3
Silhueta de uma nuvem de pontos via reconstrução local de malha

Neste capítulo descrevemos um método de geração de silhueta a partir da reconstrução local de uma malha. Ao final, mostramos também um outro tipo de linhas características, as Linhas Laplacianas.

3.1
Reconstrução local de malha

Conforme comentamos na seção 2.1, curvas silhuetas são bem definidas em malhas poligonais. Porém, em nuvens de pontos que representam superfícies não são conhecidos os vizinhos de cada ponto ou de que maneira estes se conectam. Assim, uma das maneiras de extrair a silhueta de uma nuvem de pontos é reconstruir uma malha tendo os pontos da nuvem como vértices.

Neste trabalho, optamos por uma reconstrução local da malha, já que o cálculo para a extração de silhueta é feito localmente. Portanto, nesta seção apresentamos o processo de reconstrução local em torno de cada ponto P da nuvem.

Em linhas gerais, seguimos o processo descrito por Olson (11), no qual a reconstrução tem como base a determinação de uma estrela para cada ponto P da nuvem, ou seja, determinamos triângulos em torno deste ponto de modo que os outros vértices são também pontos da nuvem.

Para este fim, tomamos como dado inicial uma nuvem N de pontos munidos de suas normais. Para o cálculo das normais, podemos aplicar a técnica descrita em (11).

Para determinar a estrela de um ponto P da nuvem ($star(P)$), seguimos dois passos:

1. Definir os vizinhos mais próximos.

2. Gerar a estrela.

Definição dos vizinhos de um ponto

Dado um ponto $P \in N$, munido de sua normal $\mathbf{n}_P$, definimos como plano tangente à nuvem em P o plano $\pi(P)$, determinado pela normal $\mathbf{n}_P$ e pelo ponto P. Determinamos o conjunto $K(P)$, formado pelos k pontos que estão mais próximos de P segundo a distância euclidiana, e que estão mais próximos também do plano tangente à nuvem em P. Isto é, se $Q \in K(P)$, com $Q \neq P$, então o ângulo α entre o vetor $\overrightarrow{PQ}$ e $\mathbf{n}_P$ deve ser próximo de 90^o. Geometricamente, isto significa que descartamos todos os pontos que estão dentro de um cone, como mostra a Figura 3.1.

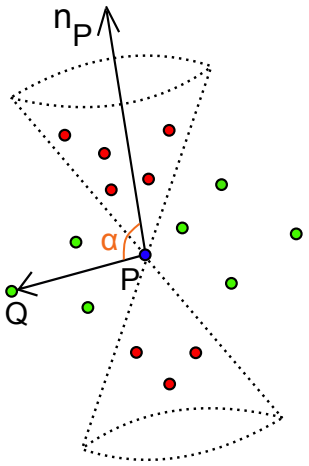


Figura 3.1: Filtro dos pontos vizinhos: os pontos vermelhos, que estão dentro do cone, são descartados.

O objetivo deste filtro é reduzir possíveis problemas na geração de estrelas em regiões que apresentem possíveis ruídos ou com alta curvatura. Em modelos conhecidos, notamos que, ao fazermos a reconstrução de estrelas de pontos em regiões com esta característica, as estrelas definidas para alguns desses pontos não respeitam a provável superfície do modelo. Isto ocorre porque, nessas regiões, podemos considerar inicialmente como vizinhos de um ponto P pontos que não seriam naturalmente conectados a P e, ao projetarmos este ponto no plano tangente à superfície em P, sua projeção fica mais próxima do ponto P do que a projeção dos vizinhos que o modelo exige. Assim, após a triangulação de Delaunay, este ponto fará parte da estrela de P, causando resultados indesejáveis.

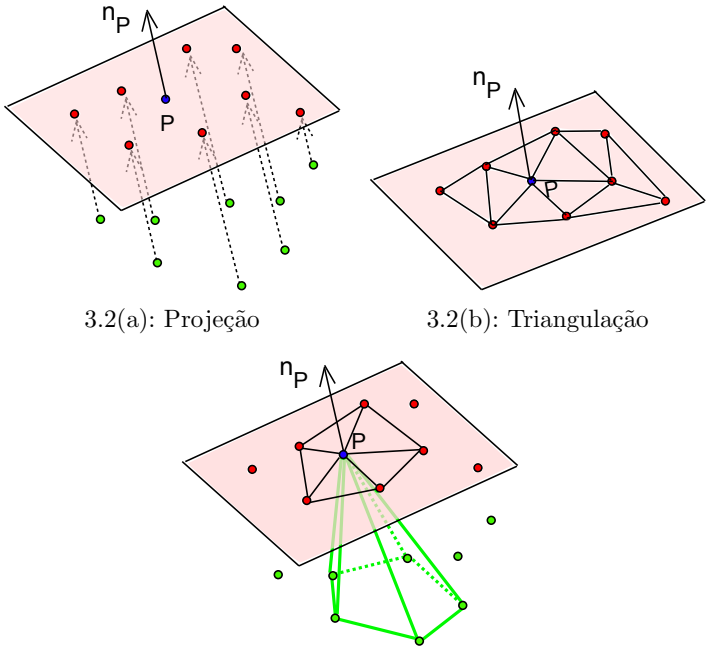
Obtemos, assim, o conjunto $K(P)$ de vizinhos de P, ao qual incluímos também o ponto P.

