MAC 747 GEOMETRIA COMPUTACIONAL

1. Introdução

A procura de algoritmos para resolver problemas geométricos vem desde a época da antiguidade. (Algumas motivações práticas para a busca por tais algoritmos foram os impostos sobre o uso da terra e as construções de edificações.) São bem conhecidas as construções geométricas de Euclides que usavam como instrumentos régua e compasso e consistiam de algumas *operações primitivas* que podiam ser realizadas com esses instrumentos. Um dos problemas algorítmicos[1] em geometria foi o chamado Problema de Apollonius (cerca de 200 A.C.) no qual três circunferências arbitrárias no plano eram dadas e pedia-se uma quarta circunferência que fosse tangente às três circunferências dadas. Euclides apresentou um algoritmo que resolve este problema.

Dentre todos os problemas algorítmicos em geometria (usando construções geométricas de Euclides) um que atraiu grande atenção foi o problema da construção de um polígono regular de n lados. Para alguns valores de n (e.g. $n = 3, 4, 5, 6$) a solução é conhecida desde a antiguidade. Entretanto, para heptágonos regulares ($n = 7$) prova-se que o problema não tem solução.[2]

Em 1902, Emile Lemoine introduziu uma medida de *simplicidade* para os algoritmos que usam as construções de Euclides. Esta medida é baseada no número de operações primitivas realizadas pelo algoritmo. Para Lemoine o algoritmo mais simples é aquele que faz menos operações primitivas. (A solução de Euclides para o Problema de Apollonius requer 508 dessas operações enquanto que um algoritmo proposto por Lemoine requer menos do que duzentas.) Estava portanto introduzido em geometria um conceito que é, pelo menos em essência, o que hoje em dia chamamos de complexidade de um algoritmo.

Em geometria computacional também estamos interessados em desenvolver algoritmos eficientes para resolvermos problemas geométricos. Pelo que foi exposto acima vemos que não é algo novo. Com a diferença de que as operações primitivas usam um instrumento diferente da régua e do compasso: usam um computador. Um pouco mais precisamente, em geometria computacional estamos interessados em encontrar algoritmos eficientes, ou

---

[1] Diremos que um problema é *algorítmico* se este problema pede como resposta um algoritmo para resolver um determinado problema. Em geometria clássica esses problemas são conhecidos como *Problemas de Construções Geométricas*.

[2] Aos 17 anos Carl Friedrich Gauss (1777-1855) mostrou que *não* existe um algoritmo que usando somente as operações primitivas de Euclides construa um heptágono regular. Na realidade Gauss mostrou mais que isso, ele mostrou que existe um algoritmo para construir um p-gon, p primo, se e somente se p é um primo da forma $2^{2^n} + 1$.

procedimentos computacionais, para resolver problemas geométricos. Muitos desses problemas tem sua origem em outras áreas como computação gráfica, robótica, computer-aided design e processamento de imagens. No desenvolvimento de tais algoritmos são comumente utilizados resultados em: geometria euclidiana, combinatória, teoria dos grafos, estruturas de dados e análise de algoritmos.

Se a tese de M.I. Shamos em 1978 (cf. [20]) for aceita como o início da geometria computacional (pelo menos da maneira como ela será tratada nesta disciplina[3]), então a área tem apenas 19 anos. Apesar disso existem pelo menos 5 livros na área, 4 jornais.

De acordo com O'Rourke [17] (página *xi*):

> " ... Not all open problems [em geometria computacional] are necessarily difficult; some are simply awaiting the requisite attention ... ".

Este pode ser um bom motivo para ficarmos de olhos abertos durante a disciplina e talvez tentar fazer alguma contribuição para a área. Divirtam-se!

[O que foi brevemente tratado nesta introdução pode ser encontrado no Capítulo 3 de [4], em [11] e no Capítulo 1 de [19].]

2. Objetivos da disciplina

O objetivo desta disciplina é apresentar técnicas, algoritmos e estruturas de dados usados para a resolução de problemas geométricos por computador. Pretendemos mostrar estratégias clássicas de solução de problemas geométricos, assim como apresentar possivelmente temas de pesquisa.

