Um pouco sobre visualização de imagens médicas

Prof. Alexandre Xavier Falcão

Aula 05 - Parte II

1 Projeção

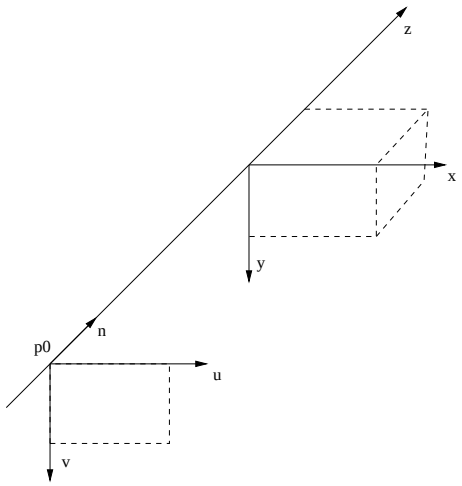

Figura 1: Sistemas de coordenadas da imagem 3D (x, y, z) e do plano de visualização (u, v, n) em $p_0 = (0, 0, -D/2)$ em relação ao (x, y, z), onde D é a diagonal da imagem 3D (para evitar cortes).

No caso de imagens médicas, o observador busca rotacionar a imagem 3D em torno do seu centro e depois projetar os voxels no plano de visualização (Figura 1) [1]. Isto equivale a uma translação $\mathbf{T}(-p_c)$, onde $p_c = (x_c, y_c, z_c)$ é o centro da imagem 3D, seguida normalmente de rotações em torno de x (tilt) e y (spin), $\mathbf{R_y}(\theta_y)\mathbf{R_x}(\theta_x)$, e da projeção no plano de visualização (com eliminação de voxels escondidos durante a projeção). As coordenadas (u, v) da projeção são então transladadas de $(D/2, D/2)$ para que a imagem final tenha apenas coordenadas positivas. As equações abaixo representam a transformação que leva um voxel (x_1, y_1, z_1) a um

[1] Observador e fonte de luz no $z = -\infty$ e raios de projeção paralelos e orgonais ao plano de visualização.

pixel (u, v) na imagem final.

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \mathbf{R_y}(\theta_y)\mathbf{R_x}(\theta_x)\mathbf{T}(-p_c)\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

$$\begin{aligned} u &= x_2 + D/2 \\ v &= y_2 + D/2 \end{aligned}$$

1.1 Remoção de voxels escondidos

A remoção de voxels escondidos está associada à opacidade que atribuimos aos voxels. No caso mais simples, temos um objeto opaco na imagem 3D e, portanto, apenas os voxels mais próximos do plano de visualização devem ser apresentados (voxels da superfície do objeto).

A distância do voxel ao plano de visualização no instante da projeção é dada por $(z_2 + D/2)$. Esta distância é usada para selecionar o voxel mais próximo do plano. A seleção é feita com o auxílio de uma imagem auxiliar $\hat{Z} = (D_Z, Z)$ $(z - buffer)$, $|D_Z| = D \times D$, a qual mantém armazenada em $Z(u, v)$, a menor distância entre o plano e o último voxel mais próximo, que foi projetado nesta posição.

A geração da imagem final requer propriedades associadas aos voxels projetados. Estes voxels podem ser diretamente acessados na imagem 3D se nós também mantivermos uma imagem de indexação $\hat{Ib} = (D_{Ib}, Ib)$ $(indexingbuffer)$, onde $|D_{Ib}| = D \times D$ e $Ib(u, v)$ armazena o índice do voxel projetado.

2 Tonalização

O processo de projeção com formação da imagem final $\hat{J} = (D_J, J)$, $|D_J| = D \times D$, é denominado *rendering*. Neste caso temos o rendering da superfície do objeto 3D. A intensidade do pixel $p = (u, v)$ correspondente em $\hat{J}$ é calculada pela fórmula de Phong:

$$\begin{aligned} J(p) &= 255k_a + I_Z(p)(k_d \cos(\alpha) + k_s \cos^{n_s}(2\alpha)) \\ I_Z(p) &= 255(1 - \frac{Z(p)}{D}) \end{aligned}$$

onde $Z(p)$ é a distância armazenada no $z - buffer$, $I_Z(p)$ é a intensidade baseada em profundidade (depth shading), $0 \leq k_a \leq 1$ (e.g., 0.1) é a constante de ambiente, $0 \leq k_d \leq 1$ (e.g., 0.7) é a constante de reflexão difusa, $0 \leq k_s \leq 1$ (e.g., 0.2) é a constante de reflexão especular, $k_a + k_s + k_d = 1$, e α é o ângulo entre o vetor normal à superfície no ponto (x_2, y_2, z_2) e o vetor de visualização $\vec{V} = (0, 0, -1)$ que aponta para o observador e fonte de luz (α é obtido pelo produto interno entre estes vetores). O expoente n_s (e.g., 5) aumenta e diminui o efeito especular. O voxel mais próximo só é projetado se $0 \leq \alpha < \pi/2$ e a parte especular só é calculada se $0 \leq \alpha < \pi/4$.

2.0.1 Vetor normal

O vetor normal $\vec{N}$ pode ser calculado para cada voxel $p = (x_1, y_1, z_1)$ da superfície do objeto 3D na imagem $\hat{I} = (D_I, I)$ pela equação de gradiente abaixo, quando existe forte contraste entre o brilho do objeto e o fundo.

$$\vec{N}(p) = \sum_{\forall q \in A(p)} (I(q) - I(p))\vec{pq}$$

onde $A(p)$ é uma adjacência esférica (e.g., raio 1.8) com centro em p e $\vec{pq}$ é o vetor unitário na direção de p para o adjacente q. Note que o vetor gradiente pode estar apontando para dentro ou para fora do objeto. Para o cálculo de α, ele deve ser considerado sempre para fora.

