Disciplina INF2604 – Geometria Computacional

RELATÓRIO DE TRABALHO II:
Diagrama de Voronoi

Aluno: Gustavo Coelho

1. Objetivo

O objetivo deste relatório é descrever as abordagens usadas na implementação de um Diagrama de Voronoi para um dado conjunto de pontos. Para a construção do diagrama, usaremos o método baseado no Algoritmo Incremental.

Primeiro será feita uma breve descrição teórica do método e seus passos para a construção de cada célula. Em seguida, demonstraremos os resultados de sua implementação.

2. Método

Como já mencionado, o método escolhido é baseado no Algoritmo Incremental de Voronoi. O método inicia a execução escolhendo os dois primeiros pontos do conjunto de pontos e criando o diagrama inicial. Diagramas com apenas dois pontos são triviais, sendo apenas necessário traçar a bissetora. Para a definição da bissetora, usaremos coordenadas homogêneas, ou seja, definimos a reta bissetora através de dois pontos "infinitos" de coordenadas (x, y, 0). Para a definição desses pontos, primeiro consideramos o ponto M, localizado no centro da distância entre os dois ponto. Ou seja:

$$M = \left(\frac{P_{0_x} + P_{1_x}}{2}, \frac{P_{0_y} + P_{1_y}}{2}\right)$$

Em seguida, definimos o vetor $\overrightarrow{V_1}$ como $P_0 - M$, ou seja:

$$\overrightarrow{V_1} = \left(P_{0_x} - M_x, P_{0_y} - M_y\right)$$

Definimos então o vetor $\overrightarrow{V_2}$ que será usado efetivamente na construção da bissetora e é perpendicular a $\overrightarrow{V_1}$, ou seja, $\overrightarrow{V_1}.\overrightarrow{V_2} = 0$. Portanto, temos:

$$\overrightarrow{V_1}.\overrightarrow{V_2} = V_{1_x}V_{2_x} + V_{1_y}V_{2_y} = 0$$

Desenvolvendo a equação, temos:

$$V_{2_y} = -\frac{V_{1_x}V_{2_x}}{V_{1_y}}$$

Ou seja:

$$\overrightarrow{V_2} = \left(t, -\frac{V_{1_x}t}{V_{1_y}}\right)$$

Os pontos que formam a bissetora serão formados a partir de M, deslocados através de $\overrightarrow{V_2}$, nas duas possíveis direções, ou seja, considerando o valor de t positivo e negativo. Portanto, temos:

$$Bissetora_{(P_0,p_1)} = \left(\left(M_x + t, M_y - \frac{V_{1_x}t}{V_{1_y}}\right), \left(M_x - t, M_y + \frac{V_{1_x}t}{V_{1_y}}\right)\right)$$

Para efeitos práticos, consideramos o valor de t suficientemente alto para que seja detectável por outras retas distantes. Usando o valor de t = 1000 e considerando coordenadas homogêneas, temos:

$$Bissetora_{(P_0,p_1)} = \left(\left(M_x + 1000, M_y - \frac{1000V_{1_x}}{V_{1_y}}, 0\right), \left(M_x - 1000, M_y + \frac{1000V_{1_x}}{V_{1_y}}, 0\right)\right)$$

O resultado pode ser visto na figura abaixo:

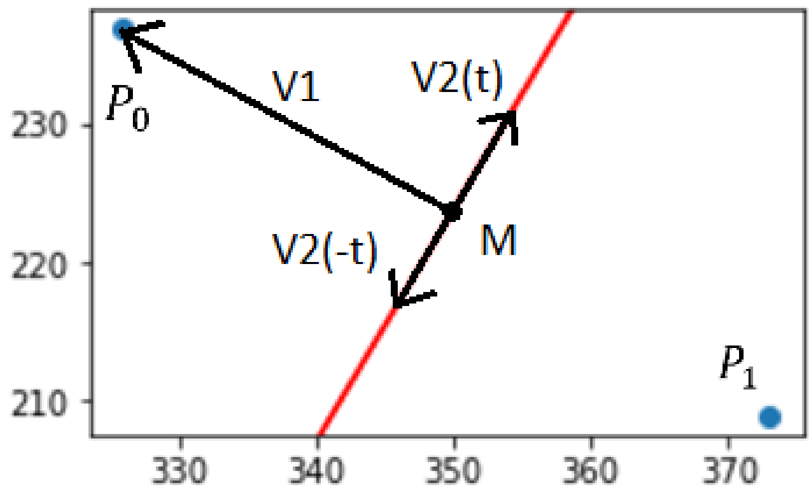

Figura 1: Definição da bissetora de dois pontos

É importante notar que para casos em que P_0 e P_1 formam uma linha horizontal, V_{1_y} será nulo, o que resultará em uma coordenada y de V_2 igual a infinito. Nestes casos, consideramos $V_2 = [0,1]$. O custo em tempo de execução para esta operação será constante ($O_{(1)}$).

