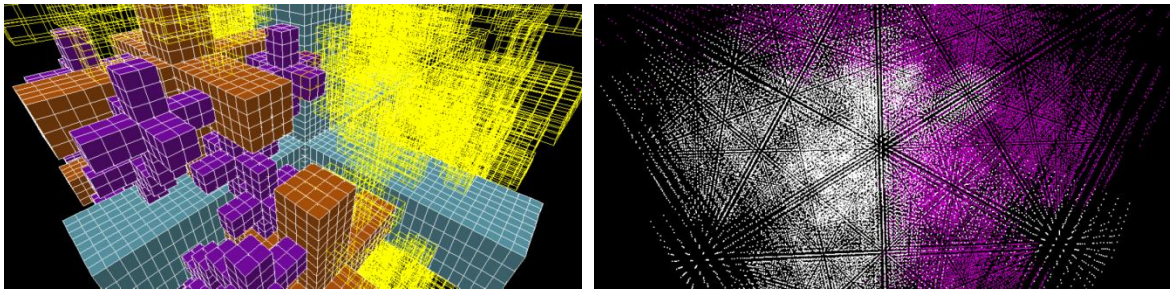
Testes de Pertinência Generalizada de Pontos em GPU

Luciano Silva

Universidade Mackenzie, Faculdade de Computação e Informática,
Laboratório de Processamento Gráfico e Mídias Digitais, Brasil



(a) Superfície recursiva (b) Pontos classificados

Figura 1: Resultados de classificação de pertinência de pontos contra uma superfície tridimensional complexa.

Resumo

O problema de pertinência de pontos em objetos matematicamente bem definidos, como curvas e superfícies, encontra-se entre os problemas mais elementares em Computação Gráfica. A especificação e implementação de predicados de pertinência corretos em critérios geométricos e numéricos, bem como eficientes, é de grande interesse computacional e possui um amplo espectro de aplicações em operações como *filling*, *stroking* e *antialiasing*. Apesar de bem posto, o problema da pertinência ainda carece de métodos que alcancem grandes classes de geometrias. Este trabalho apresenta um algoritmo de pertinência de pontos que alcança o domínio das hipersuperfícies discretas de dimensão arbitrária, possui baixa complexidade, estruturas de dados simples e, adicionalmente, realiza uma implementação em CUDA, com comparação a processadores *single core* e *multicore*.

Palavras-chave: Pertinência de Pontos, Superfícies Discretas, Geometria Discreta, Algoritmos Discretos, GPU, Geometria Computacional.

Contato do Autor:
luciano.silva@mackenzie.br

1. Introdução

Um dos objetivos centrais da computação gráfica é produzir imagens a partir de um modelo, processo denominado *rendering*. Uma classe bastante importante de algoritmos de *rendering* é denominada *algoritmos baseados em imagem*, cujos representantes mais conhecidos são os algoritmos de *filling*, *stroking* e *antialiasing*. Esta classe frequentemente necessita de uma operação bastante elementar, denominada *pertinência de pontos*, que, em linhas gerais, permite decidir se um determinado ponto está no interior de um objeto matematicamente bem definido, como curvas, superfícies ou volumes. Apesar de bem posto, o problema da pertinência de pontos encontra vários obstáculos como robustez geométrica e numérica, estabilidade e alcance a grandes classes de geometrias.

[FORREST 1985] reafirmou a necessidade de um algoritmo robusto e eficiente para pertinência de pontos. No caso bidimensional, uma resposta foi dada por [CORTHOUT and POL 1992] que implementaram um predicado com robustez geométrica e numérica garantidas *a priori*, além de alcançarem uma classe bastante geométrica bastante complexa – curvas discretas com auto-intersecção – e construirem uma implementação em *hardware* de um predicado de pertinência, através do *chip Pharos*. Poucos algoritmos de pertinência têm sido publicados, principalmente em dimensões mais elevadas.

