UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Relatório Projeto 2

Geração de Malhas - SME - 5827

Professor: Antônio Castelo
Alunos: Filomen Incahuanaco Quispe
Jaqueline Alvarenga Silveira
Rosalía Taboada Leiva

Novembro
2017

1 Introdução

A rasterização é processo de conversão de algo que é contínuo como, por exemplo, um segmento de reta, numa coleção de quadrículas que tem natureza discreta. Um rasterizador determina como colorir pixels do limite de um polígono, mas os pixels próximos ao contorno são difíceis de classificar. Simplesmente colorir os pixels internos a um polígono de uma cor e os externos de outra cor, faz perder parte da informação existente e o resultado disso é uma imagem rasterizada com bordas irregulares, o qual é uma forma de *aliasing* como pode ser observado na Figura 1.

Fig. 1: Rasterização com *aliasing*.

Aliasing é um termo geral para qualquer efeito de amostragem de um sinal que contém frequências superiores à metade da taxa de amostragem. Na rasterização de polígono, o sinal é uma função χ_M que é 1 dentro do polígono e 0 fora. Além dos polígonos rasterizantes, pode-se rasterizar os limites curvos. Por exemplo, fontes e imagens vetoriais geralmente descrevem regiões delimitadas por curvas de *Bézier*. No entanto, mesmo os métodos que calculam o preciso *anti-aliasing* de polígonos apenas aproximam curvas por peças lineares, o que leva a artefatos de *aliasing* para essas formas curvas.

Embora normalmente pensemos em rasterização como um problema 2D, podemos estender a ideia ao 3D. Em 3D, a operação equivalente é calcular a ocupação das células do cubo a partir dos triângulos que encerram um volume. Essa rasterização volumétrica (*voxelização*) de um objeto cria uma representação implícita que é útil para operações, como detecção de colisões, geometria construtiva sólida e simulação de fluido, que é mais fácil de calcular sobre volumes que sobre limites.

Em simulações de fluidos, é natural modelar a tensão superficial usando uma malha de superfície, enquanto a pressão e a advecção são calculadas melhor em uma grade volumétrica. Juntamente com métodos de contorno de superfície, como *Marching Cubes* ([1]), métodos eficientes para implantar objetos podem acelerar simulações de fluidos com interfaces ar-água ([2]). O cálculo de ocupação celular precisa é importante para preservar o volume na simulação e para manter uma superfície lisa.

Com isto, neste trabalho apresentaremos uma comparação entre os diversos métodos existentes *anti-aliasing* de imagens em 2D e 3D. Os métodos trabalhados que discutiremos são *Haar Wavelets*, *Box Filter*, *Median Filter*, *Sinc* e *método Gaussian* para 2D e para 3D veremos o método de *Haar Wavelets* com *Marching Cubes*.

2 Referencial Teórico

Nesta seção descreve-se os métodos utilizados para a realização deste trabalho.

2.1 Flood Fill

O algoritmo *flood fill* [3] é um algoritmo de rasterização de imagens. O algoritmo inicialmente precisa de três parâmetros: um ponto que esteja dentro do contorno do objeto, a cor que representa o fundo da imagem e a nova cor que o algoritmo *flood fill* usará para recolorir os pixels. O algoritmo procura por todos os pontos que estão conectados ao inicial por um caminho da cor que representa o fundo da imagem e então recolore os pixels com a nova cor.

2.2 Box Filter

Box Filter [4] é um filtro para suavizar imagens, ou seja, reduzir a quantidade de variação de intensidade entre um pixel e o próximo. É usado para reduzir ruídos em imagens. A ideia do filtro é substituir cada valor de um pixel na imagem com o valor médio dos seus vizinhos, incluindo ele mesmo. Isto tem efeito de eliminar valores de pixel que não são representativos ao seu redor. Em geral, é usado com um filtro de convolução e assim como outras convoluções é baseado em torno de um *kernel*, o qual representa a forma e o tamanho da vizinhança a ser amostrada ao calcular o filtro. Neste trabalho, nós usamos um *kernel* como pode ser observado na Figura 2. A escolha do tamanho do *kernel* foi para não engrossar muito as bordas.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Fig. 2: *Kernel* 3×3 usado no *Box Filter* deste trabalho.

2.3 Gaussian Blur

Um *Gaussian Blur*[5] é o resultado de borrar uma imagem por meio de uma função gaussiana. O efeito visual desta técnica de desfocagem é um borrão suave que se assemelha o de visualizar a imagem através de uma tela translúcida, distintamente diferente do efeito *bokeh* (quando parte da imagem fica borrado) produzido por uma lente fora do foco ou a sombra de um objeto sob iluminação usual. Matematicamente, aplicar um borrão gaussiano a uma imagem é o mesmo que convolver a imagem com uma função Gaussiana. A equação de uma função Gaussiana em duas dimensões é dada por:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e \frac{x^2 + y^2}{2\sigma^2} (1)$$

2.4 Median Filter

Median Filter [6] é muito usado para remover ruído de uma imagem ou sinal. *Median Filter* é amplamente usada em processamento digital de imagem, sob certas condições, ele preserva *edges* enquanto remove ruído. *Edges* ocorrem quando o brilho da imagem muda bruscamente e são de importância crítica para aparência visual da imagem. A principal ideia do algoritmo é percorrer o sinal de entrada pela entrada, substituindo cada entrada pela mediana das entradas

vizinhas. A vizinhança padrão é chamada de "janela", que desliza, entrada por entrada em todo sinal.

