Projeto 4: Surface Rendering

Marcos Teixeira

19 de junho de 2018

1 Descrição

Neste trabalho a tarefa é implementar o algoritmo de Cortes Curvilinear. Nesta abordagem, ao invés de calcular a iluminação de Phong [1] quando encontrarmos um ponto P válido do objeto, nós verificamos se $D(P) > t$, onde t é um threshold estabelecido e, em caso positivo, retornamos o brilho da imagem em P. Desta forma conseguimos visualizar a textura de isosuperfícies de corte de um objeto. Um exemplo desta técnica é apresentada na Figura 1.

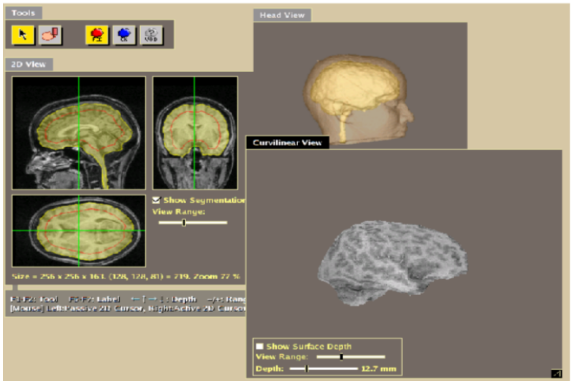

Figura 1: Exemplo de um corte de uma isosuperfície de um cérebro.

2 Cortes por Isosuperfícies

Inicialmente, precisamos aplicar um fechamento morfológico após a segmentação do objeto que estamos querendo analisar, gerando um envelope. Posteriormente, definimos um intervalo pequeno em torna da isosuperfície dado através de uma certa distância d do envelope. Por fim, executamos o algoritmo já conhecido de traçado de raios para cada $qinD_j$, de modo a projetar o brilho $I(p)$ do primeiro ponto p que satisfaz esse limite de distância. Neste projeto nós já recebemos de entrada a imagem de envelope do objeto que vamos considerar para os testes. De modo geral, esta técnica se mostra bem útil no auxílio a detecção de displasias corticais focais em pacientes com Epilepsia [2].

O pipeline de execução do alogoritmo de Corte Curvilinear utilizando pode ser expresso da seguinte maneira:

i Criar a imagem de saída R(D, D)

ii Criar a matriz de transformação M_{-1}

iii Definição das faces da cena

iv Calcular a Transformada de Distância Euclidiana (TDE)

v Aplicar M_{-1} na matriz normal

vi Para cada ponto p_i do plano de visualização, faça

(a) Aplicar M_{-1} em p_i

(b) Se houver intersecção

i. Calcular os λ válidos que dam origem a $p1$ e pn

ii. Aplicar o DDA para percorrer os pontos $p_i \in p1, ..., pn$ em direção a cena

iii. Ao encontrar um ponto p_i válido da superfície do objeto, onde $TDE(p_i) > t$, onde t é um threshold

A. Retornar o brilho de p_i

(c) Caso contrário, $R = 0$

vii Retornar R

, onde a matriz M_{-1} = T(N_x/2,N_y/2,N_z/2) * $R(\theta_x)$ * $R(\theta_y)$ * T(-D/2,-D/2,-D/2), com $D = \sqrt{nx^2 + ny^2 + nz^2}$. Como são várias estruturas que precisam ser manipuladas durante a execução do Surface Rendering, nós optamos por criar uma estrutura chamada GraphicalContext para facilitar a composição das funções. A estrutura do contexto gráfico é apresentado no pedaço de código 1.

```
typedef struct gc {
  int                numberOfObjects;
  ObjectAttributes  *object;
  iftMatrix         *Tinv;
  iftVector          vDir;
  iftImage          *tde;
  iftImage          *scene;
  iftImage          *label;
  iftVolumeFaces    *faces;
  iftImage          *normal;
} GraphicalContext;
```

Listing 1: Estrutura do Contexto Gráfico

Para cada uma das 6 faces, realizamos os cálculos apresentados na equação abaixo. Para cada um dos lambdas obtidos, nós pegamos o menor e o maior válidos para conseguir obter $p1$ e pn. O valor de menor lambda será atribuído a equação da reta para encontrar $p1$ e o maior lamba para encontrar pn. Note que é necessário garantir que $(\vec{n'} \cdot \vec{n_j})$ não sejam 0 para que os testes das equações acima funcionem.

