UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

FELIPE EINSFELD KERSTING

Domain Transform para Spherical Geometry Images

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Eduardo Simões Lopes Gastal

Porto Alegre
2018

“If I have seen farther than others,
it is because I stood on the shoulders of giants.”
— Sir Isaac Newton

AGRADECIMENTOS

Gostaria de agradecer todos os professores e funcionários da **UFRGS**, que foram muito importantes durante todo o período da graduação. Em especial, agradeço ao meu orientador Eduardo Gastal por me dar a oportunidade de começar este trabalho como bolsista de iniciação científica, e posteriormente me orientar neste trabalho de conclusão.

Por fim, agradeço minha família e meus amigos que me apoiaram durante todo este período.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

GIM	Geometry Image
SGIM	Geometry Image obtida a partir de uma parametrização esférica
DTFSGIM	Domain Transform for Spherical Geometry Images
RF	Recursive Filter
RGB	Red, Green, Blue
BFPC	The Bilateral Filter for Point Clouds
GMNF	Guided Mesh Normal Filtering

LISTA DE FIGURAS

RESUMO

Com o avanço de tecnologias como o escaneamento 3D, filtros capazes de suavizar malhas poligonais cresceram em importância, dada a necessidade da remoção de ruído destes objetos (erros e imprecisões provenientes do escaneamento, por exemplo). Além disso, filtros deste tipo são utilizados em diversas áreas da computação gráfica, como jogos e *softwares* de modelagem.

Este trabalho mostra a criação, especificação e a implementação de um filtro chamado *Domain Transform for Spherical Geometry Images*, ou *DTFSGIM*. Este filtro tem como objetivo a suavização de malhas de triângulos representadas por *spherical geometry images* e a manutenção de curvaturas importantes do objeto original.

O filtro *DTFSGIM* é baseado no filtro *domain transform*, que é um filtro de alta performance, cujo principal objetivo é suavizar imagens mantendo suas arestas importantes. Para ser possível utilizar o filtro *domain transform* em objetos tridimensionais, primeiramente a malha de triângulos é parametrizada em uma *geometry image*, que é uma imagem capaz de representar uma malha de triângulos. Esta *geometry image* é, então, filtrada, e, depois da filtragem, transformada novamente em uma malha de triângulos.

A filtragem da *geometry image* é um tanto complexa por conta de todas as peculiaridades de filtrar uma imagem que, no fundo, representa uma malha de triângulos. Neste texto, esta filtragem será explicada em detalhes, e os resultados obtidos serão mostrados e comparados com outros trabalhos.

Palavras-chave: Filtro. malha de triângulos. curvatura. DTFSGIM.

Domain Transform for Spherical Geometry Images

ABSTRACT

With the advance of technologies as 3D scanning, filters capable of smooth poligonal meshes have grown in importance, given the necessity of removing noise from these objects (errors and inaccuracies coming from scanning, for example). Besides, this kind of filter is used in several areas of computer graphics, like games and modeling software.
This work shows the creation, specification and implementation of a filter called *Domain Transform for Spherical Geometry Images*, or *DTFSGIM*. The filter's main objective is to smooth triangle meshes represented as *spherical geometry images* and at the same time keep the object important curvatures.
The *DTFSGIM* filter is based on the *domain transform* filter, which is a high-performance filter, whose main objective is to smooth images keeping its important edges. To be possible to use the *domain transform* filter with tridimensional objects, firstly the triangle mesh is parameterized as a *geometry image*, which is an image capable of representing a triangle mesh. Then, this *geometry image* is filtered and, after the filtering process, transformed again in a triangle mesh.
The *geometry image* filtering process is quite complex because of all peculiarities of filtering an image that represents a triangle mesh. In this text, this filter will be fully explained, and the obtained results will be shown and compared to other important works.

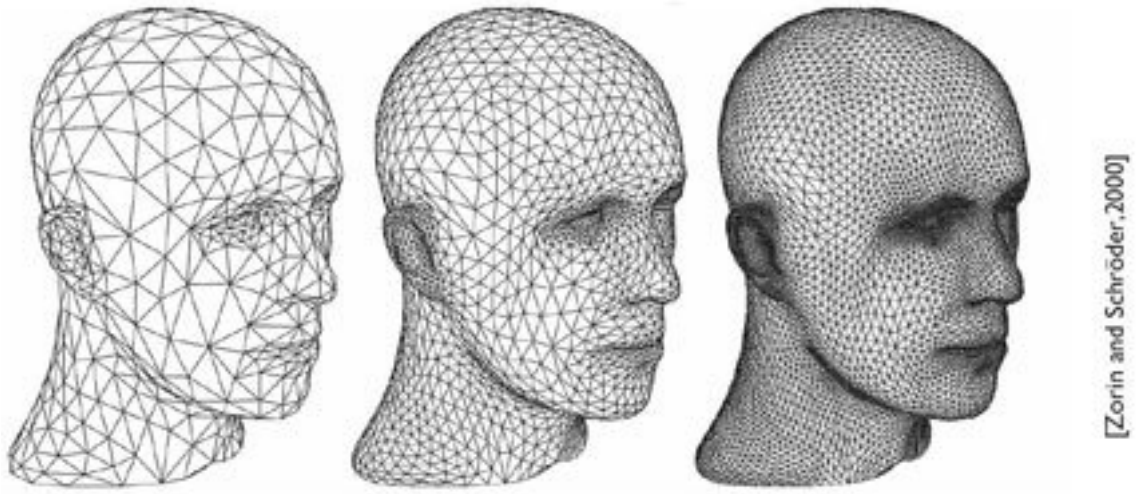
Keywords: filter, triangle mesh, curvature, DTFSGIM.

1 INTRODUÇÃO

1.1 Apresentação Teórica

Em computação gráfica, malhas de polígonos são a representação mais convencional de geometrias arbitrárias. Elas consistem em um conjunto de vértices, arestas e faces que definem um objeto tridimensional. Malhas de polígonos são utilizadas em um conjunto amplo de aplicações, que variam desde jogos digitais até aplicações de modelagem utilizadas em diversas áreas da ciência, como simulações numéricas (HO-LE, 1988). Seu amplo uso deve-se principalmente ao fato de seu processamento ser muito eficiente nas *GPUs* (*graphics processing units*) desenvolvidas atualmente e sua flexibilidade ser grande o suficiente a ponto de ser possível representar geometrias arbitrárias com qualquer tolerância de aproximação. A figura 1.1 mostra malhas poligonais que representam a cabeça de um homem.

Figura 1.1: Três malhas poligonais que representam a cabeça de um homem. À medida que o número de vértices aumenta, a aproximação se torna mais suave e mais realista.



Fonte: Subdivision for Modeling and Animation (ZORIN; SCHRöDER, 2001).

Em processamento de imagens, um filtro consiste em um operador que modifica uma imagem para, por exemplo, realçar detalhes ou suavizar variações de cores. Uma classe comum de filtros são aqueles que removem ou atenuam uma certa faixa de frequências de uma imagem. Esta técnica permite a obtenção de resultados interessantes, pois as altas frequências de uma imagem possuem uma relação direta com as arestas da mesma, dado que elas são caracterizadas por variações abruptas nas cores dos pixels. Em contraste, as baixas frequências caracterizam regiões da imagem em que há uma variação de

cores suave.

Um filtro de borramento, por exemplo, remove altas frequências de uma imagem, de modo que o resultado final consiste em uma imagem borrada. Já um filtro passa-altas é capaz de remover baixas frequências de uma imagem, de forma que apenas arestas importantes são preservadas.

Um filtro bilateral é um filtro capaz de remover seletivamente altas frequências de uma imagem, ao mesmo tempo que arestas importantes do desenho original são preservadas. Este filtro pode ser representado pela composição de dois pesos distintos: um que leva em consideração a distância entre os pixels no domínio (*grid*) da imagem e outro que leva em consideração a diferença de cor entre eles. Podemos chamar o filtro da distância de $G_{\sigma s}$ e o filtro da cor de $G_{\sigma r}$. Portanto, podemos definir matematicamente este filtro pela equação:

$$J[p] = \frac{1}{W_p} \sum_{q \in S} G_{\sigma s}(||p - q||) G_{\sigma r}(I[p] - I[q]) I[q]\,, \tag{1.1}$$

onde:

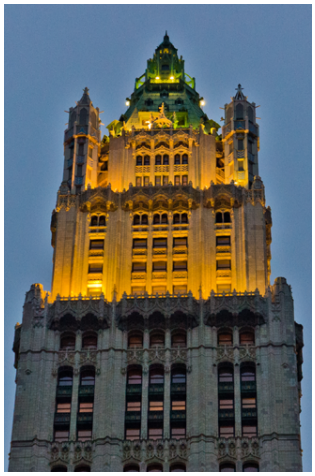
$$W_p = \sum_{q \in S} G_{\sigma s}(||p - q||) G_{\sigma r}(I[p] - I[q])\,. \tag{1.2}$$

A eq. (1.1) diz que a cor de cada pixel P da imagem filtrada J será computado como sendo a soma das cores dos seus vizinhos na imagem original I, dentro de um espaço de busca S, ponderados pelo produto de uma função $G_{\sigma s}$, que depende da distância entre o pixel central e o pixel vizinho a ser considerado e uma função $G_{\sigma r}$, que depende da diferença de cores entre os pixels considerados. O espaço S mencionado, teoricamente, refere-se a todo espaço da imagem original. Entretanto, numa implementação real, geralmente opta-se por truncar este espaço, pois as funções $G_{\sigma s}$ e $G_{\sigma r}$ acabam sendo definidas de tal maneira que pixels muito distantes do pixel central contribuem muito pouco para o resultado final. O termo $\frac{1}{W_p}$ é necessário para a normalização do resultado, isto é, para que a soma dos pesos do somatório seja igual a 1 e, portanto, o filtro preservará a intensidade (cor) média da imagem. A figura 1.2 mostra uma imagem filtrada por um filtro bilateral.

Além de imagens, filtros também podem ser aplicados em objetos tridimensionais, como malhas poligonais. Neste caso, a ideia continua sendo a mesma: a posição final de cada vértice será definida como uma mistura das posições de vértices próximos, por exemplo.

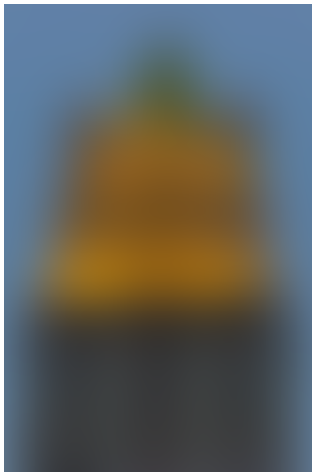
Figura 1.2: Um exemplo de um filtro bilateral. Na esquerda, a imagem original. No centro, a imagem filtrada pelo filtro bilateral, com preservação de bordas. Na direita, a imagem filtrada por um filtro de borramento regular comum. Note como o filtro bilateral permite suavizar os detalhes da imagem ao passo que as arestas importantes ainda são mantidas.

(a) *Imagem original.*



(b) *Imagem filtrada por um filtro bilateral.*

(c) *Imagem filtrada por um filtro de suavização regular comum.*



Fonte: O autor.

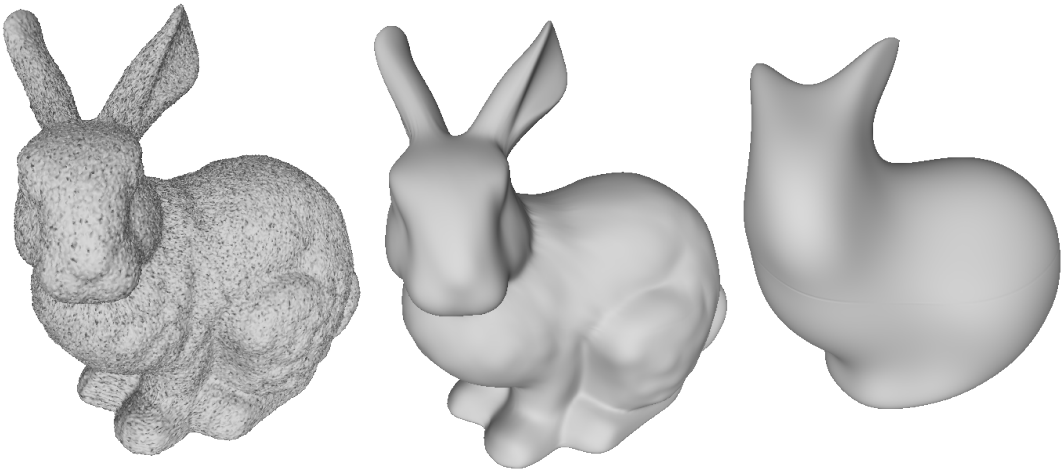
1.2 Proposta do Trabalho

Malhas poligonais, assim como imagens, podem ser suavizadas com a utilização de filtros. Entretanto, quando uma malha é suavizada sem levar em consideração suas características individuais, detalhes importantes podem ser perdidos, como curvaturas importantes do objeto original.

A ideia deste trabalho é desenvolver um filtro capaz de suavizar uma malha poligonal arbitrária descrita por uma *spherical geometry image*, removendo ruídos e componentes indesejados de alta frequência. Ao mesmo tempo, o filtro deve ser capaz de preservar detalhes importantes da malha original, de modo que o resultado final mantenha as curvas essenciais da sua entrada. Em particular, neste trabalho queremos desenvolver um filtro que explicitamente preserve regiões de alta curvatura da superfície. A figura 1.3 mostra um exemplo de uma malha poligonal suavizada com o algoritmo proposto.

Uma das dificuldades do processo de filtragem de objetos tridimensionais é a necessidade de que o algoritmo trabalhe sobre um espaço tridimensional, ao invés de um espaço bidimensional. Isto faz com que algoritmos de filtragem eficiente desenvolvidos originalmente para imagens não possam ser aplicados diretamente em malhas. Para que isto seja possível, a proposta principal deste trabalho é transformar a malha poligonal

Figura 1.3: Uma malha poligonal filtrada pelo algoritmo proposto. A imagem da esquerda mostra uma malha com ruído de alta frequência. A imagem central mostra a malha suavizada, obtida considerando informações de curvatura. A imagem da direita mostra o resultado sem considerar detalhes da geometria original.



Fonte: O autor.

em uma estrutura bidimensional regular antes do processo de filtragem, filtrar esta nova estrutura e então reconstruir a malha tridimensional novamente.

Parametrizar uma malha poligonal em um espaço bidimensional não é um processo trivial, mas algumas investigações já foram desenvolvidas neste sentido. Neste trabalho, isto será feito por meio de uma parametrização esférica (PRAUN; HOPPE, 2003) para transformar a malha em uma *geometry image* (GU; GORTLER; HOPPE, 2002). Estes trabalhos serão descritos em mais detalhes no capítulo 2.

As *geometry images* obtidas a partir da parametrização esférica serão filtradas por um algoritmo baseado no filtro *domain transform* (GASTAL; OLIVEIRA, 2011). O filtro *domain transform*, que será descrito em detalhes no capítulo 2, é um filtro de suavização com preservação de arestas desenvolvido originalmente para imagens. Por ser um filtro de suavização de imagens, modificações devem ser feitas para sua aplicação em malhas tridimensionais. Estas serão explicadas em maiores detalhes no capítulo 3.

1.3 Objetivos

Os principais fatores que serão buscados neste trabalho são:

- O filtro deve ser executado o mais rápido possível;
- O filtro deve ser capaz de filtrar qualquer malha representada por uma *spherical*

geometry image, independente da sua geometria;

- O filtro deve ser capaz de remover ruídos e detalhes indesejados de alta-frequência;
- O filtro deve ser capaz de manter curvas importantes da malha original;
- O filtro deve produzir resultados visualmente interessantes.

Os resultados obtidos serão comparados com outros trabalhos para sua avaliação e validação. O objetivo principal é verificar se os frutos deste trabalho serão melhores dos que os já existentes em algum sentido (por exemplo eficiência computacional ou qualidade visual), para então avaliar suas possíveis contribuições na filtragem de malhas poligonais.

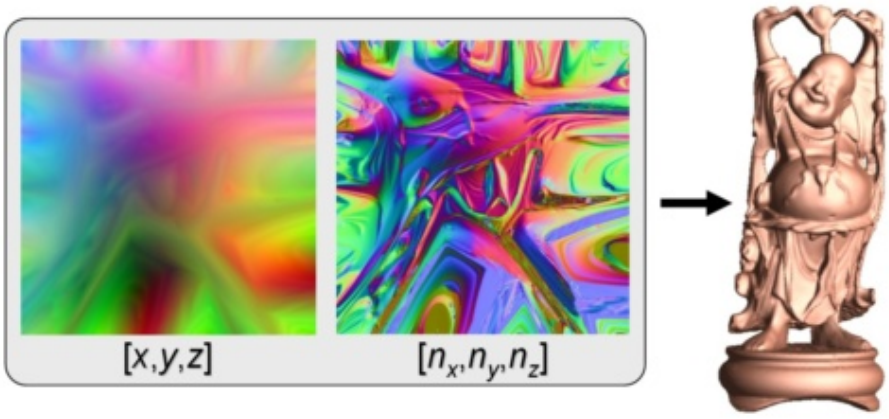
2 PESQUISA E TRABALHOS RELACIONADOS

Nesta seção serão apresentados trabalhos que serão utilizados no desenvolvimento do filtro e também outros artigos relacionados.

2.1 Geometry Images

Geometry images são imagens que representam malhas poligonais (GU; GORTLER; HOPPE, 2002). Nestas imagens, cada vértice da malha original é representado por um pixel. Os canais RGB de cada pixel representam a posição tridimensional de cada vértice, e as arestas da malhas são inferidas de acordo com a posição dos pixels na imagem. A figura 2.1 mostra um exemplo de uma *geometry image* e sua respectiva malha poligonal.

Figura 2.1: Um exemplo de uma *geometry image*. A imagem da esquerda mostra a *geometry image* da malha poligonal mostrada na direita. A imagem do centro mostra o *normal map* da mesma malha.



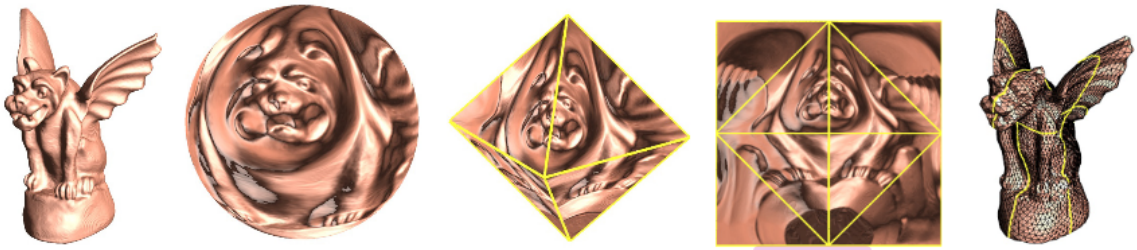
Fonte: Artigo original da *Geometry image* (GU; GORTLER; HOPPE, 2002).

