$, é a soma das áreas de qualquer conjunto de triângulos nos quais ele pode ser decomposto.*

Definição 2.13 (Área com sinal de um polígono)**.** *A área com sinal de um polígono P, denotada por $[P]$, é definida como sua própria área se ele for positivamente orientado ou então como o simétrico da sua área se ele tiver orientação negativa.*

Como demonstrado em [7], para testar se um ponto é interior a um polígono é suficiente verificar se ele é interior aos triângulos pela origem (a origem é um dos seus vértices) determinados por cada uma das arestas e a origem e somar os sinais da área de cada triângulo.

É possível testar se um ponto é interior a um triângulo pela origem utilizando o seguinte lema:

Lema 2.1. *Seja OAB um triângulo pela origem com orientação positiva e Q um ponto arbitrário do $\mathbb{R}^2$. Então Q é interior ao triângulo OAB se*

$$\text{sign } ([QOA]) \geq 0, \text{ sign } ([QAB]) \geq 0 \text{ } e \text{ sign } ([QBO]) \geq 0. \tag{2.8}$$

Demonstração. Q é interior ao triângulo se, e somente se, ele está à esquerda dos segmentos orientados AB, BO e OA, o que implica que as áreas com sinal $[QAB]$, $[QOA]$ e $[QBO]$ são positivas. Além disso, se alguma dessas áreas é zero então o ponto está na aresta correspondente do triângulo. ■

Se o triângulo OAB tem orientação negativa é suficiente estudar o triângulo OBA.

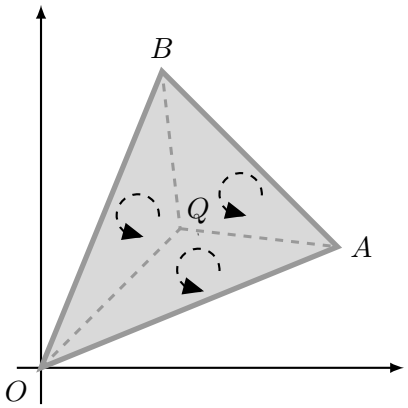


Figura 2.5: Inclusão de um ponto em um triângulo pela origem.

Para testar a inclusão de um ponto em um triângulo calculamos os determinantes através dos seus menores associados:

$$\begin{vmatrix} x_Q & y_Q & 1 \\ x_A & y_A & 1 \\ x_B & y_B & 1 \end{vmatrix} = \begin{vmatrix} x_A & y_A \\ x_B & y_B \end{vmatrix} - \begin{vmatrix} x_Q & y_Q \\ x_B & y_B \end{vmatrix} + \begin{vmatrix} x_Q & y_Q \\ x_A & y_A \end{vmatrix} \tag{2.9}$$

$$\begin{vmatrix} x_Q & y_Q & 1 \\ x_B & y_B & 1 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} x_Q & y_Q \\ x_B & y_B \end{vmatrix} \tag{2.10}$$

$$\begin{vmatrix} x_Q & y_Q & 1 \\ 0 & 0 & 1 \\ x_A & y_A & 1 \end{vmatrix} = - \begin{vmatrix} x_Q & y_Q \\ x_A & y_A \end{vmatrix} \tag{2.11}$$

A seguir as generalizações desses conceitos para três dimensões.

Definição 2.14 (Volume com sinal de um tetraedro)**.** *Sejam A, B, C e D quatro pontos do $\mathbb{R}^3$. O volume com sinal do tetraedro que tem os pontos A, B, C e D como vértices, denotado por $[ABCD]$, é definido como*

$$[ABCD] = \frac{1}{6} \begin{vmatrix} x_A & y_A & z_A & 1 \\ x_B & y_B & z_B & 1 \\ x_C & y_C & z_C & 1 \\ x_D & y_D & z_D & 1 \end{vmatrix} = \frac{1}{6} \begin{vmatrix} x_A - x_D & y_A - y_D & z_A - z_D \\ x_B - x_D & y_B - y_D & z_B - z_D \\ x_C - x_D & y_C - y_D & z_C - z_D \end{vmatrix} \tag{2.12}$$

onde $A = (x_A, y_A, z_A)$, $B = (x_B, y_B, z_B)$, $C = (x_C, y_C, z_C)$ e $D = (x_D, y_D, z_D)$.

Um tetraedro tem orientação positiva (do lado oposto a um dos vértices os demais vértices estão em sentido anti-horário) se o seu volume com sinal é positivo [21]. Se um dos vértices do tetraedro for a origem o chamamos de tetraedro pela origem.

Assim como em duas dimensões, a definição do volume com sinal permite determinar facilmente se um ponto é interior a um tetraedro ou não. Ademais, dependendo de possíveis valores nulos é possível também determinar se o ponto pertence a fronteira deste tetraedro. Esse resultado é expresso pelo lema seguinte.

Lema 2.2. *Seja $ABCO$ um tetraedro pela origem com orientação positiva e Q um ponto arbitrário do $\mathbb{R}^3$. Q é interior ao tetraedro se, e somente se,*

$$\text{sign}\,([ABCQ]) \geq 0,\ \text{sign}\,([ACOQ]) \geq 0,\ \text{sign}\,([AOBQ]) \geq 0 \text{ e } \text{sign}\,([BOCQ]) \geq 0 \tag{2.13}$$

Demonstração. Q é interior ao tetraedro se, e somente se, ele "enxerga" os três vértices de cada uma das faces com a mesma orientação que a origem "enxerga" os vértices A, B e C, o que implica que os volumes com sinal $[ABCQ]$, $[ACOQ]$, $[AOBQ]$ e $[BOCQ]$ são positivos. ■

Assim como no caso 2D, não é necessário calcular todos os determinantes para o teste da inclusão de um ponto, é possível obtê-los a partir dos seus menores associados.

$$\begin{vmatrix} x_A & y_A & z_A & 1 \\ x_B & y_B & z_B & 1 \\ x_C & y_C & z_C & 1 \\ x_D & y_D & z_D & 1 \end{vmatrix} = \begin{vmatrix} x_A & y_A & z_A \\ x_B & y_B & z_B \\ x_C & y_C & z_C \end{vmatrix} - \begin{vmatrix} x_A & y_A & z_A \\ x_B & y_B & z_B \\ x_Q & y_Q & z_Q \end{vmatrix} + \begin{vmatrix} x_A & y_A & z_A \\ x_C & y_C & z_C \\ x_Q & y_Q & z_Q \end{vmatrix} - \begin{vmatrix} x_B & y_B & z_B \\ x_C & y_C & z_C \\ x_Q & y_Q & z_Q \end{vmatrix} \tag{2.14}$$

$$\begin{vmatrix} x_A & y_A & z_A & 1 \\ x_C & y_C & z_C & 1 \\ 0 & 0 & 0 & 1 \\ x_Q & y_Q & z_Q & 1 \end{vmatrix} = - \begin{vmatrix} x_A & y_A & z_A \\ x_C & y_C & z_C \\ x_Q & y_Q & z_Q \end{vmatrix} \tag{2.15}$$

$$\begin{vmatrix} x_A & y_A & z_A & 1 \\ 0 & 0 & 0 & 1 \\ x_B & y_B & z_B & 1 \\ x_Q & y_Q & z_Q & 1 \end{vmatrix} = \begin{vmatrix} x_A & y_A & z_A \\ x_B & y_B & z_B \\ x_Q & y_Q & z_Q \end{vmatrix} \tag{2.16}$$

$$\begin{vmatrix} x_B & y_B & z_B & 1 \\ 0 & 0 & 0 & 1 \\ x_C & y_C & z_C & 1 \\ x_Q & y_Q & z_Q & 1 \end{vmatrix} = \begin{vmatrix} x_B & y_B & z_B \\ x_C & y_C & z_C \\ x_Q & y_Q & z_Q \end{vmatrix} \tag{2.17}$$

A extensão da definição para pirâmides segue. Considera-se que quando a face da pirâmide oposta ao vértice é dada em sentido anti-horário ela tem orientação positiva.

Definição 2.15 (Volume com sinal de uma pirâmide)**.** *O volume com sinal de uma pirâmide P, com vértice V e base S, denotado por $[P]$, é igual ao seu próprio volume se a pirâmide tem orientação positiva e é igual ao simétrico do seu volume se a pirâmide tem orientação negativa. Se o vértice da pirâmide é a origem dizemos que P é uma pirâmide pela origem.*

Definição 2.16 (Sinal de uma face)**.** *O sinal de uma face F de um poliedro qualquer P é o sinal do volume com sinal da pirâmide pela origem determinada pela origem e a face F.*

É possível determinar o sinal de uma face de um poliedro através do seguinte lema.

Lema 2.3. *Seja $P = OF$ uma pirâmide pela origem e $Ax + By + Cz + D = 0$ a equação do plano que contém a base (face), onde $N = (A, B, C)$ é o vetor normal a esse plano com sentido apontando para fora do poliedro. Então, o sinal da pirâmide (da face),* sign $([P])$*, é $+1$ se D é negativo, -1 se D é positivo e 0 se D é zero.*

Demonstração. Seja $Q = (x, y, z)$ um ponto qualquer no plano que contém a base da pirâmide. O vetor OQ tem coordenadas (x, y, z). O sinal do cosseno do ângulo formado por esse vetor com o normal do plano (A, B, C) pode ser calculado através do sinal do produto escalar

$$(A, B, C) \cdot (x, y, z) = Ax + By + Cz \tag{2.18}$$

e essa expressão é igual a $-D$ (porque Q está no plano). Se o ângulo é obtuso o sinal do seu cosseno é negativo e então $D > 0$. Se o ângulo é agudo, então o sinal do seu cosseno é positivo e $D < 0$. Se o plano contém a origem o valor de D é zero, sendo esta pirâmide degenerada. ■

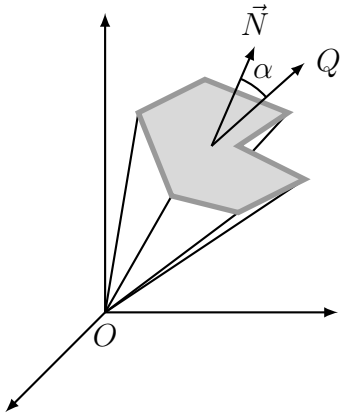
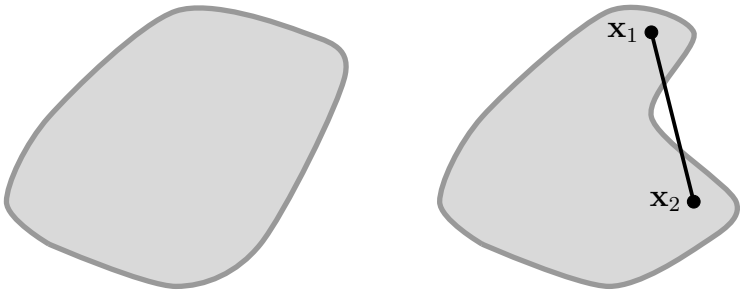


Figura 2.6: Lema 2.3.

Definição 2.17 (Conjunto convexo)**.** *Um conjunto $\mathcal{S} \subseteq \mathbb{R}^n$ é dito convexo se o segmento linear que une quaisquer dois de seus elementos está integralmente contido em $\mathcal{S}$. Em outras palavras, dados dois elementos distintos $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}$, o segmento que une $\mathbf{x}_1$ e $\mathbf{x}_2$, descrito (parametrizado) por*

$$\{\mathbf{x} \in \mathbb{R}^n \mid \lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2, \lambda \in [0,1]\}, \tag{2.19}$$

também pertence ao conjunto $\mathcal{S}$.



(a) Conjunto convexo. (b) Conjunto não-convexo.

Figura 2.7: Conjuntos convexos e não-convexos.

Definição 2.18 (Combinação convexa, afim e linear)**.** *Sejam dados os k vetores $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^n$, o conjunto definido e descrito por*

$$\left\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j, \sum_{j=1}^{k} \lambda_j = 1, \lambda_j \geq 0, j = 1, \ldots, k\right\} \tag{2.20}$$

é chamado de conjunto das combinações convexas de $\mathbf{x}_1, \ldots, \mathbf{x}_k$. *Se as condições para os multiplicadores* $\lambda_j, j = 1, \ldots, k$, *permitirem também números negativos chamamos os elementos do conjunto de combinações afins de* $\mathbf{x}_1, \ldots, \mathbf{x}_k$. *Caso as combinações exijam simplesmente que os* $\lambda_j, j = 1, \ldots, k$, *sejam reais quaisquer, os chamamos de combinações lineares de* $\mathbf{x}_1, \ldots, \mathbf{x}_k$.

Deste modo, um conjunto convexo pode ser visto como o conjunto de todas as combinações convexas de $\mathbf{x}_1$ e $\mathbf{x}_2$.

Definição 2.19 (Hiperplano)**.** *Um hiperplano* $\mathcal{H} \subset \mathbb{R}^n$ *é um conjunto da forma*

$$\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{p}^T\mathbf{x} = \alpha\}, \tag{2.21}$$

onde $\mathbf{p} \in \mathbb{R}^n$ *é um vetor não-nulo chamado de normal (ou gradiente) ao hiperplano* $\mathcal{H}$ *e* α *um escalar.*

Notemos que se $\bar{\mathbf{x}} \in \mathcal{H}$, temos $\mathbf{p}^T\bar{\mathbf{x}} = \alpha$, de modo que podemos escrever

$$\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{p}^T(\mathbf{x} - \bar{\mathbf{x}}) = 0\}. \tag{2.22}$$

Deste modo, o vetor $\mathbf{p}$ é ortogonal a todos os vetores $(\mathbf{x} - \bar{\mathbf{x}}), \forall \mathbf{x} \in \mathcal{H}$, sendo perpendicular à superfície do hiperplano $\mathcal{H}$.

Todo hiperplano define dois semiespaços abertos

$$\mathcal{H}^+ = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{p}^T\mathbf{x} > \alpha\} \quad \text{e} \quad \mathcal{H}^- = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{p}^T\mathbf{x} < \alpha\} \tag{2.23}$$

e dois semiespaços fechados

$$\overline{\mathcal{H}}^+ = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{p}^T\mathbf{x} \geq \alpha\} \quad \text{e} \quad \overline{\mathcal{H}}^- = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{p}^T\mathbf{x} \leq \alpha\}. \tag{2.24}$$

Definição 2.20 (Conjunto poliédrico)**.** *Um conjunto* $\mathcal{S} \subseteq \mathbb{R}^n$ *é chamado de poliédrico ou poliedral se é a intersecção de um número finito de semiespaços fechados, isto é,*

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{p}_i^T\mathbf{x} \leq \alpha_i, i = 1, 2, \ldots, n\}, \tag{2.25}$$

onde cada $\mathbf{p}_i$ *é um vetor não-nulo (do* $\mathbb{R}^n$*) e cada* α_i *é um escalar. Em geral, um conjunto poliédrico é descrito por*

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}, \tag{2.26}$$

onde $\mathbf{A}$ *é uma matriz* $m \times n$ *e* $\mathbf{b}$ *é um vetor de dimensão* m*.*

Hiperplanos, semiespaços e conjuntos poliédricos são exemplos de conjuntos convexos.

Lema 2.4. *Sejam* $\mathcal{S}_1$ *e* $\mathcal{S}_2$ *conjuntos convexos do* $\mathbb{R}^n$*. Então:*

(1) $\mathcal{S}_1 \cap \mathcal{S}_2$ *é um conjunto convexo.*

(2) $\mathcal{S}_1 \oplus \mathcal{S}_2 = \{\mathbf{x}_1 + \mathbf{x}_2 \mid \mathbf{x}_1 \in \mathcal{S}_1, \mathbf{x}_2 \in \mathcal{S}_2\}$ *é um conjunto convexo.*

(3) $\mathcal{S}_1 \ominus \mathcal{S}_2 = \{\mathbf{x}_1 - \mathbf{x}_2 \mid \mathbf{x}_1 \in \mathcal{S}_1, \mathbf{x}_2 \in \mathcal{S}_2\}$ *é um conjunto convexo.*

Demonstração. Sejam $\mathcal{S}_1$ e $\mathcal{S}_2$ conjuntos convexos do $\mathbb{R}^n$.

(1) Consideremos $\mathbf{x}, \mathbf{y} \in \mathcal{S}_1 \cap \mathcal{S}_2$, $\lambda \in [0, 1]$. Assim, $\mathbf{x}, \mathbf{y} \in \mathcal{S}_1$, e como $\mathcal{S}_1$ é convexo temos que $\lambda\mathbf{x} + (1 - \lambda)\mathbf{y} \in \mathcal{S}_1$. Além disso, também é verdade que $\mathbf{x}, \mathbf{y} \in \mathcal{S}_2$, e como $\mathcal{S}_2$ é convexo vale que $\lambda\mathbf{x}+(1-\lambda)\mathbf{y} \in \mathcal{S}_2$. Portanto, $\lambda\mathbf{x}+(1-\lambda)\mathbf{y} \in \mathcal{S}_1 \cap \mathcal{S}_2$ e $\mathcal{S}_1 \cap \mathcal{S}_2$ é convexo.

(2) Tomemos $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}_1 \oplus \mathcal{S}_2$, $\lambda \in [0, 1]$, com $\mathbf{x}_1 \in \mathcal{S}_1$ e $\mathbf{x}_2 \in \mathcal{S}_2$. Como $\mathcal{S}_1$ é um conjunto convexo, temos que $\mathbf{x}_1 = \lambda\mathbf{a}_1 + (1 - \lambda)\mathbf{b}_1$, para $\mathbf{a}_1, \mathbf{b}_1 \in \mathcal{S}_1$, e como $\mathcal{S}_2$ também é um conjunto convexo, temos que $\mathbf{x}_2 = \lambda\mathbf{a}_2 + (1 - \lambda)\mathbf{b}_2$, para $\mathbf{a}_2, \mathbf{b}_2 \in \mathcal{S}_2$.

Deste modo,

$$\mathbf{x}_1 + \mathbf{x}_2 = \lambda\,(\mathbf{a}_1 + \mathbf{a}_2) + (1 - \lambda)\,(\mathbf{b}_1 + \mathbf{b}_2) \in \mathcal{S}_1 \oplus \mathcal{S}_2,$$

ou seja, $\mathcal{S}_1 \oplus \mathcal{S}_2$ é convexo.

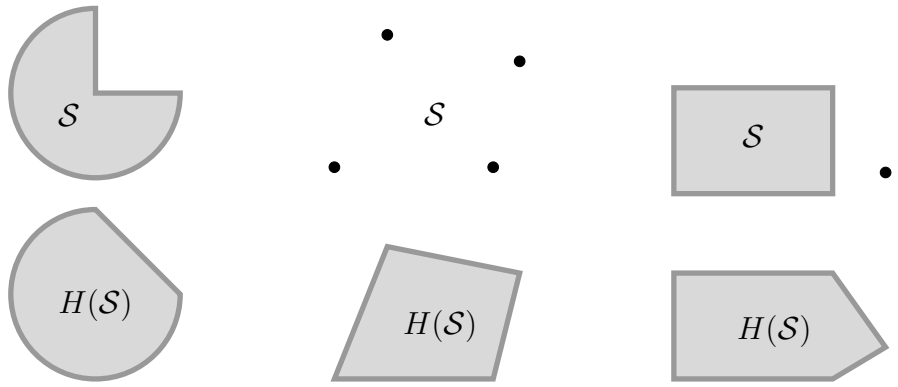
(3) Demonstração é análoga ao item 2.

■

Definição 2.21 (Envelope convexo)**.** *Seja $\mathcal{S} \in \mathbb{R}^n$ um conjunto arbitrário. O envelope convexo de $\mathcal{S}$ (também chamado de fecho convexo de $\mathcal{S}$), denotado por $H(\mathcal{S})$, é o conjunto de todas as combinações convexas de $\mathcal{S}$. Em outras palavras, $\mathbf{x} \in H(\mathcal{S})$ se e somente se $\mathbf{x}$ pode ser representado como*

$$\mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j, \tag{2.27}$$

onde $\sum_{j=1}^{k} \lambda_j = 1, \lambda_j \geq 0, j = 1, \ldots, k$, k é um inteiro positivo e $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \in \mathcal{S}$.



(a) (b) (c)

Figura 2.8: Envelope convexo.

Lema 2.5.

(1) Uma intersecção qualquer de conjuntos convexos é também um conjunto convexo.

(2) Se $\mathcal{U} \supseteq \mathcal{S}$ então $H(\mathcal{U}) \supseteq H(\mathcal{S})$.

(3) Se $\mathcal{U}$ é convexo então $H(\mathcal{U}) = \mathcal{U}$.

Demonstração.

(1) Sejam $\mathbf{x}_1, \mathbf{x}_2 \in \bigcap_{p\in\Sigma} \mathcal{U}_p$, onde $\mathcal{U}_p, p \in \Sigma$, é convexo, e seja $\lambda \in [0, 1]$.

