Universidade Federal Fluminense

ANA CAROLINA DE PAULA

Desenvolvimento de uma Ferramenta de Reconstrução 3D em Linguagem Python

VOLTA REDONDA

2024

Desenvolvimento de uma Ferramenta de Reconstrução 3D em Linguagem Python

Ana Carolina de Paula

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacio- nal em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Mode- lagem Computacional em Ciência e Tecnolo- gia. Área de Concentração: Métodos mate- máticos e computacionais aplicados à enge- nharia e ciência.

Orientador:

Weslley Luis da Silva

Assis

Coorientador:

Vanessa da Silva Garcia
Reinaldo Rodríguez
Ramos Fábio José Bento
Brum

Universidade Federal Fluminense

Ficha catalográfica automática - SDC/BEM
Gerada com informações fornecidas pelo autor

D278d De Paula, Ana Carolina
Desenvolvimento de uma Ferramenta de Reconstrução 3D em Linguagem Python / Ana Carolina De Paula. - 2024.
89 p.: il.

Orientador: Weslley Luis da Silva Assis.
Coorientador: Vanessa da Silva Garcia.
Dissertação (mestrado)-Universidade Federal Fluminense, Escola de Engenharia Industrial e Metalúrgica de Volta Redonda, Volta Redonda, 2024.

1. Processamento de imagens. 2. Computação Gráfica. 3. Seccionamento em Série. 4. Linguagem Python. 5. Produção intelectual. I. Assis, Weslley Luis da Silva, orientador. II. Garcia, Vanessa da Silva, coorientadora. III. Universidade Federal Fluminense. Escola de Engenharia Industrial e Metalúrgica de Volta Redonda. IV. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Aprovada por:

Documento assinado digitalmente
gov.br
WESLLEY LUIZ DA SILVA ASSIS
Data: 28/03/2025 09:52:12-0300
Verifique em https://validar.iti.gov.br

Prof. Weslley Luiz da Silva Assis, D.Sc. / MCCT-UFF
(Presidente)

Documento assinado digitalmente
gov.br
ELIANE DA SILVA CHRISTO
Data: 21/03/2025 20:52:59-0300
Verifique em https://validar.iti.gov.br

Prof. Eliane da Silva Christo, D.Sc. / MCCT-UFF

Documento assinado digitalmente
gov.br
MARCUS VINICIUS CARVALHO GUELPELI
Data: 24/03/2025 14:49:21-0300
Verifique em https://validar.iti.gov.br

Prof. Marcus Vinicius Carvalho Guelpeli, D.Sc. / PPGED

Volta Redonda, 27 de Dezembro de 2024.

"Dedico este trabalho aos jovens bons de coração"

O Mágico de Oz

Agradecimentos

Agradeço por ter feito este mestrado a todos que passaram pelo meu caminho, a minha família, minha mãe Ana Lúcia de Souza Corrêa, ao meu pai Carlos Alberto de Paula por me darém toda a corbetura para que eu pudesse terminar com maior tranquilhidade. Aos meus irmãos Ana Carla de Paula e Carlos Alberto Júnior por estarem presentes na minha vida. A minha psicóloga Mayara da Conceição por ter me dado ajuda para que eu mantivesse a minha saúde mental para trilhar novos caminhos. Aos amigos que conheci durante o mestrado, o Allan, o Douglas, ao Welbe e a Kethelen, que me fizeram passar por este momento com mais calma. Ao Anderson que me ajudou em partes da pesquisa pela iniciação científica. Para os professores do MCCT que nos ajudaram no nosso caminho, ao meu orientador Professor Weslley Luiz da Silva Assis que aceitou o desafio de me orientar. E agradecer a Capes, pela oportunidade de fazer pesquisa no Brasil.

Resumo

Esta dissertação apresenta o desenvolvimento de uma ferramenta computacional para a reconstrução tridimensional (3D) de objetos, utilizando a linguagem de programação Python e a técnica de seccionamento em série. O objetivo principal é a criação de uma interface intuitiva que permita a geração e processamento de imagens 2D seriadas para formar uma representação 3D fiel ao objeto original. O software foi validado em dife- rentes contextos, incluindo imagens médicas e materiais poliméricos, demonstrando sua aplicabilidade em diversas áreas, como ciência dos materiais e estudos anatômicos. A fer- ramenta visa fornecer uma solução acessível e eficiente para pesquisadores e estudantes, impulsionando novas descobertas e aplicações na análise de microestruturas e modelos complexos.

Abstract

This dissertation presents the development of a computational tool for the three-dimensional (3D) reconstruction of objects, using the Python programming language and the serial sectioning technique. The main objective is to create an intuitive interface that allows the generation and processing of serialized 2D images to form a 3D representation faithful to the original object. The software was validated in different contexts, including medical images and polymeric materials, demonstrating its applicability in various fields, such as material science and anatomical studies. The tool aims to provide an accessible and efficient solution for researchers and students, driving new discoveries and applications in microstructures analysis and complex models.