2.5 Sinc Filter

Sinc filter[7] é um filtro que remove todos os componentes de uma frequência de corte, sem afetar baixas frequências, tem resposta de fase linear. Isso ocorre porque a função *sinc* é a transformada inversa de *Fourier* da função retangular. A multiplicação da representação de frequência de um sinal com uma função retangular pode ser usada para gerar a resposta de frequência ideal, pois remove completamente as frequências acima do ponto de corte. E, uma vez que a multiplicação no domínio da frequência é equivalente com a convolução no domínio do tempo, o filtro *sinc* tem exatamente o mesmo efeito. A equação de uma função *sinc* em duas dimensões é dada por:

$$sinc(x, y) = \frac{sin(\pi x) sin(\pi y)}{\pi^2 xy}. \tag{2}$$

2.6 Wavelet - Haar Wavelet

Wavelets fornecem uma base para representar funções por meio de uma hierarquia de refinamentos localizados. Eles tem uma ampla variedade de aplicações que vai de resolver equações diferenciais até processamento digital de imagem, processamento de sinais e reconstrução de superfícies [8]. A principal vantagem de *wavelets* sobre outras representações de uma função é que *wavelets* estão localizados tanto no domínio espacial quanto no da frequência.

Deseja-se rasterizar objetos ao calcular o percentual de ocupância dos *voxels* em um grid regular. Se a área M é o conjunto de pontos dentro de um objeto com borda ∂M, representado por um conjunto de arestas λ_M é definido por:

$$\lambda_M = \begin{cases} 1, & p \in M \\ 0, & \text{caso contrário} \end{cases} \tag{3}$$

Esta função é uma representação implícita de M da qual pode-se extrair a borda de λ_M ao encontrar os pontos onde existem transições de 0 para 1. Em particular, λ_M define o conjunto de pontos que devem ser desenhados se o polígono ∂M foi rasterizado em uma resolução infinita. Levado ao limite somando super amostras sobre um pixel é equivalente a aplicar um *box filter* ou integrar λ_M sobre a área do pixel [8]. O valor de um pixel P, é portanto, dado por:

$$\frac{\int_P \lambda_M(p) dp}{\int_P dp}. \tag{4}$$

Esta equação mostra que o valor de um pixel, é idealmente, é igual a área do polígono que intersecta o pixel dividido pela área do pixel. A abordagem *wavelet* para rasterizar polígonos é calcular os coeficientes *wavelet* de λ_M para a resolução do pixel e então inverter a transformada de *wavelet* para completar a rasterização.

Wavelets fornecem uma base ortonormal que permite refinamentos locais ao adicionar funções base de alta resolução. *Wavelets* são representados por uma função escalar ϕ e uma função *wavelet* Ψ.

Manson [8] usa a seguinte construção de *wavelets* bidimensionais. Seja $\Psi^0 = \phi$, $\Psi^1 = \Psi$, E' o conjunto de vértices $\{(0,0),(0,1),(1,0),(1,1)\}$ e $E = E' \backslash \{(0,0)\}$. Para cada $e = (e_x, e_y) \in E'$, $j \in \mathbb{N}$ e $k = (k_x, k_y) \in \mathbb{Z}^2$, Manson [8] define

$$\Psi^e_{j,k}(p) = 2^j \Psi^{e_x}(2^j p_x - k_x) \Psi^{e_y}(2^j p_y - k_y) \tag{5}$$

onde $p = (p_x, p_y)$. Para toda função g que é localmente integrável em $\mathbb{R}^2$ tem a expansão *wavelet*

$$g(p) = \sum_{k\in\mathbb{Z}^2} c_{0,k}^{(0,0)} \Psi_{0,k}^{(0,0)}(p) + \sum_{j\in\mathbb{N}} \sum_{k\in\mathbb{Z}^2} \sum_{e\in E} c_{j,k}^{e} \Psi_{j,k}^{e}(p)dp, \tag{6}$$

onde cada $c_{j,k}^{e}$ é dado por

$$c_{j,k}^{e} = \iint_{\mathbb{R}^2} g(p)\Psi_{j,k}e(p)dp. \tag{7}$$

aqui, o índice e indica qual das funções base é usada e k denota sua translação em uma resolução j. Se considerarmos os coficientes de *wavelet* de λ_M e usar a definição de λ_M na equação 3, a equação 7 reduz a

