Relatório do trabalho 2
Diagrama de Voronoi

João Paulo Pizani Flor
Mauricio Oliveira Haensch

April 7, 2011

1 Introdução

Para o segundo trabalho prático da disciplina era necessário a implementação de um algoritmo para construção do diagrama de Voronoi, para ser utilizado em um conjunto de pontos 2D, assim como nos casos específicos de espirais simples e duplas.

Para um conjunto A de **n** pontos dispostos em um espaço, chamados centróides, o diagrama de Voronoi é a segmentação desse espaço de forma com que cada um dos segmentos formados está associado a um dos centróides, representando a sub-área do espaço cuja distância até esse ponto é menor que a qualquer outro ponto do conjunto A.

O algoritmo para geração do diagrama de Voronoi possui as seguintes etapas:

- Combinação de pontos do conjunto 3 a 3 para formação de circunferências, de modo com que nenhum outro ponto esteja dentro da circunferência;
- Geração de triângulos a partir dos pontos que formam as circunferências que satisfazem a condição anterior;
- Cálculo das mediatrizes de cada triângulo criado;

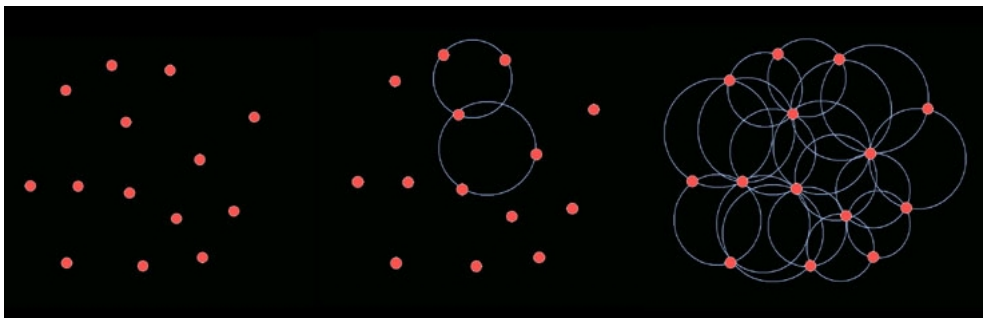


Figure 1: Conjunto de pontos inicial e geração das circunferências.

A partir do conjunto de mediatrizes encontrados, são calculados seus pontos de intersecção de forma com que novos polígonos possam ser formados. Cada um dos polígonos é um segmento do diagrama de Voronoi associada ao seu centróide.

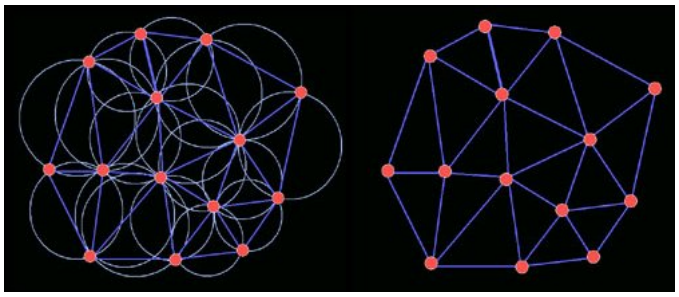


Figure 2: Formação dos triângulos a partir das circunferências válidas (com apenas 3 pontos).

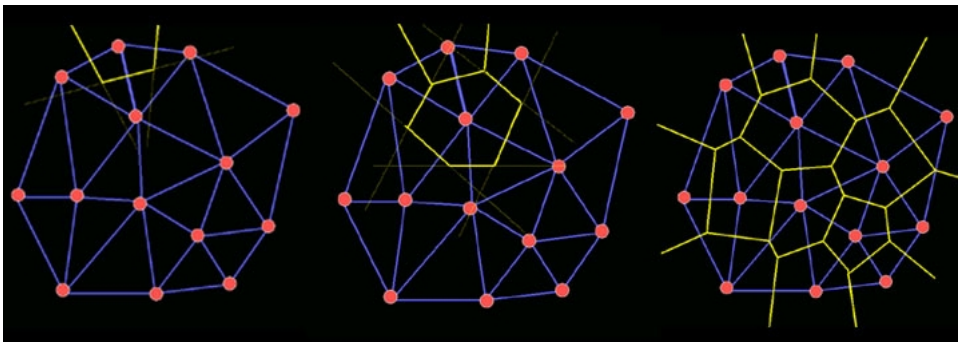


Figure 3: Cálculo das mediatrizes e suas intersecções.

