4
Silhueta de uma nuvem de pontos via superfícies implícitas

Com objetivo de extrair a silhueta de uma nuvem de pontos, em alternativa à reconstrução local de malhas poligonais, proposta por Olson (11) e exposta no capítulo 3, neste capítulo propomos aplicar uma reconstrução local usando aproximação por função implícita.

O método proposto inicia determinando uma função algébrica que aproxima a nuvem N em torno de um ponto $P \in N$. Isso é feito construindo uma *bounding box* definida pelos k vizinhos mais próximos de P e aplicando dois métodos de reconstrução (*Gradient one fitting* e *Ridge regression*).

Em seguida, para determinar pontos sobre a silhueta numa vizinhança de P, utilizamos o método de continuação numérica de Euler-Newton, com passo adaptativo.

Por fim, para extrair a curva silhueta da nuvem de pontos, buscamos dentre todas as *bounding boxes* construídas uma primeira *bounding box* que tenha um pedaço de silhueta. Então, procuramos silhueta nas *bounding boxes* vizinhas e, dessa forma, perseguimos a curva pela nuvem.

4.1
Reconstrução local das superfícies

A reconstrução utilizada foi baseada no método proposto por Mederos(10), no qual é apresentado um modo de aproximar a nuvem por uma superfície algébrica.

Definição 4.1 *Um subconjunto $O \subset \mathbb{R}^3$ é uma* ***superfície implícita*** *se existem uma função $F : \mathbb{R}^3 \to \mathbb{R}$, $O \subset \mathbb{R}^3$, e um número real $c \in \mathbb{R}$ tais que $O = F^{-1}(c)$. A superfície implícita é* ***regular*** *se F é diferenciável e satisfaz a condição de que, em cada ponto $x \in F^{-1}(c)$, o gradiente de F não se anula em x.*

Para a reconstrução local, usaremos uma superfície implícita algébrica que é imagem inversa de um polinômio, definidos como segue:

Definição 4.2 *Um* ***polinômio de grau d*** *definido em* $\mathbb{R}^3$ *é uma função* $P_d : \mathbb{R}^3 \to \mathbb{R}$ *dado pela seguinte expressão:*

$$P_d(x,y,z) = \sum_{0 \leq i+j+k \leq d} a_{i,j,k} x^i y^j z^k.$$

Definição 4.3 *Uma* ***superfície algébrica de grau d*** *é a superfície implícita* $P_d^{-1}(0)$*, cuja notação vetorial adotada será*

$$P_d(x,y,z) = \boldsymbol{v}^t_{(x,y,z)}.\boldsymbol{a}$$

onde

$$\boldsymbol{v}_{(x,y,z)} = [1, x, x^2, ..., x^d, y, xy, ..., x^{d-1}y, y^2, xy^2, ..., z, xz, ..., z^d]^t$$

e

$$\boldsymbol{a} = [a_{0,0,0}, ..., a_{d,0,0}, a_{0,1,0}, ..., a_{d-1,1,0}, ..., a_{0,0,1}, ..., a_{0,0,d}]^t.$$

Os elemetos de $\boldsymbol{a} \in \mathbb{R}^l$ *são os coeficientes de* P_d *e os elementos do vetor* $\boldsymbol{v}_{(x,y,z)} \in \mathbb{R}^l$ *são os monômios de* P_d*, sendo* $l = \frac{(d+1)(d+2)(d+3)}{6}$.

Aproximação por superfícies algébricas

Superfícies, em geral, podem ser bem aproximadas por superfícies algébricas e, de acordo com Mederos (10), podemos aumentar o grau algébrico ou utilizar reconstruções locais de superfícies para obter maior precisão na aproximação. Como nosso objetivo é extrair a silhueta local da nuvem de pontos, optamos por não fazer a reconstrução global da superfície implícita.

Vamos utilizar uma aproximação por superfícies algébricas que minimizam o erro das distâncias até os pontos da nuvem, baseado no que foi proposto por Mederos (10). Por conta disso, a reconstrução retorna um resultado satisfatório mesmo em nuvens com ruídos, como veremos no capítulo de resultados.

