FUNDAÇÃO GETÚLIO VARGAS
ESCOLA DE MATEMÁTICA APLICADA

DANIEL JACOB TONN

UM ESTUDO SOBRE FATIAMENTO E RECONSTRUÇÃO DE MODELOS 3D

Rio de Janeiro
2024

DANIEL JACOB TONN

UM ESTUDO SOBRE FATIAMENTO E RECONSTRUÇÃO DE MODELOS 3D

Trabalho de conclusão de curso apresentada para a Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Matemática Aplicada.

Área de estudo: Geometria Computacional.

Orientador: Paulo Cezar Pinto Carvalho.

Rio de Janeiro

2024

Ficha catalográfica elaborada pela BMHS/FGV

Jacob Tonn, Daniel
Um estudo sobre fatiamento e reconstrução de modelos 3D: /Daniel Jacob Tonn. – 2024.
24f.

Trabalho de Conclusão de Curso – Escola de Matemática Aplicada.

Advisor: Paulo Cezar Pinto Carvalho.
Includes bibliography.

1. Matemática 2. Aplicada 2. na matemática I. Cezar Pinto Carvalho, Paulo II. Escola de Matemática Aplicada III. Um estudo sobre fatiamento e reconstrução de modelos 3D

DANIEL JACOB TONN

UM ESTUDO SOBRE FATIAMENTO E RECONSTRUÇÃO DE MODELOS 3D

Trabalho de conclusão de curso apresentada para a Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Matemática Aplicada.

Área de estudo: Geometria Computacional.

E aprovado em 03/12/2024
Pelo próprio orientador

Paulo Cezar Pinto Carvalho.
Escola de Matemática Aplicada

À minha irmã Estela, minha *lumos* .

Agradecimentos

Agradeço ao Centro para o Desenvolvimento da Matemática e Ciências (CDMC) pelo apoio essencial prestado ao longo da minha graduação, que tornou possível a conclusão deste trabalho e a conquista do título de Bacharel.

Aos professores que contribuíram para minha formação, em especial à professora Asla Medeiros e Sá, cuja presença vai além das fórmulas e cálculos, inspirando-nos com sua dedicação e trazendo cor ao rigor das ciências exatas.

Agradeço também aos amigos de sala e de moradia que compartilharam comigo essa jornada, dividindo não apenas o aprendizado, mas também os desafios e as conquistas que nos fortaleceram ao longo do curso.

À minha família, por seu apoio constante e incondicional. E, finalmente, à minha irmã Estela, que sempre ilumina meu caminho e me lembra do que realmente importa.

"Não é fácil porque é simples.
É fácil porque você domina."
Wendell Oliveira

Resumo

Este trabalho propõe um método para fatiamento de objetos tridimensionais e extração de nuvens de pontos a partir de uma coleção de imagens paralelas. O processo envolve a captura de imagens sequenciais de fatias do objeto, gerando cortes transversais paralelos que representam a estrutura interna e externa do modelo tridimensional. A partir dessas imagens, são extraídas nuvens de pontos que servem como base para a reconstrução da superfície do objeto.

A implementação do método utiliza scripts em Python que integram ferramentas como Blender e MeshLab, automatizando o fluxo de trabalho entre a captura das seções e a extração das nuvens de pontos. Essa integração permite um processo eficiente e contínuo para a reconstrução de superfícies tridimensionais a partir de imagens paralelas.

Palavras-chave: nuvem de pontos, fatiamento tridimensional, Blender, MeshLab, Python.

Abstract

This work proposes a method for slicing three-dimensional objects and extracting point clouds from a collection of parallel images. The process involves capturing sequential images of object slices, generating parallel cross-sections that represent the internal and external structure of the 3D model. From these images, point clouds are extracted to serve as the basis for reconstructing the object's surface.