Triangulação de Delaunay e determinação das adjacências

Tendo definido o conjunto $K(P)$, projetamos os pontos de $K(P)$ em π_P, obtendo o conjunto $K_\pi(P)$ (Figura 3.2(a)).

Para obter a reconstrução local, como os pontos de $K_\pi(P)$ são coplanares, utilizamos o algoritmo de triangulação de Delaunay 2D da biblioteca CGAL (Computational Geometry Algorithms Library) (7).

Uma vez que temos a triangulação de Delaunay dos pontos de $K_\pi(P)$, estendemos as adjacências obtidas nos vértices projetados para os pontos da nuvem correspondentes. Ou seja, dado Q um ponto de $K(P)$ que após a projeção é transformado em $Q' \in K_\pi(P)$, então, se $\overline{PQ'}$ é uma aresta da triangulação de Delaunay, então Q é um ponto da nuvem que está na estrela de P, e dizemos que $Q \in star(P)$ (Figura 3.2(c)).



3.2(a): Projeção

3.2(b): Triangulação

3.2(c): Estrela do ponto P

Figura 3.2: Para determinar a estrela de um ponto P, projetamos seus vizinhos no plano tangente à nuvem em P 3.2(a), fazemos a triangulação de Delaunay 3.2(b) e daí definimos como serão as conexões na nuvem 3.2(c).

O algoritmo 3.1 resume os passos descritos acima:

Algoritmo 3.1:	Reconstrução da estrela de P
1:	Determinar os k pontos mais próximos de P.
2:	Filtrar estes pontos, obtendo o conjunto $K(P)$.
3:	Projetar $K(P)$ em π_P, plano tangente à nuvem em P.
4:	Aplicar algoritmo de triangulação de Delaunay.
5:	Determinar quais são os pontos conectados a P.

Uma das grandes dificuldades na reconstução de uma malha a partir de uma nuvem de pontos é estabelecer a conectividade entre estes pontos, uma vez que isto não está disponível. A triangulação de Delaunay nos fornece uma

malha com propriedades desejáveis. No nosso caso, a grande importância de utilizar este tipo de triangulação para reconstruir a estrela de cada ponto P da nuvem vem da ideia de estarmos buscando os pontos que estão de alguma forma mais próximos de P. Isto fica melhor esclarecido uma vez que observamos a dualidade entre a triangulação de Delaunay e o diagrama de Voronoi.

Portanto, uma vez que foi obtida a triangulação de Delaunay de um conjunto de pontos próximos a um ponto P da nuvem, se destacarmos aqueles pontos que estão conectados a P (vértices dos triângulos da estrela de P), estamos selecionando pontos da nuvem que em geral estão próximos a P.

3.2
Extração local da silhueta de uma nuvem de pontos

Estando definida a reconstrução local da malha, nesta seção descreveremos os passos para a extração da silhueta de uma nuvem de pontos.

Para a determinação da curva silhueta, buscamos pontos desta curva nas arestas da estrela de cada ponto da nuvem. Porém, dados os pontos P e Q na nuvem, tais que $Q \in star(P)$, não temos garantia alguma de que teremos também $P \in star(Q)$. Dessa forma, dado $Q \in star(P)$, para reduzir sobreposições de linhas e melhorar a qualidade da curva, a busca por pontos sobre a silhueta em uma aresta $\overline{PQ}$ de $star(P)$ é feita apenas se também $P \in star(Q)$.

Uma vez definido que uma aresta $\overline{PQ}$ deve ser processada, calculamos a iluminação em P e em Q. Caso haja mudança de sinal, usamos uma interpolação linear para localizar o ponto $\mathbf{s}$ da aresta, tal que $I(\mathbf{s}) = 0$:

$$I(P).I(Q) < 0 \Rightarrow \mathbf{s} = \frac{|I(P)|\, Q + |I(Q)|\, P}{|I(P)| + |I(Q)|},$$

e consideramos o ponto $\mathbf{s}$ um ponto sobre a silhueta.