$$\lambda_j = ((c_j - p_0) \cdot \vec{n_j})/(\vec{n'} \cdot \vec{n_j}),$$
$$p_{\prime} = p_0 + \lambda \vec{n}$$

Posteriormente, o algoritmo DDA será utilizado para encontrar todos os pontos da linha entre $p1$ e pn. Ao encontrar um ponto que p_i válido e visíveil, comparamos o valor correspondente ao voxel na TDE, verificando se é maior que um threshold t determinado. Quanto maior for o valor de t, menor será a superfície renderizada. O método de interpolação adotado é descrito na Figura 2 e é implementado na função *iftImageValueAtPoint()* da biblioteca ift.

3 Instruções

Para execução do programa **CL.c**, siga a seguinte formatação :

```
1.  $ ./CL input.csn
            input-label.csn
            output-image.png
            tilt
            spin
            threshold
```

4 Resultados

Nesta seção apresentamos os resultados obtidos na imagem de teste **brain2.scn** As imagens de saída são da forma (D, D), onde D é igual a $D = \sqrt{nx^2 + ny^2 + nz^2}$, representando a diagonal da imagem. Para padronizar as figuras nesse relatório, dado que esse D depende de cada imagem, nós limitamos a altura para 5cm e a largura para 7.5cm. Uma normalização de (0,255) é feita após a finalização do algoritmo de tonalização de Cortes Curvilinear, para melhor visualização. Inicialmente, o plano de visualização é reposicionado em $z = D/2$, para afastar um pouco da visão do observador.

Na Figura 2 mostramos os resultados do algoritmo para rotações de $R_x = 0$, $R_y = 0$, considerando $t = 0, 50, 200$. Como esperado, ao aumentar o threshold de distância os cortes vão ficando mais internos ao objeto. Note que para $t = 0$ ficam vários "buracos"na imagem resultante. De modo complementar, apresentamos resultados na Figura 3 para $R_x = 90$, $R_y = 90$ para os mesmos thresholds escolhidos anteriormente. É possível notar

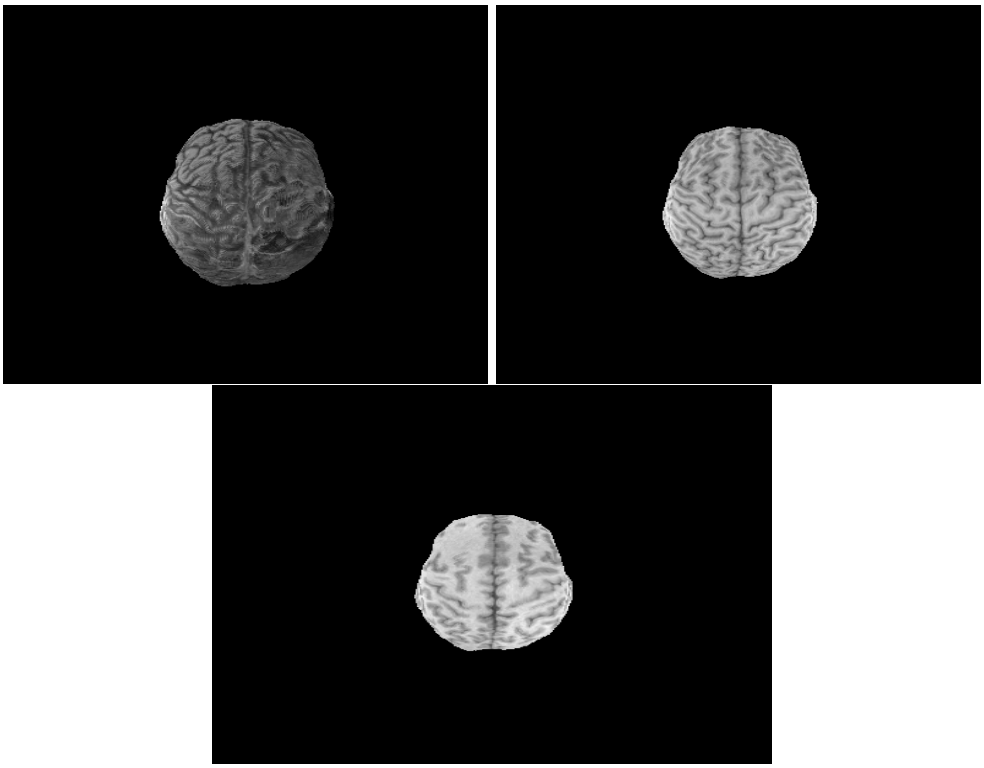

Figura 2: Resultados dos cortes de isosperfícies para thresholds de $t = 0, 20, 200$, com angulação (0,0)