O algoritmo de transformação de malhas poligonais em *geometry images* é mostrado em detalhes no trabalho de Gu et al. Como *geometry images* são imagens convencionais, algoritmos capazes de filtrar imagens podem ser utilizados para a filtragem destas. Contudo, como esta imagem representa um conjunto de vértices e suas interconexões, e não simplesmente um conjunto de pixels coloridos, problemas podem ocorrer na malha resultante de um processo de filtragem. Alguns exemplos destes problemas serão mostrados no capítulo 3. Neste texto, *geometry image* será abreviado para *GIM*.

2.2 Spherical Parametrization and Remeshing

Spherical Parametrization and Remeshing (PRAUN; HOPPE, 2003) é um trabalho que introduz um algoritmo capaz de obter uma parametrização esférica a partir de uma malha poligonal genérica. A partir da parametrização esférica, é possível se obter outros tipos de parametrizações, entre elas uma *geometry image*. A figura 2.2 mostra exemplos destas parametrizações.

Figura 2.2: Exemplos de parametrizações de uma malha. As cinco figuras, em ordem da esquerda para direita: a malha original, sua parametrização esférica, uma parametrização octaédrica obtida através da parametrização esférica, uma *geometry image* obtida através da parametrização octaédrica e a malha original reconstruída. É importante notar que as imagens mostradas são apenas uma ilustração das estruturas geradas. As cores da *geometry image* mostrada, por exemplo, não são as cores reais da *geometry image*, e sim uma textura que representa as cores da malha.



Fonte: Artigo original das *Spherical Geometry Images* (PRAUN; HOPPE, 2003).

Neste texto, uma *geometry image* obtida através de uma parametrização esférica será chamada abreviadamente de *SGIM*.

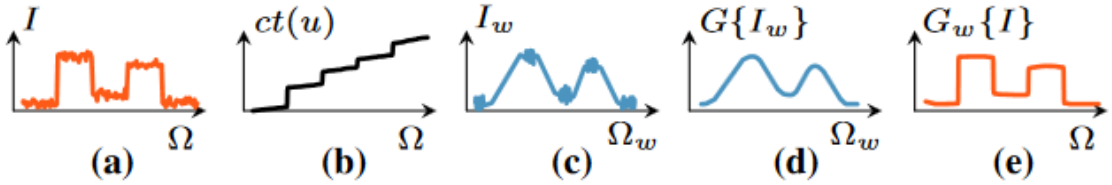
2.3 Filtro Domain Transform

Filtros bilaterais convencionais produzem bons resultados, porém são filtros computacionalmente custosos. O filtro bilateral comum apresentado na seção 1.1 realiza um somatório sobre um espaço de busca S para cada pixel da imagem, e portanto possui complexidade $O(n^2)$ quando o espaço S é composto por todos outros pixels da imagem (onde n é o número de pixels da imagem). Se o espaço S for truncado por uma vizinhança de k pixels, a complexidade seria reduzida para $O(kn)$. Uma alternativa a esses filtros é o filtro *domain transform* (GASTAL; OLIVEIRA, 2011). Este filtro é capaz de suavizar imagens mantendo informações importantes de alta frequência, assim como um filtro bilateral, porém com complexidade computacional $O(n)$.

A técnica do filtro *domain transform* é baseada em uma transformada de domínio

que permite com que os pixels da imagem original sejam representados por uma maneira alternativa. Este conceito permite com que a imagem seja filtrada de maneira mais rápida. A ideia é representar os pixels da imagem em um novo domínio que representa o quanto as cores estavam variando na imagem original. Assim, é possível obter o resultado esperado apenas com filtragens num espaço 1D. A figura 2.3 mostra as etapas de execução do filtro.

Figura 2.3: Etapas da *domain transform*. A figura (a) mostra a distribuição das cores dos pixels de, por exemplo, uma linha da imagem. A figura (b) mostra a transformada de domínio desta distribuição. Basicamente, a diferença entre os pixels são somadas, em módulo, de maneira acumulativa. A figura (c) mostra o sinal original plotado no novo domínio, enquanto que a figura (d) mostra este sinal filtrado com um filtro gaussiano 1D. A figura (e) mostra o resultado final, no domínio de origem.



Fonte: Artigo original do filtro *Domain Transform* (GASTAL; OLIVEIRA, 2011).

A transformada de domínio $ct(u)$ é dada, matematicamente, pela equação:

$$ct(u) = \int_0^u 1 + \frac{\sigma_s}{\sigma_r} \sum_{k=1}^{c} |I_k'(x)| dx \,. \tag{2.1}$$

Os parâmetros σ_s e σ_r controlam a influência dos domínios espacial e de cores no resultado final do filtro, respectivamente. O valor c é o número de canais da imagem (normalmente três, pois costuma-se trabalhar com os canais RGB), $I_k(x)$ é a função que define os valores dos pixels da imagem para o canal k e $I_k'(x)$ é sua derivada.

Para a implementação prática deste filtro, podem ser utilizadas três diferentes abordagens: *normalized convolution (NC)*, *interpolated convolution (IC)* e *recursive filtering (RF)*. Esta última permite descrever o filtro recursivamente pela equação:

$$J[n] = a^{d[n]} I[n] + (1 - a^{d[n]}) J[n-1] \,. \tag{2.2}$$

A função $d[n]$ é calculada através da transformada $ct(u)$ e é dada pela equação:

$$d[n] = 1 + \frac{\sigma_s}{\sigma_r} \sum_{k=1}^{c} |I_k[x] - I_k[x-1]| \,, \tag{2.3}$$

onde $I_k[x] - I_k[x-1]$ é aproximação da derivada $I_k'(x)$ através da primeira diferença entre pixels vizinhos.

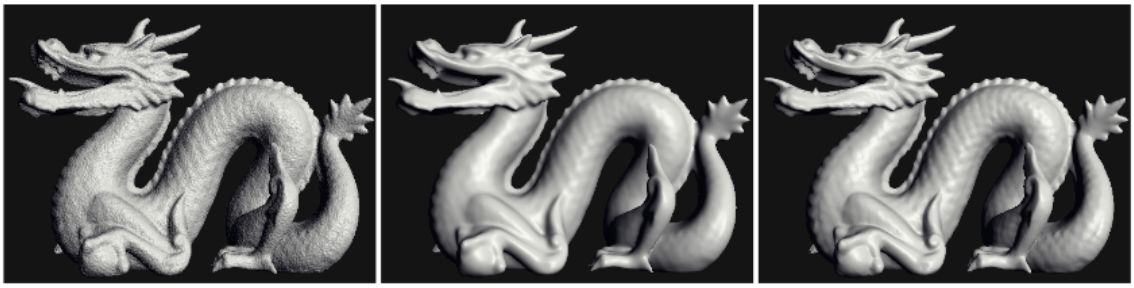
O filtro *domain transform* pode ser aplicado paralelamente ou em cascata. Entretanto, este não é um filtro separável, de modo que não é possível chegar no resultado final com apenas uma aplicação em cascata, por exemplo, deste filtro. Isto faz com que uma única filtragem introduza artefatos visuais na imagem. Portanto, para evitar o surgimento destes artefatos, é necessário aplicar o filtro várias vezes consecutivas.

2.4 Trabalhos para Suavização de Malhas

Losasso et al. apresentam um filtro de suavização de malhas a partir de *geometry images*, porém o trabalho é focado em reduzir o uso de memória e evitar a renderização de geometrias complexas. (LOSASSO et al., 2003) Por isso, é feita apenas a suavização das malhas poligonais, sem a preservação de curvaturas.

No trabalho de Adams et al., é proposto um método de aceleração de uma larga classe de filtros não-lineares, incluindo filtros bilaterais. (ADAMS et al., 2009) Baseado neste método, um filtro bilateral acelerado é implementado para a filtragem de malhas. O resultado deste trabalho é mostrado na figura 1.3 e é similar ao proposto na seção 1.2. Contudo, este método é computacionalmente custoso, levando aproximadamente 1 minuto para a filtragem mostrada na figura 2.4.

Figura 2.4: Uma malha poligonal filtrada pelo algoritmo proposto por Adams et al. (ADAMS et al., 2009) A imagem da esquerda mostra uma malha com ruído de alta frequência. A imagem central mostra a malha suavizada, ainda sem considerar detalhes da geometria original. A imagem da direita mostra o resultado final, obtido considerando informações de curvatura.



Fonte: Artigo original do trabalho de Adams et al (ADAMS et al., 2009).

Digne e de Franchis (DIGNE; FRANCHIS, 2017) apresentam um filtro bilateral para malhas. Neste trabalho, o filtro considera não apenas a distância entre os vértices, como também a distância ao longo da direção normal de cada ponto. Esta heurística permite uma preservação muito maior das arestas importantes da malha.

O trabalho *Guided Mesh Normal Filtering* (ZHANG et al., 2015) propõe um fil-

tro bilateral para malhas que atua utilizando um “sinal de orientação” ao invés do sinal original.

3 ESPECIFICAÇÃO DO PROJETO

A proposta principal do trabalho é a criação do filtro citado na seção 1.2. Este filtro será explicado em detalhes abaixo.

3.1 Filtro

O filtro de suavização proposto neste trabalho possui três etapas:

1. Parametrização da malha poligonal original em uma *geometry image*;
2. Filtragem da *geometry image*;
3. Reconstrução da malha poligonal, agora filtrada.

A primeira etapa consiste basicamente em transformar a malha poligonal original em uma *geometry image*. Para isso, a *geometry image* pode ser obtida a partir de uma parametrização esférica, como descrita na seção 2.2. A vantagem da utilização da parametrização esférica é que o resultado obtido possui propriedades mateméticas desejáveis e isto ajuda na construção do filtro bidimensional. Esta etapa já foi desenvolvida pelos trabalhos descritos no capítulo 2 e, portanto, é apenas reutilizada neste trabalho.

A segunda etapa consiste em filtrar a imagem obtida na etapa 1. O filtro desenvolvido neste trabalho é baseado no filtro *domain transform*, descrito na seção 2.3. Uma aplicação simples da *domain transform* na *geometry image* não é suficiente por alguns motivos, dentre eles:

- **Lidar com pixels de borda**: Quando a *domain transform* é implementada como um filtro recursivo (*RF*), a filtragem é realizada com uma série de filtros de primeira ordem, conforme a eq. (2.2). Originalmente, a imagem é filtrada em linhas, começando de uma borda da imagem e terminando em outra. Quando esta ideia é extendida para *geometry images* ocorre um problema, pois os pixels (vértices) localizados na borda da imagem estão, na verdade, conectados com outros vértices, de outras bordas. Isso implica que a filtragem não pode terminar nas bordas, e sim deve continuar em algum outro ponto da imagem.
- **Lidar com pixels iguais**: Na *geometry image*, pixels que estão lado a lado na imagem possuem implicitamente uma aresta definida entre eles. Entretanto, como uma imagem é definida em um espaço bidimensional, há um problema para definir

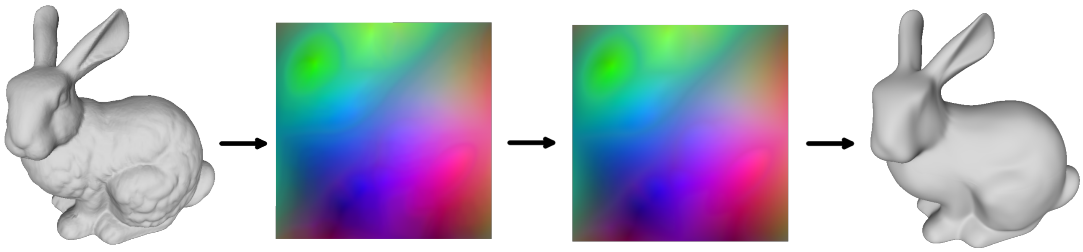
arestas entre os pixels que se encontram na borda da imagem. Na prática, pixels que estão na borda da *geometry image* costumam ser idênticos a outros pixels em alguma outra posição na borda. Pixels idênticos representam o mesmo vértice, de modo que durante o processo de filtragem é necessário ter cuidado para transferir adequadamente os resultados para pixels que representam os mesmos vértices.

- **Heurística para manter curvaturas da malha**: O filtro original da *domain transform* mantém arestas baseado na diferença de cor entre os pixels da imagem. No caso de uma *geometry image*, os canais RGB de cada pixel representam a posição de cada vértice no espaço tridimensional, portanto neste caso a diferença de cor dos pixels representa apenas a distância dos vértices e não a curvatura da malha. Desta maneira, é necessário determinar outra heurística para o filtro tridimensional, que consiga representar e preservar a curvatura.

Portanto, todos estes problemas devem ser corrigidos com a alteração da *domain transform* e a criação de novas heurísticas.

A terceira e última etapa consiste em reconstruir a malha a partir da *geometry image* filtrada. Esta etapa é trivial. A figura 3.1 ilustra as etapas do filtro citadas nos últimos parágrafos.

Figura 3.1: Etapas do filtro proposto. As imagens, em ordem da esquerda para direita: a malha original, a *geometry image* obtida a partir da malha original, a *geometry image* filtrada e a malha resultante reconstruída.



Fonte: O autor.

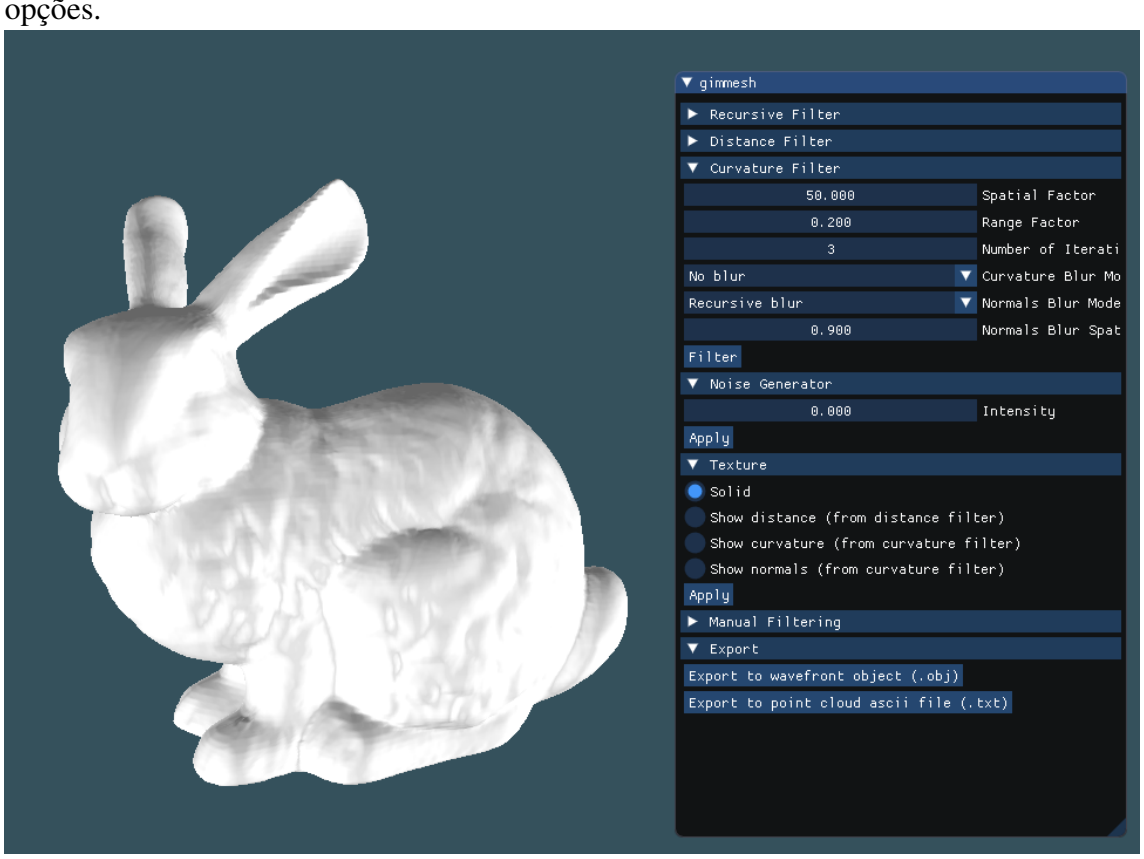
Este novo filtro, desenvolvido neste trabalho, é chamado de *Domain Transform for Spherical Geometry Images* ou, abreviadamente, *DTFSGIM*.

3.2 Aplicação

Além do filtro, também foi desenvolvida uma aplicação que permite carregamento e visualização de malhas poligonais, além da filtragem utilizando o algoritmo descrito

na seção 3.1. A aplicação também foi utilizada para efetuar testes e acelerar o desenvolvimento do trabalho. A aplicação foi desenvolvida em *C*, utilizando *OpenGL* e possui suporte para ambos *Windows* e *Linux*. A figura 3.2 mostra uma imagem do software desenvolvido.

Figura 3.2: Aplicação desenvolvida neste trabalho. O menu permite que o usuário filtre a malha carregada com diferentes parâmetros, exporte-a para outros formatos, entre outras opções.



Fonte: O autor.

4 PROPRIEDADES DE FILTRAGEM DE *GEOMETRY IMAGES*

O filtro *DTFSGIM* foi desenvolvido de modo que a filtragem deve ocorrer em uma *GIM* gerada a partir de uma *spherical parametrization* (parametrização esférica). Como já explicado no capítulo 2, *geometry images* são imagens capazes de representar uma malha poligonal por completo. Estas imagens podem ser obtidas de diferentes maneiras, uma sendo a partir de uma parametrização esférica, como explicado no trabalho *Spherical Parametrization and Remeshing* (PRAUN; HOPPE, 2003). O filtro *DTFSGIM* requer que a *GIM* recebida seja fruto de uma parametrização esférica, por motivos que serão apresentados a seguir.

A figura 2.2 mostra todas as etapas envolvidas para a geração de uma *geometry image* a partir de uma parametrização esférica. Como mostrado, a malha poligonal esférica obtida a partir do objeto original deve ser submetida a um novo processo de parametrização, desta vez para um octaedro. Este octaedro deve então ser parametrizado novamente para a obtenção da *geometry image*.

Uma *GIM* obtida a partir deste processo possui características interessantes e que são exploradas pelo filtro *DTFSGIM*. A seguir, serão mostradas as características comuns a todas *geometry images* e, em seguida, serão destacadas as vantagens da utilização de uma *SGIM*, ao invés de uma *GIM* comum.