Vale ressaltar que fizemos a determinação da silhueta localmente. Segundo Olson (11), a extração da silhueta global depende de propriedades da conectividade entre os pontos e esta deve satisfazer alguns critérios topológicos que não podem ser garantidos com uma reconstrução local.

O método de reconstrução utilizado é sensível a ruídos, o que é uma limitação. Portanto, para determinar uma silhueta em exemplos com ruídos, propomos um novo método, descrito no Capítulo 4.

A reconstrução usada neste método é muito dependente de como estão amostrados os pontos da nuvem. Os dados obtidos por meio de scanner 3D podem ser dados de alta complexidade. Contudo, o processo de captura da nuvem de pontos que representa o objeto escaneado pode gerar também

amostras esparsas, com buracos e ruídos e, ao processarmos dados deste tipo, fica aparente uma limitação deste método. O método descrito no Capítulo 4 propõe a ultilização de outro tipo de reconstrução, buscando extrair a silhueta mesmo no caso da nuvem apresentar ruídos.

3.3 Linhas Laplacianas de uma nuvem de pontos

Nesta seção estendemos a ideia de silhueta numa nuvem de pontos, aplicando a reconstrução descrita na seção 3.1 para a extração das Linhas Laplacianas de uma nuvem de pontos.

Definição 3.1 *As **Linhas Laplacianas** são um conjunto de pontos $\boldsymbol{p}$ de uma superfície $S \subset \mathbb{R}^3$ (de classe C^3), nos quais o Laplaciano da iluminação ΔI é igual a zero e a magnitude do gradiente $\|\nabla I\|$ é maior do que um limite τ definido pelo usuário, isto é,*

$$\Delta I(\boldsymbol{p}) = 0 \; e \; \|\nabla I(\boldsymbol{p})\| \geq \tau,$$

onde Δ é o operador Laplace-Beltrami em S (16).

Nos exemplos que serão exibidos, utilizamos $\tau = 0$.

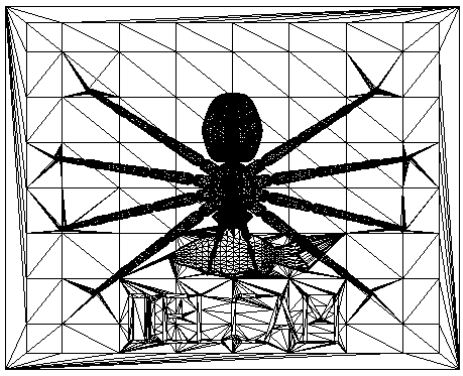
As Linhas Laplacianas são, assim como a silhueta, curvas que representam um objeto, suprimindo aspectos secundários ao que se deseja representar. Entretanto, segundo Zhang (16) a curva silhueta é um pouco limitada no que diz respeito a capturar a estrutura e a complexidade da forma interior dos objetos. Na Figura 3.3 podemos observar que as Linhas Laplacianas mostram mais detalhes do objeto do que a curva silhueta.

A determinação de linhas em 2D que representam superfícies 3D, como a silhueta e as Linhas Laplacianas, é caracterizada pela detecção de mudanças significantes na iluminação. Para determinar o Laplaciano da iluminação $\Delta I(P)$, utilizamos a definição de Zhang (16):

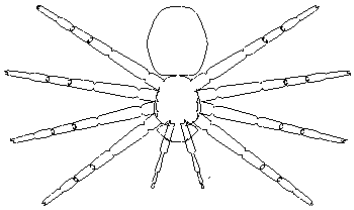
$$\Delta I(P) = (\Delta \mathbf{n}_P).\boldsymbol{v}$$

Observe que $\Delta \mathbf{n}_P$ independe da posição do observador e, por isso, pode ser calculado em uma etapa de pré-processamento. Dessa forma, a complexidade da extração das Linhas Laplacianas é semelhante à complexidade da extração da silhueta, pois basta substituirmos $\mathbf{n}_P$ por $\Delta \mathbf{n}_P$.