Com o diagrama inicial definido, em seguida adicionamos um terceiro ponto, e identificamos em qual célula ele se encontra. Para isso, buscamos o ponto com menor distância entre os existentes no diagrama e retornamos sua célula como o resultado da busca. Essa operação terá um custo de tempo de execução potencialmente $O_{(n)}$.

Com a definição da célula de chegada do novo ponto, definimos a bissetora deste ponto em relação ao ponto da célula em que ele se encontra. O cálculo é similar ao caso anterior, porém, neste caso, ao definir os valores de t, precisamos definir se há intersecção entre a nova bissetora e alguma reta, semirreta ou segmento de reta existente no diagrama. Para os dois sentidos da bissetora, se houver intersecção, a reta deve ser interrompida neste ponto. Caso contrário, estende-se o ponto para o infinito, ou seja, atribuindo à t um valor suficientemente grande e definindo a terceira coordenada como zero.

Para definir o ponto de intersecção entre a nova bissetora e as arestas da célula, consideramos que para cara aresta formada pelo ponto A e B, podemos definir sua formula (desconsiderando por hora seus limites de x e y) como:

$$r = A + sV_3 \text{ , onde } V_3 = B - A$$

Da mesma forma, podemos descrever a equação da bissetora:

$$r = M + tV_2$$

A intersecção das retas se dará onde suas coordenadas se coincidirem, ou seja:

$$A_x + sV_{3_x} = M_x + tV_{2_x}$$
$$A_y + sV_{3_y} = M_y + tV_{2_y}$$

Desta forma, temos um sistema de duas equações e duas incógnitas:

$$sV_{3_x} - tV_{2_x} = M_x - A_x$$
$$sV_{3_y} - tV_{2_y} = M_y - A_y$$

Usando a regra de Cramer, temos:

$$D = \begin{bmatrix} V_{3_x} & -V_{2_x} \\ V_{3_y} & -V_{2_y} \end{bmatrix} = V_{3_x}V_{2_y} + V_{3_x}V_{2_y}$$

$$D_s = \begin{bmatrix} M_x - A_x & -V_{2_x} \\ M_y - A_y & -V_{2_y} \end{bmatrix} = -(M_x - A_x)V_{2_y} + \left(M_y - A_y\right)V_{2_x}$$

$$D_t = \begin{bmatrix} V_{3_x} & M_x - A_x \\ V_{3_y} & M_y - A_y \end{bmatrix} = \left(M_y - A_y\right)V_{3_x} - (M_x - A_x)V_{3_y}$$

$$s = \frac{D_s}{D}, \qquad t = \frac{D_t}{D}$$

Caso D seja nulo, podemos dizer que não há intersecção. Caso contrário, definimos inicialmente a intersecção apenas substituindo os valores encontrados de t ou s em uma das retas.

Com o ponto de intersecção encontrado, ainda precisamos verificar se ela ocorre nos limites da aresta. Caso contrário, desconsideramos a intersecção.

Após realizar este procedimento para todas as arestas, podemos garantir que para qualquer célula, a nova bissetriz terá ao menos uma intersecção, e no máximo duas. No primeiro caso, estendemos a bissetora, definindo uma semirreta. No segundo caso, definimos um segmento de reta delimitado pelos dois pontos encontrados.

Após a definição da primeira aresta da nova célula, repetimos o mesmo procedimento para as células vizinhas, sempre tomando como base o novo ponto adicionado, conforme abaixo:

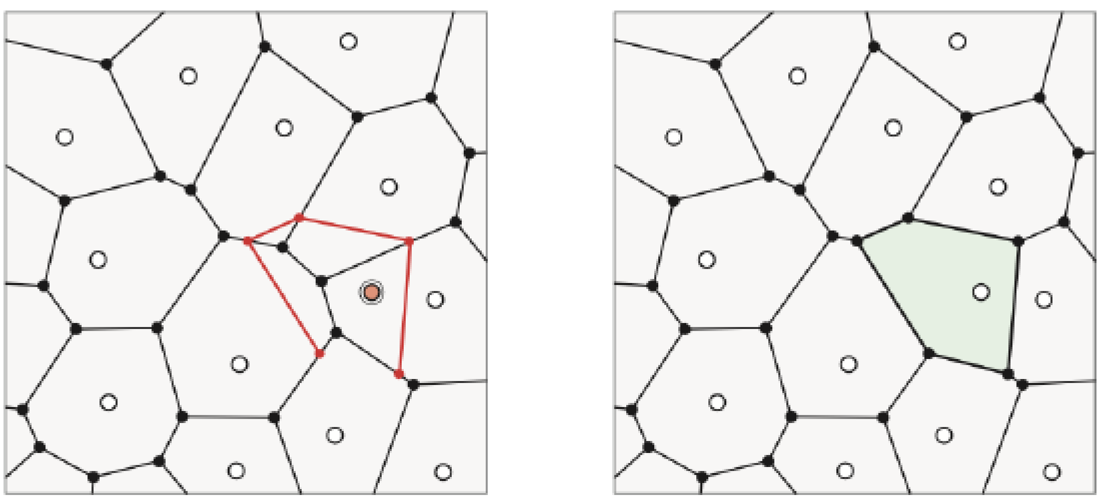

Figura 2: Processo de definição das arestas de uma nova célula.