The method's implementation utilizes Python scripts that integrate tools such as Blender and MeshLab, automating the workflow between section capture and point cloud extraction. This integration enables an efficient and continuous process for the reconstruction of 3D surfaces from parallel images.

Keywords: point cloud, 3D slicing, Blender, MeshLab, Python.

Lista de ilustrações

Sumário

1 Introdução

A reconstrução de objetos tridimensionais a partir de seções bidimensionais paralelas é um problema clássico na área de processamento de imagens e visão computacional. Essa abordagem permite a representação fiel de estruturas intricadas, especialmente aquelas com partes internas que são difíceis de detectar por outros métodos. A técnica de fatiamento destaca-se particularmente em cenários onde o objeto não é completamente denso, ou seja, contém partes ocas ou cavidades internas que não podem ser capturadas por métodos convencionais de escaneamento ou modelagem superficial. Contudo, a qualidade e precisão da reconstrução dependem diretamente da adequação das seções utilizadas.

A aplicação dessa técnica é evidente em áreas como a medicina, onde as tomografias geram seções paralelas do corpo humano. A partir dessas seções, é possível reconstruir novamente estruturas internas, como o cérebro, em modelos tridimensionais. Ferramentas como o 3D Slicer, por exemplo, realizam reconstruções de estruturas complexas a partir de dados de imagem médica, como tomografias e ressonâncias magnéticas, demonstrando a aplicabilidade e a relevância desse tipo de técnica em diferentes domínios.

Este trabalho foca no processo de fatiamento e extração da nuvem de pontos a partir de uma coleção de imagens planas, utilizando objetos de menor complexidade para desenvolvimento e testes. Inicialmente, uma base de dados é criada de forma automatizada, integrando Python e Blender através da biblioteca Blender Python API (bpy), permitindo a manipulação de objetos 3D no Blender por meio de scripts Python.

Com as imagens obtidas, aplicamos o filtro de Sobel para isolar o objeto de interesse em cada imagem. Em seguida, construímos uma nuvem de pontos que representa a estrutura tridimensional do objeto.

Este estudo visa descrever o processo de fatiamento de objetos simples, contribuindo para o avanço das técnicas de processamento de imagens e reconstrução 3D.

2 Revisão de Literatura

2.1 Estrutura de imagens computacionais

Um ponto bidimensional é expresso na forma de par ordenado (x_i, y_i). No caso de imagens, x e y representam a posição de cada pixel.

Uma imagem digital em tons de cinza pode ser representada por uma matriz I de dimensões $M \times N$, onde cada elemento $I(x_i, y_j)$ da matriz corresponde ao valor de intensidade do pixel localizado na posição (x_i, y_j) na imagem. Esse valor é geralmente um número inteiro que representa a intensidade luminosa do pixel, com valores variando de 0 (preto) a 255 (branco) para imagens de 8 bits por pixel (GONZALEZ; WOODS, 2008).

Para imagens coloridas, cada pixel é representado por um vetor tridimensional (R, G, B), onde R, G, e B são os valores de intensidade para os canais de vermelho, verde e azul, respectivamente (GONZALEZ; WOODS, 2008).

2.1.1 Formato de Imagens PNG

O formato de imagem chamado Portable Network Graphics (PNG) é valorizado por sua compressão sem perda, o que permite preservar todos os detalhes da imagem original sem qualquer degradação na qualidade. Isso é especialmente útil em contextos onde a precisão visual e a fidelidade dos detalhes são cruciais, como na reconstrução de objetos tridimensionais. Diferente de formatos como JPEG, que comprometem a nitidez para reduzir o tamanho do arquivo, o PNG mantém a integridade de cada pixel, garantindo que os detalhes essenciais sejam preservados para análises ou manipulações subsequentes (BOUTELL, 1997).