Como cada $\mathcal{U}_p$ é convexo, temos que $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{U}_p, \forall p \in \Sigma$ e, então,

$$\mathbf{x} = \lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in \bigcap_{p\in\Sigma} \mathcal{U}_p. \tag{2.28}$$

Portanto, $\bigcap_{p\in\Sigma} \mathcal{U}_p$ é convexo.

(2) Seja $\mathcal{U} \supseteq \mathcal{S}$ e $\mathbf{x} \in H(\mathcal{S})$. Então, $\exists k \in \mathbb{Z}_+, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \lambda_1, \lambda_2, \ldots, \lambda_k$ tais que

$$\mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j, \text{ com } \sum_{j=1}^{k} \lambda_j = 1, \lambda_j \geq 0 \text{ e } \mathbf{x}_j \in \mathcal{S}, j = 1, \ldots, k. \tag{2.29}$$

Como $\mathcal{U} \supseteq \mathcal{S}$, então $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \in \mathcal{U}$. Portanto, $\mathbf{x} \in H(\mathcal{U})$ (com os mesmos k e λ_j) e $H(\mathcal{U}) \supseteq H(\mathcal{S})$.

(3) Seja $\mathcal{U}$ convexo.

$(H(\mathcal{U}) \subseteq \mathcal{U})$

Seja $\mathbf{x} \in H(\mathcal{U})$. Por definição, existe $k \in \mathbb{Z}_+$ tal que $\mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j$, com $\sum_{j=1}^{k} \lambda_j = 1$, $\lambda_j \geq 0$ e $\mathbf{x}_j \in \mathcal{U}$, para $j = 1, \ldots, k$.

Como $\mathcal{U}$ é convexo, toda combinação convexa de elementos de $\mathcal{U}$ também pertence ao conjunto. Logo, $\mathbf{x} \in \mathcal{U}$.

$(H(\mathcal{U}) \supseteq \mathcal{U})$

Seja $\mathbf{x} \in \mathcal{U}$. Tomemos $k = 1$, $\lambda_1 = 1$ e $\mathbf{x} = \lambda_1\mathbf{x}_k$, onde $\mathbf{x}_1 = \mathbf{x} \in \mathcal{U}$. Deste modo, $\mathbf{x} \in H(\mathcal{U})$.

Portanto, concluímos que $H(\mathcal{U}) = \mathcal{U}$.

■

Lema 2.6. *O envelope convexo de um conjunto $\mathcal{S}$ é*

(1) o menor conjunto convexo que contém $\mathcal{S}$.

(2) a intersecção de todos os conjuntos convexos que contêm $\mathcal{S}$.

Demonstração.

(1) Seja $\mathcal{U} \supseteq \mathcal{S}$. Pelo lema 2.5, temos que $H(\mathcal{U}) \supseteq H(\mathcal{S})$. Ademais, como $\mathcal{U}$ é convexo, temos também pelo lema 2.5 que $H(\mathcal{U}) = \mathcal{U}$. Portanto, $\mathcal{U} \supseteq H(\mathcal{S})$.

(2) Seja $\mathcal{W} = \bigcap\{\mathcal{U} \mid \mathcal{U} \supseteq \mathcal{S}, \mathcal{U}$ é convexo$\}$.

($\mathcal{W} \subseteq H(\mathcal{S})$)

Seja $\mathbf{x} \in \mathcal{W}$. Então, $\mathbf{x}$ pertence a todo $\mathcal{U}$ convexo que contém $\mathcal{S}$, em particular, tal é válido para $\mathcal{U} = H(\mathcal{S})$, ou seja, $\mathbf{x} \in H(\mathcal{S})$.

($\mathcal{W} \supseteq H(\mathcal{S})$)

Seja $\mathbf{x} \in H(\mathcal{S})$. Pela parte (1), $\mathbf{x}$ pertence a qualquer conjunto convexo $\mathcal{U}$ que contenha $\mathcal{S}$. Então, $\mathbf{x} \in \mathcal{W}$, que é a intersecção de todos os $\mathcal{U}$.

Portanto, temos que $\mathcal{W} = H(\mathcal{S})$ [3].

■

Definição 2.22 (Coordenadas baricêntricas)**.** *Sejam A, B e C os vértices de um triângulo ABC e Q um ponto qualquer do plano ($\mathbb{R}^2$). Dizemos que α, β e γ são as coordenadas baricêntricas de Q em relação ao triângulo ABC se*

$$Q = \frac{\alpha A + \beta B + \gamma C}{\alpha + \beta + \gamma}, \tag{2.30}$$

onde $\alpha+\beta+\gamma \neq 0$. Assim, o ponto Q fica definido como uma média ponderada entre os vértices do triângulo ABC, com pesos α, β e γ, e o ponto Q pode ser identificado através da terna $Q = (\alpha : \beta : \gamma)$.

De fato, qualquer ponto do plano pode ser identificado através de coordenadas baricêntricas. Tomemos os vértices A, B e C do triângulo ABC e os seus lados AB e AC. Eles formam uma base para o plano, ou seja, dado qualquer ponto Q pertencente ao plano, o segmento AQ pode ser escrito através de uma combinação linear de AB e AC, de modo que

$$AQ = uAB + vAC, \text{ com } u, v \in \mathbb{R}. \tag{2.31}$$

Sendo $O = (0, 0, 0)$ a origem do sistema coordenado, temos

$$AO + OQ = u(AO + OB) + v(AO + OC), \tag{2.32}$$

ou seja,

$$OQ = (1 - u - v)OA + uOB + vOC, \tag{2.33}$$

e fazendo $\alpha = 1 - u - v$, $\beta = u$ e $\gamma = v$ ficamos com $\alpha + \beta + \gamma = 1$ e, portanto,

$$Q = \alpha A + \beta B + \gamma C. \tag{2.34}$$

As coordenadas baricêntricas de um ponto não são únicas. Se um dado ponto Q do plano é tal que $Q = (\alpha : \beta : \gamma)$ e outro ponto P do plano é tal que $P = (k\alpha : k\beta : k\gamma)$, com $k \neq 0$, então temos que $Q = P$. Isso fica fácil de se ver tomando como exemplo $Q = (2 : 3 : 5)$ e $P = (4 : 6 : 10)$. Com respeito a um triângulo ABC, temos

$$P = \frac{4A + 6B + 10C}{4 + 6 + 10} = \frac{2A + 3B + 5C}{2 + 3 + 5} = Q \tag{2.35}$$

Proposição 2.2. *Sejam $P_1 = (\alpha_1 : \beta_1 : \gamma_1)$ e $P_2 = (\alpha_2 : \beta_2 : \gamma_2)$ as coordenadas baricêntricas dos pontos P_1 e P_2 com relação a um triângulo ABC. Temos que $P_1 = P_2$ se, e somente se, existe um número real não-nulo k tal que $\alpha_2 = k\alpha_1$, $\beta_2 = k\beta_1$ e $\gamma_2 = k\gamma_1$.*

Demonstração. Sejam $P_1 = (\alpha_1 : \beta_1 : \gamma_1)$, $P_2 = (\alpha_2 : \beta_2 : \gamma_2)$ e $P = (\alpha : \beta : \gamma)$ as coordenadas baricêntricas dos pontos P_1, P_2 e P com relação a um triângulo referência ABC.

Sendo assim, temos

$$(\alpha+\beta+\gamma)P = \alpha A + \beta B + \gamma C \tag{2.36}$$

$$(\alpha_1+\beta_1+\gamma_1)P_1 = \alpha_1 A + \beta_1 B + \gamma_1 C \tag{2.37}$$

$$(\alpha_2+\beta_2+\gamma_2)P_2 = \alpha_2 A + \beta_2 B + \gamma_2 C, \tag{2.38}$$

onde $\alpha+\beta+\gamma \neq 0$, $\alpha_1+\beta_1+\gamma_1 \neq 0$ e $\alpha_2+\beta_2+\gamma_2 \neq 0$.

($\Rightarrow$) Suponhamos $P_1 = P_2$.

Sejam os pontos do plano $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P = (x_P, y_P)$, $A = (x_A, y_A)$, $B = (x_B, y_B)$ e $C = (x_C, y_C)$. Temos $[PP_1P_2] = 0$, de modo que

$$\begin{vmatrix} x_P & y_P & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = \begin{vmatrix} \dfrac{\alpha x_A+\beta x_B+\gamma x_C}{\alpha+\beta+\gamma} & \dfrac{\alpha y_A+\beta y_B+\gamma y_C}{\alpha+\beta+\gamma} & 1 \\ \dfrac{\alpha_1 x_A+\beta_1 x_B+\gamma_1 x_C}{\alpha_1+\beta_1+\gamma_1} & \dfrac{\alpha_1 y_A+\beta_1 y_B+\gamma_1 y_C}{\alpha_1+\beta_1+\gamma_1} & 1 \\ \dfrac{\alpha_2 x_A+\beta_2 x_B+\gamma_2 x_C}{\alpha_2+\beta_2+\gamma_2} & \dfrac{\alpha_2 y_A+\beta_2 y_B+\gamma_2 y_C}{\alpha_2+\beta_2+\gamma_2} & 1 \end{vmatrix} = 0 \tag{2.39}$$

Multiplicando a primeira linha por $\alpha+\beta+\gamma$, a segunda por $\alpha_1+\beta_1+\gamma_1$ e a terceira por $\alpha_2+\beta_2+\gamma_2$, chegamos em

$$\begin{vmatrix} \alpha x_A+\beta x_B+\gamma x_C & \alpha y_A+\beta y_B+\gamma y_C & \alpha+\beta+\gamma \\ \alpha_1 x_A+\beta_1 x_B+\gamma_1 x_C & \alpha_1 y_A+\beta_1 y_B+\gamma_1 y_C & \alpha_1+\beta_1+\gamma_1 \\ \alpha_2 x_A+\beta_2 x_B+\gamma_2 x_C & \alpha_2 y_A+\beta_2 y_B+\gamma_2 y_C & \alpha_2+\beta_2+\gamma_2 \end{vmatrix} = 0, \tag{2.40}$$

isto é,

$$\begin{vmatrix} \alpha & \beta & \gamma \\ \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \end{vmatrix} \cdot \begin{vmatrix} x_Q & y_Q & 1 \\ x_A & y_A & 1 \\ x_B & y_B & 1 \end{vmatrix} = 0 \tag{2.41}$$

Isso é equivalente a

$$\begin{vmatrix} \alpha & \beta & \gamma \\ \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \end{vmatrix} \cdot 2[ABC] = 0 \tag{2.42}$$

Logo, temos que

$$\begin{vmatrix} \alpha & \beta & \gamma \\ \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \end{vmatrix} = 0 \tag{2.43}$$

Como o ponto P é qualquer, o determinante se anula para quaisquer valores de α, β e γ, o que nos leva a concluir que a segunda e terceira linhas são múltiplas uma da outra, ou seja, existe um número real não-nulo k tal que $\alpha_2 = k\alpha_1$, $\beta_2 = k\beta_1$ e $\gamma_2 = k\gamma_1$.

($\Leftarrow$) Suponhamos agora que exista um número real não-nulo k tal que $\alpha_2 = k\alpha_1$, $\beta_2 = k\beta_1$ e $\gamma_2 = k\gamma_1$. Temos então que

$$P_2 = \frac{\alpha_2 A + \beta_2 B + \gamma_2 C}{\alpha_2 + \beta_2 + \gamma_2} = \frac{k\alpha_1 A + k\beta_1 B + k\gamma_1 C}{k\alpha_1 + k\beta_1 + k\gamma_1} = \frac{\alpha_1 A + \beta_1 B + \gamma_1 C}{\alpha_1 + \beta_1 + \gamma_1} = P_1 \tag{2.44}$$

■

Proposição 2.3. *Sejam A, B e C os vértices do triângulo ABC e Q um ponto qualquer no plano. As coordenadas baricêntricas de Q com relação ao triângulo ABC podem ser dadas por*

$$Q = ([QBC] : [QCA] : [QAB]), \tag{2.45}$$

ou seja, as coordenadas baricêntricas são proporcionais às áreas com sinal dos sub-triângulos determinados por Q e pelos vértices do triângulo ABC.

Demonstração. Sejam $A = (x_A, y_A)$, $B = (x_B, y_B)$ e $C = (x_C, y_C)$ as coordenadas cartesianas dos vértices do triângulo ABC e $Q = (x_Q, y_Q)$ as de um ponto qualquer do plano. Assim,

$$\begin{vmatrix} x_Q & x_Q & y_Q & 1 \\ x_A & x_A & y_A & 1 \\ x_B & x_B & y_B & 1 \\ x_C & x_C & y_C & 1 \end{vmatrix} = 0 \tag{2.46}$$

Desenvolvendo este determinante pela primeira coluna, obtemos

$$x_Q \cdot \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} - x_A \cdot \begin{vmatrix} x_Q & y_Q & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} + x_B \cdot \begin{vmatrix} x_Q & y_Q & 1 \\ x_A & y_A & 1 \\ x_C & y_C & 1 \end{vmatrix} - x_C \cdot \begin{vmatrix} x_Q & y_Q & 1 \\ x_A & y_A & 1 \\ x_B & y_B & 1 \end{vmatrix} = 0 \tag{2.47}$$

ou ainda

$$x_Q \cdot \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} - x_A \cdot \begin{vmatrix} x_Q & y_Q & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} - x_B \cdot \begin{vmatrix} x_Q & y_Q & 1 \\ x_C & y_C & 1 \\ x_A & y_A & 1 \end{vmatrix} - x_C \cdot \begin{vmatrix} x_Q & y_Q & 1 \\ x_A & y_A & 1 \\ x_B & y_B & 1 \end{vmatrix} = 0 \tag{2.48}$$

$$\Leftrightarrow x_Q \cdot 2[ABC] - x_A \cdot 2[QBC] - x_B \cdot 2[QCA] - x_C \cdot 2[QAB] = 0 \tag{2.49}$$

$$\Leftrightarrow [ABC] \cdot x_Q = [QBC] \cdot x_A + [QCA] \cdot x_B + [QAB] \cdot x_C \tag{2.50}$$

Analogamente, se considerarmos

$$\begin{vmatrix} y_Q & x_Q & y_Q & 1 \\ y_A & x_A & y_A & 1 \\ y_B & x_B & y_B & 1 \\ y_C & x_C & y_C & 1 \end{vmatrix} = 0 \tag{2.51}$$

obteremos a relação

$$[ABC] \cdot y_Q = [QBC] \cdot y_A + [QCA] \cdot y_B + [QAB] \cdot y_C. \tag{2.52}$$

Deste modo, podemos escrever

$$[ABC]Q = [QBC]A + [QCA]B + [QAB]C \tag{2.53}$$

Como $[ABC] = [QBC] + [QCA] + [QAB]$, obtemos

$$Q = \frac{[QBC]A + [QCA]B + [QAB]C}{[QBC] + [QCA] + [QAB]}, \tag{2.54}$$

ou seja, $Q = ([QBC] : [QCA] : [QAB])$ [9, 16]. ■

2.2 Teste de inclusão em polígonos de Feito-Torres

Como demonstrado em [7], para determinar se um ponto Q é interior a um polígono P planar qualquer é suficiente calcular a soma dos sinais das áreas com sinal de todos os triângulos pela origem que contêm Q, determinados pela origem O e cada uma das arestas do polígono P. Para determinar se este ponto está na fronteira é previamente testado se ele pertence à aresta ou não. O algoritmo segue, maiores detalhes em [7, 8].

Sejam $P = E_1E_2 \dots E_n$ um polígono em $\mathbb{R}^2$, E_i suas arestas e T_i os triângulos pela origem determinados pela origem O e a aresta E_i.

Algoritmo de Feito-Torres (versão 2D)

ENTRADA: polígono P, ponto Q

SAÍDA: 1: Q ponto interior; 0: Q ponto exterior

1: $INCLUSAO = 0$
2: **para** $i = 1$ **até** n **faça**
3: **se** $(Q \in E_i)$ **então**
4: **retorna** 1
5: **fim se**
6: **se** $(Q \in \text{int}\,(T_i))$ **então**
7: $INCLUSAO = INCLUSAO + \text{sign}\,([T_i])$
8: **fim se**
9: **se** $(Q \in \partial(T_i))$ **então**
10: $INCLUSAO = INCLUSAO + 0.5 * \text{sign}\,([T_i])$
11: **fim se**
12: **fim para**
13: **se** $(INCLUSAO == 1)$ **então**
14: **retorna** 1

15: **senão**

16: **retorna** 0

17: **fim se**

Notemos que um ponto que pertença a uma aresta pela origem comum a dois triângulos, contabiliza o valor $0.5 * \text{sign}\,([T_i])$ para cada um dos triângulos, evitando assim uma contagem dupla de inclusão.

Na figura 2.9 pode-se ver um exemplo de aplicação deste algoritmo. Os valores sucessivos de $INCLUSAO$ são mostrados na tabela 2.1. O polígono do exemplo possui orientação positiva.

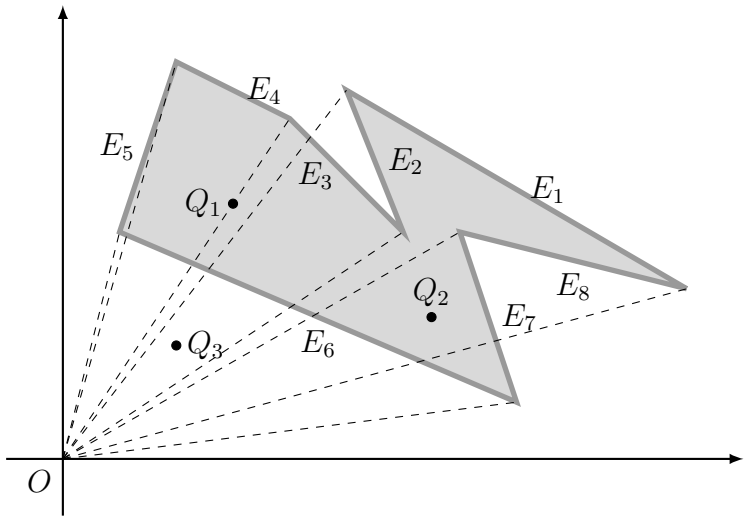


Figura 2.9: Exemplos de aplicação do algoritmo de Feito-Torres versão 2D.

Para o ponto Q_2 tem-se:

1. $Q_2 \in T_1$, portanto $INCLUSAO = 0 + \text{sign}\,([OE_1]) = 0 + 1 = 1$;
2. $Q_2 \notin T_i$, $i = 2, \ldots, 6$, portanto o valor de $INCLUSAO$ permanece sendo 1, pois $\text{sign}\,([OE_i]) = 0$ para $i = 2, \ldots, 6$;
3. $Q_2 \in T_7$, portanto $INCLUSAO = INCLUSAO + \text{sign}\,([OE_7]) = 1 + (+1) = 2$;

4. $Q_2 \in T_8$, portanto $INCLUSAO = INCLUSAO + \text{sign}([OE_8]) = 2 + (-1) = 1$.

Como o valor final de $INCLUSAO$ é 1, conclui-se que o ponto Q_2 é interior ao polígono.

Ponto	sign $([T_i])$	$INCLUSAO$
Q_1	$0, 0, 1/2, 1/2, 0, 0, 0, 0$	$0, 0, 1/2, 1, 1, 1, 1, 1$
Q_2	$1, 0, 0, 0, 0, 0, 1, -1$	$1, 1, 1, 1, 1, 1, 2, 1$
Q_3	$1, -1, 1, 0, 0, -1, 0, 0$	$1, 0, 1, 1, 1, 0, 0, 0$

Tabela 2.1: Valores do sinal dos triângulos pela origem e de $INCLUSAO$.

2.3 Teste de inclusão em poliedros de Feito-Torres

Aqui é descrita a generalização do algoritmo de Feito-Torres, especificado na seção 2.2, para o caso 3D [8]. O teste segue a mesma linha da versão planar, mas apresenta uma variedade maior de detalhes. A idéia principal continua sendo a decomposição da figura em figuras menores, mas neste caso serão usados tetraedros com vértice na origem.

São considerados tetraedros pela origem determinados pela origem, um vértice fixo de cada face e cada uma das demais arestas da mesma face que não sejam incidentes a esse vértice fixado. A seguinte terminação é utilizada:

- $V_+(Q)$: conjunto dos vértices que determinam arestas pela origem que contêm Q e ou que pertençam a tetraedros positivos ou que são pontos iniciais ou finais de faces positivas, de acordo com a ordem em que aparecem na descrição da face. $\#V_+(Q)$ denota a cardinalidade deste conjunto.

- $V_{-}(Q)$: conjunto dos vértices que determinam arestas pela origem que contêm Q e ou que pertençam a tetraedros negativos ou que são pontos iniciais ou finais de faces negativas, de acordo com a ordem em que aparecem na descrição da face. $\#V_{-}(Q)$ denota a cardinalidade deste conjunto.