Ao final do processo, precisamos ainda revisitar todas as células visitadas e refazer suas arestas que foram “cortadas” pela nova célula, ou seja, se algum dos pontos da nova aresta é colinear aos pontos de uma dada aresta. Caso positivo, eliminamos o ponto desta aresta que se situa mais próximo do novo ponto do que do ponto original da célula, e o substituímos pelo ponto colinear. Caso não haja colinearidade com nenhum ponto da nova aresta, realizamos o mesmo teste de proximidade. Caso sejam mais próximos da nova aresta, simplesmente o excluímos, e caso contrário, mantemos a aresta intacta.

Para definir a colinearidade entre um ponto da aresta adicionada e a aresta original, basta verificar se o triângulo formado por estes três pontos possui área nula, ou seja:

$$Area = \begin{bmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{bmatrix} = A_xB_y + A_yC_x + B_xC_y - C_xB_y - C_yA_x - B_xA_y$$

Onde A e B são os pontos da aresta original e C um dos pontos da aresta adicionada.

Após esta iteração, adicionamos a nova célula e repetimos o processo até o último ponto do conjunto. O custo de tempo de execução para cada uma das iterações é $O_{(n)}$. Portanto, espera-se que ao todo, o algoritmo tenha um custo de $O_{(n^2)}$.

3. Implementação

A implementação do algoritmo descrito foi implementada em linguagem Python e executada em dois conjunto de pontos pré-definidos. Os resultados podem ser vistos nas figuras abaixo:

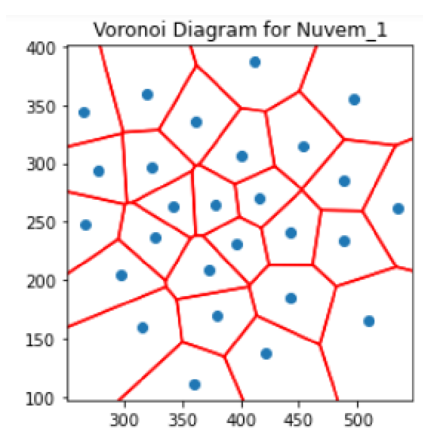

Figura 3: Implementação do algoritmo na primeira nuvem de pontos.

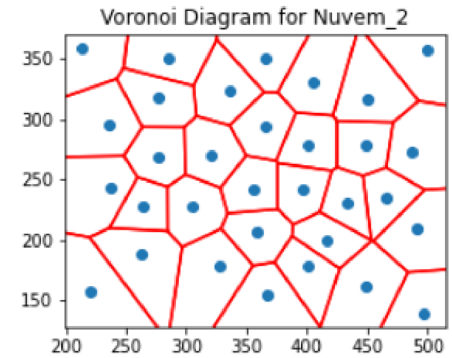

Figura 4: Implementação do algoritmo na segunda nuvem de pontos.

A variação do tempo de execução de acordo com o tamanho da entrada pode ser vista abaixo:

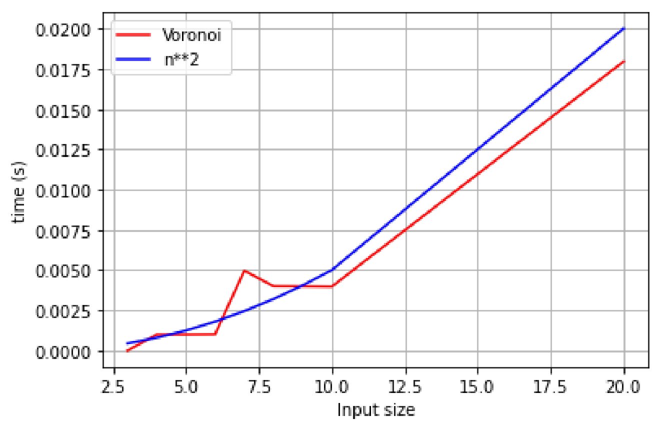

Figura 5: Relação entre tamanho da entrada e tempo de execução.

4. Conclusão

O algoritmo proposto foi implementado e como esperado, a complexidade assintótica do tempo de execução é $O_{(n^2)}$. A implementação possui desafios principalmente em relação à erros de arredondamento no momento da definição da colinearidade entre uma aresta e os pontos de uma nova aresta adicionada. Para pontos que se encontram muito distantes da nuvem de pontos, o valor do determinante que define a colinearidade tende a ser maior, mesmo em pontos colineares. Em contrapartida, para pontos muito próximos mas não colineares, o determinante tende a ser pequeno. Estas discrepâncias tendem a induzir erros ao modelo, dependendo da distribuição dos pontos.