Este artigo apresenta uma extensão da técnica de [CORTHOUT and POL 1992] para dimensões inteiras arbitrárias e alcançando uma classe bastante geométrica ainda mais complexa: hipersuperfícies discretas com autointersecção. Os algoritmos generalizados de pertinência propostos possuem baixa complexidade, utilizam estruturas de dados muito simples, qualidade que permitiu uma implementação eficiente em CUDA.

2. Hipersuperfícies Discretas

O ambiente geométrico deste trabalho é formado por objetos e hipersuperfícies discretos. A definição precisa destes elementos desempenha um papel fundamental na caracterização das noções de interior e exterior.

Complexos celulares formam a base para definição de objetos discretos. De [SPANIER 1995], um complexo celular euclideano *k*-dimensional é um objeto formado pela junção de células homeomorfas a bola fechada *k*-dimensional, chamadas de *k*-células. A partir disto, baseado em [AYALA ET AL. 1995], define-se a noção de objetos e hipersuperfícies discretos.

Definição 2.1. Um objeto discreto k-dimensional Ω é um complexo celular euclideano discreto puro k-dimensional, isto é, um complexo celular cujas 0-células têm coordenadas inteiras e toda k′-célula, $k' < k$, é bordo de de alguma k-célula. O conjunto dos objetos k-dimensionais Ω com n k-células, imersos no espaço $\mathbb{Z}_d$, é denotado por $\Omega^{k,d}{}_n$.

A Figura 2 mostra dois exemplos de objetos discretos bidimensionais e um tridimensional:

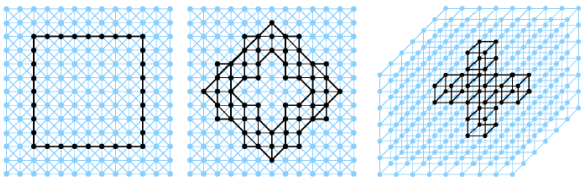


Figura 2: Exemplos de objetos discretos.

Hipersuperfícies discretas limitam o conjunto de pontos interiores a um objeto discreto e representam o principal elemento para computação de pertinência de pontos.

Definição2.2. Seja Ω um objeto discreto k-dimensional e Ω' seu sub-complexo formado por células $(k-1)$ dimensionais. O bordo combinatório de Ω, denotado por $\partial(\Omega)$, é definido pelo fecho combinatório Ω':

$$\partial(\Omega) = \mathrm{Cl}(\Omega') = \Omega' \cup \{\tau \in \Omega | \exists \tau' \in \Omega', \tau \leq \tau'\}.$$

Uma hipersuperfície discreta k-dimensional M é um subconjunto $M \subset \mathbb{Z}_{k+1}$ que é bordo combinatório de algum objeto discreto $(k+1)$-dimensional Ω imerso em $\mathbb{Z}_{k+1}$.

A Figura 3 mostra as hipersuperfícies discretas associadas aos objetos discretos da Figura 2.

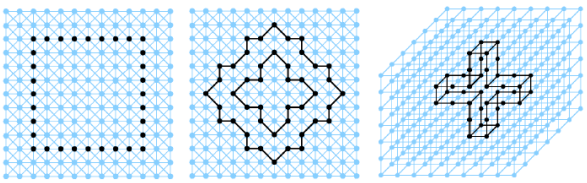


Figura 3: Exemplos de hipersuperfícies discretas.

Não é difícil observar que toda hipersuperfície *k*-dimensional *M* é também um objeto discreto *k*-dimensional.

3. Problema de Pertinência Generalizada de Pontos

O problema de pertinência generalizada de pontos pode ser posto da seguinte forma:

Dados uma hipersuperfície discreta fechada (k-1)-dimensional $M \subset \mathbb{Z}_k$ e um ponto $P \in \mathbb{Z}_k$, P pertence a alguma região interior definida por M ?