2.1.2 Conversão de Imagem para Monocromia cinza

Um método mais sofisticado e que leva em conta a percepção humana das cores é o uso de uma média ponderada, onde os canais são combinados com pesos diferentes, refletindo a sensibilidade do olho humano a diferentes cores. A fórmula mais comum, baseada na recomendação da ITU-R BT.601, é:

$$I_{\text{cinza}} = 0{,}2989 \times R + 0{,}5870 \times G + 0{,}1140 \times B.$$

Nesse método, o canal verde recebe o maior peso, seguido pelo vermelho, enquanto o azul tem o menor peso, refletindo a menor sensibilidade humana a esta cor (INTERNATIONAL

TELECOMMUNICATION UNION, 1995). Isso resulta em uma conversão que geralmente preserva mais detalhes visuais, especialmente em áreas onde o verde é dominante.

2.1.3 O Filtro de Sobel

O filtro de Sobel é uma técnica utilizada no processamento de imagens para a detecção de bordas. Ele é baseado no cálculo de gradientes de intensidade em uma imagem, que são usados para identificar regiões onde há mudanças abruptas na intensidade de cor ou luminância, típicas das bordas dos objetos (GONZALEZ; WOODS, 2008).

O gradiente de uma função $I(x, y)$, que representa a intensidade da imagem em cada ponto (x, y), é um vetor que aponta na direção de maior variação da função. Matematicamente, o gradiente é definido como:

$$\nabla I(x, y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right).$$

Aqui, $\frac{\partial I}{\partial x}$ e $\frac{\partial I}{\partial y}$ são as derivadas parciais de $I(x, y)$ em relação às direções x e y, respectivamente (SZELISKI, 2011).

Como as imagens digitais são representadas por matrizes discretas, as derivadas parciais devem ser aproximadas. O filtro de Sobel utiliza máscaras de convolução (kernels) para realizar essa aproximação. Essas máscaras são aplicadas em cada ponto da imagem para calcular as derivadas parciais aproximadas (SOBEL; FELDMAN, 1968).

As máscaras de Sobel para as direções x e y são:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}.$$

A aplicação dessas máscaras sobre a imagem $I(x, y)$ se dá por meio de uma operação de convolução. Se considerarmos uma janela 3×3 centrada no ponto (x_i, y_i) da imagem, as derivadas parciais são aproximadas por:

$$\frac{\partial I}{\partial x} \approx G_x * I(x_i, y_i), \quad \frac{\partial I}{\partial y} \approx G_y * I(x_i, y_i),$$

onde $*$ denota a operação de convolução entre a máscara G_x (ou G_y) e a submatriz de I centrada em (x_i, y_i) (GONZALEZ; WOODS, 2008).

Uma vez que as derivadas parciais foram calculadas, a magnitude do gradiente $|G(x_i, y_i)|$ e a direção $\theta(x_i, y_i)$ podem ser determinadas da seguinte forma:

- Magnitude do Gradiente:

$$|G(x_i, y_i)| = \sqrt{\left(\frac{\partial I}{\partial x} \right)^2 + \left(\frac{\partial I}{\partial y} \right)^2}.$$

Esta magnitude indica a força da borda no ponto (x_i, y_i). Quanto maior o valor, mais pronunciada é a borda (GONZALEZ; WOODS, 2008).

- Direção do Gradiente:

$$\theta(x_i, y_i) = \arctan\left(\frac{\partial I}{\partial y} \Big/ \frac{\partial I}{\partial x}\right).$$

A direção do gradiente indica a orientação da borda em relação ao eixo x. Esse ângulo é frequentemente quantizado em categorias, como 0º, 45º, 90º, e 135º, para facilitar o processamento subsequente (SZELISKI, 2011).

Na aplicação do filtro de Sobel em uma imagem real, cada pixel da imagem é substituído pela magnitude do gradiente calculado, o que resulta em uma nova imagem onde as bordas são destacadas em branco e os demais pontos são transformados em pontos pretos (SOBEL; FELDMAN, 1968).