Pretendemos cobrir o "background" básico necessário para o estudo de geometria computacional. Este background incluirá resultados em geometria Euclidiana (dimensão 2 e 3), teoria dos grafos, e desenvolvimento e análise de algoritmos (i.e. contagem do número de operações primitivas feitas pelos algoritmos).

3. Critério de avaliação

A nota final na disciplina será baseada em três componentes:

Listas de exercícios: Eu gostaria de dar cerca de 10 listas de exercício durante esta disciplina, cada lista deverá ser entregue na semana seguinte. Para efeito de nota serão descartadas as piores notas de 20% das listas (uma ou duas listas). A nota em listas de exercícios será responsáveis por 40% da nota final.

Provas: Haverá duas provas nesta disciplina. A primeira estará disponível na quarta-feira dia 1 de outubro e deverá ser entregue até segunda-feira dia 6 de outubro. A segunda prova estará disponível

[3] O termo "Geometria Computacional" tem sido usado com várias conotações distintas. Por exemplo, Geometria Computacional também foi usado para se referir a modelagem geométrica através de splines e superfícies (cf. Capítulo 1 de [19]).

na quarta-feira dia 19 de novembro e deverá ser entregue até segunda-feira dia 24 de novembro. A nota de cada prova representará 20% da nota final.

Projeto: Dependendo do seu interesse, você deverá fazer um projeto de programação ou um survey da literatura sobre algum tópico.

O projeto de programação envolverá a implementação de 2 algoritmos para um problema e a comparação dos desempenhos. Junto com o programa é esperado que você entregue um relátorio (de preferência em LaTeX) de no máximo duas páginas sobre os resultados obtidos. Existem vários problemas que são candidatos para este exercício. Alguns exemplos são: fecho convexo em três dimensões, diagrama de Voronoi, triangularização de Delaunay, etc.

O survey consistirá em escrever cerca de 10 páginas (de preferência em LaTeX) dando um histórico claro sobre a pesquisa em algum tópico em geometria computacional. Seria bacana se o tópico escolhido não fosse algo já coberto por livros. Alguns exemplos para tópicos podem ser: algoritmos aleatórios em geometria computacional, algoritmos paralelos em geometria computacional, etc.

O projeto valerá 20% da nota final.

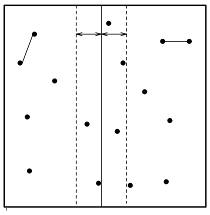
Observação. Nota final entre 8.6 e 10 corresponderá ao conceito A, entre 7.0 e 8.5 corresponderá ao conceito B, entre 5.0 e 6.9 corresponderá ao conceito C, ...

4. Tópicos que pretendemos cobrir

Alguns dos tópicos que pretendemos cobrir nesta disciplina são: fechos convexos; problemas de proximidade; partições convexas; busca geométrica; e problemas de intersecção.

Abaixo encontra-se uma breve descrição de alguns problemas que estudaremos nesta disciplina.

O Problema do par mais próximo (The closest pair problem)



Dados n pontos queremos encontrar dois cuja distância entre eles é mínima. Uma aplicação prática deste problema é em controle de tráfego aéreo: os dois aviões que estão em maior perigo de colisão são aqueles que estão mais próximos. Este problema pode ser resolvido facilmente em $O(dn^2)$ onde d é a dimensão do espaço. O problema do par mais próximo pode ser resolvido por um algoritmo do tipo divisão-e-conquista em tempo $O(dn \log n)$ (cf. Capítulo 5 de [19]) .

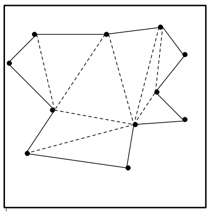
Fecho convexo de um conjunto de pontos

Dados n pontos queremos encontrar o fecho convexo desses pontos. Uma aplicações práticas deste problema se encontra em robótica. Se o fecho convexo de um robô não colide com obstáculos então o robô também não colide.

Segundo O'Rourke (cf. [17], pg. 80) talvez o primeiro artigo na área de geometria computacional tenha sido sobre fechos convexos.