$$c_{j,k}^{e} = \iint_{M} \Psi_{j,k}^{e}(p)dp. \tag{8}$$

Integrar sobre o domínio de M é difícil. Usa-se o teorema da divergência para referir a integral sobre M para uma integral sobre sua borda ∂M. O teorema da divergência afirma que:

$$\iint_{M} \bigtriangledown . F_{j,k}^{e}(p)dp = \oint_{p\in\partial M} F_{j,k}^{e}(p).n(p)d\sigma. \tag{9}$$

onde $F = (f_x, f_y)$ de valor vetorial em $\mathbb{R}^2$, n(p) é a unidade externa normal para a curva ∂M no ponto p e $d\sigma$ é o tamanho diferencial de ∂M. Ao encontrar funções $F_{j,k}^{e}$ que satisfaz $\bigtriangledown . F_{j,k}^{e} = \Psi_{j,k}^{e}$, pode-se calcular os coeficientes *wavelet* de λ_M usando apenas a borda do polígono nas integrais de linha

$$c_{j,k}^{e} = \sum \int_{0}^{1} F_{j,k}^{e}(P_i(t)).n(P_i(t))\|P_i^{'}(t)\|dt \tag{10}$$

onde P_i representa o i^{th} segmento polinomial da borda.
As funções F para o caso 2D são:

$$\begin{aligned} F^{(0,0)} &= \frac{1}{2}(\Phi(p_x)\phi(p_y), \phi(p_x)\Phi(p_y)) \\ F^{(1,0)} &= (\Psi(p_x), 0) \\ F^{(0,1)} &= (0, \Psi(p_y)) \\ F^{(1,1)} &= (\Psi(p_x)\psi(p_y), 0) \end{aligned} \tag{11}$$

onde a função escalar

$$\phi(t) = \begin{cases} 1, & 0 \leq t \leq 1 \\ 0, & \text{caso contrário} \end{cases} \tag{12}$$

gera a base *Haar*, e ψ é dado por

$$\psi = \phi(2t) - \phi(2t-1) \tag{13}$$

Mais detalhes sobre a rasterização utilizando *wavelets* pode ser encontrados no paper de Manson [8].

2.7 Marching Cubes

Marching cubes é um algoritmo da computação gráfica, publicado em 1987 nos artigos do SIGGRAPH por Lorensen e Cline, para a extração de malhas geométricas de isosuperfícies a partir de um campo escalar tridimensional (algumas vezes chamados de *voxels*). Esse artigo é um dos mais citados no campo da computação gráfica. As aplicações desse algoritmo são mais focadas com visualização médica, tais como tomografia computacional axial e leitura de imagens por ressonância magnética, assim como efeitos especiais ou modelagem 3D com o que se chama metaballs ou outras meta-superfícies.

O algoritmo prossegue através do campo escalar, levando oito locais vizinhos ao mesmo tempo (formando assim um cubo imaginário), determinando o(s) polígono(s) necessário(s) para representar a parte da isosuperfície que passa por este cubo.

Cada vértice dos polígonos gerados é colocado na posição apropriada ao longo da borda do cubo interpolando linearmente os dois valores escalares que estão conectados por essa borda.

O gradiente do campo escalar em cada ponto da grade também é o vetor normal de uma isosuperfície hipotética que passa desse ponto. Portanto, essas normais podem ser interpoladas ao longo das bordas de cada cubo para encontrar as normas dos vértices gerados que são essenciais para sombrear a malha resultante com algum modelo de iluminação.

3 Metodologia

Neste trabalhamos nós estudamos os artigos [9], [8] e [10] e replicamos os testes destes artigos. Sendo assim, temos duas abordagens uma 2D e uma 3D. O pipeline 2D e 3D do trabalho do Manson [8] pode ser observado na Figura 4. Na Figura 3 pode-se observar o pipeline para gerar os coeficentes *wavelet*. Este pipeline corresponde ao trabalho de Manson [9] e tem como objetivo reconstruir uma função indicadora discreta, de uma nuvem de pontos, ao aproximar os coeficientes *wavelet* da função indicadora λ_M. O *grid* gerado por este trabalho é usado como entrada para outro trabalho de Manson [10]. Este trabalho consiste em modificar o *Marching Cubes* para trabalhar melhor do que o *Marching Cubes* original com a função indicadora, visto que o mesmo gera artefatos ao tentar extrair a superfície. O pipeline deste trabalho pode ser observado na Figura 4 na rasterização 3D. Para a rasterização 2D, precisa-se apenas do contorno da imagem que deseja-se rasterizar como pode-se observar na Figura 4 na rasterização 2D.