2.2 Representação computacional de objetos tridimensionais

Em teoria, uma superfície é uma entidade contínua, descrita matematicamente por uma função contínua $S(u, v)$, onde u e v são parâmetros bidimensionais que percorrem a superfície. No entanto, a continuidade de $S(u, v)$ não pode ser diretamente representada em um sistema computacional, que só pode lidar com dados discretos (BOTSCH et al., 2010).

A fim de representar uma superfície contínua em um ambiente digital, precisamos discretizar essa superfície, o que significa substituí-la por uma coleção finita de elementos que possam ser manipulados e armazenados. A discretização é o processo de amostragem da superfície contínua em pontos finitos, resultando em uma aproximação da superfície real. Matemática e computacionalmente, essa discretização pode ser feita de diversas maneiras, dentre estas, as descritas no livro *Polygon Mesh Processing* (BOTSCH et al., 2010):

- **Discretização por Malhas**: Uma malha é uma estrutura que aproxima a superfície contínua através de um conjunto de polígonos (geralmente os mais simples, como triângulos ou quadriláteros). Esses polígonos são formados por vértices, que são pontos discretos no espaço 3D. Dado um conjunto de pontos amostrados da superfície, a malha conecta esses pontos em uma rede de arestas e faces, criando uma aproximação discreta da superfície original.

- **Discretização por Nuvens de Pontos**: Em vez de conectar os pontos discretos em uma estrutura de malha, uma nuvem de pontos representa a superfície como um conjunto não estruturado de pontos no espaço 3D. Cada ponto $P_i = (x_i, y_i, z_i)$ é uma amostra da superfície. A nuvem de pontos não contém informações explícitas sobre

como os pontos são conectados, mas oferece uma base para reconstruir a superfície utilizando técnicas de interpolação ou aproximação, como triangulação de Delaunay ou superfícies implícitas.

Em suma, ao discretizar o objeto por meio das imagens, a coleção de pontos advinda do posicionamento dos pixels é a informação mais completa disponível. Essa estruturação oferece uma base sólida para a reconstrução tridimensional precisa e detalhada do objeto, crucial para análises e aplicações (BOTSCH et al., 2010).

3 Desenvolvimento

3.1 Repositório de Código

Como parte do desenvolvimento deste trabalho, foi criado um repositório público no GitHub para armazenar o código desenvolvido e os recursos computacionais utilizados durante as etapas de fatiamento e reconstrução de objetos tridimensionais. Esse repositório, intitulado *Slicing and 3D Model Reconstruction*, está disponível em `https://github.com/SeuUsuario/NomeDoRepositorio` e possui a seguinte estrutura:

- `automatic_slicer.ipynb`: Contém os sripts responsáveis pelos processos de fatiamento e extração da nuvem de pontos.
- `blender_arquives/`: Diretório para armazenar objetos 3D do blender.
- `planes_intersections/`: Onde estão as pastas que contém as seções fatiadas dos objetos.
- `meshlab_arquives/`:Arquivos .ply contendo nuvem de pontos.

Demais requisitos e comentários estão listadas no próprio repositório.

3.2 Fatiamento

O primeiro passo é a criação de uma base de dados de imagens bidimensionais, utilizando o Blender (BLENDER ONLINE COMMUNITY, 2018), um software de código aberto amplamente utilizado para modelagem 3D, animação e renderização. Esse processo é realizado de forma automatizada, integrando o Python e o Blender por meio da biblioteca Blender Python API (bpy), que permite um controle programático sobre as funcionalidades do Blender.

Esses scripts permitem a criação de seções bidimensionais paralelas ao objeto 3D, que são essenciais para a posterior reconstrução tridimensional. Através do bpy, temos controle preciso sobre a posição e orientação dos planos de corte, garantindo que as imagens resultantes sejam adequadas e de alta qualidade para a reconstrução tridimensional.