- $T_{or}(OV_0V_1V_2)$: conjunto de faces (triângulos) de tetraedros pela origem determinados pela origem O e os pontos V_0, V_1 e V_2.

O teorema a seguir é empregado para pontos que não estão na fronteira do poliedro. Pode-se determinar se um ponto coplanar a uma face do poliedro é interior a face aplicando-se a versão 2D do algoritmo em uma projeção da face [8].

Teorema 2.3. *Seja $P = F_1F_2 \dots F_n$ um poliedro qualquer com faces $F_1, F_2, \dots, F_n$, onde $F_i = V_{i,1}V_{i,2} \dots V_{i,e_i}$. Então um ponto $Q \in \mathbb{R}^3$ não contido na fronteira de P é interior ao poliedro se, e somente se,*

$$\sum_{i=1}^{n}\sum_{j=1}^{e_i} \lambda_{ij} + \#V_{+}(Q) - \#V_{-}(Q) = 1, \tag{2.55}$$

onde $1 < j < e_i$ e

$$\lambda_{ij} = \begin{cases} \text{sign } (OV_{i,1}V_{i,j}V_{i,j+1}), & \text{se } Q \in \text{int}\,(OV_{i,1}V_{i,j}V_{i,j+1}) \\ 0.5 * \text{sign } (OV_{i,1}V_{i,j}V_{i,j+1}), & \text{se } Q \in \text{int}\,(T_k), \text{com } T_k \in T_{or}(OV_{i,1}V_{i,j}V_{i,j+1}) \\ 0, & \text{caso contrário} \end{cases} \tag{2.56}$$

Demonstração. É sabido que um ponto Q é interior à P se, e somente se, um raio com origem em Q tem um número ímpar de intersecções com a fronteira do polígono P, pelo teorema da Curva de Jordan (teorema 2.2). Demonstraremos que a equação (2.55) computa o número correto de intersecções.

Cada intersecção implica em uma transição entre o interior e o exterior da figura, e vice-versa. Se considerarmos um raio que contém Q a partir da origem, cada intersecção entre o raio e o poliedro se encaixa em um dos seguintes casos:

1. A intersecção é um segmento de reta que é coplanar com uma face (observe que o ponto Q não está na face, mas sim no plano determinado pela face). O sinal é considerado como sendo zero e não é computada uma transição.

2. O raio cruza uma face e existe um ponto de intersecção que é interior à face. Cada intersecção com uma face i implica em um número ímpar de cruzamentos com o tetraedro associado a esta face e, neste caso, cada uma contribui com sign ($[F_i]$) para a contagem, de modo que o número total é $+1$. Esta ocorrência é equivalente ao primeiro valor de λ_{ij}.

3. O ponto de intersecção é interior a uma face mas ele pertence a uma aresta comum a dois tetraedros, que não é pela origem. Sendo assim, o ponto Q pertence à face pela origem comum a esses dois tetraedros, e a contribuição para a fórmula é $1/2 + 1/2 = 1$ se ambos os tetraedros são positivos, $1/2 - 1/2 = 0$ se um é positivo e o outro negativo ou $-1/2 - 1/2 = -1$ se ambos forem negativos. Este caso é tratado pelo segundo valor de λ_{ij} e é similar ao caso bidimensional exemplificado pelo ponto Q_1 da figura 2.9.

4. O ponto de intersecção pertence a uma aresta comum a várias faces. Esse número será 2 para poliedros geometricamente regulares ou um múltiplo de 2 para poliedros geometricamente não-regulares. Duas possibilidades podem ocorrer: haver uma transição, de modo que $+1$ intersecção deve ser contada (figura 2.10(b)); ou não haver uma transição, de modo que nenhuma intersecção é contada (figura 2.10(a)). Para poliedros não-regulares o raciocínio é análogo (figura 2.10(c) e 2.10(d)). Este caso é tratado pelo segundo valor de λ_{ij}.

5. O ponto de intersecção coincide com um vértice do poliedro e podemos contar uma transição ou não. Se uma transição é considerada,

significa que o vértice pertence ao conjunto $V_+(Q)$ e não pertence ao conjunto $V_-(Q)$, ou vice-versa (figura 2.10(e)). Em qualquer uma das duas situações, para este vértice $\#V_+(Q) - \#V_-(Q)$ é $+1$ ou -1, o que é necessário para termos uma transição, de modo que λ_{ij} é zero. Se não há transição alguma, então o vértice pertence à $V_+(Q)$ e à $V_-(Q)$ e $\#V_+(Q) - \#V_-(Q) = 0$, de modo que λ_{ij} também é zero (figura 2.10(f)).

Se Q não é interior ele é exterior, e já é assumido que ele não pertence à fronteira do poliedro. Então, cada raio que tem origem em Q intercepta o poliedro um número ímpar de vezes. ■

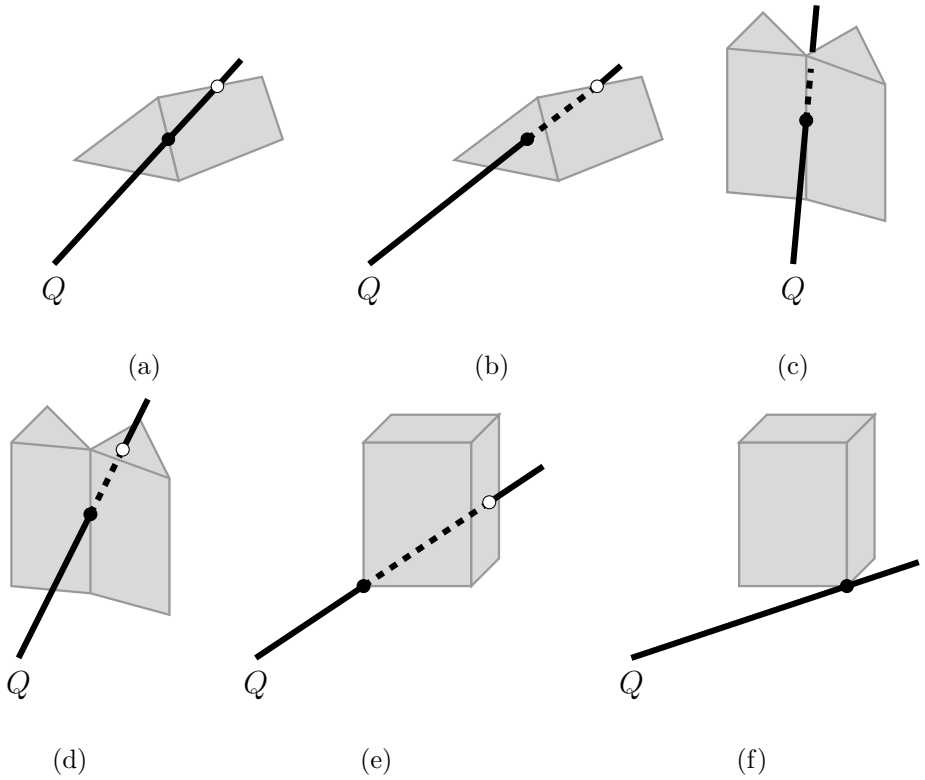


Figura 2.10: Demonstração do teorema 2.3.

Este teorema permite decidir se um ponto pertence a um poliedro qualquer e é usado na implementação do algoritmo, que avalia a equação (2.55). O

esquema deste algoritmo é mostrado a seguir e maiores detalhes podem ser encontrados em [8].

ALGORITMO DE FEITO-TORRES (VERSÃO 3D)

ENTRADA: poliedro P, ponto Q

SAÍDA: 1: Q ponto interior; 0: Q ponto exterior

1: $INCLUSAO = 0$
2: $V_+(Q) =$ vazio
3: $V_-(Q) =$ vazio
4: **para** $i = 1$ **até** n **faça**
5: **se** $(Q \in F_i)$ **então**
6: **retorna** 1
7: **fim se**
8: **se** $((Q \in \text{int}\,(OV_{i,1}))$ E
9: $((\text{sign}\,([OF_i]) > 0$ E $V_{i,1} \notin V_+(Q))$ OU
10: $(\text{sign}\,([OF_i]) < 0$ E $V_{i,1} \notin V_-(Q))))$ **então**
11: $INCLUSAO = INCLUSAO + \text{sign}\,([OF_i])$
12: **se** $(\text{sign}\,([OF_i]) < 0)$ **então**
13: $V_-(Q) = V_-(Q) + V_{i,1}$
14: **senão**
15: $V_+(Q) = V_+(Q) + V_{i,1}$
16: **fim se**
17: **senão se** $((Q \in \text{int}\,(OV_{i,e_i}))$ E
18: $((\text{sign}\,([OF_i]) > 0$ E $V_{i,e_i} \notin V_+(Q))$ OU
19: $(\text{sign}\,([OF_i]) < 0$ E $V_{i,e_i} \notin V_-(Q))))$ **então**
20: $INCLUSAO = INCLUSAO + \text{sign}\,([OF_i])$
21: **se** $(\text{sign}\,([OF_i]) < 0)$ **então**
22: $V_-(Q) = V_-(Q) + V_{i,e_i}$
23: **senão**

24: $V_+(Q) = V_+(Q) + V_{i,e_i}$

25: **fim se**

26: **senão**

27: **para** $j = 2$ **até** $e_i - 1$ **faça**

28: **se** $((Q \in \text{int}\,(OV_{i,1}V_{i,j}))$ OU $(Q \in \text{int}\,(OV_{i,j}V_{i,j+1}))$ OU

29: $(Q \in \text{int}\,(OV_{i,j+1}V_{i,1})))$ **então**

30: $INCLUSAO = INCLUSAO + 0.5 * \text{sign}\,([OV_{i,1}V_{i,j}V_{i,j+1}])$

31: **senão se** $((Q \in \text{int}\,(OV_{i,j}))$ E

32: $((\text{sign}\,([OV_{i,1}V_{i,j}V_{i,j+1}]) > 0$ E $V_{i,j} \notin V_+(Q))$ OU

33: $(\text{sign}\,([OV_{i,1}V_{i,j}V_{i,j+1}]) < 0$ E $V_{i,j} \notin V_-(Q))))$ **então**

34: $INCLUSAO = INCLUSAO + \text{sign}\,([OV_{i,1}V_{i,j}V_{i,j+1}])$

35: **se** $(\text{sign}\,([OV_{i,1}V_{i,j}V_{i,j+1}]) < 0)$ **então**

36: $V_-(Q) = V_-(Q) + V_{i,j}$

37: **senão**

38: $V_+(Q) = V_+(Q) + V_{i,j}$

39: **fim se**

40: **senão se** $(Q \in \text{int}\,(OV_{i,1}V_{i,j}V_{i,j+1}))$ **então**

41: $INCLUSAO = INCLUSAO + \text{sign}\,([OV_{i,1}V_{i,j}V_{i,j+1}])$

42: **fim se**

43: **fim para**

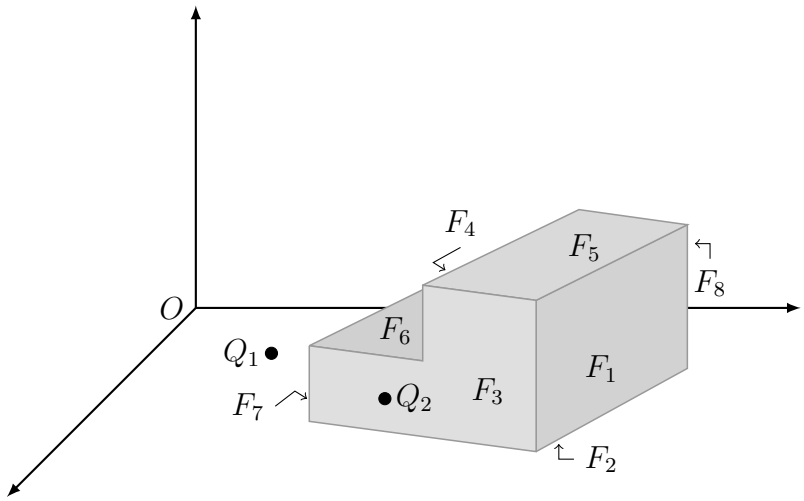
44: **fim se**

45: **fim para**

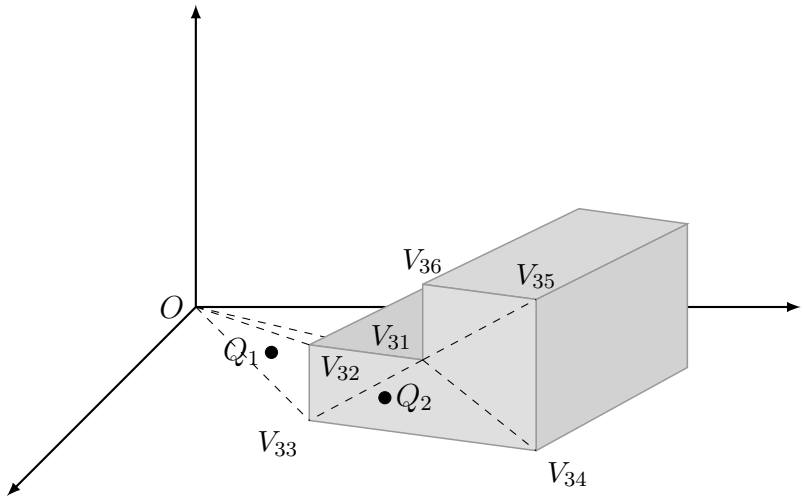
46: **retorna** $INCLUSAO$

Na figura 2.11 é mostrado um exemplo de aplicação do algoritmo [8]. É suposto que Q_1 pertença aos tetraedros $OV_{31}V_{32}V_{33}$ e $OV_{71}V_{72}V_{73}$ (onde $V_{32} = V_{71}$) e a mais nenhum tetraedro. Para o ponto Q_2 é suposto que ele seja interior ao tetraedro $OV_{31}V_{32}V_{34}$ e a mais nenhum outro. A tabela 2.2 mostra os sucessivos

valores de $INCLUSAO$ para esses dois pontos. Note que o algoritmo é eficaz com poliedros não-regulares, como pressuposto no teorema 2.3.



(a)



(b)

Figura 2.11: Exemplos de aplicação do algoritmo de Feito-Torres versão 3D.

Ponto	$INCLUSAO$
Q_1	$0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0$
Q_2	$0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1$

Tabela 2.2: Valores de $INCLUSAO$.

2.4 Algoritmo de Badouel

Esse algoritmo determina primeiramente se a intersecção do segmento com o triângulo ocorre e, posteriormente, as coordenadas desse ponto de intersecção com respeito aos vértices do triângulo. Para tal, essa intersecção e o triângulo são projetados nos planos xy, yz ou xz, com o objetivo de resolver o problema em duas dimensões usando as suas coordenadas baricêntricas (figura 2.12). Esses parâmetros podem ser usados também para o cálculo do vetor normal que tem como origem o ponto de intersecção, caso ele seja útil.

O vetor normal ao plano que contém o triângulo pode ser calculado

$$\vec{N} = \overrightarrow{V_0V_1} \times \overrightarrow{V_0V_2} \tag{2.57}$$

Para cada ponto P do plano, a quantidade $P \cdot N$ é constante. O valor dessa constante é dado por $d = -V_0 \cdot N$. Assim, temos a representação implícita do plano que contém o triângulo:

$$N \cdot P + d = 0 \tag{2.58}$$

Seja a representação paramétrica de um segmento dada por

$$R(t) = O + tD, \tag{2.59}$$

onde O é o seu ponto de origem e D a sua direção (não necessariamente normalizada).

O parâmetro t correspondente a intersecção entre o triângulo e o segmento pode ser obtido através das equações (2.58) e (2.59), e é dado por:

$$t = -\frac{d + N \cdot O}{N \cdot D} \tag{2.60}$$

Sendo assim, os seguintes testes são feitos:

- Se o triângulo e o segmento são paralelos ($N \cdot D = 0$), a intersecção é rejeitada.

- Se a intersecção está atrás da origem do segmento ($t \leq 0$), a intersecção é rejeitada.

- Se uma intersecção mais próxima de O, correspondente a um $t_R < t$, já foi encontrada, a intersecção também é rejeitada.

Seja $V_0V_1V_2$ um triângulo. A posição de um ponto P no plano desse triângulo pode ser expressa da seguinte forma:

$$\overrightarrow{V_0P} = \alpha\overrightarrow{V_0V_1} + \beta\overrightarrow{V_0V_2} \tag{2.61}$$

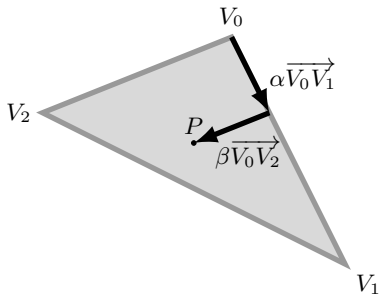


Figura 2.12: Representação paramétrica do ponto P.

Se $\alpha \geq 0$, $\beta \geq 0$ e $\alpha + \beta \leq 1$, então o ponto P pertence ao interior do triângulo. A equação (2.61) pode ser reescrita como

$$x_P - x_0 = \alpha(x_1 - x_0) + \beta(x_2 - x_0) \tag{2.62}$$

$$y_P - y_0 = \alpha(y_1 - y_0) + \beta(y_2 - y_0) \tag{2.63}$$

$$z_P - z_0 = \alpha(z_1 - z_0) + \beta(z_2 - z_0) \tag{2.64}$$

onde $P = (x_P, y_P, z_P)$ e $V_i = (x_i, y_i, z_i)$.

O triângulo é projetado em um dos planos primários xy, xz ou yz dependendo do maior valor absoluto dos coeficientes na equação do plano que o contém. Entretanto, se o triângulo for perpendicular a um desses planos sua projeção será um segmento de reta. A fim de evitar esse problema e garantir que a projeção

seja a maior possível, é determinado o eixo dominante do vetor normal do triângulo, e o plano perpendicular a esse eixo é usado. Seja i a coordenada rejeitada quando o triângulo é projetado e sejam j e k as demais coordenadas, escolhidas através do seguinte processo, onde $N = (x_N, y_N, z_N)$ é vetor normal calculado em (2.57):

$$i = \begin{cases} 0, & \text{se } |x_N| = \max\{|x_N|, |y_N|, |z_N|\} \\ 1, & \text{se } |y_N| = \max\{|x_N|, |y_N|, |z_N|\} \\ 2, & \text{se } |z_N| = \max\{|x_N|, |y_N|, |z_N|\} \end{cases} \tag{2.65}$$

As coordenadas não rejeitadas j e k representam o plano usado para a projeção do triângulo. Definimos as coordenadas (u, v) dos vetores $\overrightarrow{V_0P}$, $\overrightarrow{V_0V_1}$ e $\overrightarrow{V_0V_2}$ nesse plano bidimensional como segue:

$$\begin{aligned} u_0 &= P_{(j)} - V_{0(j)}, \quad u_1 = V_{1(j)} - V_{0(j)}, \quad u_2 = V_{2(j)} - V_{0(j)} \\ v_0 &= P_{(k)} - V_{0(k)}, \quad v_1 = V_{1(k)} - V_{0(k)}, \quad v_2 = V_{2(k)} - V_{0(k)} \end{aligned} \tag{2.66}$$

As equações (2.66) se reduzem a

$$u_0 = \alpha u_1 + \beta u_2, \quad v_0 = \alpha v_1 + \beta v_2 \tag{2.67}$$

e as soluções são dadas por

$$\alpha = \frac{\begin{vmatrix} u_0 & u_2 \\ v_0 & v_2 \end{vmatrix}}{\begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}} \qquad \text{e} \qquad \beta = \frac{\begin{vmatrix} u_1 & u_0 \\ v_1 & v_0 \end{vmatrix}}{\begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}} \tag{2.68}$$

Maiores detalhes podem ser encontrados em [2, 15].

Este algoritmo fornece as coordenadas baricêntricas do ponto P com relação ao triângulo $V_0V_1V_2$. Reescrevendo (2.61)

$$\begin{aligned} P - V_0 &= \alpha(V_1 - V_0) + \beta(V_2 - V_0) \\ &= \alpha V_1 - \alpha V_0 + \beta V_2 - \beta V_0 \qquad (2.69) \\ \Leftrightarrow P &= (1 - \alpha - \beta)V_0 + \alpha V_1 + \beta V_2 \qquad (2.70) \end{aligned}$$

Assim, $P = (1-\alpha-\beta : \alpha : \beta)$, e é possível decidir se um ponto pertence a alguma das arestas do triângulo ou ainda se ele coincide com algum dos seus vértices. Isso fica resumido na tabela a seguir.