A noção de região interior é intuitiva quando *M* não tem auto-intersecção. Mesmo nesta situação, é necessário o uso de uma ferramenta matemática chamada *número de rotação (winding number)*. Existem duas maneiras de definir n´umeros de rotação: uma *versão analítica*, que pode empregar integrais complexas de linha [SPANIER 1995], formas diferenciais [GUILLEMIN e POLLACK 1974] ou a Integral de Kronecker [PRICE 1984], e uma *versão geométrica*, que conta o número de intersecções diretas com uma semi-reta [EDELSBRUNER 1987, GUIBAS et al. 1983]. Algoritmos de pertinência de pontos utilizam, normalmente, a versão geométrica, dada a sua viabilidade para implementação.

Existem poucos métodos de pertinência de pontos em Computação Gráfica. As limitações destes métodos incluem geometrias reduzidas, tratamento exaustivo de casos especiais e problemas numéricos. Alguns métodos somente caracterizam as regiões interiores e exteriores, mas não disponibilizam nenhum algoritmo para pertinência.

No ambiente bidimensional, grande parte dos métodos computa ou conta o número de intersecções entre semiretas e partes da hipersuperfície discreta. Para estes métodos, geometrias complexas representam a principal barreira e, normalmente, necessitam de um tratamento extensivo de casos especiais. [EDELSBRUNER 1987, GUIBAS et al. 1983] são exemplos bastante representativos nessa linha. [CORTHOUT e POL 1992] desenvolveram um método para geometrias dimensionais complexas, com robustez geométrica e numérica provadas *a priori* e será o ponto de partida para a estensão proposta neste artigo. Mesmo com implementação em hardware, através do *chip Pharos*, o método ainda possui uma dependência quadrática em relação à resolução.

[FABRIS et al. 1997, FABRIS et al. 1998] reduziram significantemente a complexidade de utilização do método de [CORTHOUT e POL 1992] para operações de *filling* e *stroking*. Uma linha alternativa de tratamento para o ambiente bidimensional é apresentada por [TANG 1988], que transforma a integral de linha da definição analítica de número de rotação para uma integral dupla, utilizando o Teorema de Green.

Poucos resultados têm sido publicados para dimens˜oes maiores ou iguais a 3. Grande parte destes resultados ainda somente caracterizam interiores e exteriores, não disponibilizando predicados de decisão de pertinência para as regiões. Nesta linha, [MORGENTHALER e ROSENFELD 1981] apresentaram uma versão discreta do bem-conhecido Teorema de Jordan-Bröwer e [AYALA ET AL. 1995] estenderam este resultado para o domínio das variedades discretas. Numa linha mais computável, [FEITO ET AL. 1997] desenvolveram um método para poliedros gerais, que utiliza cálculo de determinantes para decidir a pertinência de pontos. Este método inclui modelos *non-manifold* e sua extensão para geometrias mais complexas não é clara. [MORGENTHALER e ROSENFELD 1981] descrevem um complexo algoritmo para determinar as componentes do complemento de uma variedade discreta $(n-1)$-dimensional, imersa em $\mathbb{Z}_n$. Mesmo no caso tridimensional, a implementação deste método é uma tarefa bastante árdua.

A próxima seção descreve uma extensão realizada por [SILVA 2004], a partir do trabalho de [CORTHOUT e POL 1992], para espaços discretos de dimensão arbitrária e que forma a base da implementação em GPU.

3. Pertinência Generalizada

A noção de interior e exterior, para qualquer dimensão, pode ser realizada através dos números de rotação (*winding numbers*), definidos através da Integral de Kronecker:

Definição 3.1: Seja $s : A \subset \mathbb{R}^{d-1} \rightarrow \mathbb{R}^d$ uma hipersuperfície parametrizada, suave por partes e sem bordo, e $P \in \mathbb{R}^d$ um ponto não contido no traço $s(u) = (s_1(u), \ldots, s_d(u)), u \in A$. O número de rotação $w(s,P)$ de s com relação a P é dado pela fórmula:

$$(-1)^d \frac{1}{A_{d-1}(1)} \int_A \frac{\det[s(u)-P, D_1 s(u), \ldots, D_{d-1} s(u)]}{|s(u)-P|^d} du_1 \ldots du_{d-1},$$

onde $A_{d-1}(1)$ representa a área da esfera unitária $(d-1)$-dimensional.