Especificamente, em cada seção paralela, são gerados dois planos de corte paralelos ao plano XY do Blender. O primeiro plano divide o objeto em partes superior e inferior, omitindo a parte superior, enquanto o segundo plano omite a parte inferior do corpo, a uma distância suficientemente próxima do plano anterior. Com a câmera posicionada

estrategicamente, é capturada apenas uma fatia visível do objeto, que é então adicionada à coleção de imagens das seções.

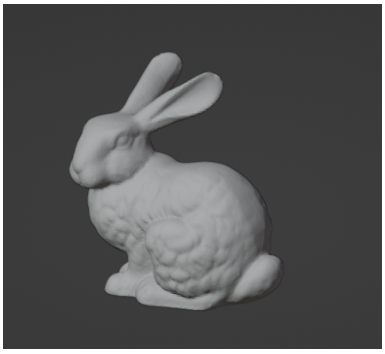


Figura 1 – Stanford Bunny.

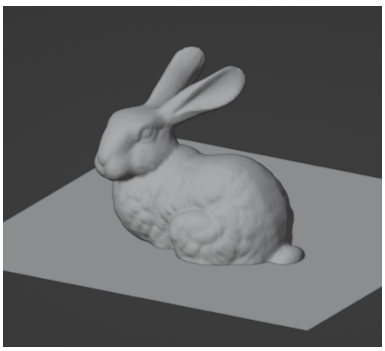


Figura 2 – Plano fatiador.

3.3 Criação de bordas com o filtro de Sobel e extração da nuvem de pontos

Diferentemente das imagens médicas, como as de tomografia, que muitas vezes apresentam estruturas complexas e sobrepostas (como o cérebro, vasos sanguíneos e crânio) que não são claramente distinguíveis, optamos por começar com objetos mais simples. As imagens médicas contêm muito ruído e suas fronteiras não são bem definidas, o que torna o processo de reconstrução mais desafiador. Ao trabalhar inicialmente com imagens de objetos simples, conseguimos testar e validar os métodos de detecção de bordas e extração de nuvens de pontos de maneira controlada e precisa, evitando as complicações adicionais encontradas nas imagens médicas.

Com a coleção de imagens bidimensionais em mãos, a próxima etapa envolve a aplicação do filtro de Sobel para a detecção de bordas. O filtro de Sobel calcula a derivada da intensidade da imagem em duas direções (horizontal e vertical), produzindo uma imagem de gradiente que destaca as bordas onde ocorrem mudanças abruptas de intensidade.

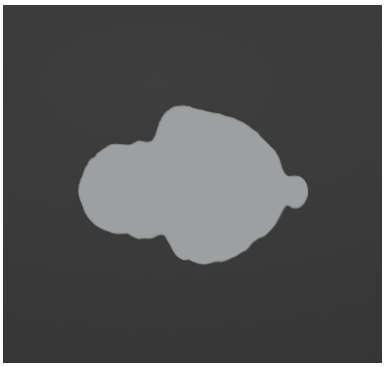


Figura 3 – Intersecção entre o plano e o objeto da Figura 2.

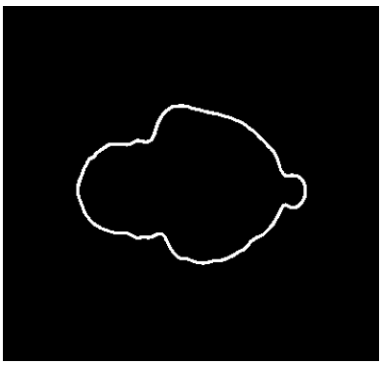


Figura 4 – Borda da seção do objeto da Figura 3.