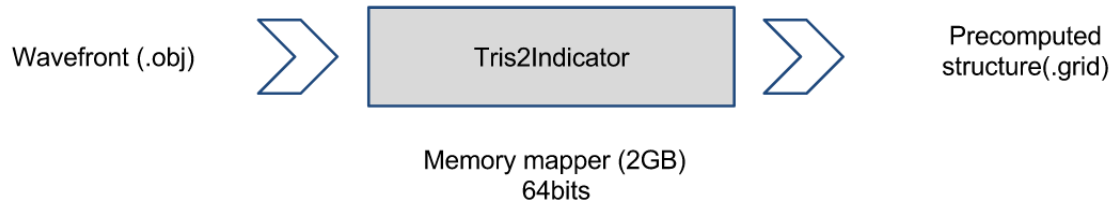


Fig. 3

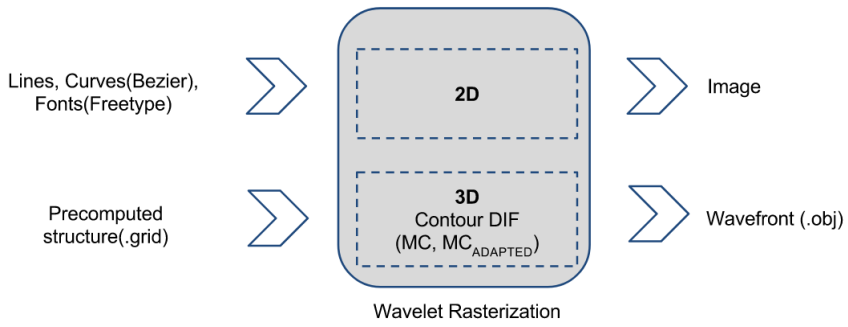


Fig. 4

3.1 Abordagem 2D

Na abordagem 2D, o Manson [8] menciona que o método proposto é mais eficiente do que determinados filtros de processamento de imagem. No entanto, os autores não mostram em nenhum momento o comportamento de outros filtros em relação ao método proposto por eles. Sendo assim, para confirmar essa afirmação feita pelas autores, replicamos os testes realizados pelos autores no seu próprio método, só que ao invés de usar *wavelets* usamos filtros de processamento de imagem para remover *aliasing*.

Na Figura 5 pode-se observar o pipeline para remover *aliasing* de imagens já rasterizadas. Primeiramente, implementamos as funções que geram contorno. Em seguida, aplicamos o algoritmo *flood fill* descrito na seção 2 para rasterizar a imagem, ele pinta de preto o que está dentro do contorno e de branco o que está de fora. Em seguida, como uma tentiva de melhorar o aspecto da imagem nas bordas aplica-se filtros de processamento de imagem.

Fig. 5: Pipeline 2D de como os filtros foram aplicados nos exemplos descritos no paper[8].

3.2 Abordagem 3D

Na abordagem 3D, também replicamos alguns testes. Na Figura 6, pode-se observar o pipeline 3D para rasterizar uma nuvem de pontos. Sendo assim, dada uma nuvem de pontos (arquivo *obj* com pontos e faces), primeiramente precisa-se calcular os coeficientes *wavelet* para cada ponto. Os coeficientes *wavelet* são encontrados de acordo com as funções apresentadas na subseção 2.6. Após o cálculo dos coeficientes é possível encontrar a função indicadora também descrita na subseção 2.6. Por meio desta função, é então possível reconstruir a superfície obtendo o dual do *Marching Cubes*. Após gerar a função indicadora discreta, essa será a entrada para o algoritmo de *Marching Cubes*. Neste caso, aplicamos duas versões do *Marching Cubes*, uma é o *Marching Cubes* apresentado em aula e o outro é o *Marching Cubes Modificado* que tira proveito dos coeficientes *Wavelet*.

Na Figura 7 pode-se observar o passo a passo da função indicadora. À esquerda tem-se uma nuvem de pontos, no centro, tem-se a rasterização por meio da função indicadora descrita na subseção 2.6. E à direita, uma aproximação da função indicadora, por meio dos coeficientes *wavelet*.

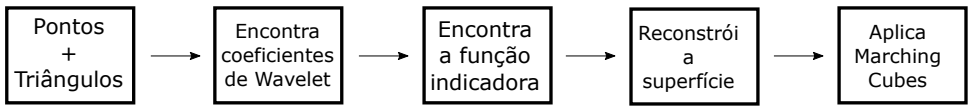


Fig. 6: Pipeline 3D para rasterizar um objeto 3D.

Fig. 7: Passo a passo da função indicadora.

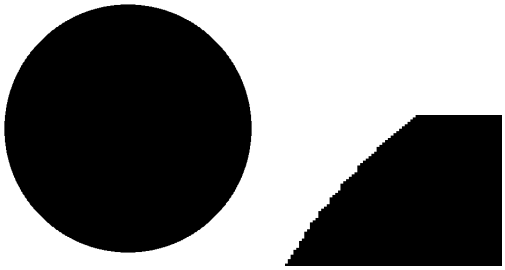
4 Resultados

Nós apresentamos os resultados obtidos usando a abordagem 2D e 3D, em termos de análise visual. Primeiramente, apresentamos e discutimos os resultados para 2D e em seguida para 3D.