Condições			**Localização**
$\alpha \neq 0$	e	$\beta = 0$	$P \in V_0V_1$
$\alpha = 0$	e	$\beta \neq 0$	$P \in V_0V_2$
$\alpha, \beta \neq 0$	e	$\alpha + \beta = 1$	$P \in V_1V_2$
$\alpha = 0$	e	$\beta = 0$	$P = V_0$
$\alpha = 1$	e	$\beta = 0$	$P = V_1$
$\alpha = 0$	e	$\beta = 1$	$P = V_2$

Tabela 2.3: Ponto em vértice ou aresta no algoritmo de Badouel.

Segue o algoritmo, adaptado à notação do texto.

ALGORITMO DE BADOUEL

ENTRADA: triângulo $V_0V_1V_2$, origem O, direção D, tolerância ε

SAÍDA: 1: O ponto interior; 0: O ponto exterior

1: $E_1 = V_1 - V_0$
2: $E_2 = V_2 - V_0$
3: $N = E_1 \times E_2$
4: Den $= N \cdot D$
5: **se** ((Den $> -\varepsilon$) E (Den $< \varepsilon$)) **então**
6: **retorna** 0 ▷ Rejeição 1
7: **fim se**
8: $T = V_0 - O$
9: $t = N \cdot T/\text{Den}$
10: **se** $(t < 0)$ **então**
11: **retorna** 0 ▷ Rejeição 2

12: **fim se**

13: **se** $(|x_N| > |y_N|)$ **então**

14: **se** $(|x_N| > |z_N|)$ **então**

15: $j = y$

16: $k = z$

17: **senão**

18: $j = x$

19: $k = y$

20: **fim se**

21: **senão**

22: **se** $(|y_N| > |z_N|)$ **então**

23: $j = x$

24: $k = y$

25: **senão**

26: $j = x$

27: $k = z$

28: **fim se**

29: **fim se**

30: $P_{(j)} = O_{(j)} + t * D_{(j)}$

31: $P_{(k)} = O_{(k)} + t * D_{(k)}$

32: $u_0 = P_{(j)} - V_{0(j)}$

33: $v_0 = P_{(k)} - V_{0(k)}$

34: $u_1 = V_{1(j)} - V_{0(j)}$

35: $u_2 = V_{2(j)} - V_{0(j)}$

36: $v_1 = V_{1(k)} - V_{0(k)}$

37: $v_2 = V_{2(k)} - V_{0(k)}$

38: **se** $((u_1 > -\varepsilon)$ E $(u_1 < \varepsilon))$ **então**

39: $\beta = u_0/u_2$

40: **se** $\beta < 0$ ou $\beta > 1$ **então**

41: **retorna** 0 ▷ Rejeição 3
42: **fim se**
43: $\alpha = (v_0 - \beta * v_2)/v_1$
44: **senão**
45: $\beta = (v_0 * u_1 - u_0 * v_1)/(v_2 * u_1 - u_2 * v_1)$
46: **se** $((\beta < 0)$ OU $(\beta > 1))$ **então**
47: **retorna** 0 ▷ Rejeição 3
48: **fim se**
49: $\alpha = (u_0 - \beta * u_2)/u_1$
50: **fim se**
51: **se** $((\alpha < 0)$ OU $(\alpha + \beta > 1))$ **então**
52: **retorna** 0 ▷ Rejeição 4
53: **fim se**
54: **retorna** 1 ▷ Intersecção ocorre e ponto é interior ao triângulo

2.5 Algoritmo de Möller

Esse algoritmo é baseado numa transformação da face triangular e da origem do raio $R(t) = O + tD$. Essa transformação produz a distância t da origem O até o ponto de intersecção com a face e as coordenadas baricêntricas deste ponto. Para fazer isso, o método translada o triângulo até a origem do sistema de coordenadas e o reescalona a fim de obter um triângulo unitário no plano yz e que o raio esteja alinhado com eixo x.

Um ponto T em um triângulo $V_0V_1V_2$ pode ser representado por

$$T(u, v) = (1 - u - v)V_0 + uV_1 + vV_2, \tag{2.71}$$

onde u e v determinam as suas coordenadas baricêntricas, isto é, $T = (1 - u - v : u : v)$, com $u \geq 0$, $v \geq 0$ e $u + v \leq 1$. Encontrar a interseccção $T(u, v)$ entre o

raio $R(t)$ e o triângulo é equivalente a fazer $R(t) = T(u, v)$, ou seja,

$$O + tD = (1 - u - v)V_0 + uV_1 + vV_2. \tag{2.72}$$

Reescrevendo esta equação obtemos o sistema:

$$\begin{bmatrix} -D & V_1 - V_0 & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0. \tag{2.73}$$

Assim, u, v e t podem ser encontrados simplesmente resolvendo esse sistema linear. As transformações aplicadas ao triângulo e ao raio ao transladar o primeiro para a origem e ao transformá-lo em um triângulo unitário nas coordenadas y e z e ao segundo ao alinhá-lo com o eixo x são ilustradas na figura 2.13, onde $M = \begin{bmatrix} -D & V_1 - V_0 & V_2 - V_0 \end{bmatrix}$.

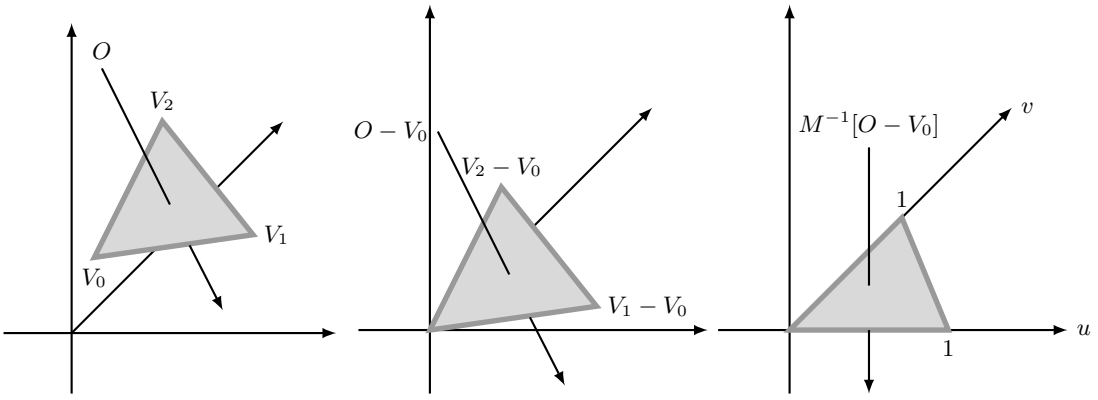


Figura 2.13: Translação e mudança de base do raio no algoritmo de Möller.

Observamos que esse algoritmo não utiliza o vetor normal do triângulo, mas somente os seus vértices V_0, V_1 e V_2 e a direção D do raio parametrizado. Denotando $E_1 = V_1 - V_0$, $E_2 = V_2 - V_0$ e $R = O - V_0$, podemos escrever a solução

da equação (2.73) utilizando a regra de Cramer:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\begin{vmatrix} -D & E_1 & E_2 \end{vmatrix}} \begin{bmatrix} \begin{vmatrix} R & E_1 & E_2 \end{vmatrix} \\ \begin{vmatrix} -D & R & E_2 \end{vmatrix} \\ \begin{vmatrix} -D & E_1 & R \end{vmatrix} \end{bmatrix} \tag{2.74}$$

Podemos reescrever o determinante $\begin{vmatrix} A & B & C \end{vmatrix} = -(A \times C) \cdot B = -(C \times B) \cdot A$. Assim, a equação (2.74) é equivalente a:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times E_2) \cdot E_2} \begin{bmatrix} (R \times E_1) \cdot E_2 \\ (D \times E_2) \cdot R \\ (R \times E_1) \cdot D \end{bmatrix} = \frac{1}{P \cdot E_1} \begin{bmatrix} Q \cdot E_2 \\ P \cdot R \\ Q \cdot D \end{bmatrix}, \tag{2.75}$$

onde $P = D \times E_2$ e $Q = R \times E_1$. Esses fatores são calculados e armazenados previamente na implementação para evitar redundâncias. Em [1] é descrito um algoritmo similar, porém utilizando o vetor normal à face. Maiores detalhes podem ser encontrados em [19].

Desse modo ficam determinados o parâmetro t e as coordenadas baricêntricas u e v do ponto T com relação ao triângulo $V_0V_1V_2$, a saber

$$t = \frac{Q \cdot E_2}{P \cdot E_1} \tag{2.76}$$

$$u = \frac{P \cdot R}{P \cdot E_1} \tag{2.77}$$

$$v = \frac{Q \cdot D}{P \cdot E_1} \tag{2.78}$$

Como as coordenadas baricêntricas do ponto são conhecidas, é fácil decidir se o ponto de intersecção pertence a alguma das arestas ou ainda se ele é algum dos vértices. O caso é idêntico ao algoritmo de Badouel (seção 2.4), e é mostrado na tabela a seguir.

Condições			Localização
$u \neq 0$	e	$v = 0$	$T(u,v) \in V_0V_1$
$u = 0$	e	$v \neq 0$	$T(u,v) \in V_0V_2$
$u, v \neq 0$	e	$u + v = 1$	$T(u,v) \in V_1V_2$
$u = 0$	e	$v = 0$	$T(u,v) = V_0$
$u = 1$	e	$v = 0$	$T(u,v) = V_1$
$u = 0$	e	$v = 1$	$T(u,v) = V_2$

Tabela 2.4: Ponto em vértice ou aresta no algoritmo de Möller.

O algoritmo segue.

ALGORITMO DE MÖLLER

ENTRADA: triângulo $V_0V_1V_2$, origem O, direção D, tolerância ε

SAÍDA: 1: O ponto interior; 0: O ponto exterior

1: $E_1 = V_1 - V_0$
2: $E_2 = V_2 - V_0$
3: $P = D \times E_2$
4: $\det = P \cdot E_2$
5: **se** $(\det > \varepsilon)$ **então**
6: $\quad R = O - V_0$
7: $\quad u = P \cdot R$
8: $\quad$ **se** $((u < 0)$ OU $(u > \det))$ **então**
9: $\quad\quad$ **retorna** 0 ▷ Rejeição 2
10: $\quad$ **fim se**
11: $\quad Q = R \times E_1$
12: $\quad v = Q \cdot D$
13: $\quad$ **se** $((v < 0)$ OU $(u + v > \det))$ **então**
14: $\quad\quad$ **retorna** 0 ▷ Rejeição 3

15: **fim se**

16: **senão se** $(\det < -\varepsilon)$ **então**

17: $R = O - V_0$

18: $u = P \cdot R$

19: **se** $((u > 0)$ OU $(u < \det))$ **então**

20: **retorna** 0 ▷ Rejeição 2

21: **fim se**

22: $Q = R \times E_1$

23: $v = Q \cdot D$

24: **se** $((v > 0)$ OU $(u + v < \det))$ **então**

25: **retorna** 0 ▷ Rejeição 3

26: **fim se**

27: **senão**

28: **retorna** 0 ▷ Rejeição 1

29: **fim se**

30: $\text{inv\_det} = 1/\det$

31: $t = (Q \cdot E_2) * \text{inv\_det}$

32: **se** $((t < 0)$ OU $(t > 1))$ **então**

33: **retorna** 0 ▷ Rejeição 4

34: **fim se**

35: $\alpha = u * \text{inv\_det}$

36: $\beta = v * \text{inv\_det}$

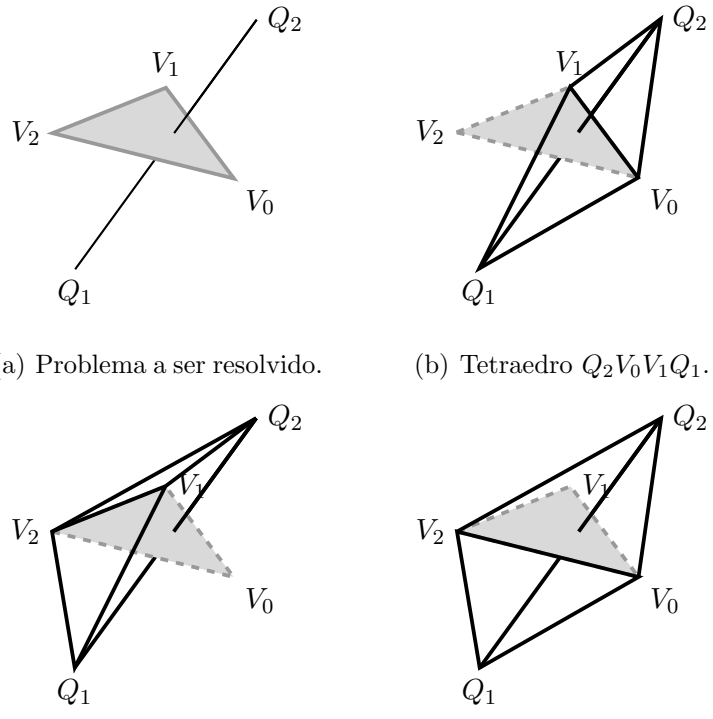
37: **retorna** 1 ▷ Intersecção ocorre e ponto é interior ao triângulo

2.6 Algoritmo de Segura

Esse algoritmo é baseado no estudo do volume com sinal de diferentes tetraedros formados com os vértices do triângulo e dois pontos pertencentes ao raio

teste $R(t) = O + tD$. Sejam V_0, V_1 e V_2 os vértices de um triângulo $V_0V_1V_2$ em $\mathbb{R}^3$ e Q_1 e Q_2 as extremidades do segmento $Q_1Q_2 \in R(t)$, os quais estão em lados opostos em relação ao plano que contém o triângulo $V_0V_1V_2$. A orientação dos vértices é feita de tal forma que o tetraedro $Q_1V_0V_1V_2$ tenha orientação positiva (isto é, o seu volume com sinal $[Q_1V_0V_1V_2]$ seja positivo). Assim, o segmento Q_1Q_2 intercepta o triângulo $V_0V_1V_2$ se, e somente se,

$$\begin{aligned} &\text{sign } ([Q_2V_0V_1Q_1]) \geq 0 \text{ e} \\ &\text{sign } ([Q_2V_2V_1Q_1]) \geq 0 \text{ e} \\ &\text{sign } ([Q_2V_0V_2Q_1]) \geq 0. \end{aligned} \tag{2.79}$$



(a) Problema a ser resolvido.

(b) Tetraedro $Q_2V_0V_1Q_1$.

(c) Tetraedro $Q_2V_2V_1Q_1$.

(d) Tetraedro $Q_2V_0V_2Q_1$.

Figura 2.14: Interpretação do algoritmo de Segura.

O algoritmo indica se a intersecção de um segmento com um triângulo existe ou não, mas originalmente não calcula as coordenadas baricêntricas do ponto

de intersecção. Para tal, é necessário aplicar algum outro algoritmo clássico de intersecção de reta/plano. Os cálculos usados para a determinação da intersecção booleana entre o segmento e o plano são também necessários para o cálculo do sinal da expressão sign $([Q_1V_0V_1V_2]) \geq 0$, o que é equivalente a determinar se o tetraedro associado tem orientação positiva. Se o tetraedro $Q_1V_0V_1V_2$ não tem orientação positiva, então o segmento Q_1Q_2 intercepta o triângulo $V_0V_1V_2$ se, e somente se:

$$\begin{gathered} \text{sign } ([Q_2V_0V_1Q_1]) \leq 0 \text{ e} \\ \text{sign } ([Q_2V_2V_1Q_1]) \leq 0 \text{ e} \\ \text{sign } ([Q_2V_0V_2Q_1]) \leq 0 \end{gathered} \tag{2.80}$$

Mais detalhes podem ser encontrados em [15, 29].

Apesar do algortimo não produzir as coordenadas baricêntricas do ponto de intersecção em relação ao triângulo $V_0V_1V_2$, é possível mesmo assim saber se o ponto se localiza na fronteira do triângulo [29]. A tabela a seguir resume os casos possíveis.

Condições	**Localização**
sign $([Q_2V_0V_1Q_1]) = 0$ e sign $([Q_2V_0V_2Q_1])$ = sign $([Q_2V_2V_1Q_2])$	$P \in V_0V_1$
sign $([Q_2V_0V_2Q_1]) = 0$ e sign $([Q_2V_0V_1Q_1])$ = sign $([Q_2V_2V_1Q_2])$	$P \in V_0V_2$
sign $([Q_2V_2V_1Q_1]) = 0$ e sign $([Q_2V_0V_1Q_1])$ = sign $([Q_2V_0V_2Q_2])$	$P \in V_1V_2$
sign $([Q_2V_0V_1Q_1]) = 0$ e sign $([Q_2V_0V_2Q_1]) = 0$	$P = V_0$
sign $([Q_2V_0V_1Q_1]) = 0$ e sign $([Q_2V_2V_1Q_1]) = 0$	$P = V_1$
sign $([Q_2V_0V_2Q_1]) = 0$ e sign $([Q_2V_2V_1Q_1]) = 0$	$P = V_2$

Tabela 2.5: Ponto em vértice ou aresta no algoritmo de Segura.

O pseudo-código do algoritmo segue.

Algoritmo de Segura

ENTRADA: triângulo $V_0V_1V_2$, segmento Q_1Q_2, tolerância ε

SAÍDA: 1: Q_1 ponto interior; 0: Q_1 ponto exterior

1: $E_1 = V_1 - V_0$
2: $E_2 = V_2 - V_0$
3: $N = E_1 \times E_2$
4: $D = Q_1 - Q_2$
5: $\det = -N \cdot D$
6: **se** $((\det > -\varepsilon)$ E $(\det < \varepsilon))$ **então**
7: **retorna** 0 ▷ Rejeição 1
8: **fim se**
9: $T = V_0 - Q_1$
10: t_param $= (N \cdot T)/\det$
11: **se** ((t_param < 0) OU (t_param > 1) **então**
12: **retorna** 0 ▷ Rejeição 2
13: **fim se**
14: $A = V_1 - Q_2$
15: $B = V_0 - Q_2$
16: $W_1 = A \times D$
17: $s = -B \cdot W_1$
18: **se** $(\det > 0)$ **então**
19: **se** $(s < 0)$ **então**
20: **retorna** 0 ▷ Rejeição 3
21: **fim se**
22: $C = V_2 - Q_2$
23: $t = C \cdot W_1$
24: **se** $(t < 0)$ **então**
25: **retorna** 0 ▷ Rejeição 4

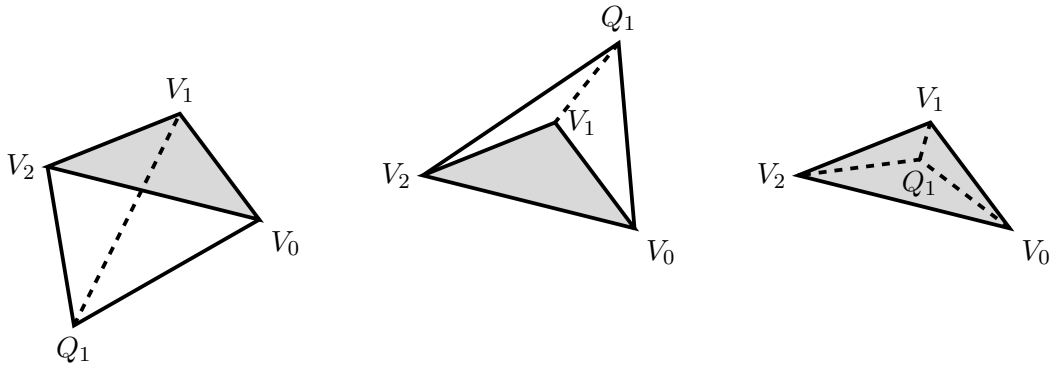
26: **fim se**
27: $W_2 = C \times D$
28: $u = B \cdot W_2$
29: **se** $(u < 0)$ **então**
30: **retorna** 0 ▷ Rejeição 5
31: **fim se**
32: **senão**
33: **se** $(s > 0)$ **então**
34: **retorna** 0 ▷ Rejeição 3
35: **fim se**
36: $C = V_2 - Q_2$
37: $t = C \cdot W_1$
38: **se** $(t > 0)$ **então**
39: **retorna** 0 ▷ Rejeição 4
40: **fim se**
41: $W_2 = C \times D$
42: $u = B \cdot W_2$
43: **se** $(u > 0)$ **então**
44: **retorna** 0 ▷ Rejeição 5
45: **fim se**
46: **fim se**
47: **retorna** 1 ▷ Intersecção ocorre e ponto é interior ao triângulo

2.7 Novo Algoritmo de Segura

Um novo algoritmo foi proposto por Segura [15] para determinar a intersecção de um segmento com um triângulo. Sua estratégia é encontrar as coordenadas

baricêntricas de um extremo do segmento Q_1Q_2, por exemplo Q_2, com respeito ao tetraedro $Q_1V_0V_1V_2$ e verificar os seus sinais.