A partir das imagens com as bordas destacadas, construímos uma nuvem de pontos que representa a estrutura tridimensional do objeto. Em resumo, desejamos uma aplicação

$$\phi : P(x_i, y_i) \to P(x_i, y_i, z_i).$$

Aqui, o ordenamento das imagens do objeto é essencial. Afinal, tomando os pontos da imagem correspondente a uma das extremidades do objeto como altura $z_i = 0$ temos, para a i-ésima imagem a altura $z_i = \epsilon i$, onde ϵ denota o espaçamento entre as fatias. É importante destacar que o espaçamento $\epsilon \in \mathbb{N}$ afinal, a distância entre dois pontos originários da imagem é no mínimo 1, pois trata-se de quantidade de pixels.

Cada ponto na nuvem é derivado das coordenadas das bordas detectadas nas imagens bidimensionais, associadas às suas respectivas posições no espaço tridimensional. A precisão na detecção de bordas é crucial nesta etapa, pois determina a qualidade da nuvem de pontos gerada.

Para facilitar a visualização e a manipulação da nuvem de pontos gerada, os pontos são salvos em um arquivo no formato .ply, suportado por diversas ferramentas de visualização 3D. Embora o Blender tenha funcionalidades experimentais para visualizar nuvens de pontos, optamos por utilizar o MeshLab — uma ferramenta de código aberto especializada na manipulação e visualização de nuvens de pontos — devido à sua estabilidade e recursos avançados. O uso do MeshLab garante uma visualização precisa e confiável da estrutura 3D, facilitando a análise e o refinamento da nuvem de pontos obtida.

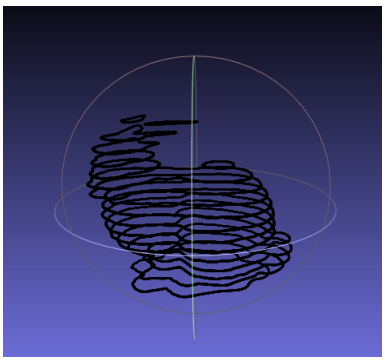


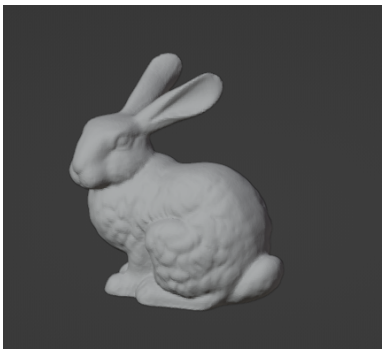
Figura 5 – Nuvem de pontos gerada do objeto da Figura 1.

4 Resultados

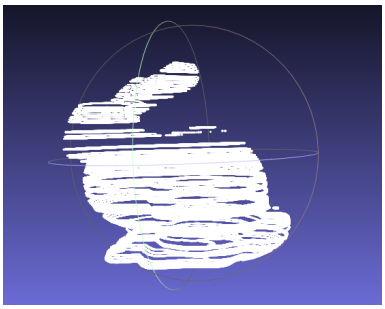
4.1 Comparativo visual entre o objeto e a nuvem de pontos extraída

Nesta seção, apresentamos uma análise comparativa entre o objeto original e a nuvem de pontos extraída das seções paralelas do objeto para quatro objetos distintos, com o objetivo de avaliar o impacto da aplicação do método proposto. Cada par de imagens representa a aparência original do objeto e o resultado após o processamento da nuvem de pontos.

4.1.1 Stanford Bunny



Objeto 3D.

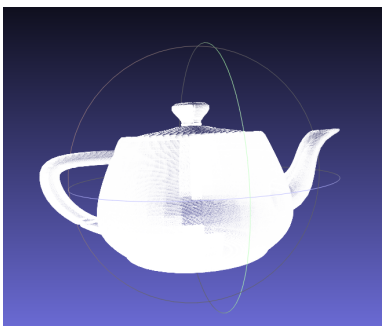


Nuvem de pontos.

Figura 6 – Stanford Bunny.

4.1.2 Teapot

Objeto 3D.



Nuvem de pontos.

Figura 7 – Teapot.