Definição da vizinhança de um ponto para a reconstrução

Diferente do que propõe Mederos (10), que usa a estrutura de *octree*, optamos por reconstruir uma superfície implícita em torno de cada ponto da nuvem. Assim, para cada ponto P da nuvem, construímos uma *bounding box*, denotada por bb_P e delimitada pelos k pontos da nuvem mais próximos de P, fixado $k \in \mathbb{N}$, segundo a distância euclidiana. O valor de k é fixo e em nossos exemplos testamos diferentes valores.

Para uma boa reconstrução, é necessário um número razoável de pontos dentro de cada *bounding box*, dado que a superfície é uma aproximação desses pontos. Porém, uma vez que pontos próximos têm muitos vizinhos em comum, há auto-interseção das *bounding boxes* (Figura 4.1) e, consequentemente, das superfícies reconstruídas. Na seção 4.2 mostraremos que, para extrair a silhueta e evitar sobreposições de linhas, não é necessário reconstruir superfícies em todos os pontos da nuvem e que a vizinhança em torno de cada ponto será reduzida.

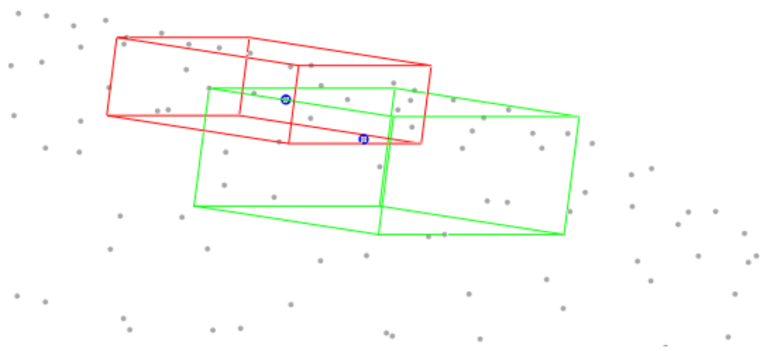


Figura 4.1: Pontos vizinhos têm muitos pontos próximos em comum, o que gera auto-interseção de suas *bounding boxes*. Aqui, mostramos as caixas de dois pontos vizinhos destacados em azul.

Métodos de reconstrução

Dada uma *bounding box*, aplicamos dois métodos de reconstrução: *Gradient one fitting* (3) e *Ridge regression* (13).

Para aplicar estes métodos, consideramos como dado de entrada uma nuvem de pontos com suas normais unitárias.

O método *Gradient one fitting* é descrito como um problema de mínimos quadrados com pesos, utilizado para evitar problemas de continuidade e sensibilidade causados por pequenas perturbações que podem ser feitas na nuvem de pontos (10). Dessa forma, a utilização deste método se mostra bastante conveniente para a nossa proposta de extrair a silhueta de uma nuvem de pontos com ruídos.

Reconstruir a superfície $P_d^{-1}(0)$ que aproxima a nuvem em cada *bounding box* depende da determinação do vetor dos coeficientes do polinômio P_d, o vetor $\mathbf{a}$. Para isso, em cada *bounding box* baseado em Mederos (10), resolvemos o problema dado por:

$$min_{\mathbf{a}}\{\mathbf{a}^t(\mathbf{S} + \mu\mathbf{S_N})\mathbf{a} - 2\mu\mathbf{a}^t\mathbf{g_N} + \mu k\},$$

onde:

- k é o número de pontos na *bounding box*;
- $\mathbf{S} = \sum_{i=1}^{k} \frac{1}{k}\mathbf{v}_i.\mathbf{v}_i^t$ é uma matriz $l \times l$
- $\mathbf{g_N} = \sum_{i=1}^{k} D_i n_i$, sendo $D_i = [\frac{\partial \mathbf{v}_i}{\partial x}\ \frac{\partial \mathbf{v}_i}{\partial y}\ \frac{\partial \mathbf{v}_i}{\partial z}]$ e n a normal do ponto;
- $\mathbf{S_N} = \sum_{i=1}^{k} \mathbf{g_N}\mathbf{g_N}^t$ é uma matriz $l \times l$.

