Teorema da Galeria de Arte e Triangularização de Polígonos e Pontos no Plano

Aluno: Lucas Piva Rocha Corrêa
Orientador: Carlos Eduardo Ferreira

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

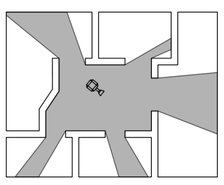
CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

1. Introdução

Uma Galeria de Arte pode ser representada como um polígono simples no plano. Uma necessidade natural que surge é a de vigiar essa galeria, usando, por exemplo, câmeras de segurança. Evidentemente, quanto menos câmeras usarmos, mais fácil e barato será essa tarefa. O problema de descobrir quantas câmeras são necessárias e suficientes para vigiar qualquer polígono de n vértces deu origem ao famoso *Teorema da Galeria de Arte*, que estudamos nesse trabalho. Do problema original, surge o estudo da triangularização de polígonos.

Extendemos o conceito de triangularização para que a nossa entrada seja um conjunto de pontos no plano, e estudamos propriedades de uma triangularização particular, a *Triangularização de Delaunay*, que possui diversas aplicações dentro de geometria computacional.

2. Teorema da Galeria de Arte



Dizemos que uma câmera enxerga um ponto se o segmento de reta que liga a câmera ao ponto está dentro do polígono. Um polígono é coberto se todo ponto é enxergado por alguma câmera.

Seja $g(P)$ o menor número de câmeras necessário para cobrir o polígono P:

$$g(P) = min_S|\{S : S \text{ cobre } P\}|$$

onde S é um conjunto de câmeras e $|S|$ é a cardinalidade de S. Seja P_n um polígono com n vértices. $G(n)$ é o máximo de $g(P_n)$ sobre todos os polígonos de n vértices:

$$G(n) = max_{P_n} g(P_n)$$

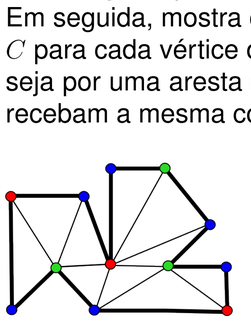
O que queremos é calcular quanto vale $G(n)$. Dizemos que esse número é necessário e suficiente: necessário no sentido de que precisamos de pelo menos esse número de câmeras para pelo menos um polígono e suficiente no sentido de que esse número é suficiente para qualquer polígono de n vértices.

Teorema *Para cobrir um polígono de n vértices, $\lfloor n/3 \rfloor$ câmeras são sempre suficientes e ocasionalmente necessárias.*

Esse resultado foi provado por Chvátal em 1975 [3]. Uma prova mais concisa foi dada por Fisk em 1978 [4].

Necessidade: A figura mostra uma classe de polígonos que necessita de $\lfloor n/3 \rfloor$ câmeras, uma para cada ponta de três vértices. Portanto $G(n) \geq \lfloor n/3 \rfloor$.

Suficiência: A prova de Fisk envolve particionar o polígono em triângulos pela adição de diagonais não intersectantes. Em seguida, mostra que podemos atribuir uma cor A, B ou C para cada vértice de forma que dois vértices adjacentes, seja por uma aresta do polígono ou por uma diagonal, não recebam a mesma cor.



Assim, três vértices de um triângulo obrigatoriamente recebem cada uma das três cores, caso contrário dois vértices adjacentes receberiam a mesma cor. Podemos notar que posicionar uma câmera em qualquer vértice de um triângulo é suficiente para cobrí-lo. Finalmente, concluímos que para cobrir todo polígono basta colocarmos câmeras em todos os vértices que receberam uma mesma cor. Como uma das cores é usada no máximo $\lfloor n/3 \rfloor$ vezes, $G(n) \leq n/3$.

3. Triangularizando Polígonos

Naturalmente, gostaríamos de desenvolver algoritmos para triangularizar polígonos. Um algoritmo simples pode ser desenvolvido usando as idéias apresentadas na seção anterior.