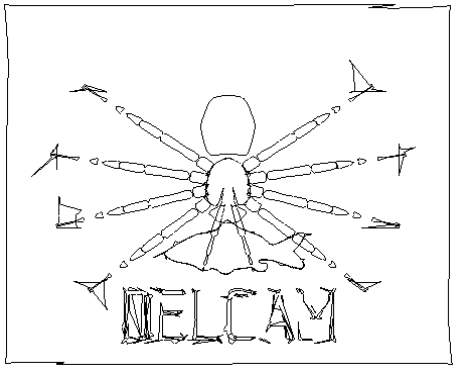
Portanto, como essa é a diferença essencial para a silhueta, o cálculo do Laplaciano das normais é um passo importante da extração das Linhas Laplacianas. Embora muitos operadores Laplacianos sejam conhecidos, Zhang



3.3(a): Malha



3.3(b): Silhueta



3.3(c): Linhas laplacianas

Figura 3.3: A partir de uma malha poligonal, foram extraídas a silhueta e as Linhas Laplacianas.

(16) utiliza um Laplaciano *para malhas* de forma a tornar robusto seu algoritmo de extração das linhas. Tal Laplaciano é definido como segue:

Definição 3.2 *Dada uma função $f : M \to \mathbb{R}$, definida na malha M, o operador* ***Laplaciano na malha*** *L^h_M em um vértice $\boldsymbol{p}$ da malha é definido como*

$$L^h_M f(\boldsymbol{p}) = \frac{1}{4\pi h^2(\boldsymbol{p})} \sum_{\triangle_i \in M} \frac{A(\triangle_i)}{3} \sum_{\boldsymbol{q} \in \triangle_i} e^{\frac{\|\boldsymbol{q}-\boldsymbol{p}\|^2}{4h(\boldsymbol{p})}} (f(\boldsymbol{q}) - f(\boldsymbol{p})),$$

onde $A(\triangle_i)$ é a área do triângulo $\triangle_i$ e $h(\boldsymbol{p})$ é um valor positivo que está relacionado ao tamanho da vizinhança considerada do ponto $\boldsymbol{p}$.

Este Laplaciano discreto utiliza um peso que prioriza os pontos mais próximos do ponto no qual será calculado. O parâmetro h, na definição anterior, está diretamente relacionado ao número de Linhas Laplacianas que serão extraídas: quanto maior o valor de h, menos linhas serão desenhadas (Figura 3.4).

A determinação do Laplaciano da normal de um ponto P é feita utilizando a definição 3.2 para cada uma das coordenadas da normal.

3.4(a): h = 0.01

3.4(b): h = 0.03

3.4(c): h = 0.05

Figura 3.4: Linhas Laplacianas para diferentes valores de h.

Para o caso de nuvem de pontos, estamos propondo uma adaptação do cálculo deste Laplaciano. Foram testadas duas estratégias para o cálculo do Laplaciano em um vértice:

- utilizar apenas a estrela do vértice;
- utilizar todos os triângulos gerados na reconstrução das estrelas de todos os vértices.

No Capítulo 5, mostramos que os resultados obtidos utilizando a primeira estratégia não foram satisfatórios.

Como não temos disponível uma malha, utilizamos a reconstrução via triangulação de Delaunay (seção 3.1) para gerar uma lista L de triângulos. Isto é, para cada ponto da nuvem, adicionamos os triângulos de sua estrela a uma lista, tomando o cuidado de não adicionar triângulos que já estão em L. Assim, podemos utilizar uma variação do cálculo do Laplaciano da definição 3.2, onde, em vez de utilizarmos todos os triângulos da malha M, utilizamos os triângulos desta lista.

Para cada triângulo da lista oriunda da reconstrução, processamos suas arestas em buscas de pontos $\mathbf{s}$ tais que $\Delta I(\mathbf{s}) = 0$. Para tanto, dada uma aresta

$\overline{PQ}$ em $star(P)$, temos:

$$\Delta I(P).\Delta I(Q) < 0 \Rightarrow \mathbf{s} = \frac{|\Delta I(P)|\, Q + |\Delta I(Q)|\, P}{|\Delta I(P)| + |\Delta I(Q)|},$$

semelhante ao que foi feito na seção 3.2, substituindo I por ΔI.

O algoritmo 3.3 mostra os passos da extração das Linhas Laplacianas de uma nuvem de pontos:

Algoritmo 3.3:	Extração das Linhas Laplacianas de uma nuvem de pontos
Pré–processamento:	
1:	Para cada $P \in N$, reconstruir $star(P)$.
2:	Para cada $P \in N$, para cada triângulo t de $star(P)$: se $t \notin L$, incluir t na lista L.
3:	Para cada $P \in N$, calcular $\Delta\mathbf{n}_P$.
Fim pré–processamento	
4:	Para cada $t \in L$, calcular ΔI em cada vértice; detectar os zeros de ΔI em cada aresta de t via interpolação; desenhar aresta da Linha Laplaciana.