Seja $Q_2 = \alpha Q_1 + \beta V_0 + \gamma V_1 + \delta V_2$, $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ a equação da posição de Q_2 com relação ao tetraedro $T = Q_1V_0V_1V_2$, de modo que $Q_2 = (\alpha : \beta : \gamma : \delta)$. Assim, podemos escrever $\delta = 1 - \alpha - \beta - \gamma$. Para o cálculo dessas coordenadas, usamos o volume com sinal do tetraedro $T = Q_1V_0V_1V_2$. O sinal deste volume depende da ordem em que os pontos são listados: se os triângulos do tetraedro têm orientação positiva (sentido anti-horário), o sinal do volume é positivo; caso contrário é negativo, ou ainda é zero caso os quatro pontos sejam coplanares.



(a) Positivo. (b) Negativo. (c) Nulo.

Figura 2.15: Volume com sinal do tetraedro $Q_1V_0V_1V_2$.

Dado um tetraedro $T = Q_1V_0V_1V_2$, o ponto Q_1 fica definido como vértice de origem. Um tetraedro $T = Q_1V_0V_1V_2$ tem orientação positiva se o triângulo $V_0V_1V_2$ tem orientação positiva em relação ao vértice de origem e os triângulos $Q_1V_1V_0$, $Q_1V_2V_1$ e $Q_1V_0V_2$ orientação positiva com relação aos demais vértices.

De posse desta definição de volume com sinal, as coordenadas baricêntricas de Q_2 em relação ao tetraedro $T = Q_1V_0V_1V_2$ são calculadas como:

$$\alpha = \frac{|Q_2V_0V_1V_2|}{|Q_1V_0V_1V_2|} \tag{2.81}$$

$$\beta = \frac{|Q_2Q_1V_2V_1|}{|Q_1V_0V_1V_2|} \tag{2.82}$$

$$\gamma = \frac{|Q_2Q_1V_0V_2|}{|Q_1V_0V_1V_2|} \tag{2.83}$$

$$\delta = \frac{|Q_2Q_1V_1V_0|}{|Q_1V_0V_1V_2|} \tag{2.84}$$

Sendo assim, a posição de um ponto com relação a esse tetraedro pode ser determinada através desses valores. Um ponto Q_2 é interior ao tetraedro $T = Q_1V_0V_1V_2$ se α, β, γ, $\delta \in [0, 1]$ e $\alpha + \beta + \gamma + \delta = 1$.

Ademais, essas coordenadas podem ser usadas para determinar de qual lado o ponto está localizado com relação aos planos-suporte dos triângulos das faces do tetraedro, fornecendo uma interpretação geométrica do valor de cada coordenada obtida.

Proposição 2.4. *Seja $T = ABCD$ um tetraedro e $P = (\alpha : \beta : \gamma : \delta)$ um ponto descrito através das coordenadas baricêntricas com relação a T. Três casos são possíveis:*

1. *$\alpha = 0$: o ponto está no plano-suporte do triângulo BCD.*

2. *$\alpha > 0$: o ponto está no mesmo lado que o ponto A em relação ao plano-suporte do triângulo BCD.*

3. *$\alpha < 0$: o ponto está no lado oposto ao ponto A em relação ao plano-suporte do triângulo BCD.*

Essa mesma interpretação pode ser dada para os sinais de β, γ e δ com respeito aos planos-suportes dos triângulos ADC, ADB e ACB, respectivamente.

Demonstração. Podemos mostrar este fato usando a interpretação geométrica das coordenadas baricêntricas de um ponto com respeito a um tetraedro. Seja o ponto $P = (\alpha : \beta : \gamma : \delta)$ descrito em coordenadas baricêntricas com respeito ao tetraedro $ABCD$. A coordenada α representa o volume com sinal normalizado do tetraedro $PBCD$ (figura 2.16). Um tetraedro tem volume com sinal nulo se os quatro pontos pertencem ao mesmo plano. Se o tetraedro $ABCD$ tem volume positivo, o volume de $PBCD$ também será positivo se os triângulos de $PBCD$ tiverem orientação positiva. Isso acontece se o ponto P está no mesmo lado que o ponto A em relação ao plano suporte de BCD. Se o ponto está no lado oposto, os triângulos que formam o tetraedro $PBCD$ terão orientação negativa, e assim o volume de $PBCD$ será negativo. Para um tetraedro com volume negativo, o raciocínio é o mesmo.

Analogamente, para as coordenadas β, γ e δ pode ser feita uma análise similar para os planos-suportes dos triângulos ADC, ADB e ACB, respectivamente. ■

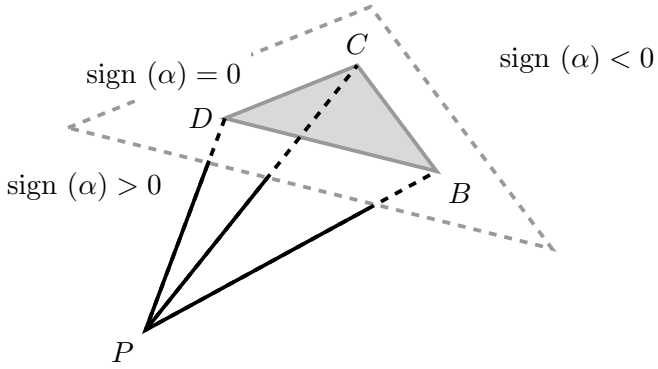


Figura 2.16: Interpretação geométrica do sinal de α.

Para determinar se há intersecção entre o segmento Q_1Q_2 e o triângulo $V_0V_1V_2$, necessitamos do seguinte teorema:

Teorema 2.4. *Seja $V_0V_1V_2$ um triângulo e Q_1Q_2 um segmento tal que Q_1 não é coplanar com o plano definido por $V_0V_1V_2$. Então, o segmento Q_1Q_2 intercepta o*

triângulo $V_0V_1V_2$ se, e somente se

$$\text{sign }(\alpha) \leq 0 \;\; e \;\; \text{sign }(\beta) \geq 0 \;\; e \;\; \text{sign }(\gamma) \geq 0 \;\; e \;\; \text{sign }(\delta) \geq 0, \tag{2.85}$$

onde $Q_2 = (\alpha : \beta : \gamma : \delta)$ em relação ao tetraedro $Q_1V_0V_1V_2$.

Mais além, o segmento Q_1Q_2 não intercecpta o triângulo $V_0V_1V_2$ se, e somente se

$$\text{sign }(\alpha) > 0 \;\text{ ou }\; \text{sign }(\beta) < 0 \;\text{ ou }\; \text{sign }(\gamma) < 0 \;\text{ ou }\; \text{sign }(\delta) < 0 \tag{2.86}$$

Podemos escrever

$$a_0 = |Q_1V_0V_1V_2| = -|V_2Q_1V_0V_1| = -\begin{vmatrix} Q_1 - V_2 & V_0 - V_2 & V_1 - V_2 \end{vmatrix} \tag{2.87}$$

$$a_1 = |Q_2V_0V_1V_1| = -|V_2Q_2V_0V_1| = -\begin{vmatrix} Q_2 - V_2 & V_0 - V_2 & V_1 - V_2 \end{vmatrix} \tag{2.88}$$

$$a_2 = |Q_2Q_1V_2V_1| = -|V_2Q_2V_1Q_1| = -\begin{vmatrix} Q_2 - V_2 & V_1 - V_2 & Q_1 - V_2 \end{vmatrix} \tag{2.89}$$

$$a_3 = |Q_2Q_1V_0V_2| = -|V_2Q_2Q_1V_0| = -\begin{vmatrix} Q_2 - V_2 & Q_1 - V_2 & V_0 - V_2 \end{vmatrix} \tag{2.90}$$

$$a_4 = |Q_2Q_1V_1V_0| = -|V_0Q_2Q_1V_1| = -\begin{vmatrix} Q_2 - V_0 & Q_1 - V_0 & V_1 - V_0 \end{vmatrix} \tag{2.91}$$

Definindo

$$\begin{aligned} A = Q_1 - V_2, \quad B = V_0 - V_2, \quad C = V_1 - V_2, \quad D = Q_2 - V_2 \\ E = Q_2 - V_0, \quad F = Q_1 - V_0, \quad G = V_1 - V_0 \end{aligned} \tag{2.92}$$

é possível obter os valores através do sistema

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \frac{1}{-\begin{vmatrix} A & B & C \end{vmatrix}} \begin{bmatrix} -\begin{vmatrix} D & B & C \end{vmatrix} \\ -\begin{vmatrix} D & C & A \end{vmatrix} \\ -\begin{vmatrix} D & A & B \end{vmatrix} \\ -\begin{vmatrix} E & F & G \end{vmatrix} \end{bmatrix} = \frac{1}{A \cdot (B \times C)} \begin{bmatrix} D \cdot (B \times C) \\ -C \cdot (D \times A) \\ B \cdot (D \times A) \\ G \cdot (E \times F) \end{bmatrix} \tag{2.93}$$

Como Q_1 não é coplanar com $V_0V_1V_2$, sabemos que $a_0 \neq 0$ e, portanto,

$$\text{sign}\left(\frac{a_i}{a_0}\right) = \frac{\text{sign}\,(a_i)}{\text{sign}\,(a_0)} = \text{sign}\,(a_i)\cdot\text{sign}\,(a_0)\,, \quad i = 1,2,3,4 \tag{2.94}$$

Assim,

$$\text{sign}\,(i)\cdot\text{sign}\,(j) = \begin{cases} \text{sign}\,(j)\,, & \text{se sign}\,(i) > 0 \\ 0, & \text{se sign}\,(i) = 0 \\ -\text{sign}\,(j)\,, & \text{se sign}\,(i) < 0 \end{cases} \tag{2.95}$$

e, portanto,

$$\begin{bmatrix} \text{sign}\,(\alpha) \\ \text{sign}\,(\beta) \\ \text{sign}\,(\gamma) \\ \text{sign}\,(\delta) \end{bmatrix} = \text{sign}\,(A\cdot(B\times C))\cdot \begin{bmatrix} \text{sign}\,(D\cdot(B\times C)) \\ \text{sign}\,(-C\cdot(D\times A)) \\ \text{sign}\,(B\cdot(D\times A)) \\ \text{sign}\,(G\cdot(E\times F)) \end{bmatrix} = \text{sign}\,(w)\cdot \begin{bmatrix} \text{sign}\,(s) \\ \text{sign}\,(t) \\ \text{sign}\,(u) \\ \text{sign}\,(v) \end{bmatrix} \tag{2.96}$$

onde

$$\begin{aligned} s = D\cdot(B\times C), \quad t = -C\cdot(D\times A), \quad u = B\cdot(D\times A) \\ v = G\cdot(E\times F), \quad w = A\cdot(B\times C) \end{aligned} \tag{2.97}$$

Além do mais, $\alpha+\beta+\gamma+\delta = 1$, ou seja,

$$\frac{s}{w} + \frac{t}{w} + \frac{u}{w} + \frac{v}{w} = 1 \tag{2.98}$$

e

$$v = w - s - t - u. \tag{2.99}$$

O parâmetro tt da distância entre a origem do segmento Q_1 e a intersecção é calculado como segue, onde N é o vetor normal de $V_0V_1V_2$:

$$tt = \frac{N\cdot(Q_1 - V_2)}{N\cdot(Q_2 - Q_1)} = \frac{-A\cdot(B\times C)}{-(B\times C)\cdot(D-A)} = \frac{A\cdot(B\times C}{(B\times C)\cdot D - (B\times C)\cdot A)} = \frac{w}{s-w} \tag{2.100}$$

Tendo o valor desse parâmetro, o algoritmo é capaz de determinar as coordenadas baricêntricas do ponto de intersecção P com respeito ao triângulo $V_0V_1V_2$. Sendo $P = (\alpha_P : \beta_P : \gamma_P : \delta_P)$ descrito em coordenadas baricêntricas com respeito ao tetraedro $Q_1V_0V_1V_2$, quando $\alpha = 0$ o ponto estará no triângulo $V_0V_1V_2$.

Dado que $Q_1 = (1 : 0 : 0 : 0)$ e $Q_2 = (\alpha : \beta : \gamma : \delta)$, temos

$$P = Q_1 + tt \cdot (Q_2 - Q_1) \tag{2.101}$$

$$\alpha_P = 1 + tt \cdot (\alpha - 1) = 0 \tag{2.102}$$

$$\beta_P = 0 + tt \cdot (\beta - 0) = tt \cdot \beta \tag{2.103}$$

$$\gamma_P = 0 + tt \cdot (\gamma - 0) = tt \cdot \gamma \tag{2.104}$$

$$\delta_P = 0 + tt \cdot (\delta - 0) = tt \cdot \delta = 1 - \beta - \delta \tag{2.105}$$

Assim, $P = (\beta_P : \gamma_P : \delta_P)$ é a descrição de P em coordenadas baricêntricas com respeito ao triângulo $V_0V_1V_2$. Para calcular os seus valores, efetuamos

$$\beta_P = tt \cdot \beta = \frac{w}{s-w} \cdot \frac{t}{w} = \frac{t}{s-w} \tag{2.106}$$

$$\gamma_P = tt \cdot \gamma = \frac{w}{s-w} \cdot \frac{u}{w} = \frac{u}{s-w} \tag{2.107}$$

Desse modo, somente é necessário calcular uma divisão $\dfrac{1}{s-w}$ e uma multiplicação para obtermos cada uma dessas coordenadas, reutilizando os cálculos já feitos. Maiores detalhes são encontrados em [15].

A seguir apresentamos o algoritmo do método.

Novo Algoritmo de Segura

ENTRADA: triângulo $V_0V_1V_2$, segmento Q_1Q_2, tolerância ε

SAÍDA: 1: Q_1 ponto interior; 0: Q_1 ponto exterior

1: $A = Q_1 - V_2$
2: $B = V_0 - V_2$
3: $C = V_1 - V_2$
4: $W_1 = B \times C$
5: $w = A \cdot W_1$
6: $D = Q_2 - V_2$
7: $s = D \cdot W_1$
8: **se** $(w > \varepsilon)$ **então**
9: **se** $(s > \varepsilon)$ **então**
10: **retorna** 0 ▷ Rejeição 2
11: **fim se**
12: $W_2 = A \times D$
13: $t = W_2 \cdot C$
14: **se** $(t < -\varepsilon)$ **então**
15: **retorna** 0 ▷ Rejeição 3
16: **fim se**
17: $u = -W_2 \cdot B$
18: **se** $(u < -\varepsilon)$ **então**
19: **retorna** 0 ▷ Rejeição 4
20: **fim se**
21: **se** $(w < s + t + u)$ **então**
22: **retorna** 0 ▷ Rejeição 5
23: **fim se**
24: **senão se** $(w < -\varepsilon)$ **então**
25: **se** $(s < -\varepsilon)$ **então**

26: **retorna** 0 ▷ Rejeição 2
27: **fim se**
28: $W_2 = A \times D$
29: $t = W_2 \cdot C$
30: **se** $(t > \varepsilon)$ **então**
31: **retorna** 0 ▷ Rejeição 3
32: **fim se**
33: $u = -W_2 \cdot B$
34: **se** $(u > \varepsilon)$ **então**
35: **retorna** 0 ▷ Rejeição 4
36: **fim se**
37: **se** $(w > s + t + u)$ **então**
38: **retorna** 0 ▷ Rejeição 5
39: **fim se**
40: **senão** ▷ $w = 0$, troca Q_1 e Q_2
41: **se** $(s > \varepsilon)$ **então**
42: $W_2 = D \times A$
43: $t = W_2 \cdot C$
44: **se** $(t < -\varepsilon)$ **então**
45: **retorna** 0 ▷ Rejeição 3
46: **fim se**
47: $u = -W_2 \cdot B$
48: **se** $(u < -\varepsilon)$ **então**
49: **retorna** 0 ▷ Rejeição 4
50: **fim se**
51: **se** $(-s < t + u)$ **então**
52: **retorna** 0 ▷ Rejeição 5
53: **fim se**
54: **senão se** $(s < -\varepsilon)$ **então**

55: $W_2 = D \times A$

56: $t = W_2 \cdot C$

57: **se** $(t > \varepsilon)$ **então**

58: **retorna** 0 ▷ Rejeição 3

59: **fim se**

60: $u = -W_2 \cdot B$

61: **se** $(u > \varepsilon)$ **então**

62: **retorna** 0 ▷ Rejeição 4

63: **fim se**

64: **se** $(-s > t + u)$ **então**

65: **retorna** 0 ▷ Rejeição 5

66: **fim se**

67: **senão**

68: **retorna** 0 ▷ Rejeição 1

69: **fim se**

70: **fim se**

71: $tt = 1/(s - w)$ ▷ t_param $= w/(s - w)$

72: $\alpha = tt * t$

73: $\beta = tt * u$

74: **retorna** 1 ▷ Intersecção ocorre e ponto é interior ao triângulo

3 IMPLEMENTAÇÃO COMPUTACIONAL

Neste capítulo são expostos detalhes das implementações dos algoritmos e da geração de pontos aleatórios em torno de um sólido, assim como são apresentados brevemente alguns tipos arquivos usados para representação de malhas discretizadas.

3.1 Discretizações e testes aleatórios

Em Geometria Computacional e aplicações, freqüentemente um sólido é representado usando estruturas de dados que são construídas a partir de informação da digitalização em 3D de sua superfície, construindo-se assim uma lista de vértices e uma lista de faces poligonais. Adicionalmente, pode ser considerada uma lista de vetores unitários normais de face, que pode ser obtida por pós-processamento a partir das duas listas básicas, ou ainda obtida diretamente durante a aquisição de dados, dependendo do hardware/software digitalizador. Por essa razão, também é objetivo deste trabalho investigar a implementação computacional usando estruturas de dados que sejam de simples gerenciamento e cujo acesso (endereçamento) seja rápido e eficaz. Todas as informações das discretizações dos sólidos utilizados durante este trabalho foram carregadas a partir de arquivos no formato STL (*STereoLithography*) e OFF (*Object File Format*).

O formato de arquivo STL tem originem em aplicativos de estereolitografia em ambientes CAD (uma documentação completa pode ser encontrada em [24]). Esse formato é amplamente utilizado em inúmerAs conseqüentes aplicações de manufatura auxiliada pelo computador (CAM), como, por exemplo projetos e impressões 3D. A superfície geométrica em três dimensões é descrita usando faces triangulares juntamente com vetores que representam direções normais às faces,

usando coordenadas cartesianas. Dois tipos de representação são permitidos: ASCII e binário. O tipo binário é mais comum sobretudo porque resulta em arquivos de tamanho menor. Neste trabalho foram construídas rotinas em linguagem C para a leitura de arquivos no formato STL, tanto em suas versões ASCII quanto binária, porém em todos os testes foram usados STLs binários.

As faces definem a superfíce de um objeto 3D. Sendo assim, cada face é parte da fronteira entre o interior e o exterior do objeto. A orientação das faces caracteriza qual lado é qual de duas maneiras reduntantes, as quais devem ser consistentes: a direção do vetor normal a cada face aponta para o exterior e os vértices são listados no sentido anti-horário quando vistos sob um ponto de referência de fora do objeto (regra da mão direita).

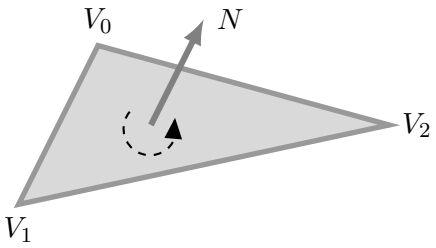
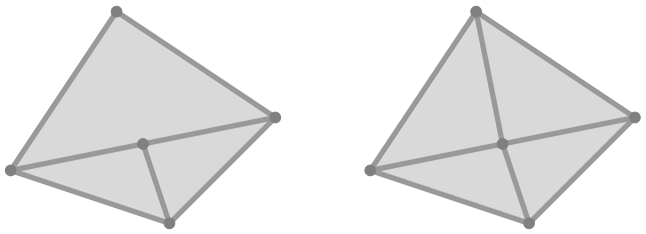


Figura 3.1: Orientação de uma face, regra da mão direita.

Além disso, cada triângulo deve ter dois vértices em comum (isto é, uma aresta) com seus adjacentes. Isso significa que um vértice de um triângulo não pode pertencer a aresta de um outro.



(a) Inválida. (b) Válida.

Figura 3.2: Configurações de triângulos adjacentes em arquivo STL.

Todo arquivo STL do tipo ASCII começa com a linha `solid nome`, onde `nome` é uma *string* opcional. No entanto, se esse parâmetro for omitido, ainda é necessário haver um espaço depois de `solid`. O arquivo continua com um número arbitrário de faces, sendo listados os seus vértices junto com o vetor normal a cada uma delas. A indicação de final de arquivo é a linha `endsolid nome`.