3.1 Algoritmo simples

Uma *orelha* é um vértice do polígono cujo segmento que une seus dois vizinhos é uma diagonal. Pré-processamos o polígono e guardamos os vértices que são orelhas em uma fila. Removemos uma orelha pela adição de uma diagonal e arrumamos a nossa fila observando que os únicos vértices que podem ter sido alterados são os vértices conectados pela diagonal adicionada. Repetimos o processo recursivamente.

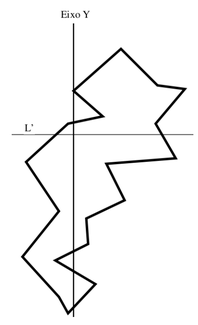
TRIANGULARIZAÇÃO ORELHA(P)

```
1  D ← ∅ ▷ Conjunto de diagonais adicionadas
2  Inicialize o status de orelha para cada vértice em P
3  enquanto n > 3
4     faça
5          Localize uma orelha v_i
6          D ← v_{i-1}v_{i+1}
7          P ← P − v
8          v_{i-1} ← ORELHA(P, v_{i-1})
9          v_{i+1} ← ORELHA(P, v_{i+1})
10 devolva D
```

O algoritmo consome tempo proporcional a $O(n^2)$.

3.2 Algoritmo mais eficiente

Primeiro precisamos definir um polígono *y-monótono*. Um polígono é *y-monótono* se para qualquer linha horizontal, a intersecção dessa linha com o polígono é conexa, ou seja, vazia, um ponto ou um segmento de reta.



Lema *Um polígono y-monótono pode ser triangularizado em tempo proporcional a $O(n)$.*

Assim, temos um interesse especial em polígonos *y-monótonos*. Se soubéssemos como particionar um polígono arbitrário em polígonos *y-monótonos* eficientemente, teríamos um algoritmo melhor. Felizmente, temos o seguinte resultado:

Lema *Um polígono arbitrário pode ser particionado em polígonos y-monótonos em tempo proporcional a $O(n * \log n)$.*

Existem dois tipos especiais de vértices, que podem ser considerados como a fonte de não-monotonicidade de um polígono: os *vértices de quebra* e os *vértices de junção*. Ambos possuem ângulo interno maior que π, mas os vértices adjacentes ao primeiro possuem menor y-coordenada e ao segundo possuem maior y-coordenada.

O algoritmo estudado move uma linha horizontal no plano (=*linha de varredura*), de cima para baixo, e vai processando os vértices conforme passa por eles. A idéia é adicionar diagonais para vértices de quebra assim que a linha de varredura cruza com eles e diagonais para os vértices de junção depois de passar por eles.

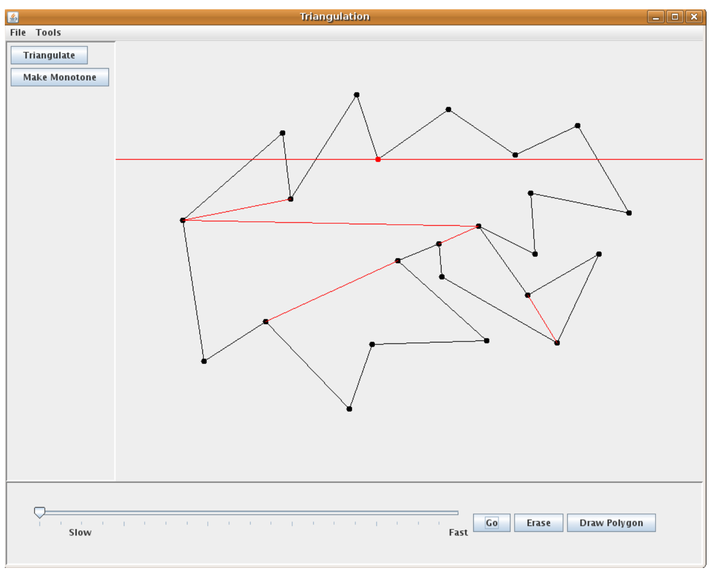


Figura 2: *Algoritmo de monotonização.(Fonte: Programa desenvolvido durante o projeto)*