4.1 Características de uma *Geometry Image* comum

Toda *GIM* possui as seguintes características fundamentais:

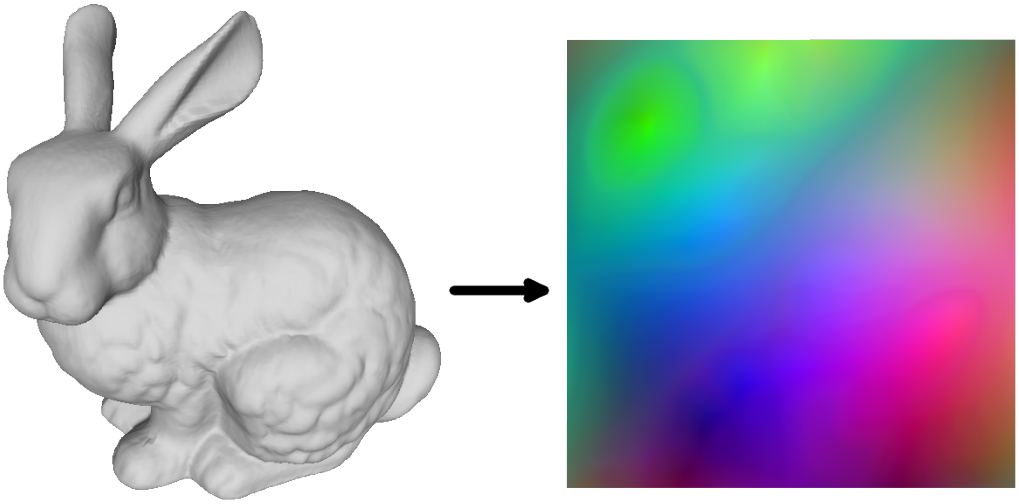
- Representa uma certa malha de triângulos (*triangle mesh*) tridimensional.
- Cada pixel da imagem representa um vértice da malha original. Para isto, as coordenadas *XYZ* do vértice são mapeadas diretamente para os canais *RGB* do pixel. Para obter-se uma melhor precisão, é possível aumentar o número de bits dos canais *RGB* de modo que seja suficiente para a representação desejada da malha. Além disso, os canais *RGB* da *geometry image* podem ser representados de diversas maneiras, como números inteiros, números em ponto flutuante, números negativos, etc. Todas *geometry images* utilizadas neste texto foram codificadas com a utilização de números em ponto flutuante de qualquer sinal.
- As arestas dos triângulos da malha são representadas pela distribuição dos pixels na

imagem. Cada grade 2x2 de pixels da imagem define implicitamente a existência de dois triângulos, e a diagonal utilizada para a criação dos triângulos é sempre a menor das duas no espaço tridimensional da malha. Isto significa que, para descobrir qual diagonal utilizar em cada grade 2x2 de pixels na imagem, é necessário considerar os quatro vértices definidos pelos valores *RGB* dos pixels da grade e calcular, no espaço tridimensional, qual é a distância entre os vértices que definem as duas diagonais. A diagonal que possuir a distância menor é a escolhida.

- As arestas da malha não são necessariamente as mesmas arestas definidas na malha original. O algoritmo prevê a alteração das arestas e dos vértices de modo que seja possível a parametrização.

A figura 4.1 mostra um exemplo de uma *GIM*.

Figura 4.1: Um exemplo de uma *geometry image*. Na esquerda, uma malha poligonal. Na direita, a *geometry image* gerada a partir desta malha.



Fonte: O autor.

Em uma *geometry image*, vértices que são definidos por pixels que estão na borda da imagem podem ser representados por mais de um pixel. Isto significa que mais de um pixel localizado na borda da imagem podem definir exatamente o mesmo vértice. Neste texto, pixels que estão na borda da *geometry image* serão chamados de *pixels de borda*. Além disto, sempre que um vértice for representado pelos pixels X e Y, por exemplo, será dito que o pixel X é um *match* de Y (e vice-versa).

4.2 Características de uma *SGIM*

Uma *geometry image* obtida a partir de uma parametrização esférica possui a característica fundamental de que os *matches* de todos *pixels de borda* podem ser inferidos sabendo-se apenas o tamanho da imagem, dispensando assim a necessidade de uma verificação manual. Em outras palavras, isto significa que a posição do *match* de um pixel X localizado em uma posição qualquer da borda da imagem pode ser inferida com base apenas na posição do pixel X e no tamanho da imagem.

Toda *geometry image* obtida a partir de uma parametrização esférica possui três tipos de *pixels de borda*, separados de acordo com a localização de seus *matches*. No caso mais comum, o *match* de um determinado pixel X encontra-se sempre na mesma borda de X, porém em uma posição diferente. No caso de *pixels de borda* que estão na borda esquerda ou na borda direita da imagem, por exemplo, suas posições horizontais (eixo X) não mudam, mas suas posições verticais (eixo Y) são alteradas. Esta relação pode ser definida matematicamente pela equação:

$$P_y = H - y - 1\,, \tag{4.1}$$

onde H é a altura da imagem (em pixels), y é a posição vertical do pixel X e P_y é a posição vertical do $match$.

No caso de *pixels de borda* que estão na borda superior ou inferior da imagem, suas posições verticais (eixo Y) não mudam, mas suas posições horizontais (eixo X) são alteradas. Esta relação pode ser definida matematicamente como:

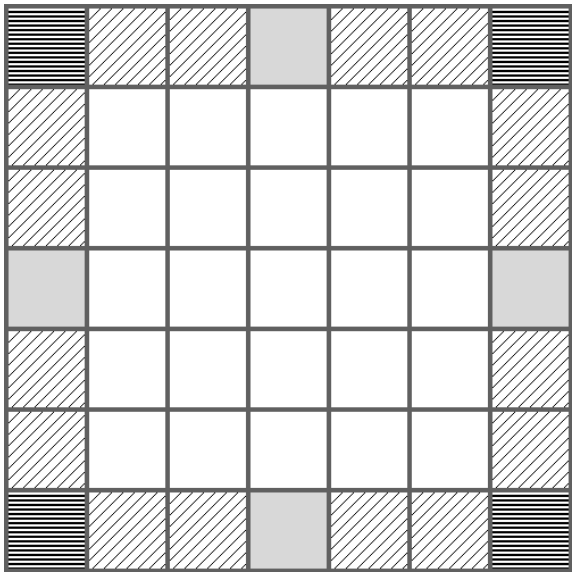
$$P_x = W - x - 1\,, \tag{4.2}$$

onde W é a largura da imagem (em pixels), x é a posição horizontal do pixel X e P_x é a posição horizontal do $match$.

A eq. (4.1) e a eq. (4.2) são válidas para quase todos os *pixels de borda* da imagem. Neste texto, *pixels de borda* que possuam estas características serão chamados de *pixels de borda tipo 1*. Como já dito, ainda existem dois outros tipos. Chamaremos de *pixels de borda tipo 2* os quatro pixels que estão nos cantos da imagem; em uma *geometry image* obtida a partir de uma parametrização esférica, estes quatro pixels sempre representarão o mesmo vértice. Isto implica que todo pixel localizado nos cantos da imagem sempre possui três *matches*, sendo estes os outros pixels que estão nos cantos da *geometry image*.

Por fim, os *pixels de borda tipo 3* são os quatro pixels localizados exatamente no centro de suas respectivas bordas. Estes pixels não possuem *matches*, de modo que são únicos na *geometry image*. A figura 4.2 ilustra as três categorias de *pixels de borda* existentes em uma *geometry image* 7x7 obtida a partir de uma parametrização esférica.

Figura 4.2: Os tipos de *pixel de borda* existentes em uma *SGIM* com 7 pixels de largura e 7 pixels de altura. Os pixels representados por linhas diagonais representam o caso mais comum: todos possuem apenas um *match*, cuja posição é dada pelas eqs. (4.1) and (4.2) (tipo 1). Os pixels representados por linhas horizontais representam o mesmo vértice, portanto são *matches* entre si (tipo 2). Por fim, os pixels representados por uma cor sólida cinza não possuem nenhum *match* (tipo 3).



Fonte: O autor.

A figura 4.3 mostra todos os *matches* da *geometry image* ilustrada na figura 4.2.

4.3 Reconstrução de uma *Geometry Image*

Para que seja possível reconstruir a malha poligonal após a *geometry image* ter sido filtrada, é necessário saber como recuperar todos os vértices e arestas do objeto tridimensional a partir da imagem. Para isto, o seguinte conjunto de regras é utilizado:

- Todo pixel da *geometry image* define um vértice na malha poligonal. Os canais R, G e B da imagem representam as coordenadas X, Y e Z no espaço tridimensional, respectivamente.
- Pixels podem possuir a mesma cor na *geometry image* e, portanto, representar o mesmo vértice. Pixels que representam o mesmo vértice devem resultar em um único vértice na malha reconstruída.

Figura 4.3: Os *matches* existentes em uma *SGIM* com 7 pixels de largura e 7 pixels de altura. Cada vértice é representado por uma letra do alfabeto. Pixels identificados pela mesma letra representam o mesmo vértice e portanto são *matches* um do outro.

A	B	C	D	C	B	A
E						K
F						L
G						M
F						L
E						K
A	H	I	J	I	H	A

Fonte: O autor.

- As arestas da malha são implicitamente definidas pelas posições dos pixels na *geometry image*. Cada *grid* 2x2 de pixels existente na *geometry image* define cinco arestas, que juntas formam dois triângulos. A diagonal escolhida para a formação dos triângulos deve ser aquela que possui menor tamanho no espaço tridimensional. A figura 4.4 ilustra as duas maneiras que os triângulos podem ser formados.

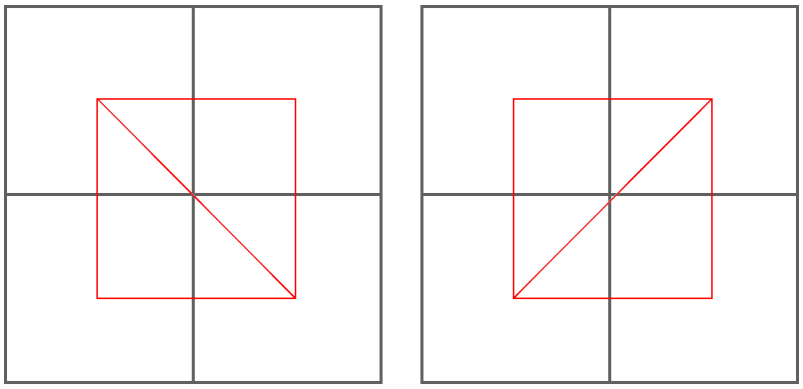
4.4 Direção de Filtragem

O filtro *DTFSGIM* deve ser aplicado diretamente sob uma *geometry image* obtida a partir de uma parametrização esférica. Além disso, é desejável que o filtro tenha natureza recursiva, assim como o filtro *domain transform*. Com isto em mente, torna-se necessário analisar as consequências de filtrar a *geometry image* em diferentes direções com relação à malha poligonal resultante.

No filtro *domain transform* implementado de maneira recursiva (*Recursive Filtering* ou *RF*), a imagem é filtrada de quatro maneiras diferentes:

- Da esquerda para a direita: Todas as linhas da imagem são filtradas de maneira independente. A filtragem de cada linha começa no pixel mais à esquerda (isto é, na borda esquerda da imagem) e termina no pixel mais à direita (isto é, na borda direita da imagem).

Figura 4.4: Uma ilustração mostrando as duas maneiras que as *grids* 2x2 de uma *geometry image* podem ser renderizadas. Na esquerda, a diagonal é formada entre o pixel superior esquerdo e o pixel inferior direito. Esta situação ocorre quando a norma do vetor formado pela diferença entre os vértices definidos pelos pixels superior esquerdo e inferior direito é menor do que a norma do vetor formado pela diferença entre os vértices definidos pelos pixels superior direito e inferior esquerdo. A imagem da direita ilustra o outro caso possível.



Fonte: O autor.

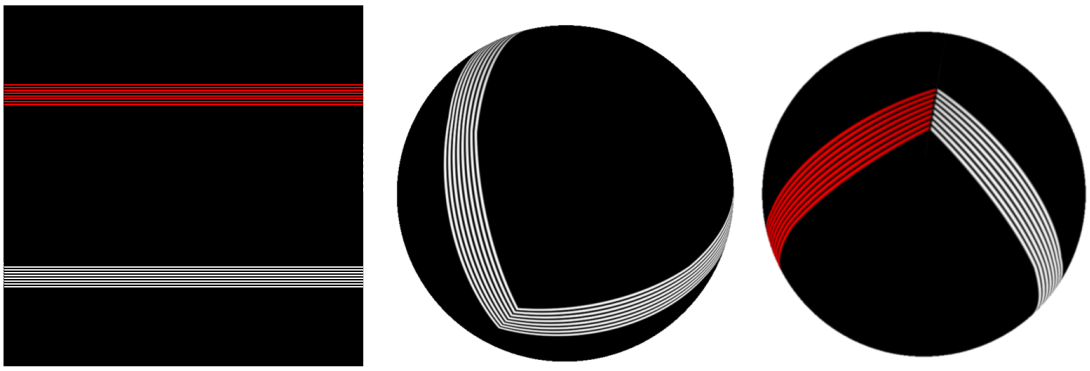
- Da direita para a esquerda: Todas as linhas da imagem são novamente filtradas de maneira independente. Todavia, desta vez a filtragem de cada linha começa no pixel mais à direita (isto é, na borda direita da imagem) e termina no pixel mais à esquerda (isto é, na borda esquerda da imagem).
- De cima para a baixo: Todas as colunas da imagem são filtradas de maneira independente. A filtragem de cada coluna começa no pixel mais acima (isto é, na borda superior imagem) e termina no pixel mais abaixo (isto é, na borda inferior da imagem).
- De baixo para a cima: Todas as colunas da imagem são novamente filtradas de maneira independente. Todavia, desta vez a filtragem de cada coluna começa no pixel mais abaixo (isto é, na borda inferior imagem) e termina no pixel mais acima (isto é, na borda superior da imagem).

Esta análise mostra que, no caso do filtro *domain transform*, é possível filtrar a imagem por completo filtrando-se apenas suas linhas e suas colunas, de maneira totalmente independente. Portanto, não é necessário filtrar a imagem em uma direção mais complexa (por exemplo, uma direção de 45 graus).

Para a construção do filtro *DTFSGIM* seria vantajoso que a filtragem da *geometry image* pudesse ser feita de maneira similar ao filtro *domain transform*. Isto evitaria a necessidade de filtrar a imagem em direções mais complexas. Para determinar se isto é possível, a figura 4.5 mostra, através de texturas, como que linhas e colunas da *geometry*

image (que representa uma esfera) são mapeadas sobre a superfície tridimensional que está sendo definida. Desta maneira, é possível obter-se uma ideia visual de como que uma filtragem nestas direções estaria filtrando a superíficie da malha poligonal.

Figura 4.5: Uma textura aplicada em uma malha poligonal esférica. Na esquerda, é mostrada a textura. Ela possui as mesmas dimensões da *geometry image* que originou a malha poligonal esférica. No centro, a malha poligonal esférica com a textura aplicada de maneira que cada pixel da textura é mapeado para o vértice definido por aquele mesmo pixel na *geometry image*. Na direita, uma outra visão da mesma malha poligonal mostrada na imagem central.

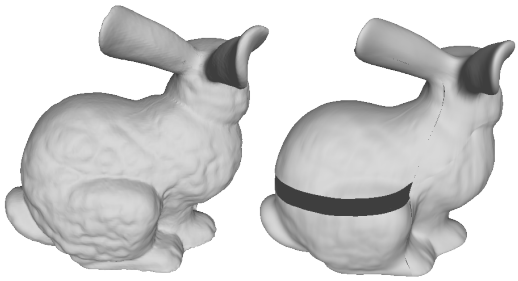


Fonte: O autor.

A figura 4.5 mostra que, de fato, uma filtragem simples de algumas linhas da *geometry image* resulta em uma filtragem contínua e consistente da superífice da malha poligonal representada por aquela imagem. É importante salientar que estas características são verdadeiras apenas quando tratando-se de *geometry images* obtidas através de uma parametrização esférica. Baseado nestas observações, pode-se então tentar filtrar a *geometry image* de diferentes maneiras para observar que tipo de resultados são obtidos.

A primeira tentativa consiste em filtrá-la aplicando um filtro recursivo de borramento simples, tratando a *geometry image* como se fosse uma imagem comum. A figura 4.6 mostra um exemplo desta tentativa. Como pode ser visto, a malha original foi suavizada, porém abriu-se um "buraco" no meio do objeto. Este resultado ilustra uma característica importante das *geometry images* e que sempre devem ser levadas em consideração: a imagem recebida pelo filtro representa uma superfície fechada, e portanto a *GIM* deve ser vista como um domínio periódico. Isto implica que quando uma linha da imagem é filtrada, por exemplo, iniciando em um pixel da borda esquerda em direção à borda direita, a filtragem não deve necessariamente parar quando o pixel da borda direita é finalmente filtrado. Este fenômeno é bem mostrado na figura 4.5. Na terceira imagem, é mostrado as linhas brancas encontrando as linhas vermelhas. Isto é causado justamente na borda da *geometry image* e é neste ponto que o "buraco" é aberto.

Figura 4.6: Um exemplo de uma filtragem simples de uma *geometry image*, tratando-a como uma imagem comum. Na esquerda, a malha poligonal reconstruída a partir da *geometry image* original. Na direita, a malha poligonal reconstruída a partir da *geometry image* filtrada.



Fonte: O autor.

Para resolver o problema descrito no parágrafo anterior, é necessário, de alguma maneira, continuar filtrando a malha poligonal, sem descontinuidades, quando a borda da *geometry image* é atingida. Como a *geometry image* representa uma malha poligonal fechada, pode-se continuar o processo de filtragem em algum outro ponto da imagem, de modo que isto resulte em uma filtragem efetivamente contínua na superfície da malha poligonal. Se isto não for feito, uma falha aparecerá na malha resultante, como já mostrado na figura 4.6.

Para que seja possível determinar como continuar o processo de filtragem ao atingir um *pixel de borda*, é necessário descobrir o próximo pixel da imagem que deverá ser filtrado e em qual direção prosseguir a filtragem. Como citado anteriormente, existem três tipos de *pixels de borda*. Pode-se, portanto, separar este problema em três casos distintos, baseados no tipo do último *pixel de borda* filtrado:

- *Pixels de borda tipo 1*: Neste caso mais simples, o próximo pixel a ser filtrado é o seu *match*. Como *pixels de borda* deste tipo possuem apenas um único *match*, ele pode ser obtido de maneira trivial, utilizando as eqs. (4.1) and (4.2). Como a *geometry image* foi obtida a partir de uma parametrização esférica, a nova direção de filtragem será a mesma direção que estava sendo utilizada, porém com o sentido invertido. A figura 4.7 (a) mostra um exemplo de *pixels de borda tipo 1*.
- *Pixels de borda tipo 2*: Neste caso, o *pixel de borda* possui três *matches* distintos. Portanto, é necessário determinar uma regra para escolher qual *match* deve ser escolhido. Para que a filtragem continue de maneira suave na malha poligonal, o seguinte conjunto de regras é definido:
    - O *match* que será escolhido sempre estará na mesma borda do último pixel

filtrado. Como sempre haverão dois pixels que satisfazem este requisito, pois um pixel que está no canto da imagem faz parte de duas bordas diferentes, a borda que deverá ser escolhida é aquela que é ortogonal à direção que a *geometry image* estava sendo filtrada.

- A nova direção de filtragem, assim como no caso dos *pixels de borda tipo 1*, será a mesma direção que estava sendo utilizada, porém com o sentido invertido.