A estrutura padrão de um arquivo STL ASCII é a seguinte:

```
solid nome                        - Cabeçalho
facet normal Nx Ny Nz             - Vetor normal
    outer loop
        vertex V0x V0y V0z        - Vértice 0
        vertex V1x V1y V1z        - Vértice 1
        vertex V2x V2y V2z        - Vértice 2
    endloop
endfacet
facet normal ...
outer loop
        ...
    endloop
endfacet
...
endsolid nome                     - Final de arquivo
```

Temos que $V_i = (V_{i_x}, V_{i_y}, V_{i_z})$, $i = 0, 1, 2$, são os vértices da face e $N = (N_x, N_y, N_z)$ é o vetor normal à face triangular (que deve apontar para fora do sólido e ser unitário). Os números são representados em ponto flutuante no formato mantissa-exponente e são todos não-negativos.

Embora a estrutura do arquivo permita o uso de outras possibilidades de discretização, como faces com mais de um *loop* ou então *loops* com mais de

três vértices, na prática a discretização com faces triangulares é utilizada na grande maioria das aplicações.

Como arquivos STL ASCII podem se tornar grandes demais conforme o número de vértices e de faces crescem na discretização do objeto, um tipo binário de STL foi criado. STLs binários têm um cabeçalho composto por uma *string* de 80 caracteres, ignorada na maioria dos casos, mas que nunca deve iniciar com "solid", o que pode fazer com que os softwares de leitura o considere como sendo um arquivo do tipo ASCII. Após o cabeçalho, há um inteiro sem sinal de 4 bytes que indica o número de faces do arquivo. Em seguida, vem a descrição de cada face, uma após a outra, e o arquivo acaba após a última face (triângulo).

A estrutura padrão de um STL binário é a seguinte:

```
UINT8[80]                - Cabeçalho
UINT32                   - Número de triângulos

(para cada triângulo)
REAL32[3]                - Vetor Normal
REAL32[3]                - Vértice 0
REAL32[3]                - Vértice 1
REAL32[3]                - Vértice 2
UINT16                   - Atributo de contagem de bytes
(fim)
```

Cada triângulo é especificado por 12 números em aritmética de ponto flutuante segundo o padrão IEEE754, cada um com precisão simples (32 bits): 3 números são para o vetor normal da face e os demais para as coordenadas de cada um dos três vértices da face triangular. Ainda, após isso, há um inteiro sem sinal que ocupa 2 bytes, que no formato padrão é igual a zero e é desconsiderado pela maioria dos softwares de leitura.

Arquivos no formato OFF são também usados para representar modelos geométricos em 3D. Todo arquivo OFF é do tipo ASCII e contém coleções de polígonos planares com possíveis vértices comuns, e começa sempre com a *string* `OFF`. Na próxima linha, há três inteiros que indicam o número de vértices, o número de faces e o número de arestas, respectivamente. Os softwares de leitura ignoram o atributo de número de arestas, mas sua presença é necessária. Em geral, é inicializado como zero.

A estrutura padrão de um arquivo OFF utilizado é a seguinte:

```
OFF                          - Cabeçalho
Nv Nf Na                     - Número de vértices, faces e arestas
X[0] Y[0] Z[0]               - Vértice 0
X[1] Y[1] Z[1]               - Vértice 1
X[2] Y[2] Z[2]               - Vértice 2
     ...
X[Nv-1] Y[Nv-1] Z[Nv-1]      - Vértice Nv-1
3 V[0] V[1] V[2]             - Face 0
3 V[0] V[1] V[2]             - Face 1
       ...
3 V[0] V[1] V[2]             - Face Nf-1
```

Os vértices de cada face poligonal são listados em suas coordenadas x, y e z com números em ponto flutuante, uma face por linha. Após a listagem dos vértices segue a lista de faces, também uma por linha. Cada face é descrita por números inteiros: o primeiro indica o número de vértices que a compõe e os demais são os índices dos seus vértices, conforme aparecem na listagem anterior. As faces podem ter um número arbitrário de vértices, mas foram utilizadas somente discretizações com faces triangulares neste trabalho. Importante notar que os vértices são indexados a partir do zero (*0-based addressing*, que é compatível com implementações

nas linguagens que derivam do ANSI-C). Existem também atributos opcionais de cores para as faces e os vértices, que podem ter vários formatos, mas que não são relevantes para o presente trabalho. O arquivo simplesmente termina após a descrição da última face.

Para a visualização das discretizações dos sólidos foi utilizado o software MESHLAB, um programa de código aberto e portável para o processamento de malhas 3D. Ele se destina a ajudar no processamento de modelos estruturados típicos que surgem de digitalizações 3D, fornecendo um conjunto de ferramentas para edição, criação, limpeza, inspeção, renderização e conversão entre diversos tipos de arquivos e malhas, entre eles STL ASCII, STL binário e OFF.

As informações sobre a discretização dos sólidos foram carregadas nas rotinas utilizando estruturas do tipo *array*. A partir do arquivo OFF foram extraídas as coordenadas dos vértices, com variáveis *double*, e a disposição das faces, com variáveis *int*. Já os arquivos STL binários foram utilizados para obter as informações dos vetores normais, armazenadas em variáveis *double*.

Para estabelecer o tamanho dos *arrays*, foi declarada no programa principal uma constante TAM_F para indicar o número máximo de faces e uma outra constante $TAM\_V = 3 \cdot TAM\_F$ para indicar o número máximo de vértices. Sendo assim, o *array* de vértices `vx` foi declarado como sendo um vetor $TAM\_V \times 3$ de *double*, o *array* de faces `fx` foi declarado como um vetor $TAM\_F \times 3$ de inteiros e o *array* de normais `nx` foi declarado como um vetor de tamanho $TAM\_F \times 3$ de *double*. Em C, uma variável do tipo *int* ocupa 4 bytes e uma variável do tipo *double* ocupa 8 bytes, então as estruturas principais ocupam um total de

$$\begin{gathered} 3 \cdot 8 \cdot TAM\_V + 3 \cdot 8 \cdot TAM\_F + 3 \cdot 4 \cdot TAM\_F = \\ 24 \cdot 3 \cdot TAM\_F + 24 \cdot TAM\_F + 12 \cdot TAM\_F = \\ (72 + 24 + 12) \cdot TAM\_F = \\ 108 \cdot TAM\_F \text{ bytes} \end{gathered} \tag{3.1}$$

Deste modo, o tamanho em memória dos programas não é um problema relevante. Por exemplo, considerando a constante TAM_F setada inicialmente como 5000 na execução, temos que as estruturas principais ocuparão um tamanho de 540.000 bytes, ou ainda, aproximadamente 528 kb.

Para os testes dos algoritmos, e as subseqüentes comparações entre eles, foram gerados lotes de pontos aleatórios em localizações específicas em relação ao sólido testado: dentro, fora ou ambos. Para os três casos, primeiro sorteia-se uma face f_i, com $i = 0, \ldots, nF - 1$, sendo nF o número total de faces triangulares da malha.

Após isso, são gerados três números aleatórios $0 \leq a_1 \leq 1$, $0 \leq a_2 \leq 1$ e $0 \leq a_3 \leq 1$, sendo todos depois normalizados, produzindo os números α, β e γ, onde

$$\alpha = \frac{a_1}{a_1 + a_2 + a_3}, \quad \beta = \frac{a_2}{a_1 + a_2 + a_3}, \quad \text{e} \quad \gamma = \frac{a_3}{a_1 + a_2 + a_3}. \tag{3.2}$$

Assim, o ponto $P = \alpha V_0 + \beta V_1 + \gamma V_2$ é um ponto interior à face triangular sorteada f_i, cujos vértices são V_0, V_1 e V_2.

É também calculado para o sólido a menor distância R entre todos os seus vértices e um ponto C_G, chamado de centro geométrico do sólido, donde

$$R = \min_{i=0,\ldots,nV-1} d(V_i, C_G), \tag{3.3}$$

$$d(V_i, C_G) = \sqrt{(x_i - x_{C_G})^2 + (y_i - y_{C_G})^2 + (z_i - z_{C_G})^2}, \tag{3.4}$$

$$C_G = \left(\frac{\sum_i x_i}{nV}, \frac{\sum_i y_i}{nV}, \frac{\sum_i z_i}{nV} \right) = (x_{C_G}, y_{C_G}, z_{C_G}), \tag{3.5}$$

$V_i = (x_i, y_i, z_i)$ são os vértices na discretização do sólido, nV o número total de vértices e $i = 0, \ldots, nV - 1$.

Ainda, é tomado um número $li \leq \sigma \leq ls$, onde os limites inferiores e superiores dependem da localização desejada do ponto a ser gerado. Seja N_i o vetor

normal da face sorteada f_i, temos então um ponto aleatório para teste

$$P_j = \alpha V_0 + \beta V_1 + \gamma V_2 + \sigma N_i. \tag{3.6}$$

A fim de garantir a geração de pontos interiores ao sólido a ser testado, é produzida com o auxílio do MESHLAB uma malha semelhante a testada, só que em escala menor, com 90% do tamanho original. Suas informações são carregadas de maneira semelhante a descrita anteriormente para o sólido original, utilizando estruturas do tipo *array* para os seus vértices, faces e normais, o que ocupa em memória o mesmo tamanho indicado em (3.1). Desta maneira, tomando um número $-0.01 \cdot R \leq \sigma \leq 0$ produzimos um ponto P_j tal qual descrito em (3.6) que está seguramente dentro da malha.

Para gerar pontos que estão localizados no exterior da malha a ser testada, consideramos o seu envelope convexo, obtido também com auxílio do software MESHLAB, e carregado em memória de maneira equivalente a já descrita em estruturas do tipo *array*. Sendo assim, tomamos $0.01 \leq \sigma \leq 0.01 \cdot R$, e geramos um ponto P_j como em (3.6), que está localizado fora do sólido em questão.

Para os métodos que utilizam o Teorema da Curva de Jordan nos seus testes escolhemos a direção $D = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$.

3.2 Estratégias algorítmicas

O código para o método de Badouel foi construído a partir de [2, 15]. Nos testes, foram utilizadas duas versões desse algoritmo, a primeira delas se valendo de pré-processamento do vetor normal, extraído do arquivo STL, e a segunda sem esta informação pré-calculada, portanto, compatível com um arquivo de entrada de dados do tipo OFF. Uma outra opção para acelerar os cálculos é o armazenamento do plano de projeção de cada face, o que não foi testado neste trabalho. Este algoritmo tem algumas desvantagens, entre elas a quantidade de cálculos a serem feitos para

os testes de rejeição e o grande número de divisões necessárias, segundo [15], o que afeta a sua robustez (eficácia).

A implementação do algoritmo de Möller foi baseada no código presente em [15, 19]. Todas as computações são atrasadas o máximo possível até o momento em que elas sejam realmente necessárias. Por exemplo, no cálculo das coordenadas baricêntricas u e v, o valor de v somente é calculado quando se tem certeza que o valor de u é válido para um ponto interior ao triângulo. Há possibilidade de melhoramentos, como o pré-processamento dos vetores E_1 e E_2. Na tentativa de eliminar instabilidade numérica, os testes que descartam raios paralelos comparam o determinante com um pequeno intervalo em torno de zero, utilizando uma tolerância que pode ser modificada de acordo com os propósitos, e que foi usada em todos os algoritmos testados. Nos testes realizados foi utilizado um limiar de precisão simples ($\varepsilon = 10^{-7}$).

Para os algoritmos de Segura, os códigos foram baseados em [15]. Ambos os algoritmos se utilizam de cálculos de volumes com sinal de tetraedros formados com o ponto testado e uma face. Há possibilidade de melhoramentos, como o aproveitamento de informações já calculadas para tetraedros adjacentes, o que requer uma mudança nas estruturas de armazenamento das informações das faces triangulares e não foi feito neste trabalho. Os dois métodos testam se um segmento de reta limitado intercepta ou não uma face triangular, e para isto é necessário que se obtenha um ponto exterior à malha para a determinação de tal segmento. Este ponto foi obtido a partir do seguinte procedimento: um número S, que indica a maior distância a partir do centro geométrico C_G (3.5) aos vértices do sólido foi calculado, sendo

$$S = \max_{i=0,\ldots,nV-1} d(V_i, C_G). \tag{3.7}$$

Após, uma direção $D_2 = \begin{bmatrix} 0 & 0 & 10S \end{bmatrix}$ foi calculada, de modo que o ponto

$$Q_2 = C_G + D_2 \tag{3.8}$$

estará localizado fora do sólido.

Todos esses métodos têm uma característica em comum: para executar o teste de inclusão é necessário considerar o Teorema da Curva de Jordan (generalizado). Isso é feito pela grande maioria dos métodos que testam inclusão [21]. Esta técnica é extemamente popular por causa da sua simplicidade e do seu amplo campo de aplicação. Assim, em todos esses métodos utilizados no trabalho as faces do sólido são percorridas de forma seqüencial, uma a uma, e o número de intersecções do raio (ou segmento) com as faces é contado: se ele for ímpar, o ponto está localizado no interior do sólido; se for par, o ponto está localizado no exterior do sólido.

Há um problema com esse tipo de teste quando casos especiais são detectados, que ocorrem quando o raio (ou segmento) intercepta uma face em uma de suas arestas ou em um de seus vértices [20]. Para contornar esse empecilho, foi estabelecida uma *flag* na implementação dos algoritmos para indicar a ocorrência de cruzamentos na fronteira, e caso ela seja identificada, perturba-se a direção testada e o teste é refeito. Caso ainda haja detecção de casos especiais, uma nova perturbação é aplicada e a partir daí aceita-se o resultado obtido. Como observado em [20], testes baseados na Curva de Jordan podem ser otimizados ao se utilizar de estruturas pré-processadas de segmentação espacial, tais como BSP ou *octrees*, no entanto esse tipo de estrutura não foi usada para os testes realizados.

A implementação do algoritmo de Feito-Torres foi bastante trabalhosa, apesar de [8] relatar que ela seja simples. Como dito em [8] as funções necessárias ao funcionamento do algoritmo são de fácil implementação, todavia elas são muito numerosas, várias delas realizando cálculos redundantes. O algoritmo foi seguido tal qual exposto em [8, 25] e algumas simplificações foram implementadas, mas com certeza o código gerado não é ótimo.

No início do algoritmo é testado se o ponto em questão é coplanar e interior a face considerada. Para isso, uma projeção da face foi feita em 2D e o algoritmo em sua versão 2D foi empregado. Rotinas que testam colinearidade, coplanaridade e que calculam sinais de face e volumes de tetraedros pela origem foram também implementadas. Foram necessários também algoritmos para o tratamento de listas ordenadas, como rotinas de inserção e de busca binária [26] implementadas utilizando estruturas *array*.

4 EXPERIMENTOS COMPUTACIONAIS

Neste capítulo são mostrados os resultados obtidos através dos testes realizados, comparando cada um dos métodos pela sua eficiência computacional (tempo de execução) e sua eficácia (número acertos). As malhas utilizadas foram obtidas através de digitalizações de pedras brutas feitas pelo Centro Tecnológico de Pedras de Soledade, RS.

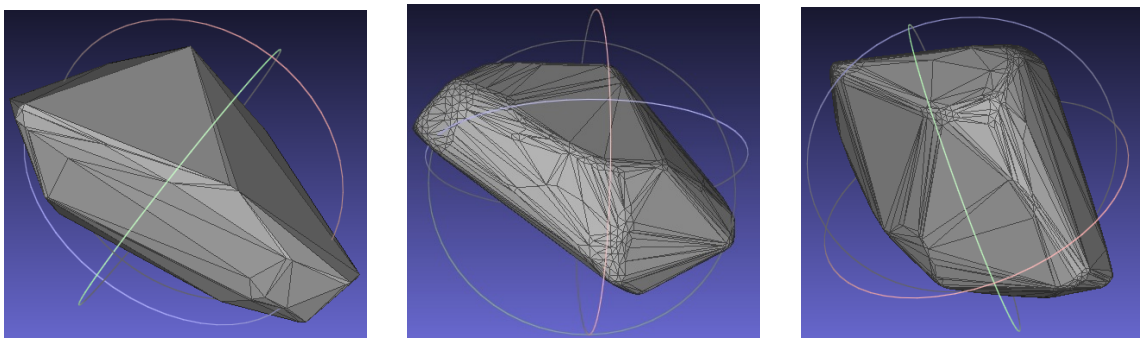
4.1 Testes de eficiência e eficácia

Três lotes de pontos foram gerados (como descrito na seção 3.1) e testados com cada um dos métodos:

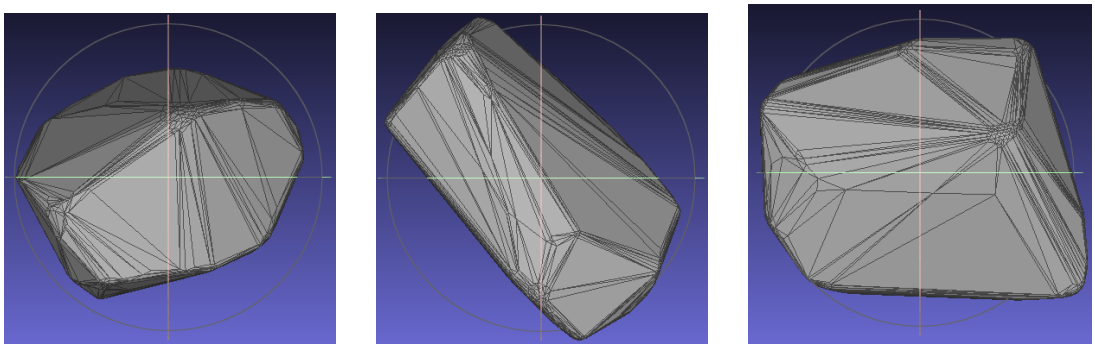
1. Pontos interiores ao sólido, sendo o tempo de execução, bem como o número de acertos, contabilizado.
2. Pontos exteriores ao sólido, sendo o tempo de execução, e também o número de acertos, contabilizado.
3. Pontos interiores e exteriores ao sólido, onde o tempo de execução e o número de pontos dentro e fora foram computados.

Um conjunto de 18 malhas de digitalização de pedras brutas foi considerado para a geração das tabelas apresentadas e então ordenado segundo o número de vértices e faces presentes nas suas representações. Ao total, seis métodos diferentes foram testados, dois deles variações do algoritmo de Badouel (um utilizando o vetor normal pré-calculado e outro não) e dois deles variações do método de Segura baseado em volumes com sinais de tetraedros. O único dos algoritmos que não é baseado no Teorema da Curva de Jordan é o método de Feito-Torres.

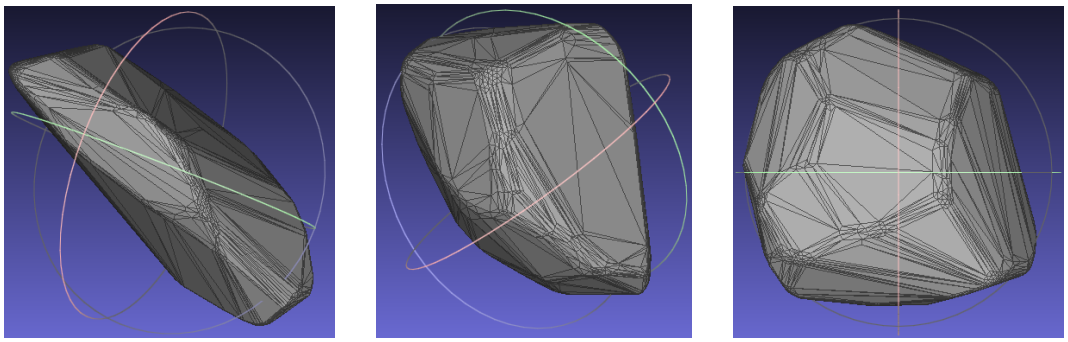
As figuras 4.1, 4.2 e 4.3 exibem as malhas testadas durante o trabalho.



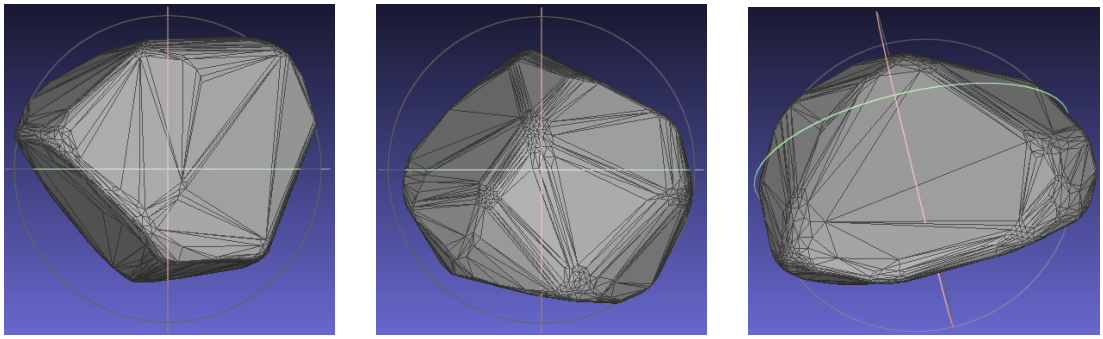
(a) Sólido 1. (b) Sólido 2. (c) Sólido 3.