Os desenvolvimentos foram baseados nos códigos originais do Manson [8] para a plataforma *Windows*, nossas implementações sobre esses códigos estão sendo desenvolvidos para *Linux*. As bibliotecas 3D basea-se em *OpenGL*, *VTK*, a parte de imagens *OpenCV* e *sci-image* do *Python*.

4.1 Resultados 2D

Para confirmar o que Manson [8] menciona em seu artigo sobre a sua técnica de rasterização replicamos os exemplos do trabalho dele e aplicamos filtros de processamento de imagem anti-aliasing. Na Figura 8a podemos que perceber que de fato, realizar uma rasterização 0 ou 1 sem um tratamento nas bordas irá gerar o efeito indesejado que é o *aliasing*. Em 8b pode-se perceber uma escada ao dar um *zoom* na imagem.



(a) (b)

Fig. 8: Em 8a, circunferência rasterizada por meio do *flood fill* sem filtro, e em 8b, deu-se um zoom para verificar o contorno.

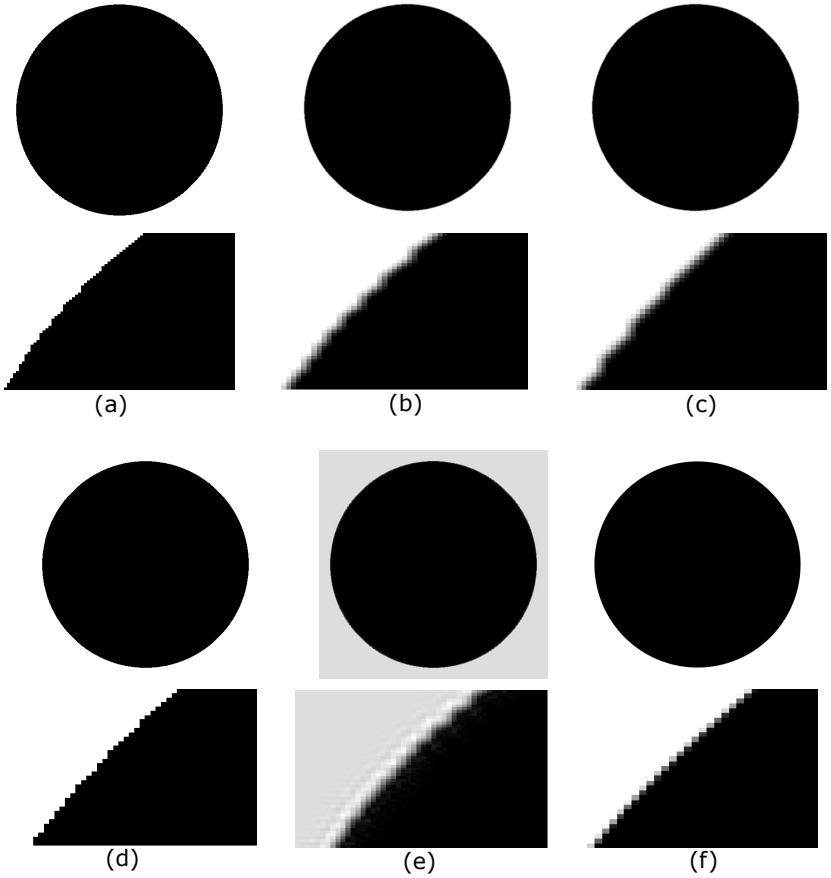


Fig. 9: Suavização das bordas da imagem. Em (a), (b), (c) e (d) tem-se respectivamente o *Box filter*, *Gaussian filter*, *Median filter* e o *Sinc filter* aplicados à uma imagem rasterizada. Em (f) tem-se a abordagem de Manson [8].

A fim de suavizar as bordas da imagem da Figura 9 (a) aplicamos alguns filtros anti-aliasing de processamento de imagens. O resultado do *Box filter* na Figura 9 (b) aplicado na imagem da Figura 9 (a) tornou a imagem borrada e não foi tão eficiente quanto o *Gaussian filter* em que o resultado pode ser observado na Figura 9 (c) e o *Sinc filter* em que o resultado pode ser observado na Figura 9 (e). O *Sinc filter* e o *Gaussian filter* obtiveram um resultado melhor do que resultado dos demais filtros que podem-ser observados nas imagens da Figura 9. De todos os filtros, o *Median filter* foi o que teve o pior resultado, praticamente mateve o aliasing da imagem. Já na Figura 9 (f) pode-se observar a rasterização por meio de coeficientes *wavelet*. É notória a qualidade da rasterização por *wavelet*, o *aliasing* torna-se praticamente imperceptível.