A princípio, o resultado se apresenta excelente. No entanto, ao verificarmos a parte inferior do objeto, podemos ver que o método não captura toda a superfície, apresentando buracos.

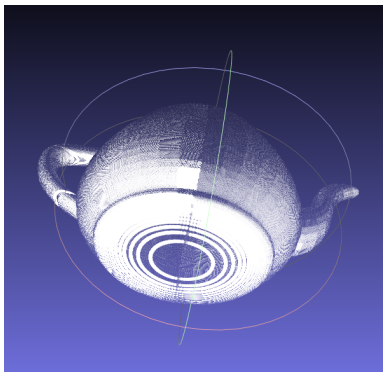
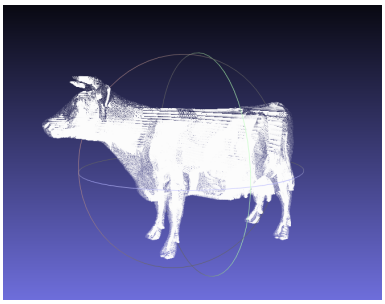


Figura 8 – Parte inferior do Teapot.

4.1.3 Cow

Objeto 3D.



Nuvem de Pontos.

Figura 9 – Cow.

4.2 Limitações do método de fatiamento

Ao utilizar o método de fatiamento paralelo para reestruturar um objeto tridimensional, surgem algumas limitações inerentes à técnica. Uma das principais limitações é a falta de distinção entre objetos maciços e ocos, ou seja, entre objetos completamente preenchidos e superfícies fechadas sem volume interno. Ambos acabam sendo tratados como sólidos. Um exemplo ilustrativo dessa limitação são os objetos descritos pelas equações $x^2+y^2+z^2 = 1$ e $x^2 + y^2 + z^2 \leq 1$, que, nesse contexto, são considerados essencialmente o mesmo. Isso ocorre porque o processo se limita a descrever a superfície do objeto, sem consideração pelo seu interior.

O espaçamento entre as fatias, denotado por ϵ, corresponde à distância fixa entre planos consecutivos. Embora este espaçamento uniforme simplifique o processo de reconstrução,

ele pode resultar na perda de informações cruciais, especialmente em regiões onde a superfície do objeto apresenta variações bruscas. Por exemplo, quando há uma mudança significativa na curvatura da superfície entre duas fatias consecutivas, tal variação pode não ser capturada de maneira adequada.

Matematicamente, para um objeto cuja superfície é representada por $S(x, y, z)$, uma variação abrupta entre as fatias $z = z_i$ e $z = z_{i+1}$ pode ser caracterizada por um grande valor da derivada parcial $\frac{\partial S}{\partial z}$ nessa região. Como a técnica de fatiamento assume que ϵ é suficientemente pequeno para que $\frac{\partial S}{\partial z} \approx 0$, descontinuidades ou mudanças abruptas na superfície não são estruturadas de maneira precisa. Em áreas com grandes variações na superfície, pode ser necessário reduzir ϵ para garantir uma representação mais fiel.

Uma solução adicional seria complementar o fatiamento paralelo com cortes em outras orientações, como os cortes sagital e coronal, reduzindo significativamente a perda de informação, mas este trabalho se limita ao método de corte horizontal.

Outro exemplo dessa limitação pode ser observado em um cilindro posicionado ao longo do eixo vertical. Ao aplicar fatiamento paralelo ao plano XY, as faces que correspondem às extremidades do objeto não são corretamente estruturadas. Como resultado, o cilindro pode ser reconstruído com extremidades abertas, assemelhando-se mais a um cano do que a um cilindro fechado. Isso ocorre porque o método de extração das bordas do objeto presentes em cada fatia não coletam informação sobre o seu interior, mas sim apenas da borda do objeto.

5 Conclusão

Este trabalho explorou o processo de fatiamento de objetos tridimensionais e a extração de nuvens de pontos utilizando uma abordagem baseada em imagens bidimensionais sequenciais. O objetivo principal foi desenvolver um método que permitisse capturar as propriedades do objeto a partir de coleções de imagens paralelas, possibilitando a representação visual de estruturas complexas.