(d) Sólido 4. (e) Sólido 5. (f) Sólido 6.

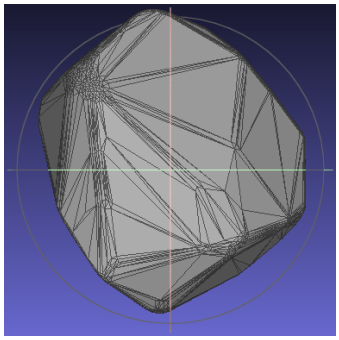


(g) Sólido 7. (h) Sólido 8. (i) Sólido 9.

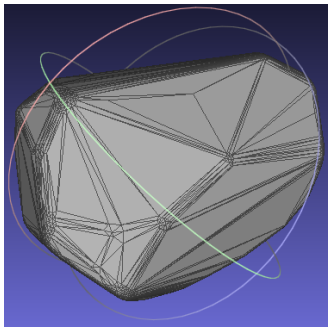


(j) Sólido 10. (k) Sólido 11. (l) Sólido 12.

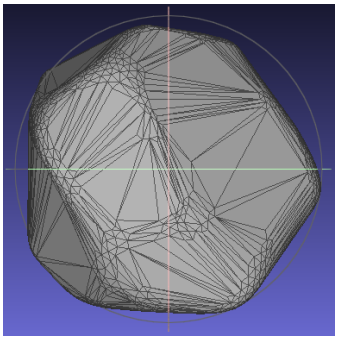
Figura 4.1: Sólidos testados.



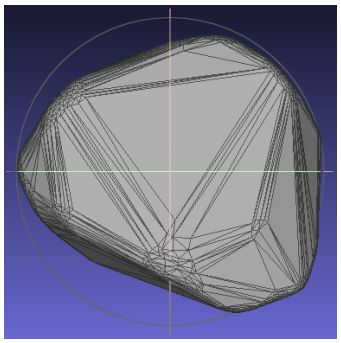
(a) Sólido 13.



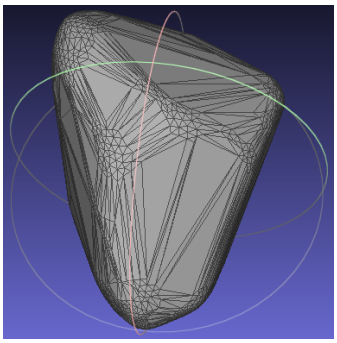
(b) Sólido 14.



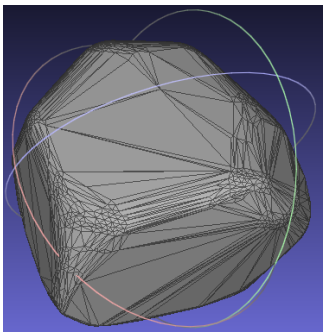
(c) Sólido 15.



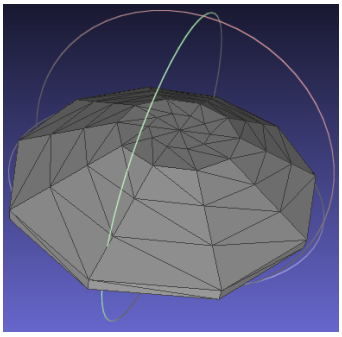
(d) Sólido 16.



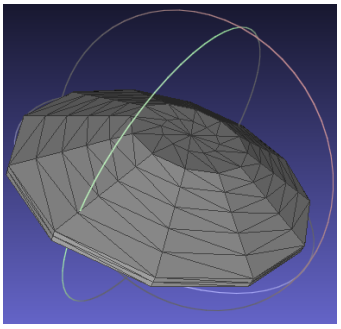
(e) Sólido 17.



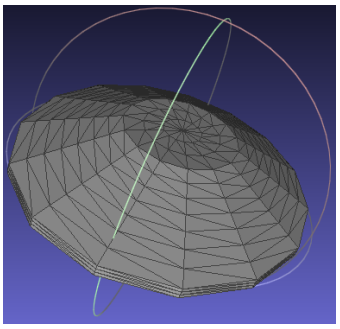
(f) Sólido 18.



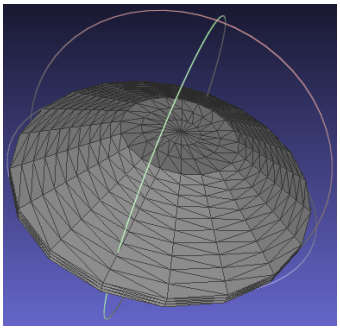
(g) Sólido regular I.



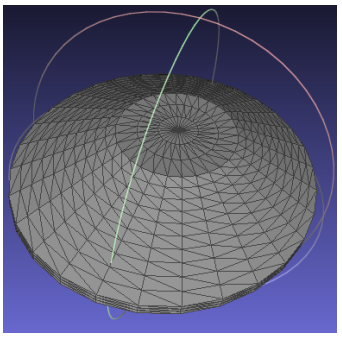
(h) Sólido regular II.



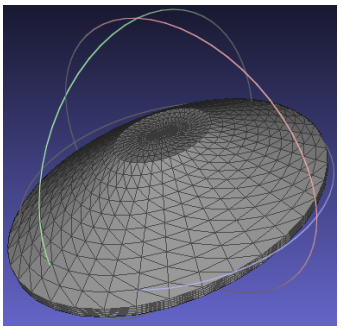
(i) Sólido regular III.



(j) Sólido regular IV.

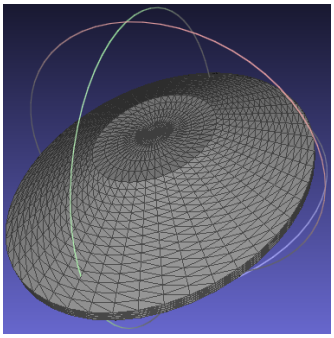


(k) Sólido regular V.



(l) Sólido regular VI.

Figura 4.2: Sólidos testados.



(a) Sólido regular VII.

Figura 4.3: Sólidos testados.

Os tempos exibidos nas tabelas 4.1, 4.2 e 4.3 a seguir são todos relativos, obtidos de uma média de 10 baterias para cada um dos testes e são normalizados a partir do tempo de execução do algoritmo de Badouel (versão que não utiliza o vetor normal pré-calculado). Os testes foram realizados utilizando um processador Intel Core i3, 2.5 GHz, 3 GB RAM, implementados em linguagem C através do compilador DevC++ (com o uso de diretivas padrão de compilação) e rodados em Windows 7, versão 64 bits. Esses tempos foram medidos utilizando a função `clock()` da biblioteca ANSI-C `time.h`.

A tabela 4.1 mostra o tempo relativo de execução dos métodos quando foram gerados aleatoriamente um total de 5000 pontos interiores ao sólido. S indica o sólido testado; V/F faz referência ao número de vértices/arestas da malha.

Na tabela 4.2 são apresentados o tempo relativo para a execução dos métodos quando gerados 5000 pontos aleatórios exteriores ao sólido.

Na tabela 4.3 estão indicados os tempos relativos de execução quando foram gerados 5000 pontos aleatórios tanto dento quanto fora do sólido em questão (teste 3). A segunda coluna da tabela designa o número de pontos dentro e fora (D/F) do sólido considerados na geração aleatória.

		MÉTODO					
		Badouel			Segura		
S	**V/F**	**s/n**	**c/n**	**Möller**	**I**	**II**	**Feito-Torres**
1	86/168	1.0000	0.8260	0.6776	0.9498	0.7584	3.1054
2	573/1142	1.0000	0.7153	0.7777	0.9502	0.8046	3.2482
3	580/1156	1.0000	0.7368	0.7964	1.0000	0.8630	3.4561
4	628/1252	1.0000	0.7001	0.7168	0.9231	0.7945	3.1534
5	741/1478	1.0000	0.7479	0.7248	0.9218	0.8109	3.1452
6	774/1544	1.0000	0.7248	0.7659	0.9627	0.8325	3.3522
7	838/1672	1.0000	0.7269	0.7850	0.9632	0.8388	3.3978
8	990/1976	1.0000	0.7282	0.7736	0.9566	0.8276	3.3527
9	1086/2168	1.0000	0.7536	0.7979	1.0096	0.8761	3.5355
10	1142/2280	1.0000	0.7183	0.7882	0.9704	0.8497	3.4049
11	1192/2380	1.0000	0.7378	0.7613	0.9749	0.8388	3.3154
12	1199/2394	1.0000	0.7461	0.7586	0.9145	0.8046	3.2163
13	1205/2406	1.0000	0.7366	0.7634	0.9797	0.8406	3.3066
14	1213/2422	1.0000	0.7238	0.7323	0.9431	0.8190	3.2097
15	1243/2482	1.0000	0.7313	0.7614	0.9708	0.8405	3.3108
16	1273/2542	1.0000	0.7230	0.7682	0.9625	0.8383	3.3395
17	1414/2824	1.0000	0.7288	0.7464	0.9577	0.8332	3.2606
18	1438/2872	1.0000	0.7213	0.7987	0.9863	0.8625	3.4826

Tabela 4.1: Teste 1: tempo relativo, 5000 pontos gerados dentro.

		MÉTODO					
		Badouel			Segura		
S	**V/F**	**s/n**	**c/n**	**Möller**	**I**	**II**	**Feito-Torres**
1	86/168	1.0000	0.9011	0.7773	0.9771	0.8637	3.3049
2	573/1142	1.0000	0.7147	0.7599	0.9311	0.8015	3.2235
3	580/1156	1.0000	0.7675	0.8015	1.0093	0.8715	3.5179
4	628/1252	1.0000	0.7167	0.7171	0.9067	0.7860	3.0910
5	741/1478	1.0000	0.7404	0.7398	0.9394	0.8156	3.2406
6	774/1544	1.0000	0.7143	0.7659	0.9443	0.8238	3.2998
7	838/1672	1.0000	0.7449	0.7796	0.9674	0.8413	3.3839
8	990/1976	1.0000	0.7437	0.7689	0.9575	0.8345	3.3716
9	1086/2168	1.0000	0.7333	0.7497	0.9457	0.8188	3.2462
10	1142/2280	1.0000	0.7353	0.7807	0.9597	0.8488	3.3820
11	1192/2380	1.0000	0.7267	0.7530	0.9595	0.8287	3.2881
12	1199/2394	1.0000	0.7280	0.7730	0.9537	0.8399	3.3538
13	1205/2406	1.0000	0.7278	0.7481	0.9535	0.8259	3.2619
14	1213/2422	1.0000	0.7358	0.7649	0.9858	0.8489	3.3227
15	1243/2482	1.0000	0.6664	0.6914	0.8742	0.7598	2.9930
16	1273/2542	1.0000	0.7172	0.7559	0.9467	0.8251	3.2703
17	1414/2824	1.0000	0.7266	0.7419	0.9445	0.8230	3.2244
18	1438/2872	1.0000	0.7082	0.7735	0.9490	0.8314	3.3464

Tabela 4.2: Teste 2: tempo relativo, 5000 pontos gerados fora.

		MÉTODO					
		Badouel			Segura		
S	**D/F**	**s/n**	**c/n**	**Möller**	**I**	**II**	**Feito-Torres**
1	2497/2503	1.0000	0.8265	0.7433	0.9333	0.8552	2.8973
2	2536/2464	1.0000	0.6823	0.6990	0.8842	0.7641	3.0357
3	2505/2495	1.0000	0.7239	0.7720	1.0021	0.8632	3.4243
4	2449/2551	1.0000	0.7142	0.7116	0.9259	0.7957	3.1375
5	2507/2493	1.0000	0.7271	0.7493	0.9342	0.8114	3.2360
6	2536/2464	1.0000	0.7157	0.7676	0.9536	0.8348	3.3268
7	2441/2559	1.0000	0.7260	0.7716	0.9552	0.8330	3.3605
8	2497/2503	1.0000	0.7439	0.7661	0.9550	0.8397	3.3868
9	2486/2514	1.0000	0.7335	0.7660	0.9683	0.8447	3.3360
10	2424/2576	1.0000	0.7247	0.7808	0.9632	0.8486	3.3816
11	2423/2577	1.0000	0.7301	0.7307	0.9273	0.8021	3.1884
12	2489/2511	1.0000	0.7330	0.7713	0.9531	0.8357	3.3593
13	2507/2493	1.0000	0.7332	0.7534	0.9684	0.8428	3.2865
14	2480/2520	1.0000	0.7383	0.7578	0.9715	0.8442	3.3099
15	2537/2463	1.0000	0.7323	0.7584	0.9674	0.8376	3.2923
16	2516/2484	1.0000	0.7232	0.7658	0.9578	0.8358	3.3198
17	2483/2517	1.0000	0.7326	0.7577	0.9662	0.8385	3.3021
18	2522/2478	1.0000	0.7313	0.7987	0.9748	0.8607	3.4599

Tabela 4.3: Teste 3: tempo relativo, 5000 pontos gerados dentro/fora.

Também foram feitos testes em uma malha regular, que simula um modelo de lapidação (na sua forma calibrada, isto é, ainda não facetada) para uma pedra. A geometria do modelo é a mesma em todos os sólidos considerados, variando apenas e número de vértices e faces utilizados para a sua representação. Os resultados para os tempos relativos nos testes 1, 2 e 3 são mostrados nas tabelas 4.4,

4.5 e 4.6, respectivamente. A segunda coluna da tabela 4.6 faz referência ao número de pontos gerados em cada localização (dentro ou fora).

		MÉTODO					
		Badouel			**Segura**		
S	**V/F**	**s/n**	**c/n**	**Möller**	**I**	**II**	**Feito-Torres**
I	102/200	1.0000	0.9011	0.7868	1.0177	0.8963	3.6761
II	182/360	1.0000	0.7159	0.7871	1.0048	0.8700	3.6934
III	332/660	1.0000	0.7105	0.7688	0.9495	0.8455	3.5097
IV	562/1120	1.0000	0.6988	0.7872	0.9888	0.8524	3.5493
V	842/1680	1.0000	0.7387	0.8131	1.0356	0.9165	3.8067
VI	1362/2720	1.0000	0.7387	0.7725	0.9939	0.8957	3.7007
VII	2202/4400	1.0000	0.7384	0.8165	1.0390	0.9346	3.8322

Tabela 4.4: Teste 1 (modelo regular): tempo relativo, 5000 pontos dentro.

		MÉTODO					
		Badouel			**Segura**		
S	**V/F**	**s/n**	**c/n**	**Möller**	**I**	**II**	**Feito-Torres**
I	102/200	1.0000	0.7277	0.7379	1.0567	0.8603	3.1943
II	182/360	1.0000	0.6850	0.7086	0.8755	0.7723	3.1976
III	332/660	1.0000	0.6827	0.7681	0.9386	0.8480	3.4884
IV	562/1120	1.0000	0.6966	0.7339	0.9590	0.8193	3.4019
V	842/1680	1.0000	0.7069	0.7817	0.9715	0.8584	3.5771
VI	1362/2720	1.0000	0.7232	0.7872	0.9990	0.9012	3.6964
VII	2202/4400	1.0000	0.7316	0.8096	1.0635	0.9626	3.7869

Tabela 4.5: Teste 2 (modelo regular): tempo relativo, 5000 pontos fora.

		MÉTODO					
		Badouel			Segura		
S	**D/F**	**s/n**	**c/n**	**Möller**	**I**	**II**	**Feito-Torres**
I	2495/2505	1.0000	0.9078	0.8289	1.0206	0.8653	3.8483
II	2414/2586	1.0000	0.7314	0.7799	0.9528	0.8272	3.5243
III	2518/2482	1.0000	0.7452	0.7653	0.9727	0.8454	3.6010
IV	2543/2457	1.0000	0.7323	0.7662	0.9709	0.8500	3.5120
V	2565/2435	1.0000	0.6992	0.7605	0.9609	0.8576	3.5708
VI	2493/2507	1.0000	0.7195	0.7903	1.0091	0.8990	3.7287
VII	2521/2479	1.0000	0.7800	0.8389	1.0337	0.9337	3.8252

Tabela 4.6: Teste 3 (modelo regular): tempo relativo, 5000 pontos dentro/fora.

A seguir, são apresentadas tabelas que mostram os tempos absolutos, em segundos, para o algoritmo de Badouel sem normal nos testes com pedras brutas (tabela 4.8) e nos testes com modelos regulares (tabela 4.7).

S	**V/F**	**Teste 1**	**Teste 2**	**Teste 3**
I	102/200	0.8490	0.9880	0.8240
II	182/360	1.4610	1.6950	1.5450
III	332/660	2.8290	2.8810	2.7870
IV	562/1120	4.6620	4.8820	4.6800
V	842/1680	6.4650	6.9090	6.9240
VI	1362/2720	10.7420	10.7560	10.6320
VII	2202/4400	14.7890	14.9910	14.8200

Tabela 4.7: Tempo absoluto (em segundos), algoritmo de Badouel (regulares).

S	V/F	Teste 1	Teste 2	Teste 3
1	86/168	0.9770	0.9610	0.9740
2	573/1142	5.3390	5.3690	5.7350
3	580/1156	4.4460	4.5090	4.4460
4	628/1252	6.0490	6.1180	6.0290
5	741/1478	7.0500	6.8440	6.8560
6	774/1544	6.9360	6.9970	6.9370
7	838/1672	7.3630	7.4220	7.4390
8	990/1976	8.8510	8.8450	8.8210
9	1086/2168	9.3290	9.9800	9.7110
10	1142/2280	9.8860	9.9680	9.9590
11	1192/2380	10.6480	10.7860	11.1910
12	1199/2394	11.0930	10.6210	10.6420
13	1205/2406	10.8100	10.9950	10.9240
14	1213/2422	11.2530	10.8510	10.9220
15	1243/2482	11.1030	12.3020	11.1450
16	1273/2542	11.3050	11.5390	11.3410
17	1414/2824	12.8780	13.0010	12.7280
18	1438/2872	12.2380	12.7840	12.2880

Tabela 4.8: Tempo absoluto (em segundos), algoritmo de Badouel (pedras brutas).

4.2 Discussão dos resultados

Como mencionado em [15, 18], é difícil desenvolver um conjunto de testes que certifiquem que um dado método é mais rápido que outros, porque a performance depende de muitos fatores distintos, tais como compilador, especificações de hardware, do tipo de aplicação e de dados disponíveis e até mesmo de detalhes na

construção dos programas. Observando as tabelas obtidas nos testes, e apresentadas na sessão anterior, podemos chegar a algumas conclusões.

Para o teste 1, analisando a tabela 4.1, observamos que o algoritmo de Möller (coluna 5) é a mais eficiente das estratégias implementadas que não usam informações obtidas por pré-processamento, isto é, desconsiderando as colunas 4 e 8 na tabela. Este algoritmo obteve resultados que são 20% a 32% superiores, aproximadamente, quando comparados ao algoritmo de Badouel (sem normais, coluna 3). A performance do método de Segura (Segura I, coluna 6) é muito semelhante a de Badouel, sendo em alguns casos 8% superior e em outros ligeiramente inferior. Em [28, 29] é atestada uma superioridade desse algoritmo perante o método de Möller, o que não foi observado em nenhum momento nos testes realizados neste trabalho. Segundo [15], essa melhor performance foi obtida nos trabalhos citados porque nos seus estudos comparativos foram utilizadas informações pré-calculadas, como por exemplo, o volume dos tetraedros e, naquele contexto, nenhuma informação pré-processada foi utilizada para o método de Möller.

O novo método proposto por Segura (Segura II, coluna 7) tem resultados mais satisfatórios comparados ao método de Badouel, variando de 16% a 24%, aproximadamente, a sua melhoria para o conjunto de malhas testadas. No entanto, quando a sua performance é comparada com a do algoritmo de Möller ela é inferior. Em [15], onde este algoritmo é apresentado, foram feitos testes em que ele obtém resultados melhores que o método de Möller, no contexto de simulação de interferência entre objetos numa mesma cena, que é o principal objetivo do artigo. É dito também em [15], porém, que o método de Möller é mais rápido quando são aplicados testes de *ray tracing* e *ray casting*, similares aos feitos neste trabalho. Isso sugere investigações adicionais futuras de possíveis melhorias no método, como pré-processamento de algumas informações que possam ser armazenadas na própria estrutura dos triângulos e compartilhada entre eles, mas que não foram levadas em conta nem em [15] nem neste trabalho.