Palavras-chave

1-Processamento de

Imagens 2-Reconstruçao

3D

3-Linguagem Python

4-Seccionamento em

Série 5- Computação

Gráfica

Keywords

1- Image

Processing 2-3D

Reconstruction

3- Python Language

4- Serial Sectioning

5- Computer Graphics

Sumário

Lista de Figuras

Lista de Símbolos, Siglas e Abreviaturas

MCCT : Modelagem Computacional em Ciências e Técnologia UFF : Universidade Federal Fluminense

PPI : *Pixels por Polegada*

RGD : *Red Green Blue*

RM : *Ressonância Magnética*

GPU : *Unidade de Processamento Gráfico*

OpenCV : *Open Source Computer Vision Library*

Tkinter : *Toolkit Interfac*

VTK : *Visualization Toolki*

dat : *Data File Format*

GRS : *Representação Geométrica Esparsa*

2D : *Bidimensional*

3D : *Tridimensional*

Capítulo 1

Introduçã

o

A reconstrução tridimensional (3D) é uma técnica amplamente utilizada em diversas áreas, como engenharia, medicina, biologia e geociências, possibilitando a visualização e análise de estruturas complexas a partir de dados bidimensionais (2D) [14]. Essa técnica é fundamental para o entendimento detalhado de microestruturas, permitindo uma repre- sentação volumétrica precisa de objetos de estudo que, de outra forma, seriam limitados pela análise puramente bidimensional, [9]. Com o avanço das tecnologias computacionais, especialmente nas áreas de processamento de imagem e computação gráfica [3],surgem novas oportunidades para o desenvolvimento de ferramentas que facilitem a reconstrução 3D com alta fidelidade e acessibilidade.

Neste contexto, a presente dissertação se concentra no desenvolvimento de uma fer- ramenta computacional em Python para reconstrução 3D, aplicando técnicas de seccio- namento em série. Essa ferramenta visa oferecer uma interface intuitiva para usuários, incluindo alunos, pesquisadores e professores, facilitando seu uso em ambientes educacio- nais e de pesquisa. Além disso, a proposta almeja alcançar uma representação tridimen- sional que seja quase fiel ao objeto real [1] garantindo assim sua aplicação em áreas como ciências dos materiais e estudos futuros envolvendo diferentes composições materiais.

O objetivo principal deste trabalho é a criação de um software de código aberto que possibilite a reconstrução 3D de objetos utilizando técnicas de seccionamento em série. Esta abordagem permite a geração de imagens 2D de diferentes cortes de um objeto, que são posteriormente integradas para formar uma representação

tridimensional. O código desenvolvido foi validado usando materiais poliméricos e imagens médicas, demonstrando sua aplicabilidade e robustez em diferentes contextos de uso [3].

Além da implementação da ferramenta, esta dissertação explora os fundamentos teó-

ricos e metodológicos que sustentam a computação gráfica e o processamento de imagem, essenciais para a reconstrução tridimensional. Entre os tópicos abordados estão as dife- rentes técnicas de representação de imagem [19], como a análise matricial e vetorial, e os métodos de interpolação utilizados para suavizar as transições entre as seções seriadas, proporcionando uma visualização mais precisa e contínua do objeto reconstruído.

A escolha da linguagem Python para o desenvolvimento do software é justificada por sua simplicidade, versatilidade e pela ampla disponibilidade de bibliotecas específicas para computação científica, como a OpenCV [20], utilizada para o processamento das imagens, e a Tkinter [21], empregada na construção da interface gráfica [22]. Essa combinação permite a criação de uma ferramenta poderosa e acessível, capaz de atender a diferentes níveis de usuários, desde iniciantes até especialistas na área.

Portanto, o desenvolvimento desta ferramenta busca contribuir para o avanço das técnicas de reconstrução 3D, proporcionando um meio eficiente e acessível de realizar essas tarefas em ambientes acadêmicos e de pesquisa[18]. A abordagem proposta visa não apenas facilitar o processo de aprendizado, mas também fomentar novas descobertas e aplicações práticas em áreas diversas, desde a análise de microestruturas de materiais até o estudo de modelos anatômicos complexos.

1.1 Objetivo Geral

Este trabalho tem como objetivo a criação e desenvolvimento de uma ferramenta em Python [23] para reconstrução 3D de objetos, via seccionamento em série, que apresente uma interface simples de entendimento para que os próximos usuários, como alunos, pes- quisadores e professores a desfrutem para seus estudos.

1.1.1 Objetivos Específicos

- Realizar a reconstrução 3D de objetos que são seccionados para a geração de imagens em 2D e a realização de um código para este fim;
- Utilizar o seccionamento em série para a validação do código usando ressonância magnética e fotos digitais para que possam ser usados em trabalhos futuros;
- Realização, pelo grupo de estudo, de uma interface mais amigável para um melhor entendimento do seu funcionamento;

- Mostrar que o código realizado gera representações fiéis 3D de objetos reais a serem

utilizados como modelo;

- Para que o código, uma vez validado, seja usado para fins de estudos para os pesquisados e alunos de universidades, por ser um código com fins educativos e livre para ser usado por todos.
- Melhorar o código de uma pesquisa anterior, feita em linguagem C++ [1], para que o programa seja fluído e com uma interface amigável.

1.2 Justificativa

A motivação para a realização desta pesquisa encontra-se na necessidade de ferramen- tas acessíveis, precisas e de fácil utilização para reconstrução tridimensional de objetos a partir de seccionamento em série. Embora existam soluções no mercado, muitas delas são complexas, de alto custo ou restritas a aplicações específicas, o que dificulta sua uti- lização em universidades e centros de pesquisa, especialmente por alunos e pesquisadores iniciantes.