Nos anos 60 uma aplicação da Bell Labs necessitava computar o fecho convexo de aproximadamente 10.000 pontos no plano e os algoritmos de complexidade de tempo $O(n^2)$ eram muito lentos. Tendo esse problema como motivação, no começo do anos 70, Graham [10] desenvolveu o primeiro algoritmo de complexidade de tempo $O(n \log n)$. O fecho convexo também pode ser construído em $O(n \log n)$ por um algoritmo de divisão-e-conquista (cf. Capítulo 3 de [PreSha]).

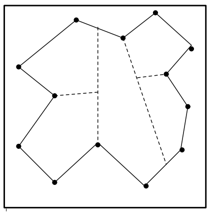
Triangularização de polígonos



Dado um polígono P queremos adicionar o maior número possível de diagonais (que não se cruzem) a P de tal forma que o interior do polígono fique particionado em triângulos. Chazelle [1] desenvolveu um algoritmo linear para este problema.

Um algoritmo para triangularizar polígonos pode ser utilizado em problemas do tipo *Art Gallery* (cf. [16]). Imagine que as salas de uma galeria de arte formem um polígono. Qual é o menor número de guardas que são necessários para tomarem conta das salas. Estamos consideramos que cada guarda fique parado em um local da galeria.

Partição de polígonos



Além de algoritmos eficientes para particionar um polígono em triângulos, também é de interesse o desenvolvimento de algoritmos que particionem um polígono em (digamos) polígonos monótonos, trapezóides e polígonos convexos. Uma motivação para particionar um polígono em polígonos convexos é reconhecimento de caracteres: um caractere pode ser representado como um polígono particionado em partes convexas.

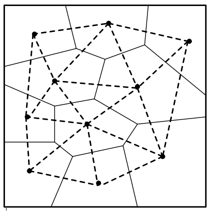
Diagramas de Voronoi

Dado um conjunto S de n pontos no plano queremos determinar para cada ponto p em S qual é a região $V(p)$ dos pontos do plano que estão mais perto de p do que de qualquer outro ponto em S. As n regiões $V(p)$ formam uma partição do plano chamada de *Diagrama de Voronoi*.

Imagine uma vasta floresta contendo vários pontos de observação de incêndio. O conjunto das árvores que estão mais próximas de um determinada posto p determina a região $V(p)$ das árvores que são de responsabilidade de ponto p.

O diagrama de Voronoi de um conjunto de n pontos pode ser contruído em $O(n \log n)$ por um (complicado) algoritmo do tipo divisão-e-conquista (cf. [21]). Em 1985, Fortune [9] desenvolveu um algoritmo de varredura (*plane-sweep algorithm*) muito elegante e simples cuja complexidade de tempo é $O(n \log n)$.

Triangularização de Delaunay



O dual geométrico (usando retas) de um diagrama de Voronoi para um conjunto S de pontos forma uma triangularização do conjunto S, chamada de *triangularização de Delaunay*. A triangularização de Delaunay tem várias propriedades geométricas interessantes, por exemplo, ela contém todas as "árvores geradoras mínimas" de S (cf. Capítulo 6 de [19]).

5. Bibliografia

Para preparar as aulas desta disciplina tenho consultado as notas de aula de Ferreira [7], Guibas [12] e Pemmaraju [18], junto com alguns livros e artigos.

O livro de Preparata & Shamos [19] é um texto clássico em geometria computacional (foi primeiro livro sobre o assunto) que coloca bastante ênfase na análise dos algoritmos apresentados. Este livro contém basicamente todos os tópicos que serão tratados nesta disciplina. Outros livros que também podem ser encontrados na biblioteca são: Edelsbrunner [6] ("The art of counting and estimating is at heart of combinatorics—and it is a necessary prerequisite for analyzing algorithms ... "; cópiado da introdução da Parte I de [6]); Figueiredo & Carvalho [8] (um livro muito claro e introdutório); e Resende & Stolfi [5] (descreve varias técnicas e algoritmos em geometria computacional). Outros livros sobre geometria computacional são: O'Roukey [17] (este livro da mais enfase ao desenvolvimento dos algoritmos e, de certa forma, menos atenção à análise); Laszlo [14] (um livro que descreve vários algoritmos em geometria computacional e apresenta trechos de implementações em `C++`); Mulmuley [15] (como o próprio título diz, este livro trata de algoritmos aleatórios em geometria computacional).