A implementação do método proposto por Feito-Torres em [8] obteve os piores resultados para o conjunto de malhas testadas neste trabalho. Em [27] foi evidenciado que esse método seria mais eficiente que qualquer método baseado no Teorema da Curva de Jordan, porém foi utilizada uma versão modificada do algoritmo, a qual não foi totalmente especificada, apenas sendo dito que estruturas de divisão espacial (*octrees*, BSP e *voxels* [23]) foram usadas. No nosso teste 1, a performance deste algoritmo ficou muito aquém de todos os outros métodos utilizados, o tempo de execução variando, aproximadamente, de 3 a 3.5 vezes mais do que o de Badouel. De acordo com [12], os testes baseados na Curva de Jordan são os mais rápidos para se testar a inclusão, sem pré-processamento, de pontos em malhas, o que foi de fato verificado no presente trabalho.

Quando o algoritmo de Badouel foi melhorado com o uso dos vetores normais às faces (obtidos por pré-processamento), os seus resultados foram na grande maioria das vezes superiores, inclusive ao método de Möller, principalmente quando o número de faces do sólido testado foi aumentando. Segundo [15], isso realmente se verifica quando há informações adicionais pré-calculadas, como os vetores normais ou então os planos de projeção das faces. Comparando sua performance a do caso sem pré-processamento, os resultados variaram de 10% a 33% de ganho de tempo, aproximadamente.

Todos os métodos são eficazes para os testes realizados, sendo a sua confiabilidade comprovada, pois nenhum ponto interno foi perdido no teste 1, realizado com o conjunto de malhas apresentado.

Nos testes 2 e 3 aplicados a pedras brutas, assim como nos testes aplicados ao modelo de lapidação usado, sob diferentes graus de detalhamento (número de faces, vértices, etc), conclusões similares foram obtidas, reforçando as já expostas. Observando as tabelas 4.2 e 4.3, percebemos que o percentual de melhoria do método de Badouel (com normal, coluna 4) variou de 10% a 33% no teste 2 e de 13% a 31% no teste 3, aproximadamente; para o método de Möller (coluna 5), a melhoria

foi de cerca de 20% a 30% para ambos os testes; o método de Segura (coluna 6) obteve desempenhos semelhantes, sendo aproximadamente até 13% mais eficiente no teste 2 e até 11% mais eficiente no teste 3, e em certos casos ligeiramente mais lento; a comparação com o outro método de Segura (coluna 7) mostra que este foi por volta de 13% a 24% mais rápido nos testes; já o algoritmo de Feito-Torres atingiu resultados que foram cerca de 3 a 3.5 vezes mais lentos para o teste 2 e de 2.9 a 3.5 vezes mais lentos no teste 3. A eficácia observada também foi de 100% para todos os métodos testados nessas baterias.

Nos modelos regulares (tabelas 4.4, 4.5 e 4.6), observamos uma diferença para melhor no método de Badouel com normal (coluna 4) que varia em torno de 10% a 30% no teste 1, de 27% a 32% no teste 2 e de 9% a 30% no teste 3; o método de Möller (coluna 5) conseguiu melhorias de aproximadamente 18% a 33% para o teste 1, de 19% a 29% para o teste 2 e de 18% a 24% para o teste 3; nos testes feitos com o algoritmo de Segura (coluna 6), o desempenho girou em torno de 4% inferior a 5% superior no primeiro teste, de 6% inferior a 12% superior no segundo teste e de 3% inferior a 5% superir no terceiro teste; o novo algoritmo de Segura (coluna 7) obteve resultados que variaram para melhor de 6% a 15% no teste 1, de 4% a 22% no teste 2 e de 7% a 15% no teste 3, aproximadamente; o algoritmo de Feito-Torres (coluna 8) obteve desempenhos piores, na ordem de 2.5 a 2.8 vezes no teste 1, de 2.2 a 2.8 vezes no teste 2 e de 2.5 a 2.85 vezes no teste 3. A eficácia observada nos testes foi de 100% em todos os casos, assim como nos testes anteriores.

Nas figuras 4.4 e 4.5 são mostrados os percentuais médios de comparação entre os métodos que utilizam o Teorema da Curva de Jordan, tendo como referência o algoritmo de Badouel na sua versão que não utiliza pré-processamento.

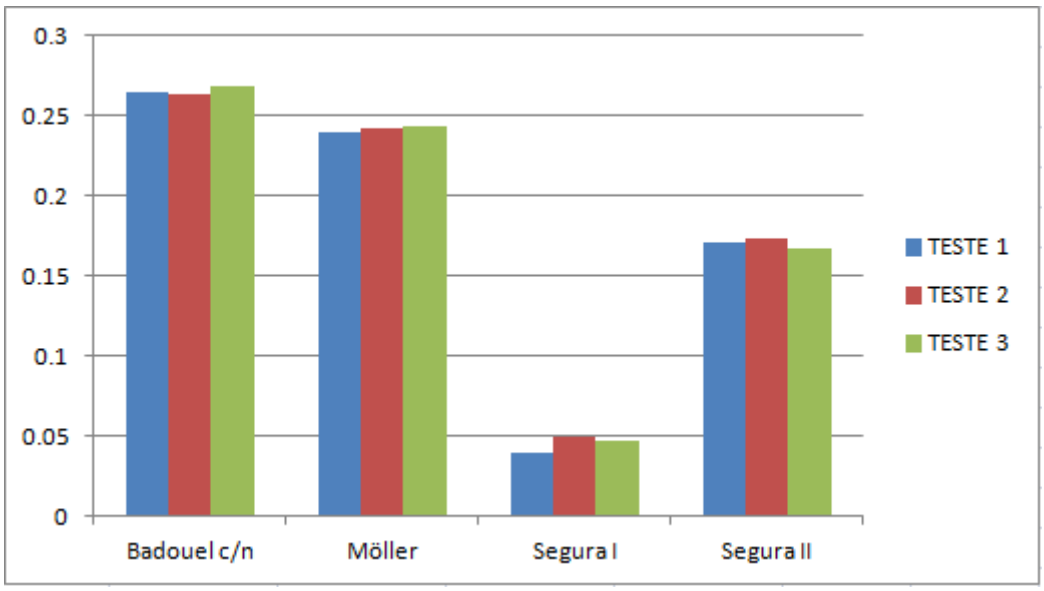


Figura 4.4: Melhora no desempenho, pedras brutas (sólidos 1 a 18).

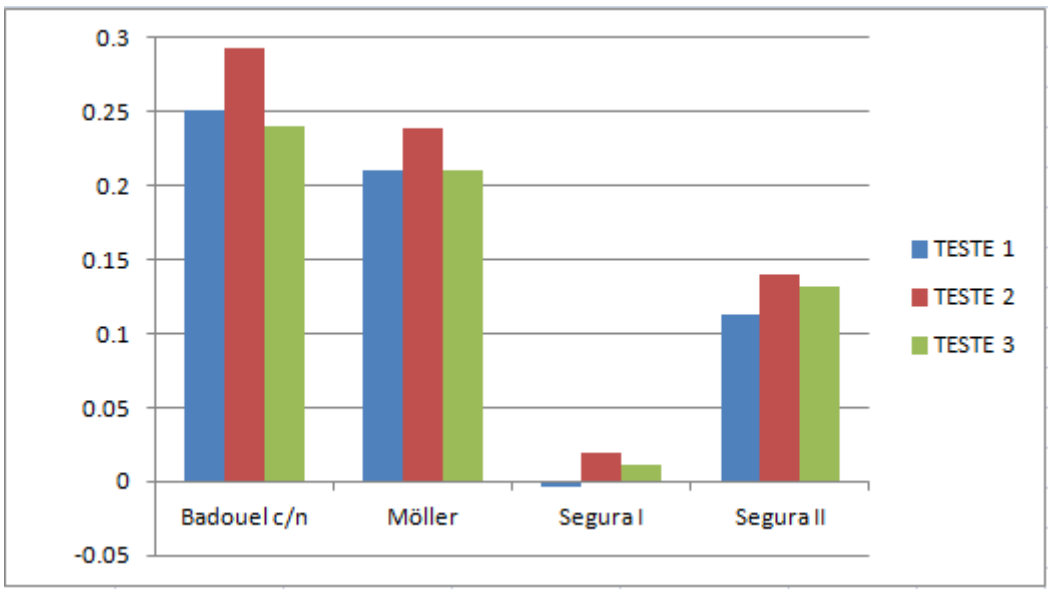


Figura 4.5: Melhora no desempenho, modelos regulares (sólidos I a VII).

Podemos verificar que as tabelas e os resultados obtidos durante a realização deste estudo confirmam que o esforço computacional é linear no número de vértices. Isso fica evidenciado nas figuras 4.6 e 4.7, que mostram gráficos do número de vértices versus o tempo decorrido, em segundos, para o teste 1.

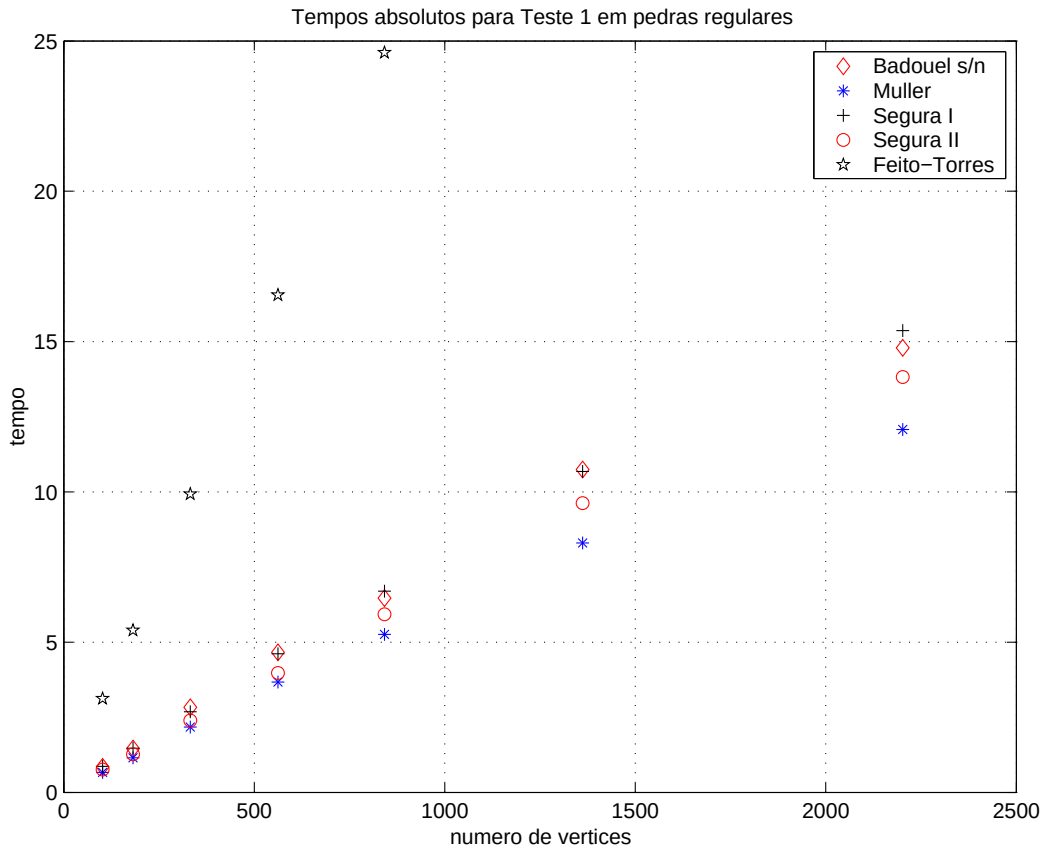


Figura 4.6: Vértices vs. tempo para o teste 1 (modelos regulares).

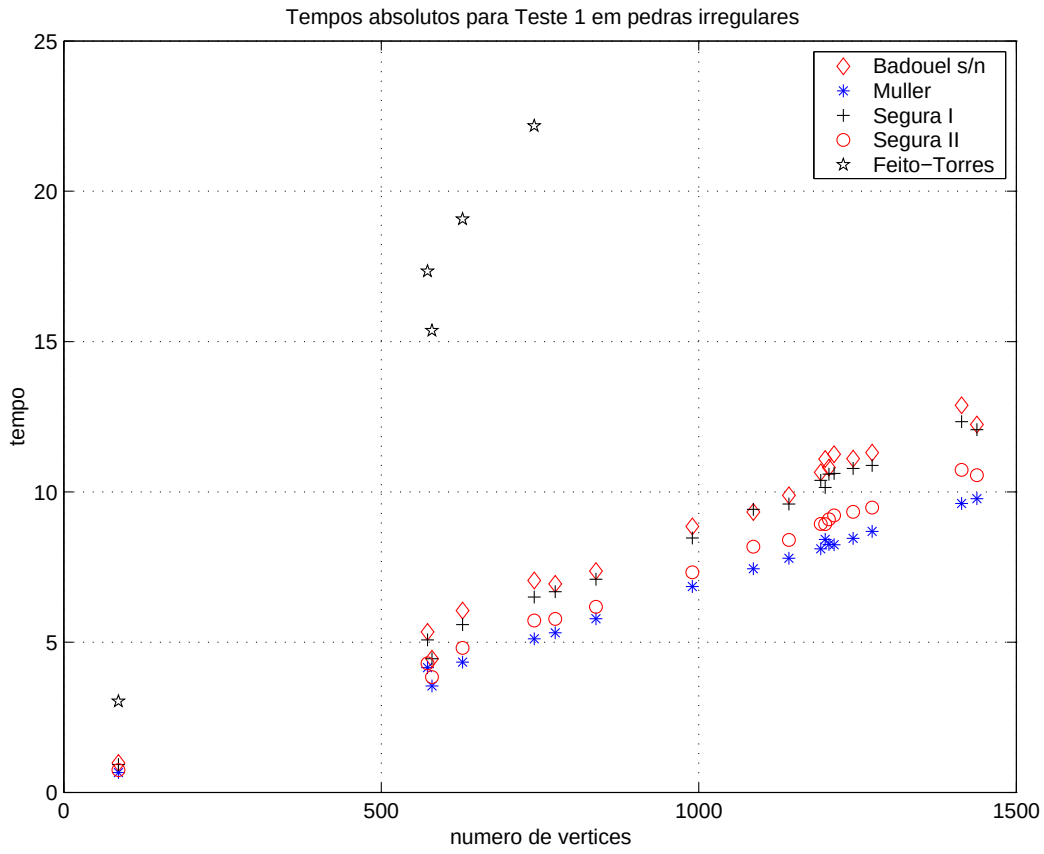


Figura 4.7: Vértices vs. tempo para o teste 1 (pedras brutas).

5 CONCLUSÕES

Com base nos resultados obtidos por este trabalho, contextualizados para inclusão/invasão entre pedras brutas e modelos regulares, fica evidente que métodos conceitualmente muito simples como os de Badouel e Möller, baseados no Teorema da Curva de Jordan, são competitivos frente a estratégias mais complexas encontradas na literatura, dentre as estratégias que não usam estruturas pré-processadas além de normais de face. Mais especificamente, se os normais exteriores das faces da digitalização não são conhecidos, por exemplo, quando somente um arquivo OFF está disponível, nosso estudo recomenda o método de Möller. Por outro lado, quando os normais exteriores das faces são conhecidos, por exemplo, quando também um arquivo STL está disponível, nosso estudo recomenda o método de Badouel melhorado. Todos os métodos implementados mostraram-se 100% eficazes nos testes realizados, que usaram estruturas de dados do tipo *array* com alocação estática de memória. Portanto, o presente trabalho fornece importante contribuição para estudos futuros envolvendo otimização do processo de lapidação de pedras brutas - preciosas, semi-preciosas ou não - como as encontradas em certas regiões de nosso estado.

O aprendizado deste trabalho sugere para estudos futuros investigação de melhorias na implementação no algoritmo de Feito-Torres, utilização de outros tipos de pré-processamento, como envelopes convexos, nos algoritmos e também o uso de estruturas de hierarquização espacial.

Referências Bibliográficas

[1] Arenberg, J. Re: Ray/triangle intersection with barycentric coordinates. *Ray Tracing News 1*, 11 (November, 4 1988). Compiled by Eric Haines.

[2] Badouel, D. An efficient ray-polygon intersection. In *Graphics Gems* (1990), Academic Press Professional, Inc., pp. 390–393.

[3] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. *Nonlinear programming: theory and algorithms*, 3 ed. John Wiley & Sons, 2013.

[4] Brusso, M. J., Silva, V., Carvalho, J., Souza, E., Lima, C., Claeyssen, J., and Beckel, C. Tecnologia 3DGemas: otimização do aproveitamento de gemas coradas digitalizadas tridimensionalmente. In *Tecnologias para o setor de gemas, joias e mineração* (2010), L. A. Hartmann and J. T. da Silva, Eds., IGEO/UFRGS, pp. 40–52.

[5] Courant, R., and Robbins, H. *What is Mathematics?: an elementary approach to ideas and methods*. Oxford University Press, 1941.

[6] Cui, S., Zhang, S., Chen, X., Pang, Z., Fu, X., and Zhang, X. Point-in-polyhedra test with direct handling of degeneracies. *Geo-Spatial Information Science 14*, 2 (2011), 91–97.

[7] Feito, F., Torres, J. C., and Urena, A. Orientation, simplicity, and inclusion test for planar polygons. *Computers & Graphics 19*, 4 (1995), 595–600.

[8] Feito, F. R., and Torres, J. C. Inclusion test for general polyhedra. *Computers & Graphics 21*, 1 (1997), 23–30.

[9] Figueiredo, J. O. d. Usando coordenadas baricêntricas para estudar a geometria do triângulo. Master's thesis, UFF, 12 2008.

[10] Glassner, A. S. *An introduction to ray tracing.* Morgan Kaufmann, 1989.

[11] Haines, E. Point in polygon strategies. *Graphics Gems IV* (1994), 24–46. Edited by Paul Heckbert.

[12] Haines, E., and Akenine-Möller, T. *Real-time rendering*, 2 ed. AK Peters, Ltd., 2002.

[13] Hilbert, D. *The Foundations of Geometry.* The Open Court Publishing Co., 2005. Translated by Townsend, E. J. – available on-line: http://www.gutenberg.org/files/17384/17384-pdf.pdf.

[14] Horn, W. P., and Taylor, D. L. A theorem to determine the spatial containment of a point in a planar polyhedron. *Computer Vision, Graphics and Image Processing 45*, 1 (1989), 106–116.

[15] Jiménez, J. J., Segura, R. J., and Feito, F. R. A robust segment/triangle intersection algorithm for interference tests. efficiency study. *Computational Geometry 43*, 5 (2010), 474–492.

[16] Lima, E. L., Carvalho, P. C. P., Wagner, E., and Morgado, A. C. *A matemática do ensino médio*, vol. 3. Rio de Janeiro: SBM, 2006.

[17] Liu, J., Chen, Y., Maisog, J. M., and Luta, G. A new point containment test algorithm based on preprocessing and determining triangles. *Computer-Aided Design 42*, 12 (2010), 1143–1150.

[18] Löfstedt, M., and Akenine-Möller, T. An evaluation framework for ray-triangle intersection algorithms. *Journal of Graphics, GPU and Game Tools 10*, 2 (2005), 13–26.

[19] Möller, T., and Trumbore, B. Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools 2*, 1 (1997), 21–28.

[20] Ogayar, C. J., Segura, R. J., and Feito, F. R. Point in solid strategies. *Computers & Graphics 29*, 4 (2005), 616–624.

[21] O'Rourke, J. *Computational geometry in C*, second ed. Cambridge University Press, 1998.

[22] Preparata, F. P., and Shamos, M. I. *Computational geometry: an introduction*, vol. 5. Springer-Verlag New York, 1985.

[23] Rogers, D. F. *Procedural elements for computer graphics*, 2 ed. McGraw-Hill, Inc., 1997.

[24] Roscoe, L., et al. Stereolithography interface specification. *America-3D Systems Inc* (1988).

[25] Rueda, A., and Ortega, L. Geometric algorithms on CUDA. *Journal of Virtual Reality and Broadcasting* (2008).

[26] Schildt, H. *C Completo e Total*, 3 ed. Makron Books, 1997.

[27] Segura, R., Feito, F. R., Ruiz de Miras, J., Ogáyar, C., and Torres, J. C. An efficient point classification algorithm for triangle meshes. *Journal of Graphics, GPU, and Game Tools 10*, 3 (2005), 27–35.

[28] Segura, R. J., and Feito, F. R. An algorithm for determining intersection segment-polygon in 3d. *Computers & Graphics 22*, 5 (1998), 587–592.

[29] Segura, R. J., and Feito, F. R. Algorithms to test ray-triangle intersection. comparative study. In *WSCG (Short Papers)* (2001), pp. 76–81.

[30] Tverberg, H. A proof of the jordan curve theorem. *Bull. London Math. Soc 12*, 1 (1980), 34–38.