4. Triangularização de Pontos no Plano

Um *PPS* (*planar point set*) é um conjunto de pontos no plano. Uma *PPST* (*planar point set triangulation*, ou *triangularização de um conjunto de pontos no plano*) é uma subdivisão do plano pela adição de um conjunto maximal de arestas não intersectantes. Ou seja, é um conjunto S de arestas tal que toda aresta que não está em S intersecta alguma aresta de S.

Um *PPS* admite várias triangularizações distintas. Podemos estabelecer propriedades desejáveis e tentar obter uma *PPST* boa em relação à essas propriedades. No nosso trabalho, estudamos uma *PPST* bem conhecida, a *Triangularização de Delaunay*, cuja propriedade é maximizar o menor ângulo dos triângulos gerados.

Formalmente, podemos definir a triangularização procurada da seguinte forma: Seja T uma *PPST* de um *PPS* P, e suponha que possua m triângulos. Agora, ordenamos os $3m$ ângulos de T em ordem crescente. Seja $A(T) := (\alpha_1, \alpha_2, ..., \alpha_{3m})$ a sequência ordenada, $\alpha_i \leq \alpha_j$ para todo $i < j$. Chamamos $A(T)$ o *vetor-ângulo* de T. Seja T' outra triangularização de P com vetor-ângulo $A(T') := (\alpha'_1, \alpha'_2, ...\alpha'_{3m})$. Dizemos que o vetor-ângulo de T é maior que o vetor-ângulo de T' se para algum índice i com $1 \leq i \leq 3m$, temos

$$\alpha_j = \alpha'_j \text{ para todo } j < i, \text{ e } \alpha_i > \alpha' i.$$

Em outras palavras, no primeiro índice i em que α_i e α'_i diferem, $\alpha_i > \alpha'_i$. Escrevemos $A(T) \succ A(T')$. Uma triangularização T é chamada *ângulo-ótima* se $A(T) \succeq A(T')$ para toda triangularização T' de P.

Lema *A Triangularização de Delaunay produz uma PPST ângulo-ótima.*

5. Triangularizações de Steiner

Mesmo com boas triangularizações, uma *PPST* ainda pode apresentar triângulos com características indesejáveis. Uma forma de melhorar a qualidade da triangularização é inserir pontos artificialmente no nosso conjunto. Chamamos esses pontos de *pontos de Steiner* e a triangularização resultante do novo conjunto de *triangularização de Steiner*. Intuitivamente, não queremos adicionar muitos pontos, já que a adição de muitos triângulos pode prejudicar o propósito da aplicação.

6. Conclusão

Existe um algoritmo que consome tempo linear para a triangularização de polígonos [5] que não foi estudado no projeto. No entanto, os algoritmos estudados já se mostram bastante complicados de implementar, especialmente quando começamos a lidar com casos degenerados.

Uma implementação da *Triangularização de Steiner* estudada mostra melhoras significativas com relação à *Triangularização de Delaunay* [6]. Podemos exigir, por exemplo, que nossa triangularização não contenha nenhum triângulo com ângulo menor que uma constante.

Referências

[1] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed., 2005.

[2] Joseph O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1998.

[3] V. Chvátal, *A combinatorial theorem in plane geometry*, Journal of Combinatorial Theory B **vol. 18** (1975), pp. 39:41.

[4] S. Fisk, *A short proof of Chvátal's watchmen theorem*, Journal of Combinatorial Theory B (1978).

[5] B. Chazelle, *Triangulating a simple polygon in linear time*, Discrete and Computational Geometry **6** (1991), no. 1, 485–524.

[6] M. and Eppstein Bern D. and Gilbert, *Provably good mesh generation*, Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on, 1990.