Na Figura 10 aplicou-se na imagem (a) (sem filtro), os mesmos filtros descritos anteriormente. Para este tipo de imagem o *box filter*, *Gaussian filter* e o *sinc filter* tiveram resultados muito similares, como pode ser observado, respectivamente, na Figura 10 (b), (c) e (e). Ao aplicar *Median filter* novamente obteve-se a pior imagem em relação aos demais filtros. Na Figura 10 (f) observa-se novamente que a técnica de wavelet apresentou um resultado superior em relação aos filtros *anti-aliasing* de processamento de imagens. A mesma análise pode ser obtida ao aplicar os mesmos filtros na imagem da Figura 11. Neste sentido, diante dos resultados obtidos, concluímos que sem dúvida a abordagem de Manson [8] é muito eficiente. Em todos os testes as imagens obtiveram resultado superior aos demais filtros anti-aliasing analisados, demonstrando assim, a importância de se utilizar *wavelets* para realizar as rasterização utilizando funções indicadoras discretas.

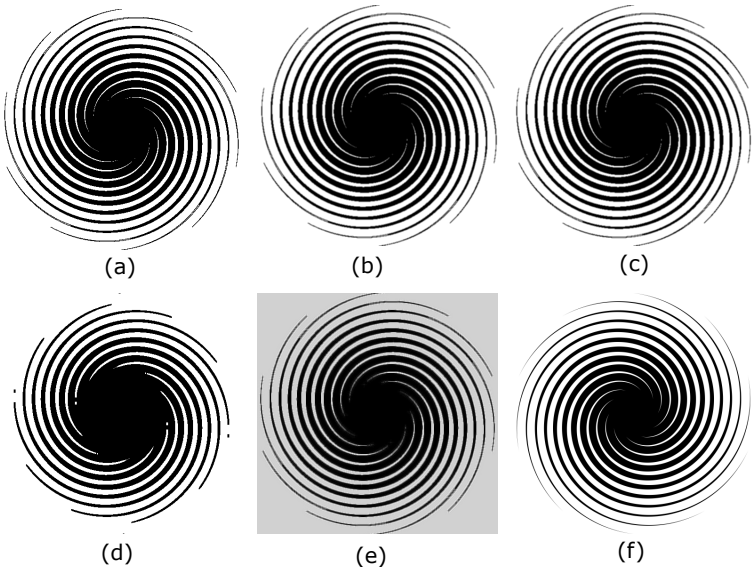


Fig. 10: Suavização das bordas da imagem. Em (a), (b), (c) e (d) tem-se respectivamente o *Box filter*, *Gaussian filter*, *Median filter* e o *Sinc filter* aplicados à uma imagem rasterizada. Em (f) tem-se a abordagem de Manson [8].

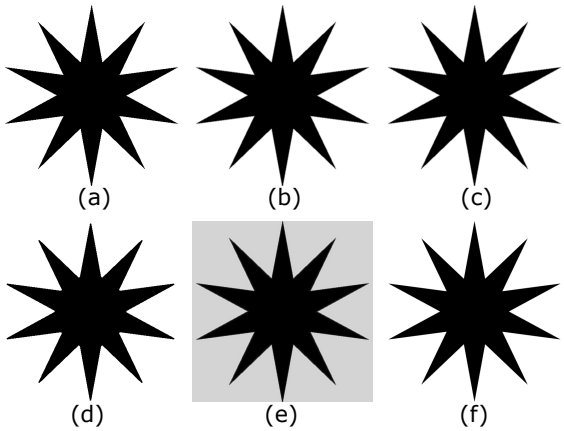


Fig. 11: Suavização das bordas da imagem. Em (a), (b), (c) e (d) tem-se respectivamente o *Box filter*, *Gaussian filter*, *Median filter* e o *Sinc filter* aplicados à uma imagem rasterizada. Em (f) tem-se a abordagem de Manson [8].

4.2 Resultados 3D

Para o caso 3D temos a nuvem de pontos da figura (12) na qual podemos observar que a parte superior da cabeça desta imagem esta faltando e vamos aplicar *Marching Cubes* puro, sem aplicar antes por *Wavelets*, e com isso obtemos a figura (13)

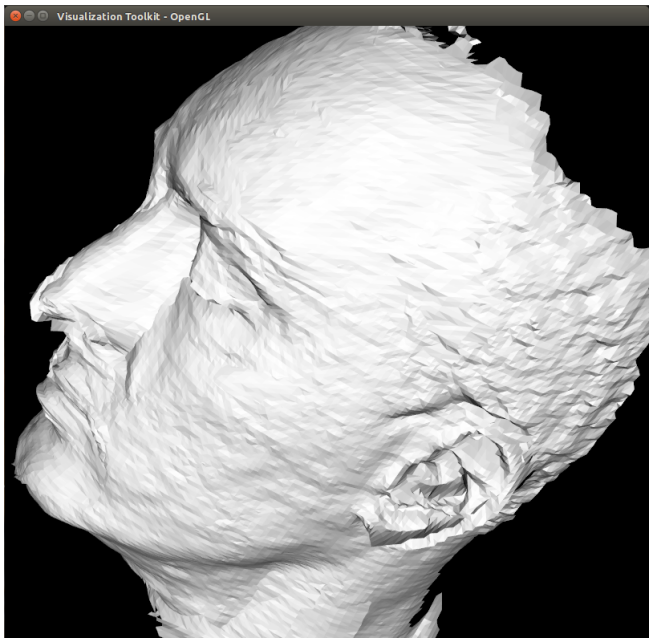


Fig. 12

na qual pode ser observado que tem certos artefatos que aparecem distorcendo a imagem, com isso podemos concluir que não pode ser aplicado *Marching Cubes* para reconstruir superfícies com muitas irregularidades.