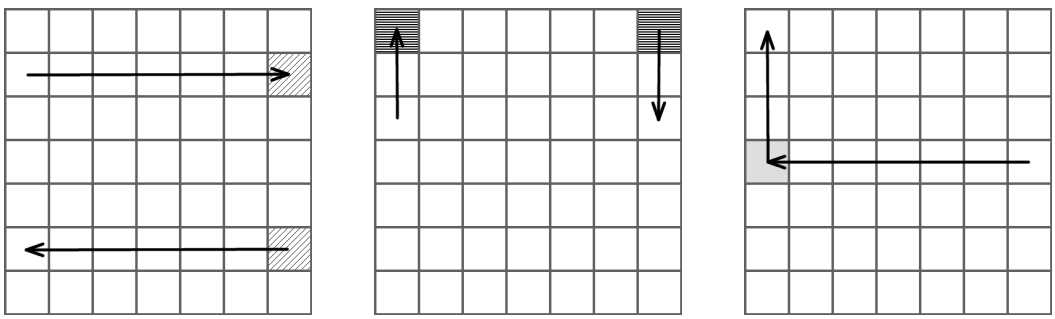
A figura 4.7 (b) mostra um exemplo de *pixels de borda tipo 2*.

- *Pixels de borda tipo 3*: Pixels deste tipo não possuem *matches* e portanto não é óbvio qual posição deve ser utilizada para continuar a filtragem. Felizmente, como a *geometry image* é fruto de uma parametrização esférica, a filtragem pode continuar de maneira contínua e suave na malha poligonal apenas alterando a sua direção, sem necessidade de escolher uma nova posição de filtragem. A direção deve ser alterada de modo que a filtragem continue ao longo da borda que contém o *pixel de borda* atingido. Isto significa que a nova direção formará um ângulo de 90 graus com a direção anterior. Neste caso, o sentido de filtragem é arbitrário, pois independente do sentido escolhido o mesmo conjunto de vértices estará sendo filtrado, uma vez que há uma simetria de *matches* nas bordas da *geometry image* (esta simetria pode ser vista na figura 4.3). A figura 4.7 (c) mostra um exemplo de *pixels de borda tipo 3*.

Com esta análise, é sempre possível continuar o processo de filtragem, mesmo quando um *pixel de borda* é filtrado, independente do seu tipo. Isto resolve o problema apresentado na figura 4.6.

Figura 4.7: Ilustração mostrando como o processo de filtragem deve continuar quando um *pixel de borda* é filtrado. Cada imagem ilustra um tipo diferente de *pixel de borda*.

(a) *Pixel de borda tipo 1*. (b) *Pixel de borda tipo 2*. (c) *Pixel de borda tipo 3*.



Fonte: O autor.

4.5 Pixels de Borda Repetidos

Como já mostrado, *pixels de borda* em geral possuem *matches*, que são outros *pixels de borda*, em posições diferentes da *geometry image*, que representam o mesmo vértice da malha poligonal. Na seção 4.4, foi mostrado em detalhes como que este fenômeno pode ser explorado para ser possível determinar como continuar a filtragem ao chegar na borda da *geometry image*. Entretanto, um outro problema emerge desta situação: como diferentes *pixels de borda* podem, potencialmente, representar o mesmo vértice, toda vez que um destes pixels é filtrado, seu *match* também estará sendo implicitamente filtrado, pois ambos representam o mesmo vértice da malha poligonal. Para assegurar-se de que isto aconteça, os valores dos *matches* devem ser atualizados à medida que os *pixels de borda* são filtrados.

A partir desta análise, aparece ainda um outro problema: uma filtragem comum similar ao filtro *domain transform* apresentará um desbalanceamento em relação à quantidade de vezes que cada pixel da imagem é filtrado. Por exemplo, se a primeira linha da imagem for filtrada por completo da esquerda para a direita, todos os pixels terão sido filtrados, na prática, duas vezes, com exceção do pixel central que possui *tipo 3* (este é filtrado apenas uma vez). Por exemplo, se a primeira linha da *geometry image* mostrada na figura 4.3 for filtrada da esquerda para a direita, os vértices A, B e C serão filtrados duas vezes, enquanto que o vértice D será filtrado apenas uma vez. Como se isto não bastasse, os outros dois pixels que estão nos cantos da *geometry image* e que não fazem parte desta linha também representam o vértice A, e portanto eles também terão sido filtrados duas vezes.

O desbalanceamento citado no parágrafo anterior causa dois grandes problemas. O primeiro é que a filtragem em excesso resultará em um erro. Este erro será acumulado à medida que a filtragem ocorre e isto resultará em uma malha resultante defeituosa. O segundo problema é que filtrar uma borda da imagem por completo não é correto do ponto de vista da malha poligonal, pois esta filtragem causaria uma quebra no *caminho de filtragem*. Na prática, os vértices seriam corretamente filtrados até o ponto em que o pixel central da borda é filtrado. A partir daí, o sentido de filtragem seria invertido, e os mesmos vértices da malha que já haviam sido utilizados serão novamente filtrados, desta vez, porém, no sentido contrário. É necessário, portanto, alterar o método de filtragem utilizado no filtro *domain transform* para ser possível levar em consideração todas peculiaridades existentes em uma *geometry image* construída a partir de uma parametrização

esférica.

5 FILTRO *DTFSGIM*

O objetivo do filtro *DTFSGIM* é filtrar uma *geometry image* obtida a partir de uma parametrização esférica de modo que os objetivos citados na seção 1.3 sejam atingidos. Nesta seção, serão explicados em detalhes todos os métodos e algoritmos de filtragem utilizados para atingir estes objetivos. Toda a análise descrita no capítulo 4 será crucial para o desenvolvimento deste filtro.

5.1 Etapas de Filtragem

Como já explicitado durante o texto, o filtro *DTFSGIM* será implementado como um filtro recursivo. Todavia, por causa dos problemas citados na seção 4.4, é necessário definir uma nova maneira de filtrar a *geometry image* que leve em consideração todas suas peculiaridades. O filtro *DTFSGIM* é, portanto, dividido em quatro *etapas* ou *sub-filtros*:

- *Etapa H*: Nesta etapa, todas as linhas da *geometry image* são filtradas, exceto a linha superior, a linha inferior e a linha central. A filtragem ocorre de tal modo que ao atingir um *pixel de borda*, seu *match* é encontrado e a filtragem continua conforme explicado na seção 4.4. Ao atingir o segundo *pixel de borda*, considera-se a linha totalmente filtrada e inicia-se a filtragem da próxima linha. Todos os *pixels de borda* envolvidos nesta etapa são do *tipo 1*. É importante notar que o segundo *pixel de borda* atingido será sempre idêntico ao *pixel de borda* que iniciou a filtragem desta linha, dadas as propriedades descritas na seção 4.2. Além disso, na *Etapa H* apenas metade das linhas precisam ser filtradas, tendo em vista que, na prática, a filtragem de uma linha acarreta em duas linhas sendo filtradas, como já descrito. Esta etapa é mostrada visualmente na figura 5.1 (a).
- *Etapa V*: Nesta etapa, todas as colunas da *geometry image* são filtradas, exceto a coluna mais à esquerda, a coluna mais à direita e a coluna central. Assim como na *Etapa H*, ao atingir um *pixel de borda* durante a filtragem, seu *match* é encontrado e a filtragem continua conforme explicado na seção 4.4. Todos os *pixels de borda* envolvidos nesta etapa são do *tipo 1*. Ao atingir o segundo *pixel de borda*, considera-se a coluna totalmente filtrada e inicia-se a filtragem da próxima coluna. O segundo *pixel de borda* atingido também será sempre idêntico ao *pixel de borda* que iniciou a filtragem desta coluna e, novamente, apenas metade das colunas pre-

cisam ser filtradas, por motivos análogos ao da *etapa H*. Esta etapa é mostrada visualmente na figura 5.1 (b).

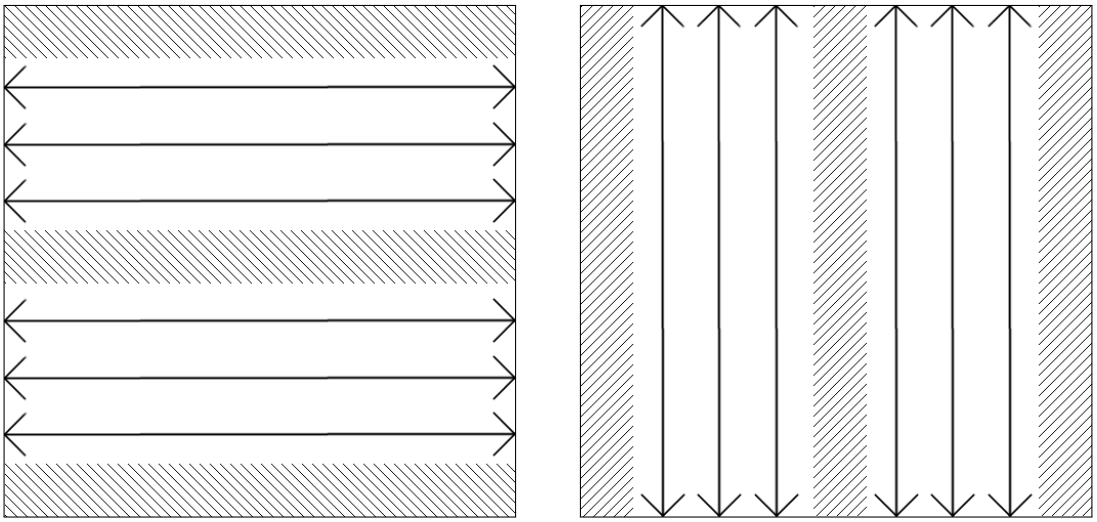
- *Etapa C*: Nesta etapa, a coluna central, a linha superior e a linha inferior da *geometry image* são filtradas. Na prática, apenas metade da linha superior e da linha inferior são filtradas. A outra metade das linhas acaba sendo filtrada de maneira implícita, pelos motivos descritos na seção 4.5. Desta maneira, assegura-se que os *pixels de borda* das linhas superior e inferior não serão filtrados mais vezes do que deveriam. Esta etapa envolve *pixels de borda* do *tipo 2* e do *tipo 3*. Considerando-se, por exemplo, que a *etapa C* é iniciada no pixel central da borda inferior da *geometry image* em direção ao pixel central da borda superior para filtrar a coluna central, no final da filtragem desta coluna será encontrado um *pixel de borda* do *tipo 3*. Como descrito na seção 4.4, para este caso de *pixel de borda tipo 3*, pode-se escolher tanto o pixel à esquerda do *pixel de borda* quanto o pixel à direita para dar prosseguimento ao processo de filtragem. Supondo-se que o pixel da direita foi escolhido, começa-se a filtragem da linha superior, até o encontro de um *pixel de borda tipo 2*, no canto da imagem. Neste caso, é descrito na seção 4.4 que a filtragem deve continuar a partir do pixel que está no canto inferior direito da *geometry image*. Por fim, a linha inferior é filtrada até chegar no *pixel de borda* inicial. Esta etapa é mostrada visualmente na figura 5.1 (c).
- *Etapa* π: Nesta etapa, a linha central, a coluna mais à esquerda e a coluna mais à direita da *geometry image* são filtradas. Esta etapa é análoga à *etapa C*, assim como a *etapa V* é análoga à *etapa H*. Esta etapa é mostrada visualmente na figura 5.1 (d).

A figura 5.1 mostra visualmente as quatro etapas citadas. Em todas as etapas a filtragem ocorre em ambos os sentidos, como é mostrado na imagem por uma seta bidirecional. Isto implica que, após uma aplicação dos quatro *sub-filtros* em sequência, todos os pixels da *geometry image* terão sido filtrados quatro vezes, com exceção dos quatro *pixels de borda tipo 3*, que terão sido filtrados apenas duas vezes. Estes quatro pixels são chamados de *singularidades* do filtro *DTFSGIM*, já que acabam sendo filtrados menos que os outros pixels.

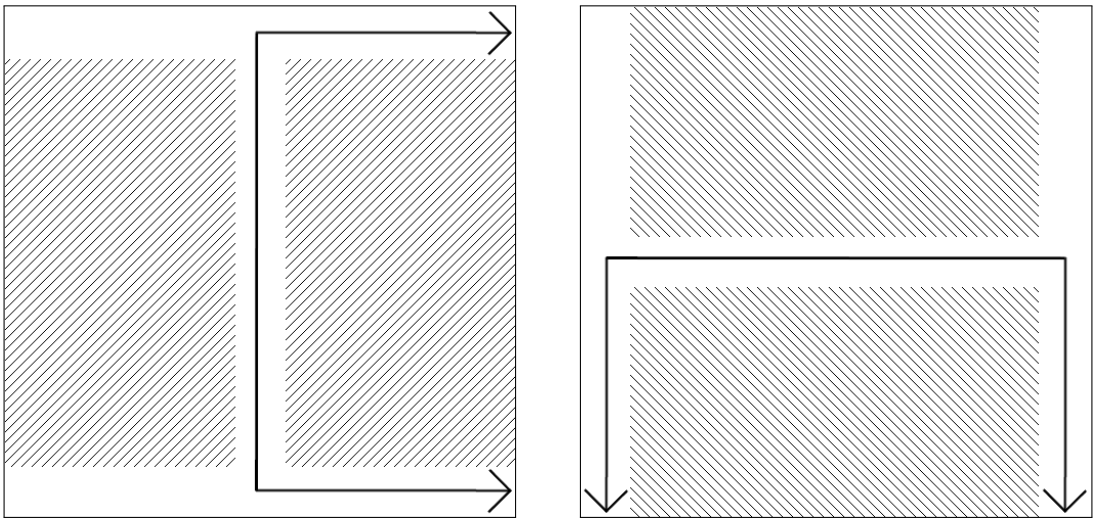
A separação do filtro em quatro etapas visa garantir que todos os caminhos de filtragem escolhidos durante a filtragem da *geometry image* resultem em caminhos contínuos e suaves na superfície da malha poligonal. Durante a aplicação dos quatro *sub-filtros*, todos os *pixels de borda* filtrados devem ter seus valores copiados para todos seus *matches*, de modo que não haja inconsistências na filtragem.

Figura 5.1: Uma ilustração do funcionamento dos quatro *sub-filtros* do filtro *DTFSGIM*.

(a) *Sub-filtro H* (b) *Sub-filtro V*



(c) *Sub-filtro C* (d) *Sub-filtro* π



Fonte: O autor.

5.2 Curvatura

Um dos objetivos mais importantes do filtro *DTFSGIM* é que ele seja capaz de preservar a curvatura da malha poligonal ao filtrar a *geometry image*. Para isto ser possível, é necessário que o filtro seja capaz de extrair informações relacionadas à curvatura da malha poligonal a partir da *geometry image* recebida.

Para preservar arestas importantes da imagem, o filtro *domain transform* define uma *transformada de domínio*, como explicado na seção 2.3. Esta transformada permite com que, antes da filtragem, um valor, definido pela eq. (2.2), seja atribuído a cada pixel da imagem. Este valor é proporcional à diferença entre os canais de cores do pixel de inte-

resse e do pixel imediatamente anterior a este, considerando alguma direção na imagem. A transformada de domínio é definida matematicamente pela equação eq. (2.1). É importante notar que o valor da *transformada de domínio* depende do pixel que foi considerado o anterior ao pixel de interesse, de modo que, dependendo do sentido da filtragem, valores diferentes devem ser utilizados. Por este motivo, no filtro *domain transform* é necessário calcular as *tranformadas de domínio* horizontal e vertical de cada pixel. O sentido da filtragem determinará qual valor deve ser usado durante a aplicação do filtro.

No caso do filtro *DTFSGIM*, também busca-se atribuir um valor para cada pixel, de maneira similar ao que é feito no filtro *domain transform*. Neste caso, porém, este valor deve fornecer informações sobre a curvatura da malha poligonal no vértice definido por aquele pixel. Como este valor também representa uma *transformada de domínio*, ela será chamada da mesma maneira para o filtro *DTFSGIM*. Nesta subseção será explicado, em detalhes, como a *transformada de domínio* é calculada para este filtro.

5.2.1 Cálculo das Normais

O primeiro passo para encontrar a *transformada de domínio* de cada pixel é definir, com base em algum método, o vetor normal à cada pixel da imagem (lembrando que cada pixel representa um ponto sobre a superfície tridimensional definida pela malha). Para isto, primeiramente os vértices e arestas da malha são recuperados a partir da *geometry image* e, em seguida, os vetores normais são calculados seguindo o método de Gouraud (GOURAUD, 1971). O algoritmo completo pode ser descrito em alto nível pelo algoritmo 1.

O algoritmo 1 mostra um método genérico de definir vetores normais para todos os pixels da *geometry image*. Com base nestes vetores normais, pode-se determinar uma maneira de definir uma *transformada de domínio* para cada pixel.

5.2.2 Transformada de Domínio

A *transformada de domínio* do filtro *DTFSGIM* é definida como:

$$d[x] = 1 + \frac{\sigma_s}{\sigma_r}\|N[x] - N[x-1]\|dx\,, \tag{5.1}$$

onde $\|N[x] - N[x-1]\|$ é a norma do vetor $N[x] - N[x-1]$ e:

Algorithm 1 Definição de vetores normais para cada pixel da *geometry image*

Entrada: *SGIM* X

Saída: *Buffer* Y, contendo os vetores normais de X

1: A partir da *geometry image* recebida, todos vértices e arestas da malha original são extraídos, com base no que foi descrito na seção 4.3.
2: Aloca-se o buffer Y, que irá conter as normais de X, de tamanho suficiente para armazenar o vetor normal de todos pixels de X. Todos vetores são inicializados como vetores nulos.
3: Para cada triângulo existente na malha com vértices *A*, *B* e *C*, sua normal é calculada da seguinte maneira:
   1. Um vetor $\vec{x}$ é definido como sendo $\vec{x} = B - A$.
   2. Um vetor $\vec{y}$ é definido como sendo $\vec{y} = C - A$.
   3. Um vetor $\vec{n}$ é definido como sendo o produto vetorial entre $\vec{x}$ e $\vec{y}$, ou seja, $\vec{n} = \vec{x} \times \vec{y}$. O vetor $\vec{n}$ é definido como sendo a normal não-normalizada do triângulo *ABC*.
4: Cada vetor $\vec{n}$ obtido é somado ao vetor normal do vértice *A*, do vértice *B* e do vértice *C* no *buffer* Y.
5: Todos vetores normais existentes no *buffer* Y são normalizados, como segue: $\vec{v'} = \frac{\vec{v}}{\|\vec{v}\|}$, onde $\vec{v}$ é o vetor normal de um vértice armazenado em Y e $\vec{v'}$ é o seu novo valor.