2 Ferramentas utilizadas

A linguagem utilizada para implementação do programa era de livre escolha dos alunos, assim como outras ferramentas que pudessem ser utilizadas para representação gráfica dos padrões, por exemplo. As ferramentas escolhidas foram as mesmas utilizadas para a implementação do trabalho anterior:

Linguagem de programação - Python: escolhida por ser uma linguagem já bem conhecida pela equipe, com boas bibliotecas e documentação, além de agilizar o desenvolvimento.

Interface gráfica - Qt + PyQt: PyQt é um *binding* do framework gráfico Qt para a linguagem Python, escolhida por já ser conhecida pelos integrantes.

NumPy: Biblioteca da linguagem Python utilizada para computação científica, com diversas estruturas e funções otimizadas. Para este trabalho, utilizamos algumas de suas funções de álgebra linear, para fazer operações com matrizes.

O trabalho foi desenvolvido no ambiente Linux/Ubuntu e para executá-lo são necessários alguns pacotes[1] para a interface gráfica:

[1] os nomes dos pacotes citados são específicos para a distribuição Ubuntu, para outras distribuições devem ser procurados os pacotes correspondentes.

- pyqt4-dev-tools
- libqtgui4
- libqtcore4

3 Resultados

A geração de um conjunto inicial de pontos de 2 dimensões e a geração de espirais simples e duplas foi mantida do trabalho anterior, assim como a maior parte da interface gráfica. Poucos ajustes nas funções já existentes foram realizados e adicionamos uma funcionalidade de adição de um ponto por clique na tela, facilitando a adição de um conjunto de pontos para teste.

Para desenvolver o algoritmo do diagrama de Voronoi, seguimos a etapa inicial com o cálculo das circunferências normalmente. São geradas combinações 3 a 3 dos pontos no conjunto de pontos inicial e é verificado se essa circunferência não possui nenhum outro ponto interno a ela, caso em que a circunferência é descartada.

A partir do conjunto de circunferências válidas, são gerados triângulos formados pelos 3 pontos que descrevem a circunferência. Nesse ponto do nosso algoritmo, possuímos uma estrutura que guarda todos os conjuntos (centro, raio) que descrevem uma circunferência válida e uma outra estrutura que mapeia cada centróide (conjunto de pontos inicial) para os triângulos adjacentes a eles. Essas estruturas são usadas no passo posterior para gerar os polígonos do diagrama de Voronoi.

O algoritmo apresentado em sala realizava o cálculo das intersecções das medianas (reta perpendicular a um dos lados do triângulo, passando pelo ponto médio desse lado) para descobrir as bordas de cada polígono que definem uma célula do diagrama do Voronoi. Porém, vimos que esses pontos de intersecção são os circuncentros dos triângulos (ponto de encontro das 3 medianas), que é também o centro do círculo circunscrito ao triângulo.

Como já tínhamos os centros dos círculos armazenados, bastava que soubéssemos quais deles deveriam ser ligados para que fossem formadas as células, e por isso guardamos o mapa ligando cada centróide aos triângulos que o formam. Desse modo, para encontrar a célula de um centróide no diagrama de Voronoi, bastava ligar os circuncentros dos triângulos adjacentes a ele na ordem correta. Para encontrar essa ordem, utilizamos o algoritmo de fecho convexo de Graham[2].

Esse método resolve o problema para todos os centróides internos do conjunto de pontos inicial. Já para os pontos mais externos, é necessário encontrar as células abertas que definem o diagrama de Voronoi. Para isso, separamos o conjunto de pontos inicial em centróides internos e centróides externos, com a ajuda do algoritmo de Graham (executando o algoritmo no conjunto total de pontos, ele formará um polígono fechado com os pontos mais externos, então calculamos a diferença entre o conjunto inicial e o resultado da varredura do algoritmo para acharmos os pontos internos). Os pontos internos são tratados como explicado anteriormente enquanto aplicamos um tratamento especial aos pontos externos.