A linguagem Python [23] foi escolhida devido à sua simplicidade, versatilidade e pela vasta disponibilidade de bibliotecas específicas para processamento de imagens e desen- volvimento de interfaces gráficas. Exemplos incluem a OpenCV, empregada no processa- mento das imagens, e a Tkinter [21], utilizada na construção da interface gráfica. Essa combinação viabiliza a criação de uma ferramenta poderosa, acessível e adaptável a dife- rentes áreas de conhecimento.

A ferramenta proposta busca contribuir para o avanço das técnicas de visualização tridimensional, promovendo uma solução acessível e eficiente que poderá ser aplicada em áreas como:

Medicina: reconstrução de imagens obtidas por ressonância magnética ou tomografia; Ciências dos Materiais: análise de microestruturas e materiais poliméricos; Biologia e Anatomia: estudo detalhado de amostras biológicas; Engenharia: modelagem e visualiza- ção de componentes complexos.

1.2.1 Relevância da Pesquisa

A relevância desta pesquisa está associada à criação de uma solução educacional e científica inovadora, que facilitará a reconstrução 3D de objetos a partir de imagens 2D. A interface amigável e a robustez da ferramenta proposta garantirão sua ampla utilização,

tornando-se um recurso essencial para estudantes, professores e pesquisadores interessados em reconstruções tridimensionais. Embora existam diversas soluções no mercado para reconstrução 3D, muitas delas são proprietárias e têm custos elevados, o que dificulta sua acessibilidade para estudantes e pesquisadores. Ferramentas como MATLAB, 3D Slicer e Amira são exemplos populares, mas apresentam restrições devido ao licenciamento ou complexidade de uso. O presente trabalho busca oferecer uma alternativa de código aberto, mais acessível e focada em aplicações educacionais e científicas

Capítulo 2

Revisão da Literatura

2.1 Fundamentos da Computação Gráfica

Primeiro precisa-se explicar, de uma maneira geral, o que é a Computação Gráfica [22]. A Computação gráfica refere-se à área da Ciência da Computação que estuda a formação, o conceito, a manipulação e a busca da análise de uma imagem digital através de um computador.A [22] computação gráfica tem tido uma enorme expansão nos últimos tempos devido ao aumento do progresso de construção dos hardwares. A Computação Gráfica se propõe a criação de algoritmos para a geração de imagens para o seu processamento. Esta área está presente em vários campos, desde a indústria automobilística a área médica.

A [2] Computação Gráfica trabalha com dois tipos de informações: a descritiva e a visual. A informação visual é o que se vê em um computador, enquanto a informação descritiva é o modelo matemático que é empregado para a construção dessa imagem. Ainda [2], a área de processamento de imagens, que está relacionada à Computação Grá- fica, abrange operações matemáticas usadas na imagem para a representação da imagem. Computação Gráfica envolve operações de síntese de imagem, isto é, a criação de uma visualização do modelo. Na figura 1 abaixo, pode-se ter uma compreensão sobre a relação da forma visual e descritiva dentro do conceito de processamento de imagens.

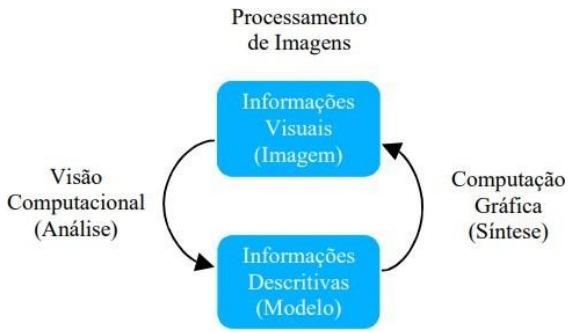

Figura 1: Fluxograma sobre o processamento de uma imagem digital

2.2 Processamento de Imagem

Este tópico será discutido o conceito de imagem digital, cores, função contínua, função discreta, fotografia digital.

2.2.1 Imagem Digital

A imagem digital [24] é uma representação de uma imagem codificada para uma forma digital, que pode ser processada, manipulada, transmitida e armazenada de uma forma eletrônica. Ela é composta de várias composições de seus pixels, diferente de uma imagem analógica, cuja característica é ser contínua [25]. A imagem é deslocada, perdendo seu sentido intrínseco, sendo seu significado demonstrado a partir do ambiente e pela forma que ela foi construída [26].Sua forma de captura [27] pode ser de forma direta ou indireta ou até semi-indireta, na qual ajuda na sua possibilidade de manipulação, além de uma maior segurança de armazenamento para usos futuros.

2.2.2 Paradigma dos Quatro Universos

O paradigma dos quatro universos no processamento de imagem é um modelo que descreve as abstrações em diferentes níveis envolvidas na manipulação e representação de dados gráficos. Para facilitar a compreensão desse conceito, dividiu-se em quatro conjuntos: Universo físico, Universo matemático, Universo de representação e Universo de implementação, cada um com sua definição específica.

1- Universo Físico: ele é definido [28] pelos objetos ou fenômenos físicos que deseja- se representar digitalmente [1], sendo grandezas analógicas não digitais que deseja-se

representar.

2- Universo Matemático: o [3] universo matemático é o que se faz possível a represen- tação abstrata, com conjunto de ferramentas matemáticas, transformar objetos analógicos físicos em um modelo matemático. Dito por [28], é o que envolve a modelagem matemática de objetos físicos.

3- Universo de Representação: o universo de representação é o que transforma os mo- delos matemáticos dos objetos físicos em objetos digitais [1]. É onde permite implementar os processos de quantização [3].