Cormen, Leiserson & Rivest [3] é um livro enciclopédico sobre análise de algoritmos que trata de geometria computacional no Capítulo 35.

Na biblioteca também podem ser encontrados alguns surveys sobre geometria computacional, veja por exemplo: Chazelle [2]; Graham & Yao [11]; Guibas & Stolfi [13]; e Yao [22].

Artigos em geometria computacional podem ser encontrados em várias revistas, incluindo *ACM Transactions on Graphics*, *Algorithmica*, *Journal of Algorithms*, *Journal of the ACM*, e *SIAM Journal on Computing*. Uma revista que é particularmente dedicada à área é *Discrete and Computational Geometry* e mais recentemente temos *International Journal of Computational Geometry & Applications* e *Computational Geometry, Theory and Applications*.

Existe uma conferência anual em geometria computacional a *ACM Annual Conference on Computational Geometry* (alguns dos proceedings podem ser encontrados na biblioteca; veja QA758.C S989). Outras conferências que também apresentam trabalhos em geometria computacional são *STOC* (QA800.C S989), *FOCS* (QA800.C S989), *SODA* (QA758.C S989), e *ICALP*. Alguns professores do Departamento de Ciência da Computação (como o prof. José Augusto, sala A-302, e-mail `jose@ime.usp.br`) podem ter alguns proceedings que a biblioteca não tem.

6. Implementações de algoritmos

As implementações que vocês verão durante as aulas foram feitas pelos estudantes Cassio Polpo de Campos (e-mail `cassio@ime.usp.br` e URL `http://www.ime.usp.br/~cassio/`) e Eduardo Garcia de Freitas (e-mail `freitas@ime.usp.br` e URL `http://www.ime.usp.br/~freitas/`). O Cassio fez suas implementações no `Turbo C` e o Eduardo fez *applets* em `Java` (todos os programas estão disponíveis nos URLs acima). Atualmente os dois estão implementando em `Java` algoritmos de triangularização.

7. Geometria computacional na Internet

Abaixo você encontra a lista de alguns sítios de geometria computacional:

- Jeff Erickson matém uma página sobre geometria computacional no URL `http://www.cs.duke.edu/~jeffe/compgeom/`. Esta página tem apontadores para um grande número de sítios sobre geometria computacional.
- Uma fonte de aplicações de geometria computacional pode ser encontrada na página mantida por David Eppstein no URL `http://www.ics.uci.edu/~eppstein/geom.html`.
- Outra página, também mantida por David Eppstein, que vale dar uma olhada esta no URL `http://www.ics.uci.edu/~eppstein/junkyard/`
- A página de Joseph O'Rouke (`http://cs.smith.edu/~orourke/`) contém apontadores para programas em `C` e para erratas do seu livro (cf. [17]).
- O URL do *Center for Geometric Computing*, que formado por Brown University, Duke University e Johns Hopkins University, é `http://www.cs.jhu.edu/labs/cgc/home.html`.
- O URL do *The Geometry Center, University of Minnesota* é `http://www.geom.umn.edu/`.

8. Outras informações

A minha sala é a B-164, o número do meu telefone é 818-6295 e meu endereço eletrônico é `coelho@ime.usp.br`. Pretendo fazer uma lista contendo os endereços eletrônicos de todos os alunos da disciplina. Assim, por favor, se você vai cursar esta disciplina mande-me um e-mail. Também sinta-se a vontade para me escrever e fazer perguntas ou comentários sobre a disciplina. Eu manterei uma página de MAC 747 no URL `http://www.ime.usp.br/~coelho/mac747/`.

Nessa página eu colocarei o material da disciplina (como, por exemplo, listas de exercícios). Por favor, de uma olhada nesta página regularmente.