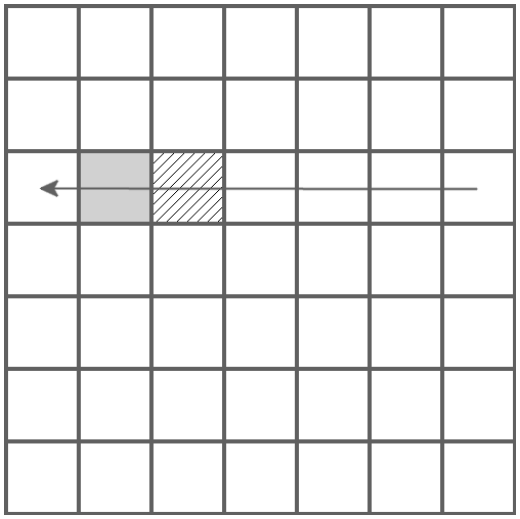
- σ_s controla a extensão espacial da filtragem. Quanto maior este valor, maior a suavização da malha.
- σ_r controla a curvatura da filtragem. Quanto menor este valor, menor será a tolerância do filtro para filtrar curvaturas importantes. Em contrapartida, quanto maior este valor, mais o filtro se aproxima de um filtro de suavização comum.
- $N[x]$ é uma função que fornece o vetor normal de um pixel x da *geometry image*. Os vetores normais são calculados como visto na subseção 5.2.1.
- $d[x]$ é o valor da *transformada de domínio* para um pixel x da *geometry image*.

A eq. (5.1) permite o cálculo das *transformadas de domínio* de todos os pixels da *geometry image*. O termo $N[x-1]$, presente na eq. (5.1), representa o vetor normal do pixel que foi filtrado logo antes do pixel atual. Portanto, o valor de $N[x-1]$ não é constante para um pixel x, pois o pixel $x-1$ depende da etapa de sub-filtragem que está sendo aplicada. Por exemplo, a figura 5.2 mostra uma situação em que o pixel $x-1$ muda dependendo do filtro que está sendo aplicado. Todas normais utilizadas na eq. (5.1) devem estar normalizadas.

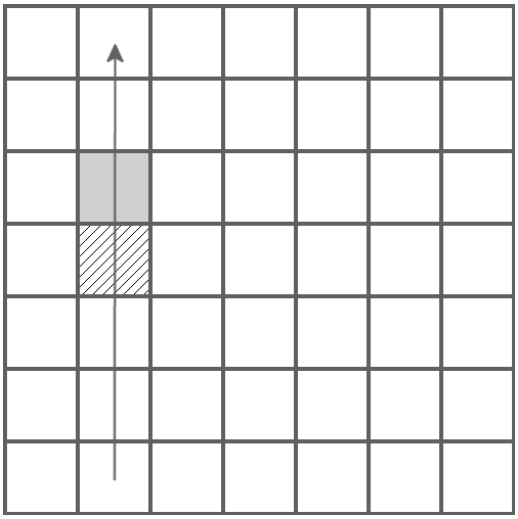
Os valores da *transformada de domínio* $d[x]$ de cada pixel da *SGIM* serão necessários durante a filtragem. Portanto, é necessário que eles estejam disponíveis de alguma forma durante a execução do filtro. Existem três maneiras de calcular as transformadas:

Figura 5.2: Duas situações onde o pixel $x-1$ considerado na eq. (5.1) muda, dependendo do sentido de filtragem considerado.

(a) Um dos passos do *sub-filtro H*. Nesta situação, o pixel $x-1$ está imediatamente à direita do pixel x.



(b) Um dos passos do *sub-filtro V*. Nesta situação, o pixel $x-1$ está imediatamente abaixo do pixel x.



Fonte: O autor.

1. A maneira mais simples, porém mais lenta, é calcular os valores de $d[x]$ dinamicamente, durante a própria filtragem. As duas principais vantagens deste método são a simplicidade e a economia de memória. O principal problema é que será necessário recalcular a *transformada de domínio* de cada pixel todas as vezes que ele for filtrado, mesmo se o valor já tiver sido calculado anteriormente.
2. A segunda opção é pré-calcular as transformadas $d[x]$ com a utilização de quatro *buffers* do tamanho da *SGIM*. Cada um dos *buffers* é responsável por armazenar os valores da *transformada de domínio* de cada pixel da imagem, porém considerando sentidos de filtragem diferentes. Os quatro sentidos de filtragens existentes nos *sub-filtros H, V, C e* π são:

   - Horizontal, da esquerda para a direita.
   - Horizontal, da direita para a esquerda.
   - Vertical, da baixo para cima.
   - Vertical, de cima para baixo.

   Com a utilização de quatro *buffers* é possível cobrir todos os sentidos possíveis. Estes *buffers* devem, portanto, ser consultados durante o processo de filtragem para evitar que as transformadas sejam recalculadas. Este método é mais eficiente do que o primeiro, porém necessita de uma quantidade considerável de memória para

o armazenamento dos *buffers*.

3. A terceira maneira é, na verdade, uma otimização da segunda opção apresentada acima. É possível utilizar apenas dois *buffers*, um para armazenar as *transformadas de domínio* horizontais e outro para armazenar as verticais. Se, por exemplo, dois *buffers* forem utilizados, um com as *transformadas de domínio* horizontais, da esquerda para a direita (*buffer 1*) e outro com as verticais, de baixo para cima (*buffer 2*), é possível obter o valor de $d[x]$ para todos pixels da imagem, independente do sentido de filtragem. Para este exemplo, a escolha deve ser feita da seguinte maneira:

   - Se o sentido de filtragem for horizontal, da esquerda para a direita, basta consultar o *buffer 1*.
   - Se o sentido de filtragem for horizontal, da direita para a esquerda, é necessário consultar o *buffer 1*, mas considerar a *transformada de domínio* do pixel $x - 1$, ou seja, do pixel imediatamente anterior a este (neste caso, o pixel à direita). Isto é possível porque a eq. (5.1) mostra que o cálculo de $d[x]$ depende do módulo da diferença $N[x] - N[x - 1]$. Como trata-se do módulo desta diferença, a alteração para $N[x - 1] - N[x]$ não deve alterar o resultado.
   - Se o sentido da filtragem for vertical, de baixo para cima, basta consultar o *buffer 2*.
   - Se o sentido da filtragem for vertical, de cima para baixo, é necessário consultar o *buffer 2*, mas considerar a *transformada de domínio* do pixel $x - 1$, ou seja, do pixel imediatamente anterior a este (neste caso, o pixel acima). Os motivos são análogos aos apresentados para o caso horizontal, da direita para a esquerda.

   Com a utilização de apenas dois *buffers*, há uma grande economia de memória e também pré-processamento quando comparado ao método 2.

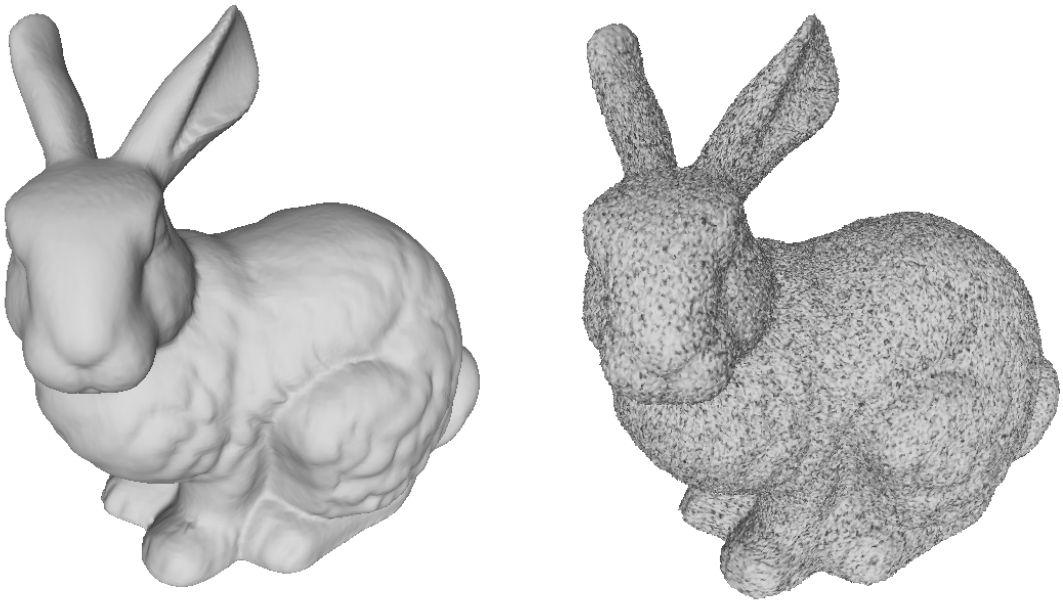
5.2.3 Pré-processamento dos *Buffers*

Na subseção 5.2.2 é explicado como obter-se a *transformada de domínio* de cada pixel e como armazenar estes resultados em dois *buffers* para a consulta posterior durante a execução do filtro *DTFSGIM*. Nesta subseção será apresentada uma técnica de pré-processamento do *buffer* das normais ($N[x]$), que é utilizado para a obtenção destes outros

dois *buffers*, de modo a melhorar o resultado do filtro e evitar determinados problemas.

A figura 5.3 mostra duas malhas, sendo uma delas altamente ruidosa. Como explicado na subseção 5.2.2, a *transformada de domínio* de cada pixel da imagem é calculada a partir das normais calculadas através do algoritmo 1. Para uma malha com baixo índice de ruído, como a mostrada na figura 5.3 (a), esta técnica apresenta ótimos resultados. Contudo, no caso de uma malha ruidosa, como a mostrada na figura 5.3 (b), as normais geradas pelo algoritmo 1 são adulteradas pelo ruído presente na malha. Isto ocorre porque a introdução do ruído na malha poligonal modifica os triângulos da malha, de modo que as normais destes triângulos mudam de direção. O erro introduzido nas normais é proporcional à intensidade do ruído adicionado, de modo que quanto maior o ruído, menor é a eficiência das *transformadas de domínio* para preservar as curvaturas relevantes da malha poligonal.

Figura 5.3: Duas malhas poligonais, sendo uma altamente ruidosa.
(a) *Uma malha poligonal regular.* (b) *Uma malha poligonal altamente ruidosa.*



Fonte: O autor.

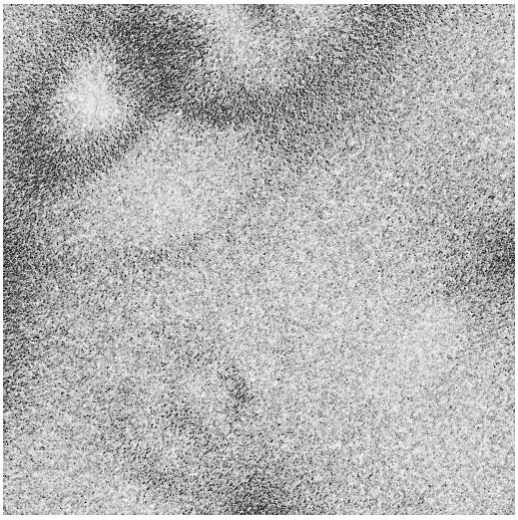
A figura 5.4 mostra as curvaturas, obtidas a partir dos *buffers* das normais das malhas apresentadas na figura 5.3, de uma maneira visualmente interessante. Como discutido, é possível perceber que as curvaturas da malha sem introdução de ruído são muito bem definidas e de fácil identificação, ao passo de que as curvaturas da malha ruidosa não apresentam informações úteis sobre como as normais se comportam localmente na malha.

A utilização das curvaturas representadas pela figura 5.4 (b) na eq. (5.1) resultará em *transformadas de domínio* com pouca informação útil, de modo que a preservação das

Figura 5.4: Uma análise visual das curvaturas (obtidas através do algoritmo 1) das malhas poligonais mostradas na figura 5.3.

(a) *As curvaturas da malha poligonal mostrada na figura 5.3 (a).*

(b) *As curvaturas da malha poligonal mostrada na figura 5.3 (b).*



Fonte: O autor.

curvaturas não será satisfatória. Por isso, é necessário submeter o *buffer* de normais a um processamento adicional antes de utilizá-lo no *filtro DTFSGIM*. Este processamento visa recuperar as informações das normais em malhas ruidosas.

O processo consiste em suavizar (borrar) o *buffer* através de um filtro recursivo comum. Como os ruídos introduzidos nas malhas poligonais costumam ter uma natureza aleatória, o borramento deve ser capaz de remover este ruído e ainda manter as curvaturas da malha original.

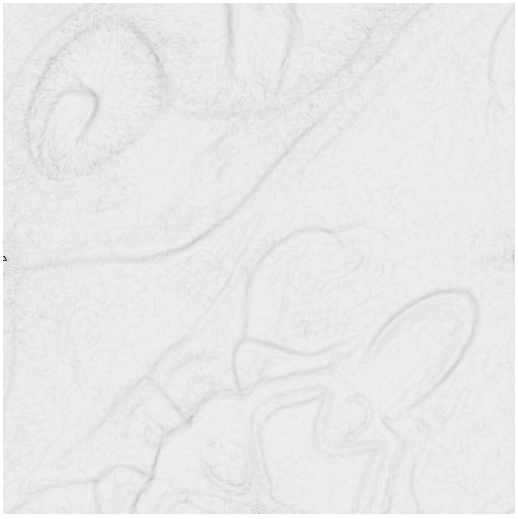
Como o *buffer* das normais é gerado a partir de uma *SGIM*, o borramento deve ser feito utilizando as mesmas etapas citadas na seção 5.1. Desta maneira, a suavização levará em conta aspectos importantes da *SGIM*, como a sua natureza periódica. A figura 5.5 mostra as curvaturas apresentadas na figura 5.4, agora suavizadas. A imagem mostra que as curvaturas da malha poligonal que possuía uma baixo índice de ruído continuaram corretas e as curvaturas da malha altamente ruidosa passaram a apresentar um resultado mais adequado.

O pré-processamento dos *buffers* das normais, como descrito nesta subseção, não é útil apenas para corrigir o problema das malhas ruidosas. O borramento das normais faz com que o filtro *DTFSGIM* apresente melhores resultados mesmo em malhas com baixo índice de ruído. Isto ocorre porque a suavização do *buffer* faz com que as *transformadas de domínio* geradas não variem de forma abrupta entre os pixels da imagem e sim de forma gradual. Isto acaba gerando malhas poligonais visualmente mais interessantes,

Figura 5.5: Uma análise visual das curvaturas (obtidas através do algoritmo 1) das malhas poligonais mostradas na figura 5.3 e posteriormente suavizadas.

(a) *As curvaturas suavizadas da malha poligonal mostrada na figura 5.3 (a).*

(b) *As curvaturas suavizadas da malha poligonal mostrada na figura 5.3 (b).*



Fonte: O autor.

pois as partes da malha em que há transição de regiões de baixa curvatura para regiões de alta curvatura acabam ficando mais suaves.

Como já citado, a suavização dos *buffers* é feita com um filtro recursivo comum, cuja equação recursiva é dada pela eq. (5.4), descrita abaixo, com $r = n_s$, onde n_s é uma constante que controla a intensidade do filtro.

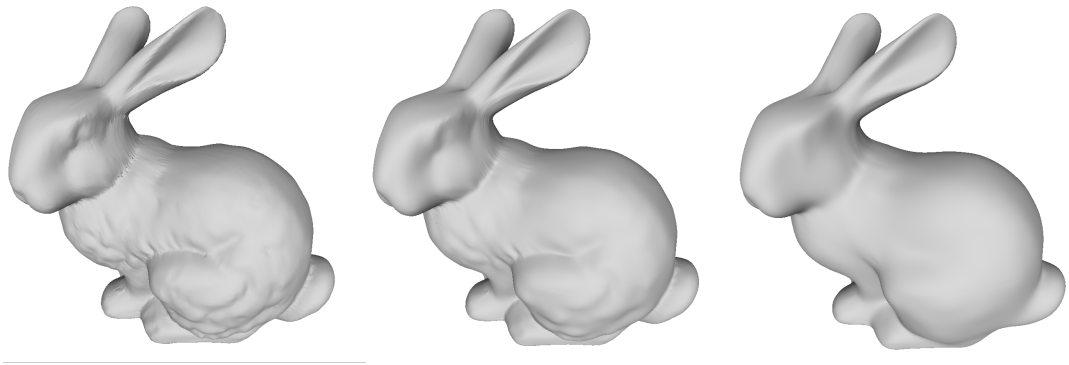
5.2.4 Impacto da escolha de n_s

A escolha do parâmetro n_s impacta na intensidade da suavização do *buffer* das normais e, portanto, também impacta nas *transformadas de domínio* de todos pixels da imagem. Um valor baixo de n_s dificultará a detecção da curvatura de malhas ruidosas e introduzirá regiões defeituosas na malha por conta das variações abruptas das *transformadas de domínio*, discutidas na subseção anterior. Já um valor alto de n_s fará com que sejam perdidas informações importantes da curvatura, por conta da alta suavização dos *buffers* das normais.

A figura 5.6 mostra o efeito da variação de n_s na filtragem de uma malha poligonal. Como observado no parágrafo anterior, um n_s baixo introduz defeitos na malha (mais visíveis no pescoço do coelho), ao passo que um n_s alto limita a preservação das curvaturas.

Figura 5.6: Uma malha poligonal filtrada com diferentes valores de n_s.

(a) $n_s = 0.1$ (b) $n_s = 0.6$ (c) $n_s = 0.9$



Fonte: O autor.

Em geral, um valor baixo de σ_r irá necessitar de um valor baixo de n_s, pois as curvaturas devem ser preservadas com maior rigor. Além disso, um valor alto de σ_r possibilita um valor alto de n_s, tendo em vista que neste caso as curvaturas possuem menor importância no processo de filtragem.

A fim de diminuir a quantidade de parâmetros do filtro *DTFSGIM* e aproveitando a relação entre n_s e σ_r, no filtro *DTFSGIM* o parâmetro n_s é obtido através do parâmetro σ_r. Para obter a relação entre n_s e σ_r, foi utilizado o apêndice do artigo original do filtro *domain transform*. Neste apêndice, é dito que um *feedback coefficient* a pode ser obtido a partir de uma variância σ_H^2 através da equação:

$$a = \exp\left(\frac{-\sqrt{2}}{\sigma_H}\right). \tag{5.2}$$

Para o caso de n_s, é necessário escolher um valor de σ_H dependente de σ_r que produza bons resultados. A partir desta análise, o valor de n_s é definido como:

$$n_s = \exp\left(\frac{-\sqrt{2}}{30\sigma_r}\right). \tag{5.3}$$

A eq. (5.3), que foi obtida através da execução de diferentes testes manuais com um certo conjunto de malhas, torna possível a obtenção do parâmetro n_s a partir de σ_r, eliminando assim a necessidade da variável n_s no filtro *DTFSGIM*. Estes testes manuais foram realizados variando o escalar que multiplica o parâmetro σ_r em diferentes malhas e observando os resultados obtidos. No resto deste texto, o parâmetro n_s sempre será obtido através da eq. (5.3), mesmo que isto não seja dito explicitamente.