4- Universo da Implementação: [1] é o universo que permite a codificação do sinal discretizado para o armazenamento do computador utilizando estrutura de dados previa- mente definida e também com algoritmos associados ao universo de representação, assim sendo viável a sua representação 3D.

De acordo com a figura 2 [1], têm-se o esquema dos paradigmas do universo:

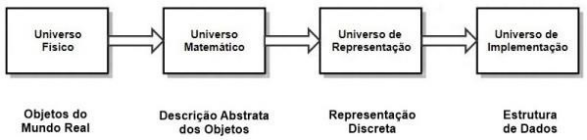

Figura 2: Fluxograma do Paradigma dos Quatro Universos [1]

Uma imagem fotográfica [2] é obtida do mundo real através de uma câmera ou algum sensor que possa captar luz. Essa imagem real é capturada por um filme fotográfico que represente bem uma imagem real que possa se transformar em uma representação digital. O filme é um plano retângulo bem definido [2], ou seja, uma imagem 2D contínua, no qual o domínio é ou seu plano e seu contra-domínio é a seu espaço de cor. A discretização é bem simples, pois segue a discretização de qualquer sinal 2D, como mostrado na figura 3.

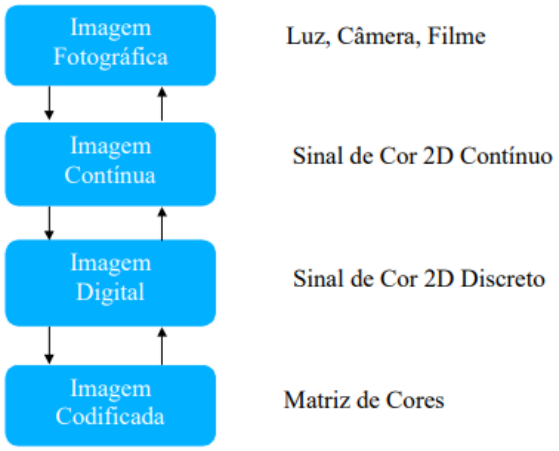

Figura 3: Fluxograma da imagem fotográfica de acordo com o paradigma dos quatro universos [2]

A imagem fotografada é modelada de acordo com a equação matemática:

$$f(x, y) = \text{Cor no ponto } (x, y)\ x \in [0, X] \text{ e } y \in [0, Y] \tag{2.1}$$

, onde x e y são números reais, em um intervalo limitado entre de 0 a X, e de 0 a Y , respectivamente [2], representados em um gráfico na figura 4 abaixo:

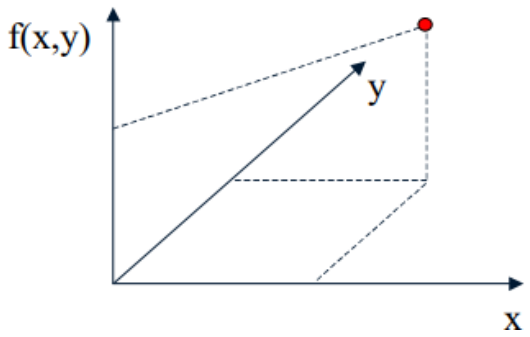

Figura 4: Gráfico da cor em um ponto [2]

Uma imagem digital é feita a partir de uma amostragem e de sua quantização dessa mesma função e pode ter sua representação pela essa mesma função [2], só que

utilizando x e y sendo números inteiros. Com o plano já mostrado, obtêm-se uma matriz de valores representantes de cor, onde cada elemento que compõe essa matriz é chamado de Pixel [2]. Nos próximas seção terá o aprofundamento sobre cores, mas vale citar que a cor é

um fenômeno físico que devê-se ser feito os mesmos passos da definição do plano para a definição da cor [2]. A cor também segue os conceitos do fluxograma da figura 3 acima, onde ela deve ser descoberta a sua definição, suas representações e transformá-las em uma estrutura de dados para ser lido em um computador [2].

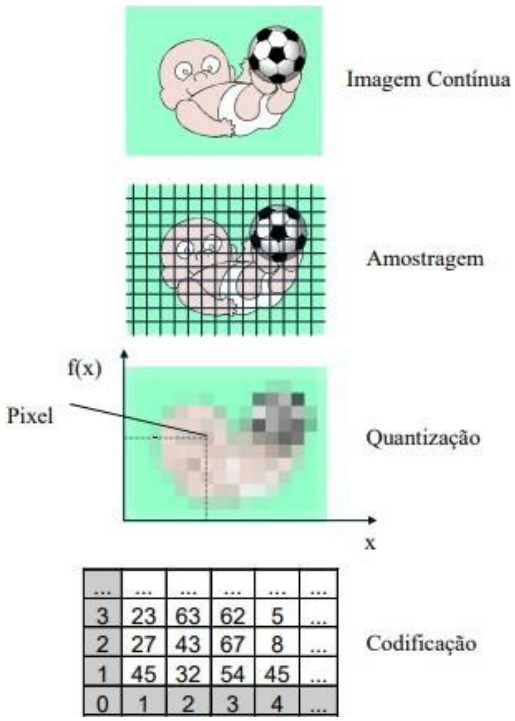

Figura 5: O processo de descritezação de uma imagem [2]

A figura 5 mostra uma simples descrição de um processo de descritezação de uma imagem e a possível codificação de sua cor por elemento de matriz.