que quando aumentamos o threshold das distâncias, a medula some da imagem final, devido ao "afinamento"do objeto.

5 Conclusão

Neste trabalho foi apresentado uma implementação para o algoritmo que realiza cortes curvilineares em imagens 3D. Através da teoria vista em sala de aula, fizemos uma modelagem que levasse em conta todos os passos do algoritmo de forma simples e objetiva. Os experimentos foram conduzidos levando em consideração os detalhes de precisão e de conversão, aprendidos através das experiências com os trabalhos anteriores. Este trabalho não apresentou muitos desafios, sendo implementado com poucas alterações a partir do algoritmo de iluminação de Phong. Foi necessário utilizar a biblioteca compilada para Mac OS disponibilizada pelo monitor para o código. Além disso, mudamos também a versão do gcc no Makefile para o gcc-7, para poder funcionar.

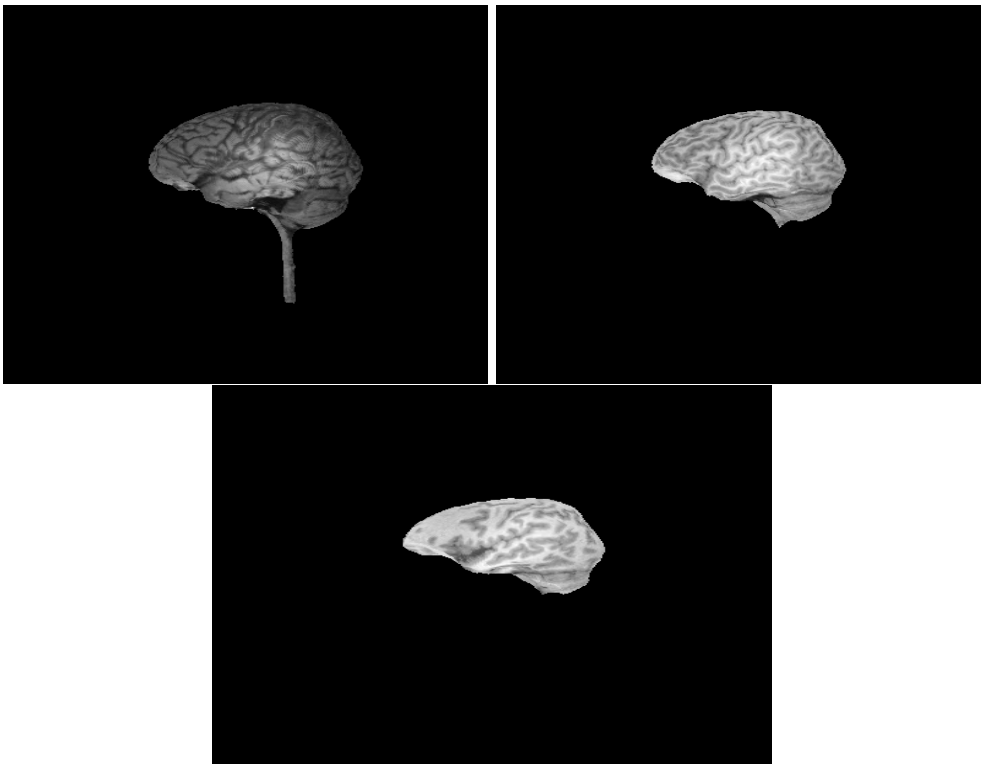

Figura 3: Resultados dos cortes de isosperfícies para thresholds de $t = 0, 20, 200$, com angulação (90,90).

Referências

[1] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975.

[2] A. Falcão, "Notas de aula mo815 visualização de imagens volumétricas," *Unicamp*.