[Silva 2004] mostrou que existe uma relação estrita entre o cálculo desta integral e a contagem de intersecções entre as faces da hipersuperfície e qualquer segmento de reta traçada a partir do ponto P que está sendo testado. Assim, para se calcular a pertinência de um ponto contra uma hipersuperfície discreta, soma-se a constribuição de cada uma destas intersecções sinalizadas (-1,0, +1):

```
WindingNumber(D)D(DiscreteHypersurface Ω, Point P)
1  w ← 0
2  for each τ_i in Ω
3  do w ← w + contribution(D)D(τ_i − P)
4  return w
```

A contribuição de cada intersecção é dada pela função abaixo:

```
contribution(D)D(Cell τ)
 1  H ← hyperplane(τ)
 2  if H ∩ {(0,...,z) : z ∈ Z} = ∅
 3    then return 0
 4    else if (0,...,0) ∈ H and (0,...,1) ∈ H
 5           then return 0
 6           else Q ← H ∩ {(0,...,z) : z ∈ Z}
 7                if Q^{v_d} ≤ 0
 8                  then τ_proj ← ProjectionX_1 ... X_{d-1}(τ)
 9                       if WindingNumber(D-1)D(τ_proj, (0,...,0)) ≠ 0
10                         then return sgn(τ · (0,...,−1))
11                         else return 0
12                  else return 0
```

A base de funcionamento desta função é reduzir, a cada passo, a dimensão do problema de pertinência através de um processo de projeção, até atingirmos a dimensão 2, onde o problema é tratado de maneira bem simples. Além disto, são feitos alguns passos de otimização para segmentos de reta paralelos ou coincidentes às faces da hipersuperfícies.

4. Pertinência Generalizada em GPU

O cálculo da contribuição de uma célula τ para a pertinência de um ponto P não depende do cálculo da contribuição de outra células. Assim, o problema de pertinência já define uma paralelização natural entre as células. Além disto, é possível se calcular a pertinência de vários pontos simultâneos, utilizando uma adaptação da técnica matricial de computação geométrica proposta por [RUEDA e ORTEGA, 2008].

Inicialmente, foi feita uma adaptação da técnica matricial para o teste de pertinência tridimensional, mostrado na figura abaixo:

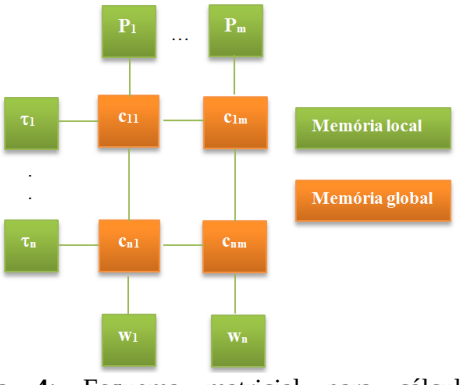


Figura 4: Esquema matricial para cálculo de pertinência tridimensional de pontos.

A implementação em CUDA 4.0 utilizou esquemas de memória local para armazenamento de pontos e, para o cálculo da contribuição cij, da célula τ_i para o ponto P_j, armazenamento em memória global. A quantidade de *threads* lançadas corresponde à dimensão da matriz. Após a computação de todas as threads, faz-se uma soma sequencial da constribuições para se encontrar o númer de rotação w_j do ponto P_j.

Para se comparar o desempenho da implementação da pertinência tridimensional em GPU com CPU, o mesmo algoritmos foi implementado em três arquiteturas diferentes: GeForce GTX 690 com CUDA 4.0, Intel i7 utilizando somente um *core* e com quatro *cores*.

Para cada uma destas arquiteturas, foram efetuados vários testes com diferentes tamanhos de superfícies. O resultado visual de classificação para uma superfície complexa de 10000 vértices é mostrada na Figura 1, no início deste artigo. A tabela abaixo mostra o desempenho para teste de apenas um ponto:

Tabela 1: Desempenhos comparativos para 1 ponto.