2.2.3 Função Contínua x Função Descritiva

O conceito de função, de modo matemática terá um bom aprofundamento nesta seção. Neste subcpítulo será apresentado o conceito matemático de uma função contínua e uma função descritiva, que são conceitos de Cálculo bem delicados a respeito de seu estudo [1]. Uma função contínua [3] é a qual para cada x existe um único valor $f(x)$, onde em seu domínio ela terá que ser totalmente contínua em todos os seus pontos e quando esse conceito não é atendido ela é chamada de função discreta ou descontínua, concluindo assim que dados contínuos são dados quantitativos que podem assumir qualquer valor numérico dentro do seu intervalo de variação [3]. Por outro lado, dados discretos são dados quantitativos que só podem assumir valores distintos, sejam eles finitos ou infinitos,

desde que sejam enumeráveis. Aprofundando [29] mais sobre os tipos de função, a função contínua tem como propriedades:

1-Sem saltos: As funções contínuas não têm interrupções, saltos ou quebras no gráfico;

2- Limite igual ao valor da função: $\lim_{x \to a} f(x) = f(a)$.

São exemplos de funções contínuas: Funções polinomiais, funções trigonométricas e funções exponenciais.

A função discreta [29] tem como propriedades:

1-As funções discretas podem ter saltos ou interrupções entre os valores de saída;

2- Muitas vezes definidas por uma sequência de valores, como $f(n)$ para n inteiro.

São exemplo de funções discretas a funções de contagem e funções indicadoras. De acordo com a figura 6 abaixo, têm-se uma comparação de gráfica das funções contínuas e funções discretas.

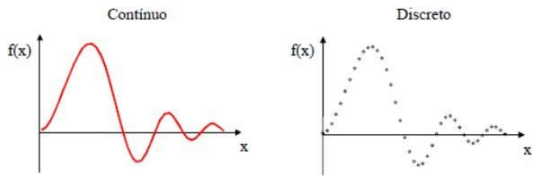

Figura 6: Diferença entre função contínua e função discreta [3]

Ainda deve-se definir a diferença de dados contínuos e dados discretos.Dados qua- litativos [1] que podem assumir todos os valores numéricos dentro de seu intervalo são chamados de dados contínuos, enquanto aqueles que podem assumir apenas um número finito ou infinito enumerável de valores distintos são denominados dados discretos. A diferença entre dados contínuos e dados discretos é fundamental para entender como os computadores lidam com informações. Dados contínuos são quantitativos e podem as- sumir todos os valores numéricos dentro de um intervalo, enquanto dados discretos são limitados a um número finito ou infinito enumerável de valores distintos [1]. Os compu- tadores armazenam dados usando bits, que têm valores binários de 0 ou 1. Para facilitar, esses bits são agrupados em bytes [2], cada um capaz de representar até 256 valores di- ferentes, variando de 0 a 255. Sistemas operacionais de 16 bits e 32 bits referem-se ao tamanho máximo de palavra que podem processar de uma vez, sendo os sistemas de 32 bits mais eficientes por lidarem com palavras maiores [2].

Como os computadores operam principalmente com números inteiros ou aproximações

de números reais (ponto flutuante) [1], eles não podem representar diretamente funções contínuas, apenas simular sua continuidade através de discretização. Esse processo en- volve tomar valores discretos ao longo do eixo x (domínio) e armazenar os correspondentes valores de f(x) (contradomínio), conhecido como amostragem e quantização, respectiva- mente [2]. Assim, [1], ao discretizar tanto o domínio quanto o contradomínio de uma função, podemos simular a continuidade de uma função no computador. Para melhor en- tendimento, a figura 7 abaixo mostra os processos de amostragem e quantização quando o domínio da função está em R^2 e o contradomínio em R^3.

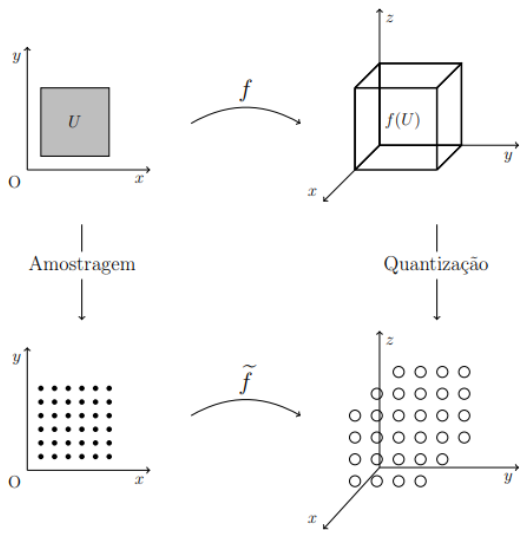

Figura 7: Uma representação de discretização da imagem [1]

A função $\tilde{f}$ segue a mesma regra da função f, porém seu domínio e imagem são "reduzidos"em comparação com os da função f. Essa notação é utilizada para destacar que o processo de amostragem e quantização foi aplicado a f, resultando em $\tilde{f}$.

2.2.4 Representação De Imagem Matricial e Vetorial

A representação de imagens é de suma importância para área de computação gráfica e processamento de imagens quandos essas são compostas por vetores e matrizes.