[2]Dado um conjunto de pontos, o algoritmo de Graham deve encontrar o maior polígono fechado possível (fecho convexo). Esse algoritmo começa pelo ponto de menor coordenada Y e varre os outros pontos, ordenando-os de acordo com o ângulo formado entre eles a reta que passa pela coordenada X do ponto inicial, em sentido anti-horário.

Para os pontos externos, ligamos os circuncentros dos triângulos adjacentes a 2 semi-retas formadas pelas medianas das bordas do polígono externo. O algoritmo de Graham é novamente utilizado nesse passo, para que seja estabelecida uma ordem na lista dos pontos obtidos, e possamos desenhar o polígono. Os segmentos de reta que se projetam "para fora" do diagrama de Voronoi são, obviamente, finitos. A ilusão de infinitude é obtida prolongando-se o segmento para coordenadas bem maiores do que as capazes de serem mostradas na interface gráfica.

A seguir, uma imagem com o resultado obtido pela aplicação do algoritmo de diagrama de Voronoi num conjunto arbitrário de pontos 2D.

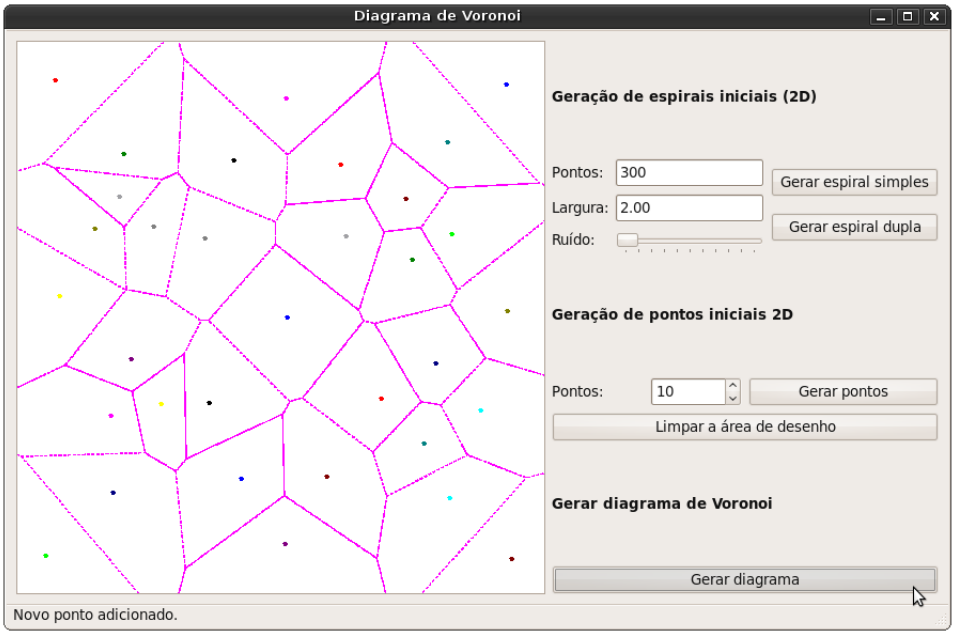
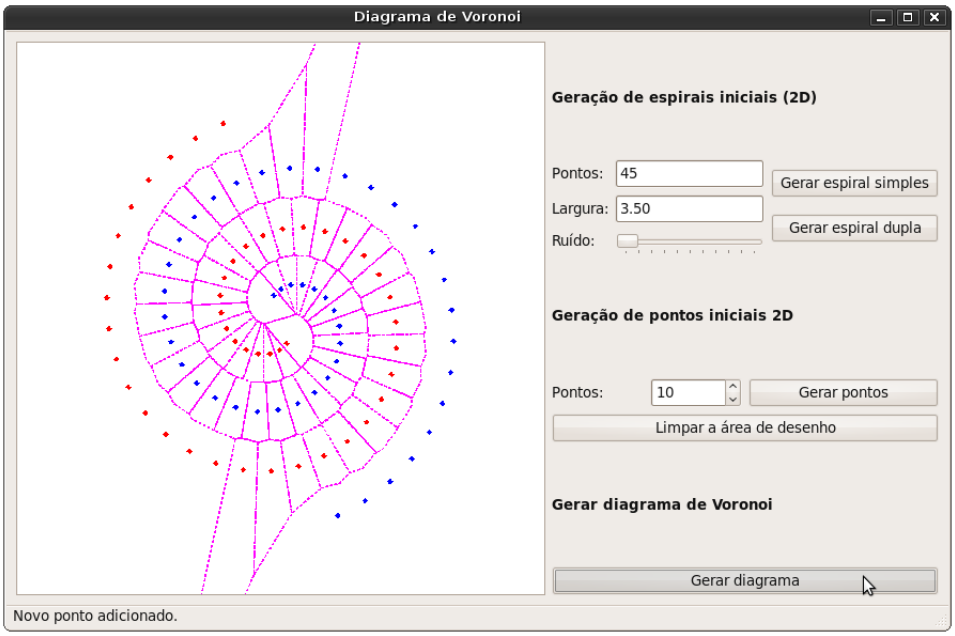
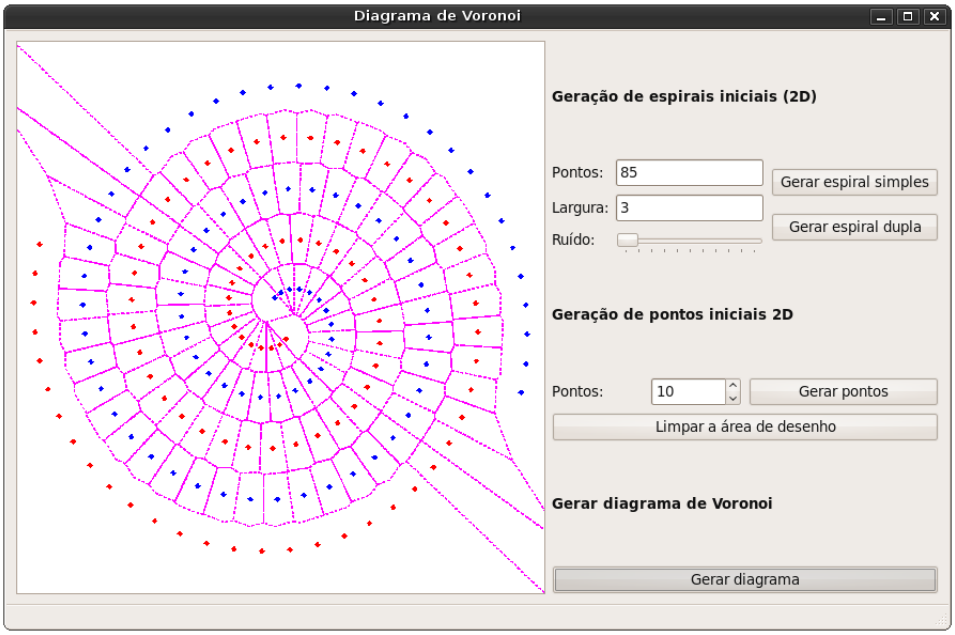


Diagrama de Voronoi sobre um conjunto arbitrário de pontos 2D.

Como casos especiais, as espirais simples e duplas deveriam ser testados com o algoritmo de diagrama de Voronoi implementado. Seguem abaixo 2 exemplos da aplicação do nosso algoritmo, considerando apenas as células fechadas (sem o cálculo das células externas).



Aplicação do algoritmo sobre uma espiral dupla, apenas com as células internas.



Outro exemplo de espiral dupla, apenas com as células internas.

4 Referências

- Python - http://www.python.org/

- Qt - http://doc.qt.nokia.com/
- pyQt - http://www.riverbankcomputing.co.uk/software/pyqt/intro