5.3 Método Recursivo de Filtragem

O algoritmo de filtragem utilizado no filtro *DTFSGIM* é inspirado no filtro *domain transform*. Como já mencionado, o filtro *DTFSGIM* possui natureza recursiva e, portanto, a filtragem de um pixel qualquer depende do valor do último pixel que foi filtrado. A filtragem de um pixel n é dada pela equação:

$$J[n] = (1 - r)I[n] + rJ[n - 1]\,, \tag{5.4}$$

onde:

$$r = a^{d[n]}\,. \tag{5.5}$$

Em um filtro recursivo comum, r assume um valor constante, ou seja, $r = a$. Isto faz com que todos os pixels da imagem sejam filtrados sem distinção. Em contrapartida, no filtro *DTFSGIM*, r não é uma constante, e seu valor é dado pela eq. (5.5). Chamaremos r de *adjusted feedback coefficient*, tendo em vista que sua função é ajustar o *feedback coefficient*, a, considerando informações de curvatura.

Na eq. (5.5), $d[n]$ é o valor da *transformada de domínio* do pixel em consideração e a é considerado o *feedback coefficient*. O valor de r irá variar dependendo de $d[n]$, o que fará com que os pesos utilizados na filtragem variem de acordo com a curvatura da malha, o que é desejado. O cálculo e a escolha da *transformada de domínio*, ou $d[x]$, é mostrado em detalhes na subseção 5.2.2. O valor do *feedback coefficient* a é calculado a partir do desvio padrão desejado do filtro e é dado por:

$$a = \exp(-\sqrt{2}/\sigma_H)\,. \tag{5.6}$$

onde σ_H é o desvio padrão do filtro. Como o filtro *DTFSGIM* possui diferentes iterações, é necessário que o desvio padrão varie a cada iteração de modo que o resultado convirja para a malha desejada. Portanto, para uma iteração i, o valor do desvio padrão σ_{Hi} é dado por:

$$\sigma_{Hi} = \sigma_H \sqrt{3} \frac{2^{N-i}}{\sqrt{4^N - 1}}\,, \tag{5.7}$$

onde σ_H é o desvio padrão total desejado do filtro e N é o número de iterações desejado.

Para o filtro *DTFSGIM*, o desvio padrão σ_H é escolhido de modo que $\sigma_H = \sigma_s$. Isto elimina a variável extra σ_H e faz com que ela seja proporcional ao fator σ_s, que controla a extensão espacial da filtragem. Esta atribuição, em geral, resulta em ótimos resultados.

O número de iterações do filtro *DTFSGIM* é chamado N. O valor de N é variável e determinado pelo usuário do filtro. Um valor baixo de N pode introduzir artefatos visuais na malha resultante, enquanto que um valor alto de N torna o processo de filtragem mais lento.

5.4 Correção

Na figura 4.6 foi mostrado que uma falha surge na malha resultante quando a *SGIM* é filtrada utilizando-se um filtro recursivo comum. Este problema, como já explicado na seção 4.4, decorre do fato que uma *SGIM* representa uma malha fechada e, portanto, trata-se de uma estrutura periódica e deve ser tratada como tal.

Um outro problema pode ocorrer também pelo fato de a *SGIM* possuir um domínio periódico. Como explicado na seção 5.1, o filtro *DTFSGIM* é dividido em quatro *sub-filtros*: *H*, *V*, *C* e π. Em todas estas etapas, diferentes *caminhos de filtragem* são escolhidos e filtrados de maneira independente. *Caminhos de filtragem* representam um caminho contínuo e suave na malha poligonal. Na *SGIM*, um *caminho de filtragem* sempre termina no mesmo pixel que a filtragem foi iniciada. Como o filtro *DTFSGIM* é um filtro recursivo, a filtragem de um determinado pixel x dependerá de todos pixels que foram filtrados anteriormente, isto é, os pixels $x-1$, $x-2$, $x-3$, etc. Aplicando-se a eq. (5.4) de maneira recursiva, temos que:

$$J[n] = (1-r)I[n] + r((1-r)I[n-1] + r((1-r)I[n-2] + r(...)))\,. \tag{5.8}$$

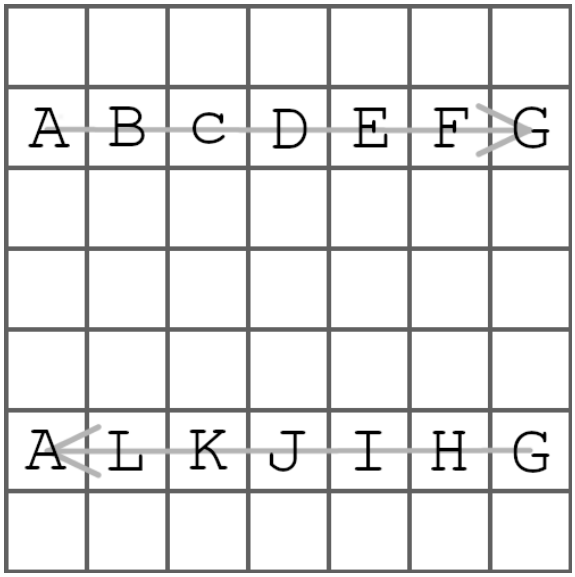
A eq. (5.8) mostra que, de fato, a filtragem de um determinado pixel depende de todos seus antecessores. Portanto, é necessário ter conhecimento prévio de todos valores de filtragem anteriores para ser possível filtrar um determinado pixel da imagem.

No caso de uma imagem comum, geralmente opta-se por condições iniciais de interesse para iniciar o processo de filtragem. Por exemplo, uma heurística comum é determinar como condição inicial um pixel com todos os canais nulos (*R = 0, G = 0, B*

= *0*). Assim, este valor é usado como o termo $J[n-1]$ na eq. (5.4) quando o primeiro pixel de cada linha/coluna é filtrado. Outro valor comum é escolher $J[n-1] = I[n]$ para o primeiro pixel.

A escolha das condições iniciais, como explicada no parágrafo anterior, irá influenciar no resultado final da filtragem. Ao contrário de uma *SGIM*, uma imagem comum não representa uma estrutura periódica, de modo que os resultados obtidos desta maneira são suficientes e corretos. Porém, no caso de uma *SGIM*, há uma dependência circular entre os pixels, de modo que a simples escolha de condições iniciais não é suficiente para a obtenção de um resultado satisfatório. A figura 5.7 mostra um exemplo de um *caminho de filtragem* existente no *sub-filtro H*. Este *caminho de filtragem* abrange doze pixels distintos, cujos quais dependem uns dos outros. Por exemplo, suponha que a filtragem inicie a partir do pixel *A*. Se a imagem fosse comum, e não uma *SGIM*, condições iniciais seriam determinadas para ser possível o início da filtragem, pois nenhum pixel foi filtrado antes do pixel *A*, já que ele é o primeiro. Contudo, como a imagem representada pela figura 5.7 é uma *SGIM*, e portanto é periódica, a filtagem do *A* depende de todos os outros onze pixels, sendo os pixels *L*, *K* e *J* os três pixels com maior influência na filtragem de *A*.

Figura 5.7: Um exemplo de um *caminho de filtragem* existente no *sub-filtro H*.



Fonte: O autor.

Esta análise mostra que a escolha de condições iniciais não é suficiente para a filtragem correta de uma *SGIM*. Ao mesmo tempo, é impossível saber-se de antemão os valores de filtragem dos outros pixels, que ainda não foram filtrados, ao filtrar-se algum pixel da imagem. Por causa deste problema, o filtro *DTFSGIM* possui um passo chamado

de *correção*, que deve ser realizado toda vez que um *caminho de filtragem* for filtrado durante qualquer um dos seus *sub-filtros*. A *correção* será explicada nesta seção e seu objetivo é corrigir todos os valores filtrados naquele *caminho de filtragem* de modo que a dependência circular seja satisfeita.

5.4.1 Derivação do Algoritmo

Dado o filtro recursivo

$$J[n] = (1-r)I[n] + rJ[n-1]\,, \tag{5.9}$$

se o mesmo está definido para $n \in \mathbb{Z}$ em um domínio periódico de período P temos que

$$J[n + kP] = J[n]\,, \tag{5.10}$$

para qualquer n e qualquer k inteiro.

Portanto, podemos descrever este problema matematicamente da seguinte maneira: dada uma sequência $I[0]$, $I[1]$, ..., $I[P-1]$ de período P devemos computar os valores de $J[0], J[1], \ldots, J[P-1]$ de acordo com as eqs. (5.9) and (5.10).

Expandindo a eq. (5.9) acima recursivamente temos

$$J[n] = (1-r)\sum_{i=0}^{n} r^i I[n-i] + r^{n+1}J[-1]\,. \tag{5.11}$$

Podemos, portanto, dizer que:

$$J[n] = G[n] + r^{n+1}J[-1]\,. \tag{5.12}$$

onde:

$$G[n] = (1-r)\sum_{i=0}^{n} r^i I[n-i]\,. \tag{5.13}$$

Fazendo-se $n = P-1$ e notando que, segundo a eq. (5.10), $J[-1] = J[P-1]$, temos que:

$$J[P-1] = G[P-1] + r^P J[P-1]\,, \tag{5.14}$$

e portanto:

$$J[P-1] = \frac{G[P-1]}{(1-r^P)}\,. \tag{5.15}$$

O termo $J[P-1]$ é chamado de *periodic boundary constant*.

Computamos assim um dos valores da sequência solução. Para computar os valores restantes $J[0]$, $J[1]$, ..., $J[P-2]$, utilizamos novamente o fato $J[-1] = J[P-1]$ na eq. (5.12) e logo:

$$J[n] = G[n] + r^{n+1}J[P-1] \tag{5.16}$$

para $n = 0, 1, \ldots, P-2$. Chamaremos o termo $r^{n+1}J[P-1]$ de *fator de correção* do pixel n.

Para computar $G[n]$, o somatório na eq. (5.13) pode ser computado recursivamente notando que $G[0] = (1-r)I[0]$ e

$$G[n] = (1-r)I[n] + rG[n-1] \tag{5.17}$$

para $n = 1, 2, \ldots, P-1$.

Portanto, para ser possível fazer a correção do filtro *DTFSGIM*, é necessário primeiramente computar os valores de $G[0], G[1], \ldots, G[P-1]$, de acordo com a eq. (5.17). Estes valores, contudo, são os resultados já obtidos quando a filtragem é feita sem correção. Em seguida, é necessário computar o valor de $J[P-1]$. Para isto, basta aplicar a eq. (5.15). Por fim, os valores de $J[0], J[1], \ldots, J[P-2]$ são obtidos de acordo com eq. (5.16) e estes são os valores finais de filtragem.

O algoritmo 2 mostra como a correção é efetuada no filtro *DTFSGIM*.

5.5 Filtro DTFSGIM

O filtro *DTFSGIM* reúne todos conceitos apresentados nas seções 5.1 to 5.4. O algoritmo 3 mostra, em alto-nível, o algoritmo completo do filtro *DTFSGIM*. Como entrada, o filtro recebe a *SGIM* que será filtrada. Como saída, o filtro retorna outra *SGIM*, sendo esta agora filtrada.

A ordem dos *sub-filtros H*, *V*, *C* e π deve ser definida ao implementar o filtro *DTFSGIM*. O algoritmo 3 utilizou a ordem *H*, *C*, *V* e π. Esta é a ordem padrão do filtro

Algorithm 2 Correção de um *caminho de filtragem*

Entrada: Buffer A, contendo todos os *adjusted feedback coefficients* $r = a^d$ utilizados durante o *caminho de filtragem* e a *SGIM* X que será corrigida.

Saída: A mesma *SGIM* X, modificada

1: Calcular a *periodic boundary constant*, como descrito na eq. (5.15) e armazenar o resultado em uma variável P.
2: $sum = 1$
3: **for** i=1 **to** i=tamanho(A)-1 **do**
4: $sum = sum * A[i]$
5: Calcular o *fator de correção* c como sendo $c = P * sum$
6: Atualizar o pixel p, cujo fator a^d é dado por $A[i]$, para o valor p', onde: $p' = p + c$ (de acordo com a eq. (5.16)).
7: Se p for um *pixel de borda*, atualizar todos seus *matches*.
8: **end for**
9: Manualmente atualizar o valor do último pixel filtrado p para $p = P$.

DTFSGIM e foi utilizada para gerar todos os resultados que serão mostrados durante este texto. Todavia, a alteração da ordem dos *sub-filtros* pode alterar levemente os resultados durante a execução do filtro. Para um N suficientemente grande, o filtro *DTFSGIM* deve convergir para um mesmo resultado independente da ordem dos seus *sub-filtros*.

5.6 Convergência do Filtro *DTFSGIM*

Como mostrado no algoritmo 3, o filtro *DTFSGIM* possui três entradas:

- σ_s: Controla a extensão espacial da filtragem. Quanto maior este valor, maior a suavização da malha poligonal.
- σ_r: Controla a manutenção das curvaturas importantes da malha durante o processo de filtragem. Quanto menor este valor, maior a preservação das curvaturas.
- N: Número total de iterações do filtro. Quanto maior este valor, maior será a remoção de artefatos visuais na malha resultante. Em contrapartida, maior será o tempo total de execução do filtro.

Para um melhor entendimento da influência dos parâmetros σ_s e σ_r no processo de filtragem, é útil observar o que acontece com a malha poligonal quando estes parâmetros são levados ao extremo:

- $\sigma_s = K$ e $\sigma_r \to \infty$: Neste caso não há nenhuma influência da curvatura da malha poligonal no resultado final, pois o valor de σ_r é muito alto. Deste modo, o filtro *DTFSGIM* passa a se comportar como um filtro de borramento comum, portanto

Algorithm 3 Filtro *DTFSGIM*

Entrada: *SGIM* X, que será filtrada, σ_r, σ_s e N
Saída: *SGIM* Y, filtrada

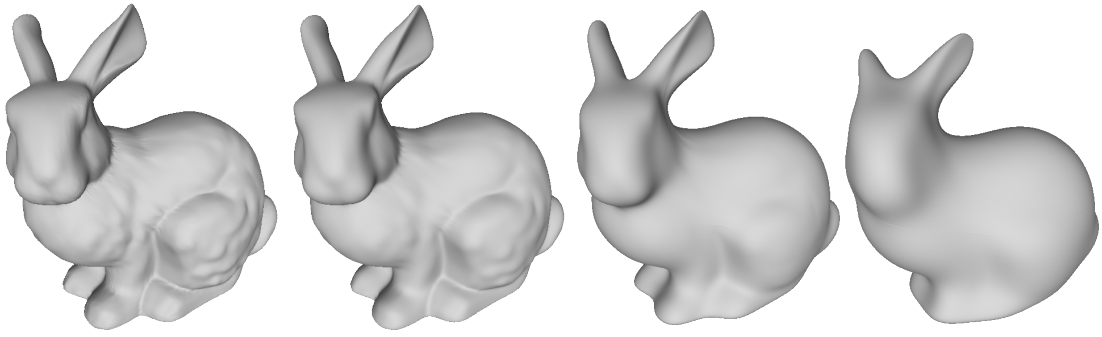
1: Aloca-se uma nova *SGIM*, Y, do tamanho de X. X é copiado para Y.
2: Aloca-se um vetor RF_{BUF} grande o suficiente para armazenar os coeficientes recursivos de todas as iterações do filtro (valor dado pelo parâmetro N).
3: **for** i=1 **to** i=N **do**
4: A partir de σ_s e N, o *feedback coefficient* da iteração i é calculado a partir das eqs. (5.6) and (5.7) e armazenado em $RF_{BUF}[i]$
5: **end for**
6: Aloca-se um *buffer* N_{BUF}, do tamanho da imagem, capaz de armazenar o vetor normal de cada pixel de X. O algoritmo 1 é utilizado para preencher o *buffer* N_{BUF}.
7: O *buffer* N_{BUF} é pré-processado, de acordo com o descrito na subseção 5.2.3.
8: Aloca-se dois vetores H_{BUF} e V_{BUF}, suficientes para o armazenamento de um valor em ponto flutuante para cada pixel de X. Estes dois vetores irão armazenar as *transformadas de domínio* horizontais e verticais de X, respectivamente.
9: Os buffers H_{BUF} e V_{BUF} são preenchidos de acordo com a subseção 5.2.2, utilizando-se os parâmetros σ_s, σ_r, o *buffer* N_{BUF} e a *SGIM* X.
10: **for** i=1 **to** i=N **do**
11: O *sub-filtro H* (seção 5.1) é aplicado em Y, utilizando-se σ_s, N, $RF_{BUF}[i]$ e as *transformadas de domínio* H_{BUF}, como descrito em seção 5.3.
12: Todos os *caminhos de filtragem* do *sub-filtro H* são corrigidos, de acordo com a seção 5.4.
13: O *sub-filtro C* (seção 5.1) é aplicado em Y, utilizando-se σ_s, N, $RF_{BUF}[i]$ e as *transformadas de domínio* H_{BUF} e V_{BUF}, como descrito em seção 5.3.
14: Todos os *caminhos de filtragem* do *sub-filtro C* são corrigidos, de acordo com a seção 5.4.
15: O *sub-filtro V* (seção 5.1) é aplicado em Y, utilizando-se σ_s, N, $RF_{BUF}[i]$ e as *transformadas de domínio* V_{BUF}, como descrito em seção 5.3.
16: Todos os *caminhos de filtragem* do *sub-filtro V* são corrigidos, de acordo com a seção 5.4.
17: O *sub-filtro* π (seção 5.1) é aplicado em Y, utilizando-se σ_s, N, $RF_{BUF}[i]$ e as *transformadas de domínio* H_{BUF} e V_{BUF}, como descrito em seção 5.3.
18: Todos os *caminhos de filtragem* do *sub-filtro* π são corrigidos, de acordo com a seção 5.4.
19: **end for**

suavizando a malha uniformemente de acordo com a constante K.

- $\sigma_s \to \infty$ e $\sigma_r = K$: Neste caso o borramento da malha passa a ser completamente limitado pelo parâmetro σ_r. Regiões com baixa curvatura serão fortemente suavizadas, enquanto que regiões de alta curvatura sofrerão uma suavização limitada por K.
- $\sigma_s \to 0$ e $\sigma_r = K$: Neste caso, como a extensão espacial da filtragem é quase nula, a malha original permanece inalterada, independentemente de K.
- $\sigma_s = K$ e $\sigma_r \to 0$: Neste caso, a tolerância de curvatura é extremamente baixa, de modo que mesmo superfícies de baixa curvatura terão uma *transformada de domínio* muito alta. Portanto, novamente a malha original permanece inalterada, independentemente de K.

A figura 5.8 mostra quatro diferentes resultados obtidos a partir de diferentes processos de filtragem. Em cada um dos resultados o parâmetro σ_r é alterado, ao passo que todos outros parâmetros permanecem constantes. Como esperado, quanto maior o valor do parâmetro σ_r, mais o filtro se aproxima de um filtro de borramento uniforme, sem preservação e curvaturas.