2.2.4.1 Análise de Matrizes Quaterniônicas em Imagem Colorida

Esse modelo de representação [19], como esprassa vetoril, utilizando matrizes quater- niônicas para analisar a preservação de imagens coloridas inerentes, como

representada pela figura 8:

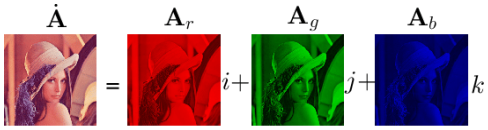

Figura 8: Representação de uma imagem por matrizes quaternions[4]

Uma representação geral comum de uma matriz quarteônica,segundo o descobridor dos quaternion pelo matemático e físico William Rowan Hamilton, em 1853[4], é dado pela equação

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{2.2}$$

,onde q_0, q_1, q_2 e q_3 são quatro números reais e i, j, k são três unidades de números imaginaríos, que é relatada como:

$$i^2 = j^2 = k^2 = ijk = -1. \tag{2.3}$$

O conjungado de q é $q^* = q_0 - q_1 i - q_2 j - q_3 k$, com a sua norma de q é igual a $\|q\| = \sqrt{q\, q^*} = \sqrt{q^2 + q^2 + q^2 + q^2}$. Para ser um quartenion puro uma das suas partes reais tem que ser igual a zero. Sendo H um campo skew de quartenions, com ordem 4 sobre R. Uma matriz conhecidas de quartenions $m \times n$ é da forma $Q = Q_0 + Q_1 i + Q_2 j + Q_3 k$ e com $Q_s \in \mathbb{R}^{m \times n}$; $s = 0, 1, 2, 3$. Uma matriz de quaternions puros é quando é cujo os elementos são zero.Em RGB, cada pixel pode ser representado por um quartenion puro $Ri + Gj + Bk$, onde R, GeB representam as cores vemelho, verde e azul, respectivamente. Uma imagem pode ser representada por matrix quartenions puros $Q = [q_{ab}]_{m \times n}$, em que cada elenebto desta é representada por $q_{ab} = R_{ab} i + G_{ab} j + B_{ab} k$, em cada um representando pixels de uma cor e R_{ab}, G_{ab} & B_{ab} são números negativos. A figura 9, representa essa parte [4]:

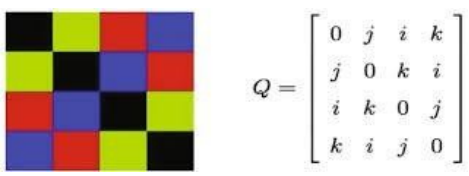

Figura 9: Representação de cada cor de pixel por método matricial[4]

Esse tipo de representação evita viés de matiz, que o torna mais eficiente para repre-

sentação de imagens coloridas, remoção de ruído, preenchimento de algumas lacunas e uma super-resolução [19].

2.2.4.2 Um método robusto de representação de imagem contra variações de iluminação e oclusão

Este método de representação propõe um modelo de GRS (Representação Geométrica Esparsa) [5]. As imagens tradicionais devem ser convertidas em vetores unidimensionais, o que pode comprometer parte da estrutura original. A figura 10 ilustra o cálculo da resposta.

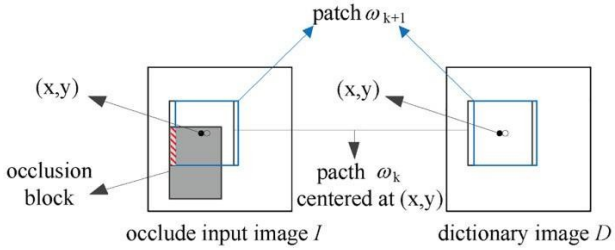

Figura 10: Ilustração do cálculo dos coeficientes de representação sob oclusão.

Propõe-se [5] a utilização do GRS como uma única imagem que preserva a imagem original, onde os coeficientes GRS têm um significado geométrico específico, revelando a estrutura geométrica intrínseca da imagem. A figura 11 ilustra a binarização esparsa de uma imagem:

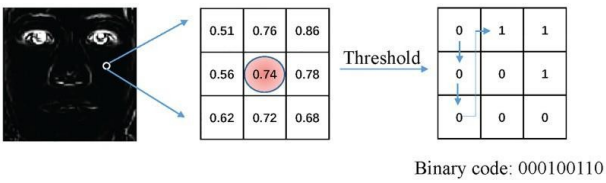

Figura 11: Codificação binária de coeficientes de representação esparsa[5]

2.2.4.3 Matrizes Vetoriais para Similaridade de Padrões

É formado por uma matriz vetorial [6], onde cada elemento desta representa um canal de cor, podendo ser usada para reconhecimento de objetos com a definição de padrões de similaridade, na qual generaliza a similaridade estrita. Um exemplo das definições [6], quando seja $k, m, n \in N$, para $k, m, n > 1$. Vetor $\boldsymbol{u}_{i,j} \in R^K$ para $1 \leq i \leq m$ e $1 \leq$

$j \leq n$.Vetor matriz $m \times n$ é defenida como

$$\boldsymbol{A} = \begin{matrix} \boldsymbol{u}_{1,1} & \boldsymbol{u}_{1,2} & \cdots & \boldsymbol{u}_{1,n} \\ \boldsymbol{u}_{2,1} & \boldsymbol{u}_{2,2} & \cdots & \boldsymbol{u}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{u}_{m,1} & \boldsymbol{u}_{m,2} & \cdots & \boldsymbol{u}_{m,n} \end{matrix} \tag{2.4}$$

,na qual a matriz pode ser escrita como $\boldsymbol{A}_{m\times n} = [\boldsymbol{u}_{i,j}]_{m\times n}$.