A solução deste problema é obtida pela resolução do sistema linear

$$(\mathbf{S} + \mu\mathbf{S_N})\mathbf{a} = \mu\mathbf{g_N}$$

Desta forma, usando apenas o método *Gradient one fitting*, já temos uma reconstrução da superfície na vizinhança de cada ponto da nuvem. Porém, em alguns exemplos foi determinada mais de uma componente em uma mesma *bounding box*, o que não desejamos. Para evitar isto, utilizamos uma combinação dos métodos *Gradiente one fitting* e *Ridge regression*, semelhante ao que foi proposto em (10), com algumas simplificações.

Ao combinar estes dois métodos, o sistema a ser resolvido passa a ser

$$(\mathbf{S} + \mu\mathbf{S_N} + \gamma\Lambda)\mathbf{a} = \mu\mathbf{g_N}$$

onde γ é uma constante real que determina o peso dado ao novo termo e Δ é uma matriz diagonal $l \times l$, como propõe (13):

$$\Lambda = diag(\delta_i), \delta_i = \frac{\sigma_i}{\sigma_i + \gamma}$$

onde σ_i são os autovalores da matriz $\mathbf{S}$.

Nos exemplos utilizados neste trabalho, em geral foi considerado $\gamma = 0$ e $\mu = 0.01$.

Desta forma, dada uma *bounding box*, determinamos uma função implícita que aproxima os pontos da nuvem nesta região.

4.2
Extração local da silhueta de uma nuvem de pontos

Nesta seção apresentamos um método de extração local da silhueta de uma nuvem de pontos, baseada no método preditor-corretor de Euler-Newton. Para determinar e visualizar a silhueta, utilizamos as superfícies reconstruídas como descrito na seção 4.1.

4.2.1
Determinação da silhueta em uma bounding box

Dado um ponto P da nuvem de pontos, considere que temos reconstruída uma superfície descrita por uma função implícita f_P conhecida, definida na *bounding box* bb_P, caixa delimitada pelos k vizinhos mais próximos do ponto P. Dessa forma, podemos calcular as normais explicitamente em qualquer ponto da superfície, derivando o polinômio.

Seja $\mathbf{g}$ o gradiente de f_P e $g = \|\mathbf{g}\|$. Então, a normal à superfície é dada por $\mathbf{n} = -\mathbf{g}^t/g$.

A determinação da silhueta sobre essa superfície depende de encontrarmos os pontos $\mathbf{s}$ tais que:

- $f_P(\mathbf{s}) = 0$ e
- $I(\mathbf{s}) = 0$, ou seja, $n_s.\mathbf{v} = 0$, sendo n_s a normal à superfície em $\mathbf{s}$ e v o vetor observador.

Portanto, definimos a função $F_P : bb_P \subset \mathbb{R}^3 \to \mathbb{R}^2$ dada por:

$$F_P(x, y, z) = (f_P(x, y, z), I(x, y, z))$$

e a curva silhueta C fica definida por $C = F_P^{-1}(0, 0)$.

Para determinar a curva silhueta C em bb_P, seguimos dois passos:

1. Encontrar um ponto inicial p_0 tal que $F_P(p_0) = (0, 0)$;

2. Utilizar um método de continuação numérica para percorrer a curva.

Para encontrar o ponto inicial p_0 sobre a curva, utilizamos o método de Newton em F_P, tendo o ponto P da nuvem como ponto de partida do método. Consideramos o ponto P uma boa escolha para inicializar o método, já que a *bounding box* bb_P foi construída com os pontos mais próximos de P. Se o método não convergir com esta condição inicial, sorteamos outro ponto em bb_P e reaplicamos Newton. Repetimos esse processo n vezes (n fixo) e, caso não haja convergência, consideramos que não há silhueta nesta caixa. Além disso, pedimos que o ponto p_0 esteja próximo de P.

O método de Newton (Figura 4.2(a)), para resolver uma equação do tipo $f(x) = 0$, com $f : \mathbb{R}^n \to \mathbb{R}^n$, se apresenta na forma

$$p_{i+1} = p_i - f'(p_i)^{-1} f(p_i).$$

Porém, no nosso caso, a matriz Jacobiana de F_P não é uma matriz quadrada e, por isso, não podemos calcular sua inversa.

Considere $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$. Aplicamos o método de Newton, como sugere (1):

$$p_{i+1} = p_i - [f'(p_i)]^{+} f(p_i),$$

sendo que $[f'(p_i)]^{+}$ corresponde à pseudo-inversa, a inversa de Moore-Penrose.