Outros professores do Departamento de Ciência da Computação que estudam geometria computacional são Carlos Eduardo Ferreira (sala A-297, e-mail `cef@ime.usp.br`, e URL `http://www.ime.usp.br/~cef/`) e José Augusto Ramos Soares (sala A-297, e-mail `jose@ime.usp.br`, e URL `http://www.ime.usp.br/~jose/`).

Se você se interessa por Computação Gráfica converse com o professor Antonio Elias Fabris (sala B-112 , e-mail `aef@ime.usp.br`). Neste semestre ele estará dando a disciplina MAC 748 SÍNTESE DE IMAGENS. Agora, se você quer saber o que é Processamento de Imagens então bata um papo com o professor Junior Barreira (sala A-290 , e-mail `jb@ime.usp.br`, e URL `http://www.ime.usp.br/dcc/people/junior.html`).

REFERÊNCIAS

1. B. Chazelle, *Triangulating a simple polygon in linear time*, Discrete and Computational Geometry **6** (1991), 485–524.
2. ———, *Computational geometry: A retrospective*, Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (Montréal, Québec, Canada), The ACM Special Interest Group for Algorithms and Computation Theory, May 1994, pp. 75–94.
3. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to algorithms*, The MIT Electrical Engineering and Computer Scienece Series, The MIT Press, MacGraw-Hill Book Company, 1990, QA758 C811i.
4. R. Courant and H. Robbins, *What is mathematics?*, Oxford University Press, New York, 1941.
5. P.J. de Resende and J. Stolfi, *Fundamentos de geometria computacional*, IX Escola de Computação, 1994.
6. H. Edelsbrunner, *Algorithms in combinatorial geometry*, EATCS Monographs on Theoretical Computer Science, no. 10, Springer-Verlag, Berlin, 1987, QA758 E21a.
7. C.E. Ferreira, *MAC 747 geometria computacional*, Notas de aula, 1995.
8. L.H. Figueiredo and P.C.P. Carvalho, *Introdução à geometria computacional*, 18º Colóquio Brasileiro de Matemática, IMPA, 1991, QA758 F475i.
9. S. Fortune, *A sweepline algorithm for Voronoi diagrams*, Algorithmica **2** (1987), 153–174.
10. R.L. Graham, *An efficient algorithm for determining the convex hull of a finite planar set*, Information Processing Letters **1** (1972), 132–133.
11. R.L. Graham and F. Yao, *A whirlwind tour of computational geometry*, The American Mathematical Monthly **97** (1990), no. 8, 687–701.
12. L.J. Guibas, *CS368: Geometric algorithms*, Stanford University, 1997.
13. L.J. Guibas and J. Stolfi, *Ruler, compass and computer: The design and analysis of geometric algorithms*, Theoretical Foundations of Computer Graphics and CAD (R.A. Earnshaw, ed.), NATO ASI Series, vol. F40, Springer-Verlag, 1988, pp. 111–165.
14. M.J. Laszlo, *Computational geometry and computer graphics in C++*, Prentice Hall, Upper Saddle River, NJ, 1996.
15. K. Mulmuley, *Computational geometry: An introduction through randomized algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1994.
16. J. O'Rourke, *Art gallery theorems and algorithms*, The International Series of Monographs on Computer Science, Oxford University Press, New York, 1987, QA830 O74a.

17. ———, *Computational geometry in C*, Cambridge University Press, Cambridge, 1993.
18. S. Pemmaraju, *Computacional geometry 22C:196*, Notas de aula, Spring 1996.
19. F.P. Preparata and M.I. Shamos, *Computational geometry: An introduction*, Texts and Monographs in Computer Science, Springer-Verlag, New York, 1985, QA758 P927c.
20. M.I. Shamos, *Computacional geometry*, Ph.D. thesis, Yale University, New Haven, 1978.
21. M.I. Shamos and D. Hoey, *Closest point problems*, Proc. 16th Annual IEEE Symposium in Foundations of Computer Science, 1975, pp. 151–162.
22. F.F. Yao, *Computational geometry*, Handbook of Theoretical Computer Science (J. van Leeuwen, ed.), vol. A, The MIT Press/Elsevier, Amsterdam, 1990, QA810.C3 V259h v.1A, pp. 343–389.