Tamanho da Malha	Intel 1 core	Intel 4 cores	GeForce 690
100	1	1.3	2.4
1000	8	2.6	0.02
10000	94	32	0.008

Para efeitos de comparação, tomou-se como referência absoluta o tempo gasto por 1 core no processamento de 100 células. Observe que, para malhas pequenas, é mais vantajoso o uso de CPU com 1 *core* devido aos custos de comunicação e distribuição de dados.

Foram, ainda, efetuados testes com classificação de pertinência de 100 pontos simultâneos com variação de tamanho de malha. Os resultados estão condensados na Tabela 2:

Tabela 1: Desempenhos comparativos para 100 pontos.

Tamanho da Malha	Intel 1 core	Intel 4 cores	GeForce 690
100	1	1.1	2.1
1000	8	2.2	0.01
10000	94	25	0.005

Devido à computação paralela dentro da matriz de pertinência e, mesmo com a soma sequencial das contribuições, CUDA permitiu um ganho de cerca de 200% de desempenho da classificação.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou um método generalizado para decidir pertinência de pontos em hipersuperfícies discretas de dimensão arbitrária, apresentando um esquema de implementação para o caso tridimensional em CUDA para GPU, com ganho muito significativo em performance.

Trabalhos futuros incluirão implementações em CUDA para testes de pertinência n-dimensionais, com utilização de núcleos recursivos de CUDA 5.0. Como o método atual de pertinência suporte apenas objetos poligonais no estado atual, ele será estendido para NURBS discretas de dimensão discreta arbitrária.

Referências

AYALA, R., FRANCÉS, A.R., QUINTERO, A. e RUBIO, R. (1995) On surfaces in digital topology. In *5th Colloquium on Discrete Geometry for Computer Imagery DGCI'95*, pp. 271–276.

CORTHOUT, M.E.A. e POL, E.J.D. (1992). *Point Containment and the Pharos Chip*. PhD thesis, Leiden University, the Netherlands.

EDELSBRUNER, H.(1987). *Algorithms in Combinatorial Geometry*. Springer-Verlag, Heidelberg.

FABRIS, A.E., SILVA, L. E FORREST, A.R. (1997). An efficient filling algorithm for non-simple closed curves using the point containment paradigm. In *SIBGRAPI'97 Conference Proceedings*, pp. 2–9.

_____.(1998). Stroking polynomial discrete bézier curves via point containment paradigm. In *SIBGRAPI'98 Conference Proceedings*, pp. 110–118.

FEITO, F.R. e TORRES, J.C. (1997). Inclusion test for general polyedra. *Computer and Graphics*, 21(1), pp. 23–30.

FORREST, A.R.(1985). *Computational geometry in practice*, pp. 707–724. Fundamental Algorithms for Computer Graphics. Springer–Verlag.

GUIBAS, L.J., RAMSHAW, L.H. E J. STOLFI. (1983). A kinetic framework for computational geometry. In *24th IEEE Symposium on the Foundations of ComputerScience*, pp. 100–111.

GUILLEMIN, V. E POLLACK, A. (1974). *Differential Topology*. Prentice-Hall, New Jersey, 1974.

MORGENTHALER, D.G. E ROSENFELD, A.(1981). Surfaces in three-dimensional images. *Information and Control*, 51, pp. 227–247.

PALKA. B.P. (1991) *An Introduction to Complex Function Theory*. Springer-Verlag, New York.

PRICE, G.B. (1984). *Multivariable Analysis*. Springer-Verlag, New York.

RUEDA, A., ORTEGA, L. (2008) Geometric Algorithms on CUDA. International Conference on Computer Graphics Theory and Applications. Pp 107-112.

SILVA, L. (2004). Pertinência de Pontos contra Superfícies Discretas de Dimensão Arbitrária. Tese de Doutorado, Instituto de Matemática e Estaística, USP.

SPANIER, E.H. (1995). *Algebraic Topology*. Springer-Verlag, New York.

TANG, G.Y. (1988). Region filling with the use of the discrete green theorem. *Computer Vision, Graphics and Image Processing*, 42, pp. 297–305.