Uma aplicação para uma imagem digital [6], na qual os pixesl são representados como uma matriz $m \times n$, e $K = 3$. As dimensões de vetores são matrizes de três dimensões com vetores $\boldsymbol{u}_{i,j} = (r_{i,j}, g_{i,j}, b_{i,j})$, onde $r_{i,j}, g_{i,j}, b_{i,j} \in \{0, 1, 2, \cdots, 255\}$. Sendo o [6] elemento $\boldsymbol{u}_{i,j}$ $= (r_{i,j}, g_{i,j}, b_{i,j})$ a representação de cada pixel na sua posição (i, j).

De acordo com a figura 12 mostra uma imagem digital de uma faculdade de matemá- tica e ciências naturais da Universidade Halu Oleo, com resolução de 620 x 1080 pixels. Esta imagem [6] é representada por um vetor matriz $\boldsymbol{B}_{620\times1108}$. Esta matriz é muito ex- tensa para ser mostrada aqui. Porém as três linhas com a suas duas colunas $\boldsymbol{B}_{620\times1108}$ é expressada pela matriz vetor A, onde

$$A = \begin{matrix} (132, 172, 207) & (138, 177, 210) \\ (135, 170, 210) & (139, 169, 207) \\ (136, 175, 208) & (135, 174, 205) \end{matrix} \tag{2.5}$$

Figura 12: Imagem digital da Universidade de Halu Oleo, 620x1080[6]

Algumas introduções algébricas em matrizes vetoriais serão mencionadas, mas não serão aprofundadas na dissertação.

Apresentam-se aqui três maneiras distintas de representar uma imagem digital em

2D, fundamentais para a compreensão das representações 3D de objetos.

2.2.5 Cores

Cores são fenômenos complexos que englobam diversos aspectos, tanto físicos quanto perceptivos. Na física, as cores são representadas por comprimentos de onda, enquanto a percepção de cores por humanos e animais é menos clara e se baseia em teorias e suposições.

O olho humano pode captar luzes [7] vísiveis, o que mostra só uma parte do espectro eletromagnético, pois os olhos não captam abaixo raios infravermelhos ou acima de raios ultravioletas. As ondas que podem ser visíveis estão compreendidas entre o meio que contém o raio infravermelho e o raio ultravioleta, no qual são faixa de comprimentos de onda que são conhecidas como cores. Essas cores são percebidas pelo olho humano, onda a faixa perceptível entre 700nm e 400nm. A figura 13 demonstra o comprimento de onda visível pelo olho humano:

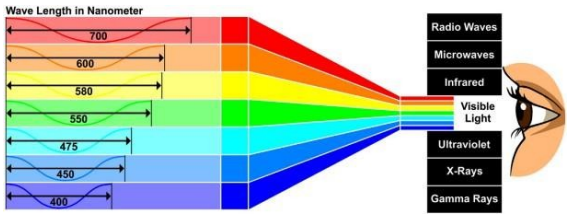

Figura 13: Comprimento de onda visível a olho nu. [7]

Outro modelo de cores conhecida é o modelo RGB (Red Green Blue), que é usado para dispositivos digitais. Este modelo de cores [30], cada cor de cada pixel aparece em um expectro primário com componentes de core RGB, na qual consiste em sua intencidade. Cada componente de cor é conhecido como canais de cor ou planos de cor [30]. Este modelo é baseado na teoria da tricometria, onde cada cor pode ser representada e perceptível por combinção de três cores primárias, sendo combinações de intensidade de luz. A aplicação do modelo RGB são em dispositivos digitais, como display de televisões, monitores de computadores, tela de celulares e etc.

O modelo RGB tem sua representação matemática, que pode ser representada como vetor tridimensional [*R, G, B*] [31], onde *R,G* e *B* são variáveis de intensidade dos com- ponentes vermelho, verde e azul, onde esses valores, de acordo com [31], vão variar de 0 a 255 em uma representação de 8 bits por cada canal, proporcionando um total de

16.777.216 (256^3) da combinação de cores possíveis.

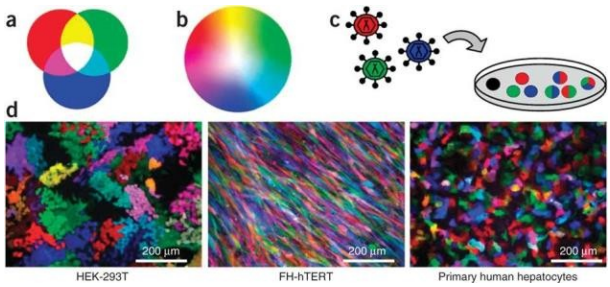

Figura 14: O principal vetor mediado de RGB feito pelo artigo [8]

A figura 14 mostra um estudo [8] que representa respectivamente: O princípio RGB mediado por vetores de marcação é descrito conforme segue:

(a) Baseando-se nas teorias da cor, a combinação de intensidades variadas de vermelho, verde e azul deve resultar na obtenção de quatro cores intermediárias: amarelo, violeta, turquesa e branco.

(b) A mistura das cores básicas em todas as intensidades possíveis gera um espectro completo de cores, cobrindo todo o intervalo do arco-íris.