Definição 4.4 *Seja A uma matriz $n \times (n+1)$ com posto máximo. Então, a **inversa de Moore-Penrose** de A é dada por*

$$A^{+} = A^t (AA^t)^{-1}$$

O produto $X = A^{+}b$, no nosso caso $X = [f'(p_i)]^{+} f(p_i)$, pode ser resolvido utilizando a decomposição QR da seguinte forma (1):

$A^t = Q \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix}$ e $A = \begin{pmatrix} R^t & \mathbf{0} \end{pmatrix}$

Daí, $A^{+} = Q \begin{pmatrix} (R^t)^{-1} \\ \mathbf{0}^t \end{pmatrix}$.

Porém, em vez de determinar $(R^t)^{-1}$, podemos:

- Resolver o sistema $R^t y = b$ por substituição progressiva, dado que R^t é triangular inferior e
- Efetuar a multiplicação $X = Q \begin{pmatrix} y \\ \mathbf{0} \end{pmatrix}$.

Assim, temos que $p_{i+1} = p_i - Q \begin{pmatrix} y \\ \mathbf{0} \end{pmatrix}$.

Uma vez que encontramos o ponto p_0, primeiro ponto determinado sobre a curva silhueta, vamos aplicar o método preditor-corretor de continuação numérica de Euler-Newton.

Para percorrer a curva, aplicamos em p_0 o método de Euler:

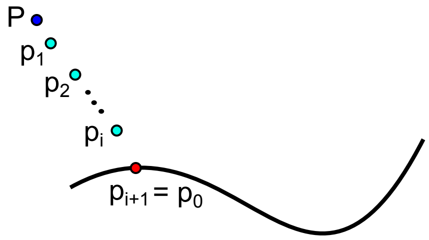
$$p_{i+1} = p_i + \lambda \mathbf{t}(p_i), i = 0, 1, 2, ...$$

sendo:

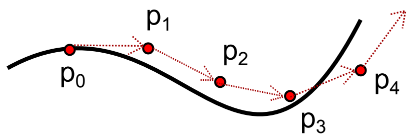
– $\lambda \in \mathbb{R}$ é o tamanho do passo do método de Euler;

– $\mathbf{t}$ o vetor tangente à curva $F_P^{-1}(0,0)$, obtido pelo produto vetorial dos gradientes ∇f_P e ∇I em p_i.

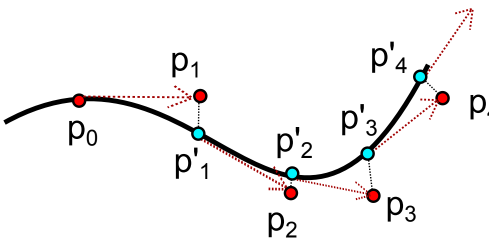
Contudo, o ponto obtido após aplicarmos o método de Euler, em geral, não pertence à curva silhueta (Figura 4.2(b)). Portanto, para retornarmos para a curva, aplicamos o método de Newton descrito anteriormente para corrigir esta distância e, assim, obtermos um ponto na curva novamente, como mostra o esquema da Figura 4.2(c).



4.2(a): Método de Newton



4.2(b): Método de Euler



4.2(c): Método Euler-Newton

Figura 4.2: Método preditor corretor de Euler-Newton.

Assim, poderíamos aplicar o método de Euler-Newton até que toda a curva fosse percorrida, ou seja, até que o método nos retornasse um ponto fora da *bounding box*. Com essa estratégia, observamos um excessivo número de linhas em uma mesma vizinhança, causado pela auto-interseção de *bounding boxes* de pontos muito próximos. Portanto, para evitar isto, extraímos a silhueta apenas em uma vizinhança reduzida do ponto inicial p_0 (Figura 4.3).

Esta vizinhança é delimitada por uma bola centrada em p_0, cujo raio é proporcional ao comprimento da diagonal da *bounding box*.