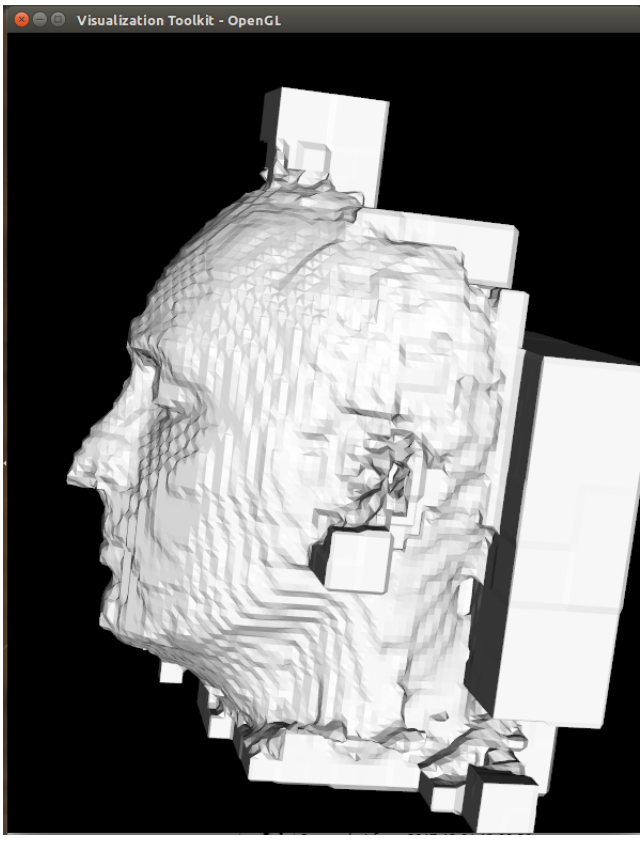


Fig. 13

Mas para no caso de aplicarmos somente o método com *Wavelets*, como podemos observar na figura (14), vemos que já tem uma melhorada com respeito ao método *Marching Cubes*, mas ainda ficaram alguns defeitos que precisam ser modificados.

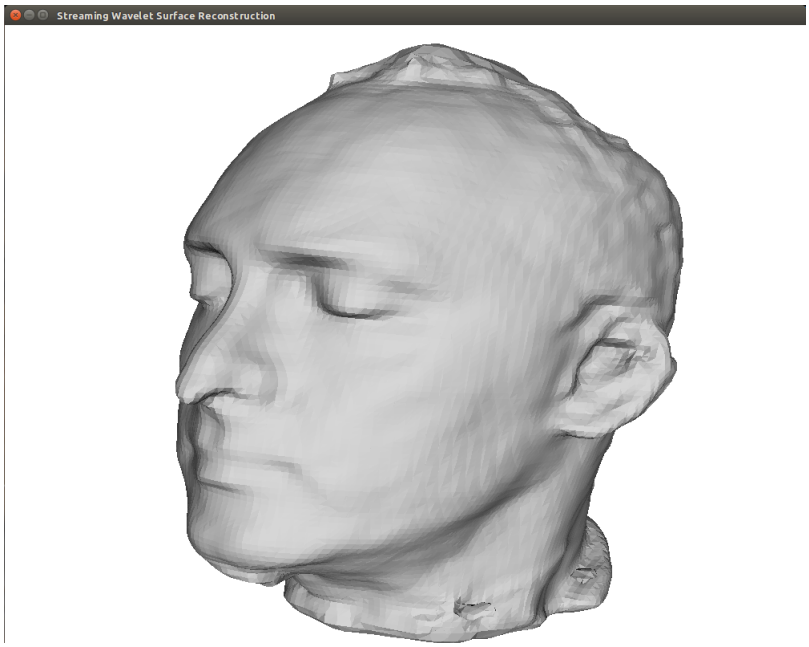
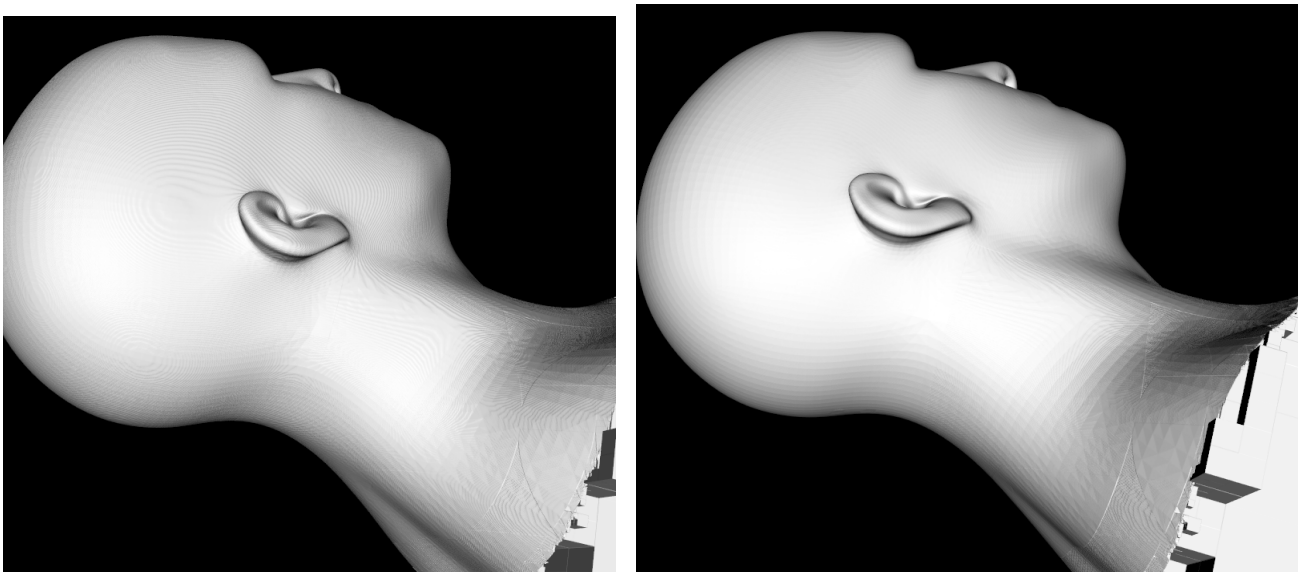