Antes de usar o vetor normal de um voxel, devemos lembrar que ele sofre a mesma rotação $\mathbf{R_y}(\theta_y)\mathbf{R_x}(\theta_x)$. Esta rotação deve ser aplicada antes do produto interno com $\vec{V}$.

Em alguns casos (e.g., ossos em ressonância magnética), o vetor gradiente não é uma boa aproximação do vetor normal. Podemos então calcular a média do produto externo entre vetores $\vec{pq}_i$ e $\vec{pq}_{i+1}$ de p a voxels adjacentes consecutivos, q_i e q_{i+1}, em $A(p)$, onde $A(p)$ considera apenas os voxels da superfície do objeto que estão em uma adjacência esférica em torno de p.

$$\vec{N}(p) = \frac{1}{n} \sum_{i=1}^{n} \vec{pq}_i \times \vec{pq}_{mod(i+1,n)}$$

onde n é o número de voxels adjacentes em $A(p)$.

Os vetores normais podem ser previamente calculados e quantizados em uma *look-up table*, onde consideramos amostras de possíveis direções em torno do centro de uma esfera de raio unitário. Neste caso, armazenamos apenas o índice da *look-up table* para cada voxel. O rendering também ficará mais rápido se armazenarmos a lista de voxels da superfície do objeto.

3 Múltiplos objetos, opacidade e cor

Para visualizar k objetos 3D com transparência, podemos considerar o rendering separado de cada um deles e depois combinar as imagens finais usando suas opacidades $0 \leq \lambda_i \leq 1$, $i = 1, 2 \ldots, k$, e as distâncias nos seus $z - buffers$. As opacidades são definidas de acordo com o nosso interesse na visualização.

$$J(p) = \sum_{i=1}^{k} \lambda_i J_i(p) \prod_{j=1}^{i-1} (1 - \lambda_j)$$

onde $J_i(p)$ é a intensidade do objeto O_i no pixel $p = (u, v)$ e os objetos são reordenados pelas distâncias ao plano, (i.e., $Z_i(p) \leq Z_{i+1}(p)$, $i = 1, 2, \ldots, k-1$) para cada pixel p.

As intensidades $J_i(p)$ calculadas podem também ser multiplicadas por coeficientes $0 \leq k_r \leq 1$, $0 \leq k_g \leq 1$, e $0 \leq k_b \leq 1$ para gerar imagens coloridas, onde a cor de cada objeto é definida por uma tripla (k_r, k_g, k_b) diferente. Neste caso, a equação acima deve ser aplicada separadamente para cada banda do vermelho, verde, e azul.

Algoritmos mais sofisticados (e.g., *raycasting*) e estruturas de dados especiais para acesso ordenado dos voxels da imagem 3D (e.g., *shell*) têm sido propostos para tornar o rendering de múltiplos objetos mais eficiente. Outros métodos, tais como *volume rendering* e projeção de intensidade máxima evitam a segmentação dos objetos, e portanto, são limitados a algumas aplicações.

4 Projeção de máxima intensidade

Uma técnica mais simples de visualização, muito útil para se ter idéia da distribuição de vasos sangüíneos em imagens de angio ressonância magnética e de ossos em imagens de tomografia de raios-X, é a projeção de máxima intensidade (MIP). Neste caso, a intensidade $J(p)$ é dada pelo brilho máximo de todos os voxels projetados em p (i.e., não existe remoção de voxels escondidos nem conceito de objeto).

5 Evitando "buracos" em transformações geométricas

Em todas as transformações vistas, o cálculo direto da transformação pode gerar uma imagem $\hat{J} = (D_J, J)$ onde alguns pixels/voxels não são associados a nenhum pixel/voxel da imagem inicial $\hat{I} = (D_I, I)$, gerando "buracos"na imagem final (Figura 2). Estes buracos podem ser evitados com a técnica de *splatting* que associa um pixel/voxel de $\hat{I}$ a mais de um pixel/voxel de $\hat{J}$. No caso da projeção, por exemplo, podemos projetar cada voxel em 3×3 pixels.

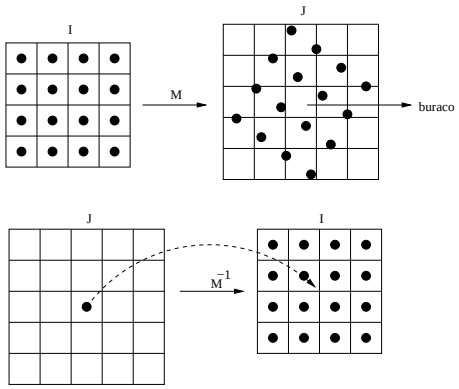

Figura 2: Buracos podem ser evitados por interpolação quando aplicamos $D_J = M^{-1}D_I$ em vez de $D_I = MD_J$.

A melhor opção, no entanto, é fazer o processo inverso. Para cada pixel p de $\hat{J}$, nós aplicamos a transformação inversa e encontramos as coordenadas adjacentes em $\hat{I}$, para as quais conhecemos as propriedades (brilho, normal, etc). A propriedade associada a p é obtida por interpolação dos valores desta propriedade em $\hat{I}$. No caso da projeção, a inversa é usada no algoritmo de *ray casting*. No caso de outras transformações, nós aplicamos a transformação inversa e encontramos o brilho por interpolação (próxima aula).