Figura 5.8: Resultados de filtragem com $\sigma_s = 20$, $N = 3$ e o parâmetro σ_r sendo variado. Em ordem: $\sigma_r = 0.1$, $\sigma_r = 0.2$, $\sigma_r = 1.0$, $\sigma_r = \infty$

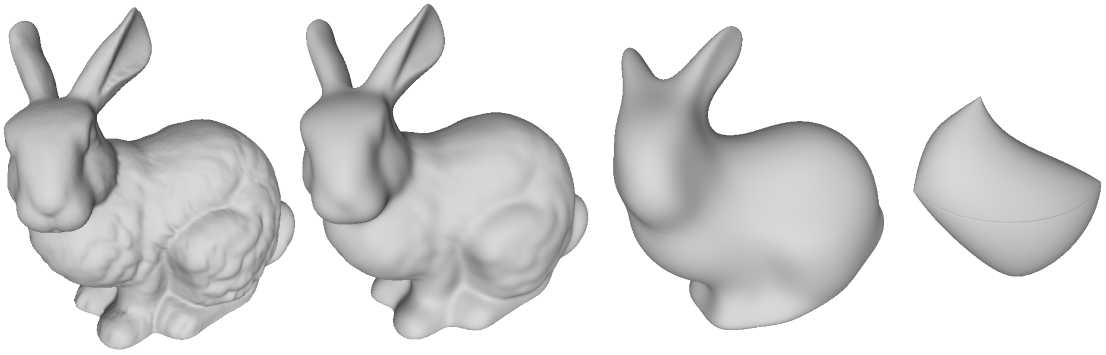


Fonte: O autor.

De modo similar, a figura 5.9 mostra quatro diferentes resultados obtidos variando o parâmetro σ_s e mantendo $\sigma_r = \infty$. Os outros parâmetros foram mantidos constantes. Como esperado, quanto maior o valor do parâmetro σ_s, maior o borramento - sempre sem preservação de curvaturas. Isto ocorre porque $\sigma_r = \infty$, portanto a curvatura da malha é totalmente desconsiderada.

Por fim, a figura 5.10 mostra outros quatro resultados obtidos também variando σ_s, com os exatos mesmos valores. Desta vez, porém, tem-se $\sigma_r = 0.8$ em todas filtragens. Como σ_r está fixado em um valor razoável, percebe-se que a partir de um determinado li-

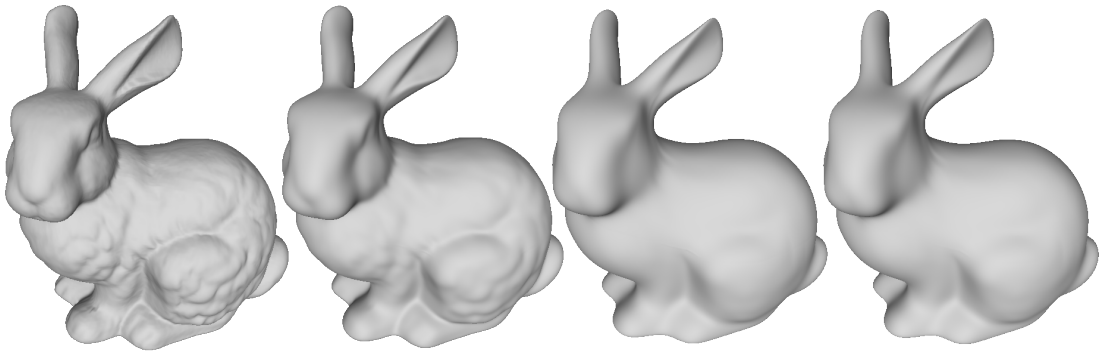
Figura 5.9: Resultados de filtragem com o parâmetro σ_s sendo variado com $\sigma_r = \infty$ e $N = 3$. Em ordem: $\sigma_s = 1$, $\sigma_s = 5$, $\sigma_s = 20$, $\sigma_s = 100$



Fonte: O autor.

mite o aumento de σ_s passa a ter um efeito muito pequeno no resultado, pois as curvaturas estão sendo mantidas.

Figura 5.10: Resultados de filtragem com o parâmetro σ_s sendo variado com $\sigma_r = 0.8$ e $N = 3$. Em ordem: $\sigma_s = 1$, $\sigma_s = 5$, $\sigma_s = 20$, $\sigma_s = 100$



Fonte: O autor.

5.7 Limitações Atuais

Nesta seção serão discutidas duas limitações atuais do filtro *DTFSGIM*.

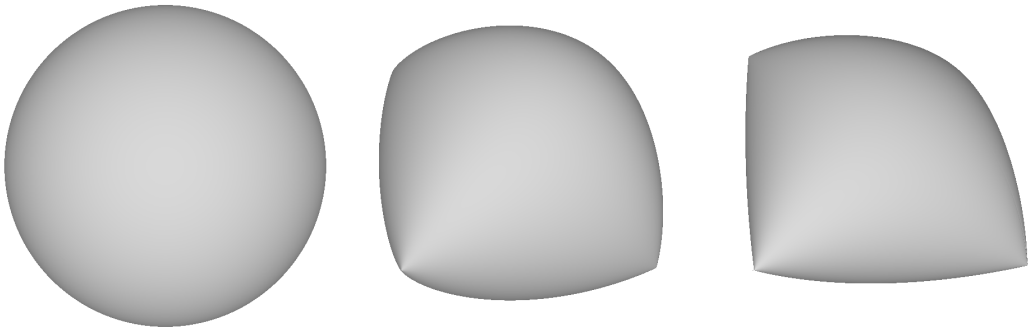
5.7.1 Convergência não-uniforme

Um dos principais problemas do filtro *DTFSGIM* é que, à medida que uma malha poligonal é suavizada, seu formato tende a se aproximar de uma estrutura específica, indesejada. Por exemplo, a figura 5.11 mostra que, quando uma esfera é suavizada utilizando o filtro *DTFSGIM* com valores bastante altos de σ_s, a esfera perde seu formato original e

se aproxima de uma estrutura com quatro pontas. Idealmente, a esfera deveria manter seu formato original, mesmo com uma filtragem intensa.

Figura 5.11: Uma análise da convergência do filtro *DTFSGIM* ao filtrar uma esfera. O parâmetro σ_s é variado, enquanto $\sigma_r \to \infty$ e $N = 3$.

(a) *Malha original* (b) $\sigma_s = 100$ (c) $\sigma_s = 200$



Fonte: O autor.

Este problema decorre do fato da própria natureza da *SGIM*. As quatro pontas, mostradas em maior detalhe na figura 5.11 (c), são definidas pelos quatro *pixels de borda tipo 3*. Todas as malhas poligonais filtradas pelo filtro *DTFSGIM* convergem para uma estrutura similar a esta.

Uma hipótese para o surgimento deste problema é que os *pixels tipo 3* são filtrados um número menor de vezes, conforme descrito na seção 5.1. Ainda estamos estudando a existência de alguma nova decomposição dos *caminhos de filtragem* que consiga corrigir este desbalanceamento.

5.7.2 Qualidade da malha poligonal

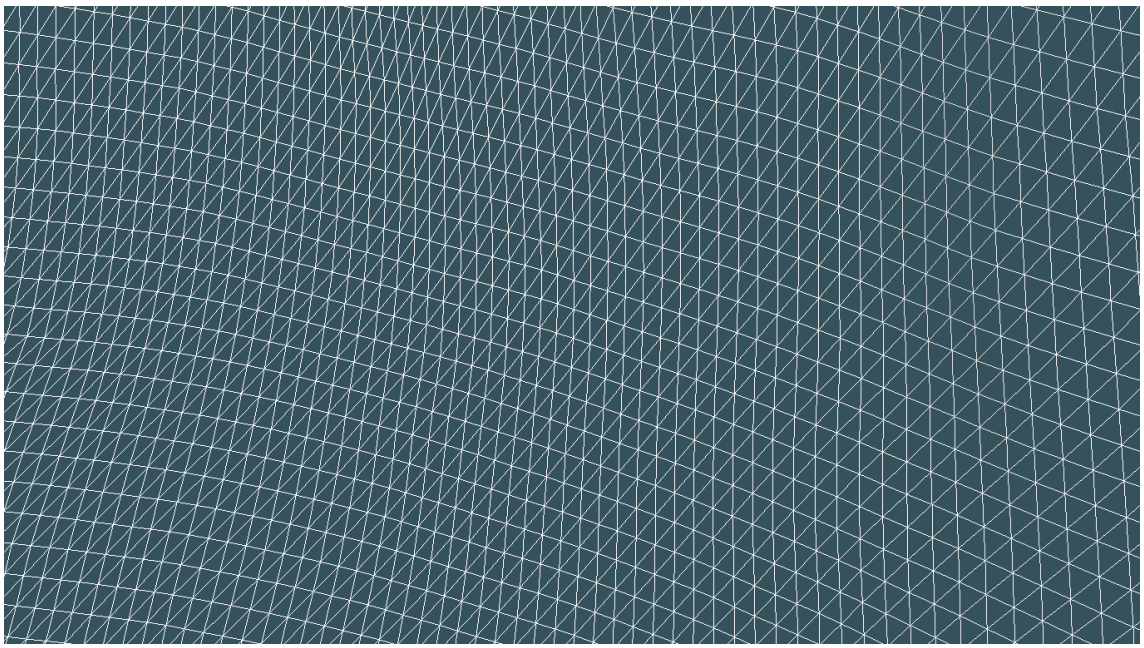
A outra limitação do filtro *DTFSGIM* é a qualidade da malha de triângulos resultante. Apesar de o filtro ser capaz de produzir superfícies interessantes e suaves, nem sempre a malha de triângulos gerada é uniforme e bem distribuída. Em particular, quanto menor o valor do parâmetro σ_r, pior a qualidade da malha de triângulos gerada. Isto ocorre porque, quando o valor de σ_r é baixo, a tolerância para a preservação das curvaturas é pequena, de modo que pequenas variações nas normais da malha resultam em variações bruscas nas *transformadas de domínio* dos pixels envolvidos. Isto faz com que pixels que estão muito próximos sejam filtrados com intensidades demasiadamente diferentes, de modo que os triângulos sejam "desregulados". Além disso, como citado na subseção 5.2.4, um valor pequeno de σ_r resulta em um baixo n_s. Isto causa uma baixa

suavização do *buffer* das normais, o que contribui mais ainda para o problema.

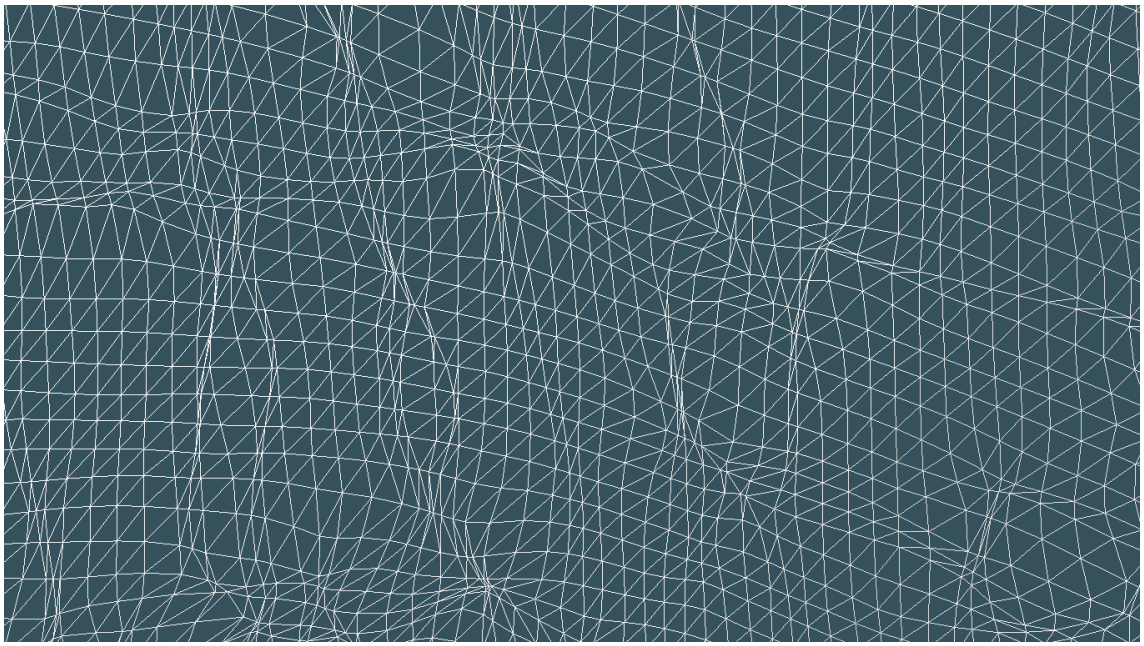
A figura 5.12 mostra um exemplo do problema citado no parágrafo anterior.

Figura 5.12: Uma comparação de como a malha de triângulos de uma malha poligonal muda quando uma filtragem é aplicada com $\sigma_r = 0.1$. Note como na imagem (b) os triângulos estão dispostos de uma maneira desregulada na malha.

(a) *Uma visão de parte da malha de triângulos da malha original.*



(b) *A mesma malha de triângulos mostrada em (a), após a filtragem com o filtro DTFSGIM.*



Fonte: O autor.

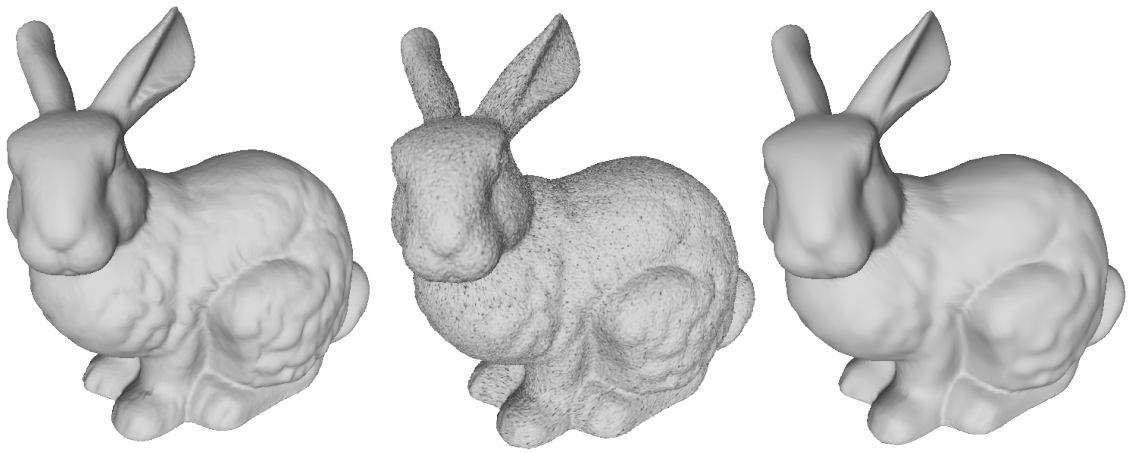
6 RESULTADOS

Para avaliar se o filtro proposto foi capaz de atingir os objetivos propostos na seção 1.3, primeiramente serão mostrados resultados obtidos a partir da aplicação direta do filtro em algumas malhas e com a escolha de um diferente conjunto de parâmetros. Por fim, implementações de trabalhos relevantes citados no capítulo 2 serão utilizadas para ser possível comparar os resultados com trabalhos já existentes.

6.1 Remoção de ruído

A figura 6.1 mostra a aplicação do filtro para a filtragem de uma malha com ruído artificial, com parâmetros $\sigma_s = 20$, $\sigma_r = 0.1$ e $N = 3$. Este ruído artificial foi gerado a partir da soma de um vetor ruído a cada vértice da malha. Este vetor foi criado com sentido e direção aleatórias e tamanho entre 0 e 1. Como pode ser observado, o ruído é removido por completo da superfície, enquanto que as curvaturas importantes da malha original são preservadas quase que em sua totalidade.

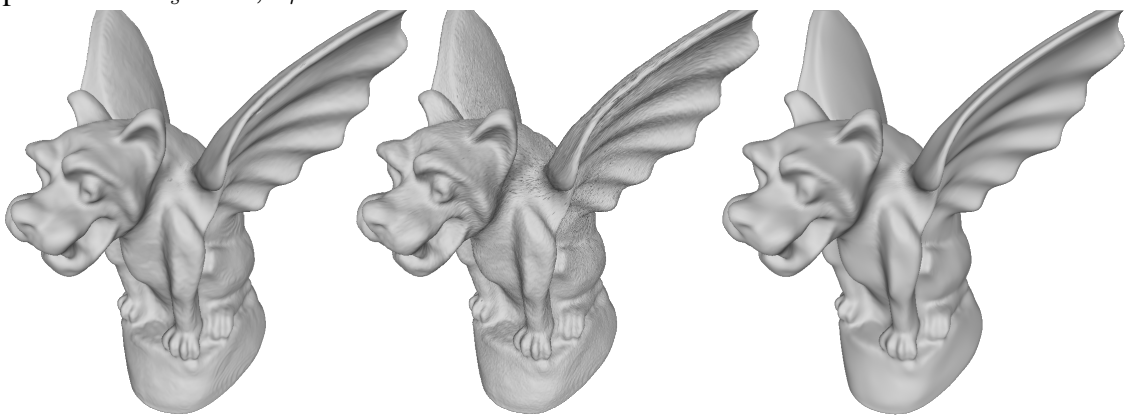
Figura 6.1: Filtragem de uma malha ruidosa. Na esquerda, a malha original. No centro, a malha ruidosa obtida a partir da malha original através da soma de um ruído gaussiano com média zero. Na direita, a malha filtrada com os parâmetros $\sigma_s = 20$, $\sigma_r = 0.1$ e $N = 3$.



Fonte: O autor.

De maneira similar, a figura 6.2 mostra a aplicação do filtro para a filtragem de uma outra malha, também com ruído artificial. Os mesmos parâmetros que foram utilizados na figura 6.1 foram novamente utilizados e mais uma vez as curvaturas foram preservadas quase que em sua totalidade.

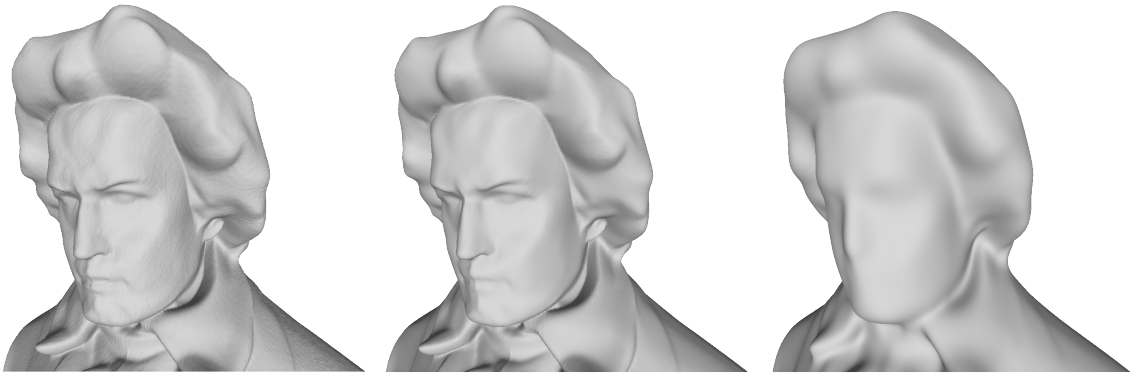
Figura 6.2: Filtragem de uma malha ruidosa. Na esquerda, a malha original. No centro, a malha ruidosa obtida a partir da malha original. Na direita, a malha filtrada com os parâmetros $\sigma_s = 20$, $\sigma_r = 0.1$ e $N = 3$.