Fig. 14

Quando a superfície é de uma forma mais contínua, como é no caso de (15a), ao aplicar *wavelets* com *Marching cubes modificado* [10] pode-se ver a diferença e a suavidade resultante (ver figura (15b)).



(a) (b)

Fig. 15

5 Conclusões e Trabalhos Futuros

Este trabalho consistiu em realizar um estudo sobre rasterização 2D e 3D em que ao final do processo utiliza-se *Marching Cubes* para extrair a superfície. Para este estudo, tivemos que estudar e entender três trabalhos [9], [10] e [8]. Além disso, realizou-se um *benchmarck* dos principais filtros *anti-aliasing* e do trabalho de Manson [8] comparando-os. Diante dos

resultados obtidos, pode-se concluir que *wavelet* é uma técnica muito poderosa (robusta e precisa) e foi muito bem empregada no contexto de rasterização, visto que, em todos os casos a mesma se mostrou mais eficiente do que qualquer filtro *anti-aliasing*. No entanto, na nossa versão para *Linux*, detectamos que a abordagem não é tão rápida quanto os autores mencionam.

Como trabalhos futuros, agora que já dominamos bastante o assunto, vamos aplicar esta técnica em fluídos para tentarmos rasterizá-los da melhor maneira possível. Outro trabalho que temos também, é adaptar os métodos do Prof. Castelo, *Marching Hyper Cubes* e *Marching Simplex* em vez de *Marching Cubes* para receberem um *grid* que uma aproximação da função indicadora discreta e comparará-los entre sim e com o *Marching Cubes*.

Referências

[1] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM.

[2] Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. A multiscale approach to mesh-based surface tension flows. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 48:1–48:10, New York, NY, USA, 2010. ACM.

[3] Lode's computer graphics tutorial. http://lodev.org/cgtutor/floodfill.html. Acessado: 03-12-2017.

[4] Mean filter. https://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm. Acessado: 03-12-2017.

[5] Gaussian smoothing. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm. Acessado: 03-12-2017.

[6] Median filter. https://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm. Acessado: 03-12-2017.

[7] Sinc filter. https://en.wikipedia.org/wiki/Sinc_filter. Acessado: 03-12-2017.

[8] Josiah Manson and Scott Schaefer. Wavelet rasterization. *Computer Graphics Forum (Proceedings of Eurographics)*, 30(2):395–404, 2011.

[9] Josiah Manson, Guergana Petrova, and Scott Schaefer. Streaming surface reconstruction using wavelets. *Computer Graphics Forum (Proceedings of the Symposium on Geometry Processing)*, 27(5):1411–1420, 2008.

[10] Josiah Manson, Jason Smith, and Scott Schaefer. Contouring discrete indicator functions. *Computer Graphics Forum (Proceedings of Eurographics)*, 30(2):385–393, 2011.