Utilizamos também um passo adaptativo para o método de Euler: se, após alguma iteração do método de Euler-Newton não conseguirmos a convergência,

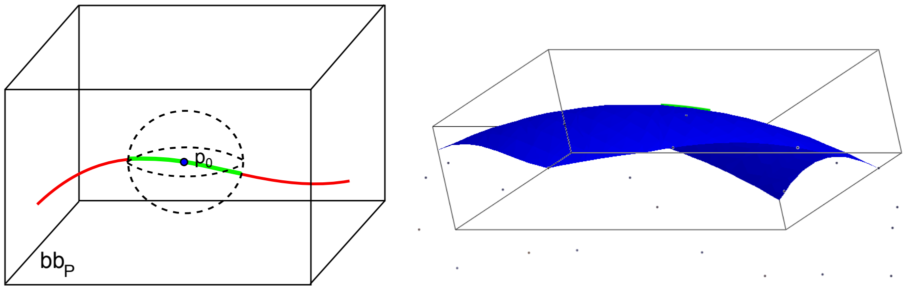


Figura 4.3: A curva em vermelho ilustra a silhueta que seria extraída se considerássemos como vizinhança toda a *bounding box*. Para reduzir sobreposições, determinamos a silhueta apenas em uma pequena vizinhança do ponto p_0 (curva em verde). Na Figura à direita, temos a superfície que foi reconstruída nesta caixa e a parte da silhueta considerada.

voltamos ao último ponto obtido sobre a silhueta e aplicamos novamente o método, reduzindo o passo utilizado. Dessa forma, conseguimos um melhor resultado, visto que, em regiões de alta curvatura, se o passo for muito grande, o método pode não convergir. Se mesmo assim, após um número de tentativas estipulado o método não convergir, finalizamos a determinação da silhueta nesta caixa.

4.2.2
Silhueta da nuvem de pontos

Para determinar a curva silhueta, definimos uma estratégia de percorrimento evitando processar as *bounding boxes* de todos os pontos da nuvem, ou seja, esta estratégia evita reconstruir a superfície e buscar pontos sobre a silhueta nas *bouding boxes* de todos os pontos da nuvem.

Inicialmente, buscamos uma primeira caixa que contenha uma curva silhueta. Para tanto, em cada uma destas caixas, devemos reconstruir a superfície e aplicar o método preditor-corretor de Euler-Newton.

Uma vez que encontramos uma *bounding box* bb_P com silhueta, vamos percorrer agora as *bounding boxes* dos vizinhos de P que estão em bb_P. Isto é, sabendo que bb_P foi delimitada pelos pontos P_1, P_2, ..., P_k vizinhos de P, vamos percorrer as caixas bb_{P_1}, bb_{P_2}, ..., bb_{P_k}, pois é natural supor que a curva silhueta passe por alguma delas, dada sua continuidade. Dessa forma, percorremos a curva, executando o processo de reconstrução apenas nas caixas de pontos vizinhos a outras caixas que contenham silhueta.

Cada caixa já visitada é sinalizada, evitando que a busca da curva silhueta seja feita novamente. Assim, se encontramos uma bounding box bb_P, cujas caixas de todos os vizinhos $P_1, P_2, ..., P_k$ já foram visitadas, finalizamos a extração desta linha da silhueta.

Para determinar outras linhas silhueta desta mesma nuvem, buscamos, dentre as caixas ainda não visitadas, outra caixa inicial com silhueta e repetimos o processo. Essa busca é feita uma quantidade fixada de vezes.

O algoritmo 4.2.2 mostra resumidamente o método descrito nesta seção.

No capítulo de resultados serão discutidas outras estratégias que foram adotadas para percorrer a curva silhueta. Além disso, mostraremos as vantagens da utilização deste método para a determinação de silhuetas em nuvens com ruídos.

Algoritmo 4.2.2:	Perseguindo a silhueta
1:	Para cada ponto Q da nuvem, gerar bb_Q.

2:	Para cada bb_Q,
	    reconstruir a superfície implícita;
	    aplicar o método de Euler-Newton;
	    se encontrar silhueta,
	        guardar os vizinhos de Q não visitados no vetor u;
	        fim Para.

3:	Enquanto bb_P for uma caixa não visitada, para cada $P \in u$,
	    reconstruir a superfície implícita;
	    aplicar o método de Euler-Newton;
	    se encontrar silhueta,
	        guardar os vizinhos de P não visitados no vetor u;
	        retirar P de u.