Os resultados obtidos mostram que o método proposto consegue capturar com precisão as bordas dos objetos fatiados e gerar nuvens de pontos coerentes com a estrutura tridimensional original. A implementação em Python, integrada com as ferramentas Blender e MeshLab, se mostrou eficaz em automatizar o processo de geração e visualização dos modelos 3D.

No entanto, algumas limitações foram identificadas. Em especial, o método demonstrou dificuldade em representar adequadamente objetos com variações abruptas de curvatura, uma vez que o espaçamento fixo entre as fatias (ϵ) pode levar à perda de detalhes importantes em regiões complexas. Adicionalmente, as áreas de grande variação da curvatura com relação ao eixo Z apresentou desafios, sugerindo que abordagens complementares, como fatiamento em outras direções, poderiam melhorar a precisão da representação final.

Para trabalhos futuros, recomenda-se explorar métodos de interpolação mais avançados e a aplicação de técnicas de aprendizado de máquina para aprimorar a precisão das bordas e a qualidade da reconstrução. Além disso, a inclusão de cortes em diferentes orientações poderia ampliar a aplicabilidade da técnica a objetos com geometrias mais complexas.

Em suma, o estudo contribui para o campo da geometria computacional ao propor uma alternativa viável para a reconstrução de superfícies a partir de fatias bidimensionais, com potencial para aplicações futuras.

Referências

3D SLICER COMMUNITY. **3D Slicer: A Multi-Platform Software for Medical Image Computing**. [S.l.: s.n.], 2024. Version 5.0. Disponível em: <https://www.slicer.org>.

BLENDER FOUNDATION. **Blender Python API**. [S.l.], 2024. Disponível em: <https://docs.blender.org/api/current/>.

BLENDER ONLINE COMMUNITY. **Blender - um pacote de modelagem e renderização 3D**. [S.l.: s.n.], 2018. Disponível em: <http://www.blender.org>.

BOTSCH, Mario et al. **Polygon Mesh Processing**. [S.l.]: A K Peters/CRC Press, 2010.

BOUTELL, Thomas. **PNG (Portable Network Graphics) Specification**. [S.l.: s.n.], 1997. Accessed: 2024-10-18. Disponível em: <https://www.w3.org/TR/PNG/>.

BRADSKI, GARY. **OpenCV: Open Source Computer Vision Library**. [S.l.], 2024. Versão 4.x. Disponível em: <https://opencv.org/>.

CIGNONI, Paolo; CALLIERI, Marco et al. MeshLab: an Open-Source Mesh Processing Tool. In_____. **Eurographics Italian Chapter Conference**. [S.l.]: The Eurographics Association, 2008. ISBN 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

CIGNONI, Paolo; MUNTONI, Alessandro et al. **MeshLab**. [S.l.]: Zenodo. DOI: 10.5281/zenodo.5114037.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital Image Processing**. 3rd. Upper Saddle River, NJ: Pearson Prentice Hall, 2008.

INTERNATIONAL TELECOMMUNICATION UNION. **Recommendation ITU-R BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios**. [S.l.: s.n.], 1995. Accessed: 2024-10-18. Disponível em: <https://www.itu.int/rec/R-REC-BT.601/>.

PYTHON SOFTWARE FOUNDATION. **Python Language Reference, version 3.10**. [S.l.: s.n.], 2023. Disponível em: <https://www.python.org>.

SOBEL, Irwin; FELDMAN, Gary. **An Isotropic 3x3 Gradient Operator for Image Processing**. [S.l.: s.n.], 1968. Accessed: 2024-10-18. Disponível em: <https://web.stanford.edu/class/ee368b/handouts/sobel.pdf>.

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. London: Springer, 2011.