Fonte: O autor.

A figura 6.3 mostra a diferença entre filtrar uma malha ruidosa considerando e não considerando informações de curvatura. É notável que a preservação das curvaturas produz resultados muito mais próximos do esperado.

Figura 6.3: Filtragem de uma malha ruidosa. Na esquerda, a malha ruidosa. No centro, a malha ruidosa filtrada pelo filtro *DTFSGIM* com os parâmetros $\sigma_s = 6$, $\sigma_r = 0.065$ e $N = 3$. Na direita, a malha filtrada sem considerar informações de curvatura, ou seja, com $\sigma_r \to \infty$. Note como a filtragem com preservação de curvaturas gera um resultado que mantém mais informações sobre a malha original.



Fonte: O autor.

6.2 Comparações com outros trabalhos

O trabalho *The Bilateral Filter for Point Clouds* (DIGNE; FRANCHIS, 2017) apresenta uma técnica de filtragem com propósitos similares aos deste trabalho. Assim como o filtro proposto, os autores apresentam um filtro capaz de remover ruídos de uma estrutura tridimensional com preservação de curvaturas importantes. Entretanto, o filtro

apresentado é capaz de filtrar apenas nuvens de pontos (*point clouds*), enquanto que o filtro *DTFSGIM* filtra *SGIMs*, que são estruturas obtidas a partir de malhas tridimensionais. Como nuvens de pontos não possuem informações de arestas, naturalmente isto deve ser levado em consideração ao compararmos ambos trabalhos. Neste texto, chamaremos este filtro de *BFPC*.

Para filtrar uma nuvem de pontos arbitrária, quatro parâmetros devem ser especificados:

1. Número de iterações (N)
2. Raio de pesquisa da vizinhança (r)
3. Peso gaussiano para a distância euclidiana (σ_d)
4. Peso gaussiano para a distância ao plano tangente (σ_n)

Para definir estes parâmetros, os autores propõem escolher $\sigma_d = \frac{1}{3}r$ e $\sigma_n = \frac{1}{3}r_n$, onde r_n define uma vizinhança normal a cada ponto. Para a escolha do parâmetro r, os autores fornecem uma maneira de calculá-lo a partir da nuvem de pontos que está sendo filtrada. Deve-se escolher $r = l\sqrt{20/N_{points}}$, onde N_{points} é o número total de pontos na *point cloud* e l é a maior dimensão da *bounding box* da nuvem de pontos. Além disto, σ_n pode ser escolhido com o mesmo valor de σ_d, de modo que $r = r_n$. Por fim, o número de iterações deve ser grande o suficiente para evitar que o filtro apresente artefatos visuais no resultado.

Baseado no parágrafo anterior, os parâmetros foram escolhidos como sugerido pelos autores. O número de iterações (N) foi variado para fins de comparação.

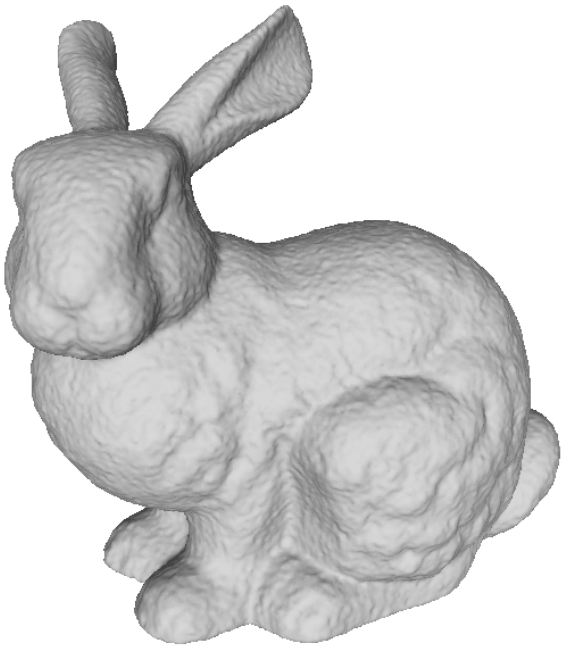
A figura 6.4 mostra uma malha com introdução artificial de ruído.

Para a comparação do filtro *DTFSGIM* com o filtro *BFPC*, a malha mostrada na figura 6.4 foi filtrada por ambos os filtros, variando o número de iterações de cada um, e os resultados foram comparados. A figura 6.5 mostra os resultados obtidos com o uso do filtro *BFPC* e a figura 6.6 mostra os resultados obtidos a partir do filtro *DTFSGIM*.

A figura 6.5 mostra que, para esta malha poligonal específica, é necessário cerca de 10 iterações para se obter bons resultados com o filtro *BFPC*. Por outro lado, a imagem figura 6.6 mostra que, no caso do filtro *DTFSGIM*, uma única iteração do filtro já é suficiente para mostrar bons resultados.

Outro trabalho utilizado foi o *Guided Mesh Normal Filtering* (ZHANG et al., 2015). Neste trabalho, os autores propõem um filtro bilateral para malhas que atua utilizando um "sinal de orientação" ao invés do sinal original. O filtro se propõe a construir

Figura 6.4: Uma malha ruidosa que possui cerca de 260000 vértices.



Fonte: O autor.

este "sinal de orientação" e é composto em duas etapas: na primeira, o filtro bilateral é aplicado às normais da malha, utilizando como auxílio um campo de normais gerado durante o processo de filtragem. Na segunda etapa, as posições dos vértices são atualizadas de acordo com as normais filtradas na etapa anterior. Neste texto, este filtro será chamado de filtro *GMNF*.

O filtro *GMNF* exige uma série de parâmetros na sua execução. Eles são:

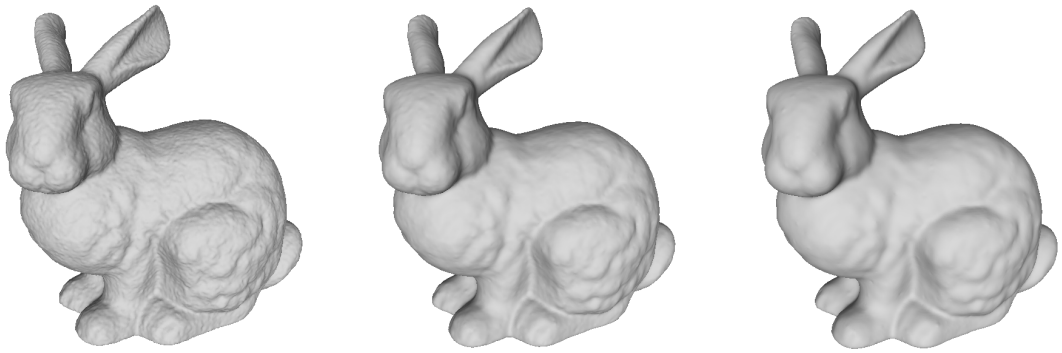
1. Número de iterações ao filtrar as normais (k_{iter})
2. Número de iterações para atualizar os vértices (v_{iter})
3. Raio para encontrar uma vizinhança geométrica (r)
4. Parâmetro de variância do *spatial kernel* (σ_s)
5. Parâmetro de variância do *range kernel* (σ_r)

Nos testes que serão apresentados, os parãmetros σ_r e σ_s foram fixados em $\sigma_r = 0.35$ e $\sigma_s = 1$. Estes valores mostraram-se suficientes para a geração de bons resultados. O parâmetro r também foi fixado em $r = 2$. Os valores de k_{iter} e v_{iter} foram variados.

A figura 6.7 mostra três resultados obtidos variando-se os parâmetros k_{iter} e v_{iter}. É notável que a primeira filtragem, ainda com valores baixos de k_{iter} e v_{iter}, não é suficiente para remover o ruído da malha. A segunda filtragem remove parcialmente o ruído, enquanto que a última produz um ótimo resultado.

Figura 6.5: Três resultados obtidos com a aplicação do filtro *BFPC* na malha poligonal mostrada na figura 6.4. Os parâmetros r, σ_d e σ_n foram escolhidos seguindo o sugerido pelos autores para esta malha específica.

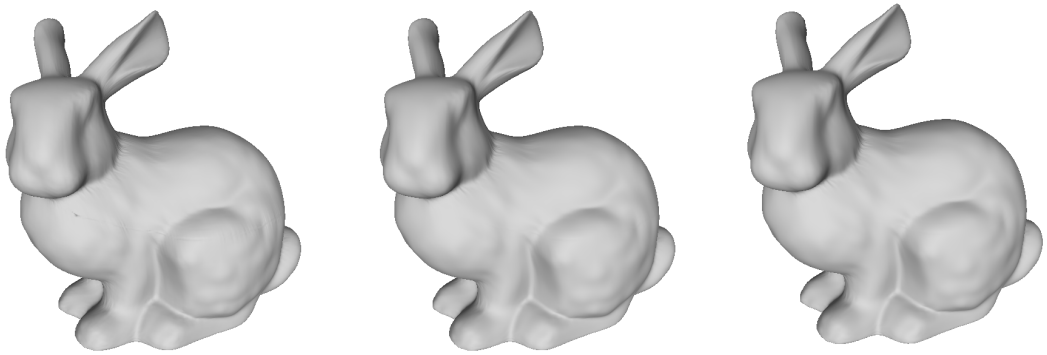
(a) *Resultado de filtragem para* $N = 1$. (b) *Resultado de filtragem para* $N = 5$. (c) *Resultado de filtragem para* $N = 10$.



Fonte: O autor.

Figura 6.6: Três resultados obtidos com a aplicação do filtro *DTFSGIM* na malha poligonal mostrada na figura 6.4.

(a) *Resultado de filtragem para* $N = 1$. (b) *Resultado de filtragem para* $N = 2$. (c) *Resultado de filtragem para* $N = 3$.



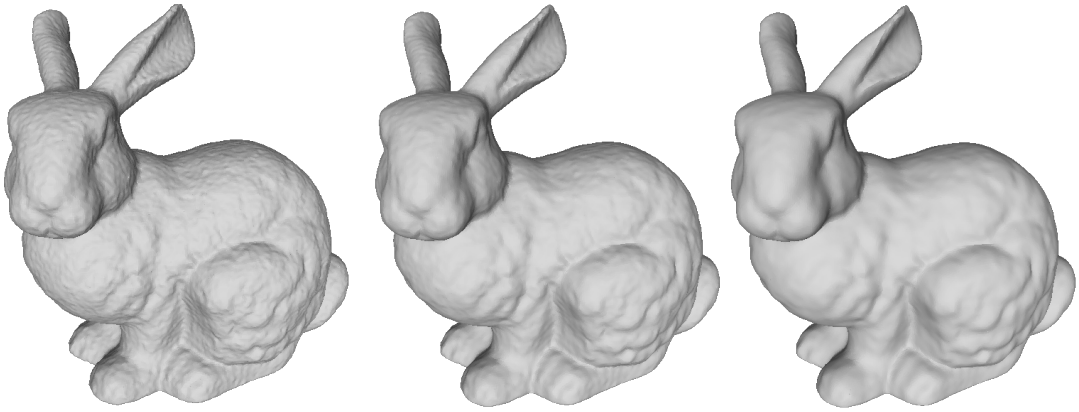
Fonte: O autor.

A tabela 6.1 mostra os tempos de filtragem obtidos para gerar as malhas mostradas nas figuras 6.5 até 6.7.

Os resultados obtidos e mostrados na tabela 6.1 foram obtidos em um processador *Intel Core i7-8705G*, de 3.10GHz. Os filtros *BFPC* e *GMNF* utilizados foram compilados utilizando os códigos fornecidos pelos autores. Os tempos foram obtidos sem considerar leituras do disco. Alocações de memória, contudo, foram consideradas.

Nenhum dos filtros foi executado em paralelo, apesar de o filtro *BFPC* possuir esta opção. Apesar de a paralelização do filtro *DTFSGIM* não ter sido explorada neste trabalho, por se tratar de um filtro recursivo, é possível a criação de uma versão paralela do filtro com o auxílio de trabalhos como o artigo *GPU-efficient recursive filtering and summed-area tables* (NEHAB et al., 2011). Neste trabalho, são explicadas técnicas para

Figura 6.7: Três resultados obtidos com a aplicação do filtro *GMNF* na malha poligonal mostrada na figura 6.4. A imagem da esquerda mostra o resultado obtido para $k_{iter} = 5$ e $v_{iter} = 2$. A imagem do centro mostra o resultado obtido para $k_{iter} = 10$ e $v_{iter} = 5$. Por fim, a imagem da direita mostra o resultado obtido para $k_{iter} = 20$ e $v_{iter} = 10$. Os outros parâmetros foram mantidos em $r = 2$, $\sigma_r = 0.35$ e $\sigma_s = 1$



Fonte: O autor.

Tabela 6.1: Tempos de filtragem da malha mostrada na figura 6.4 pelos filtros *BFPC*, *DTFSGIM* e *GMNF*. No caso do filtro *BFPC*, os valores de N considerados são 1, 5 e 10. Os valores de N considerados para o filtro *DTFSGIM* são 1, 2 e 3. Os valores de k_{iter} e v_{iter} considerados para o filtro *GMNF* são, respectivamente, 5 e 2, 10 e 5, 20 e 10.

	BFPC	DTFSGIM	GMNF
Rápido	1683ms	915ms	15316ms
Médio	8457ms	1015ms	27457ms
Lento	17034ms	1067ms	49808ms

paralelizar filtros recursivos genéricos.

Os tempos mostrados para o filtro *DTFSGIM* foram obtidos sem considerar o tempo de converter a malha para uma *SGIM*. De acordo com o artigo *Spherical Parametrization and Remeshing* (PRAUN; HOPPE, 2003), esta parametrização demora entre 7 e 25 minutos para modelos que possuem entre 25000 e 200000 faces em um processador *Pentium4* de 3GHz. Portanto, é notável que o filtro *DTFSGIM* mostra um desempenho muito superior aos outros apenas quando a *SGIM* já foi gerada anteriormente. Em contrapartida, o *overhead* da geração da *SGIM* deve ocorrer apenas uma vez; após a sua criação, diferentes filtragens podem ser feitas na *SGIM* com diferentes parâmetros sem a necessidade de sua recriação.

Analisando a tabela 6.1, é visível que o filtro *DTFSGIM* é capaz de gerar bons resultados em um tempo inferior ao dos filtros *BFPC* e *GMNF*. Em contrapartida, para um valor suficientemente grande de N, o filtro *BFPC* se mostra mais capaz de preservar as curvaturas importantes da malha. Isto é visível nas figuras 6.5 and 6.6: o filtro *DTFSGIM* deixa de preservar curvaturas muito sutis, enquanto que o filtro *BFPC* consegue preservá-

las com maior precisão. O mesmo pode ser dito do filtro *GMNF*.

7 CONCLUSÃO

Com o avanço da computação gráfica e do escaneamento 3D, filtros capazes de remover ruídos de objetos tridimensionais sem adulterar a forma do objeto original tornam-se cada vez mais importantes. O filtro *DTFSGIM*, proposto neste trabalho, visa resolver este problema e é mais uma contribuição acadêmica neste sentido. Os resultados apresentados no capítulo 6 mostram que o filtro *DTFSGIM* mostrou-se capaz de suavizar uma malha poligonal genérica mantendo suas curvaturas e com uma eficiência bastante satisfatória. Cumprindo, portanto, os objetivos citados na seção 1.3.

7.1 Futuros Trabalhos

Apesar do filtro *DTFSGIM* já ser capaz de produzir resultados bastante interessantes, os problemas citados na seção 5.7 ainda podem ser estudados e corrigidos para que o filtro seja ainda melhor. O problema citado na subseção 5.7.1 pode ser corrigido para que o filtro possua uma convergência uniforme. Já o problema citado na subseção 5.7.2 deve ser estudado em detalhes para que a superfície gerada pelo filtro *DTFSGIM* seja composta por uma malha de triângulos de melhor qualidade. Dentre as possíveis soluções propostas para este problema, tem-se:

- Após o fim do filtro *DTFSGIM*, é possível a adição de um passo extra, cujo objetivo seja regerar a malha de triângulos, sem alterar a superfície obtida.
- Durante a execução dos *sub-filtros*, pode-se adicionar passos intermediários que têm como objetivo corrigir a malha de triângulos dinamicamente, a medida que o filtro vai sendo executado.

Por fim, o filtro *DTFSGIM* pode ser implementado em *softwares* de código aberto e submetido para aprovação. Um destes *softwares* é o *MeshLab*, que possui diversos filtros capazes de modificar malhas poligonais.

REFERÊNCIAS

ADAMS, A. et al. Gaussian kd-trees for fast high-dimensional filtering. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 28, n. 3, p. 21:1–21:12, jul. 2009. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/1531326.1531327>.

DIGNE, J.; FRANCHIS, C. de. The bilateral filter for point clouds. **Image Processing On Line**, v. 7, p. 278–287, 2017.

GASTAL, E. S. L.; OLIVEIRA, M. M. Domain transform for edge-aware image and video processing. **ACM TOG**, v. 30, n. 4, p. 69:1–69:12, 2011. Proceedings of SIGGRAPH 2011.

GOURAUD, H. Continuous shading of curved surfaces. **IEEE Trans. Comput.**, IEEE Computer Society, Washington, DC, USA, v. 20, n. 6, p. 623–629, jun. 1971. ISSN 0018-9340. Available from Internet: <http://dx.doi.org/10.1109/T-C.1971.223313>.

GU, X.; GORTLER, S. J.; HOPPE, H. Geometry images. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 21, n. 3, p. 355–361, jul. 2002. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/566654.566589>.

HO-LE, K. Finite element mesh generation methods: A review and classification. **Comput. Aided Des.**, Butterworth-Heinemann, Newton, MA, USA, v. 20, n. 1, p. 27–38, feb. 1988. ISSN 0010-4485. Available from Internet: <http://dx.doi.org/10.1016/0010-4485(88)90138-8>.

LOSASSO, F. et al. Smooth geometry images. In: **Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing**. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003. (SGP '03), p. 138–145. ISBN 1-58113-687-0. Available from Internet: <http://dl.acm.org/citation.cfm?id=882370.882389>.

NEHAB, D. et al. Gpu-efficient recursive filtering and summed-area tables. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 30, n. 6, p. 176:1–176:12, dec. 2011. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2070781.2024210>.

PRAUN, E.; HOPPE, H. Spherical parametrization and remeshing. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 22, n. 3, p. 340–349, jul. 2003. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/882262.882274>.

ZHANG, W. et al. Guided mesh normal filtering. **Comput. Graph. Forum**, The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, v. 34, n. 7, p. 23–34, oct. 2015. ISSN 0167-7055. Available from Internet: <http://dx.doi.org/10.1111/cgf.12742>.

ZORIN, D.; SCHRöDER, P. **Subdivision for Modeling and Animation**. [S.l.: s.n.], 2001.