(c) A introdução simultânea de três proteínas fluorescentes que emitem vermelho, verde e azul—como mCherry (vermelho), Venus (amarelo-verde) e Cerulean (azul)—deve, teoricamente, possibilitar a visualização de até sete tipos distintos de células transduzidas. De acordo com o princípio RGB, a expressão simultânea de diferentes cores básicas nas células deve resultar em cores mistas (ver também Figura Complementar 2).

(d) A marcação RGB mediada por vetores lentivirais foi realizada nas células-alvo utilizando os vetores LeGO-C2, LeGO-V2 e LeGO-Cer2 (Figura 1 suplementar). As ima- gens obtidas mostram sobreposições de fotografias coloridas tiradas sequencialmente com filtros vermelho, verde e azul, usando um microscópio de fluorescência. Após o plaque- amento, todas as células foram cultivadas por aproximadamente 72 horas; enquanto os clones 293T mantiveram-se agrupados durante a cultura, os clones FH-hTERT se espa- lharam pelo recipiente de cultura de tecidos. Observou-se que os hepatócitos primários não se replicaram durante o período de cultivo.

Este estudo médico de demarcação de células tumorais que demosntram a sua de-marcação de cores durante um período de tempo [8]. A precisão na utilização de cores é fundamental em diversas áreas, como imagem e design, para a reprodução efetiva do usuário.A pesquisa de calibração de cor tem sido um campo bastante


estudado para a ga-

rantia de cores consistentes em dispositivos [31]. Muitos avanços tem sido feitos na área de estudo de cores e RGB, para que haja uma melhor acomodação perceptível, como o uso de monitores 4k, Rec 2020 e 8K [32], e o estudo de espaço entre as cores continua sendo um campo de pesquisa para melhor definição para a imagem digital seja mais definida para os usuários.

2.2.6 Pixel

Pixel é a representação da menor unidade da imagem digital, que se mostra essencial para a representação da própria imagem em dispositivos eletrônicos. Cada um dos pixels possui uma posição específica e um valor de cor, que combinados, forma-se a imagem digital. A imagem digital tem sua medida de resolução em PPI (Pixels Por Polegadas), o que ajuda na característica melhor qualidade visual da imagem. A resolução espacial da imagem [33], é representada pelo número de pixels que são disponíveis para tal composição. Quando as imagens tem uma alta resolução, o detalhe fino é preservado e percebido por um observador, enquanto as imagens com baixas resoluções saem distorcidas e perde alguns detalhes. Na figua 15 , mostra as etapas de construção de uma TV LED, onde contém uma placa de malha pixels e placas RGB para a composição de suas cores.

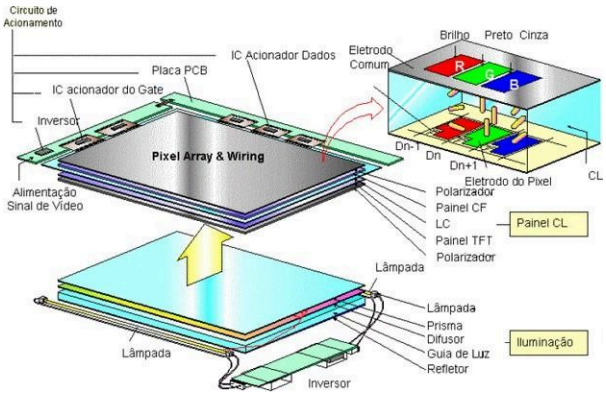

Figura 15: Esquema de montagem de uma TV LED

O pixel tem uma identidade fundamental espacial em imagens, como em sensores remotos, possui desvantagens que deverão ser abordadas para a elaboração de sensores, [34],Muitos profissionais em sensoriamento remoto não compreendem o caráter de res- posta da radiação sensora, apesar de ser um conceito conhecido na área de sensores de imagem. Além disso, a conversão de dados por pixels frequentemente resulta em perda de informações, pois muitos processos de visualização podem descartar aspectos importantes

[35]. Na figura 16 tem-se um exemplo de pixels.

Figura 16: Exemplo de pixels em uma imagem

Os pixels possuem um papel de suma importância no processo de renderização, As transformações tridimensionais em imagens bidimensionais são exibidas em um monitor. Cada pixel contém uma cor do sistema RGB, onde as combinações de cores geram uma cor principal e sua intensidade. A manipulação de pixels é relevante em diversas áreas, como visão computacional e técnicas de *Deep Learning*. A transformada de Fourier, utilizada para restauração de imagens, depende da manipulação de pixels para melhorar a resolução e qualidade, mas este tema não será abordado nesta pesquisa.

Na reconstrução de imagem é crítica o uso de pixels e sua manipulação. Como exem- plo, o uso das imagens médicas, os algoritmos trabalham diretamente com pixels e re- constroem sua imagens 3D em várias seções 2D, e sua resolução depende desse perfeito trabalho com pixels. Usando técnicas, como interpolação bilinear de pixels, a qualidade da imagem reconstruída pode dar um salto em sua definição [36], o que suaviza a imagem visualizada.

2.3 Reconstrução Tridimensional

Esta seção abordará a realização de reconstrução 3D, o seccionamento em série uti- lizado em diversos campos e sua relação com o tema do capítulo, incluindo ciências e engenharia e a possibilidade de reconstrução.

2.3.1 Seccionamento em Série

O seccionamento em série é uma técnica utilizada em histologia e ciência dos mate- riais que gera imagens tridimensionais a partir de múltiplos cortes bidimensionais de um objeto. Esse método é crucial para compreender microestruturas e relações espaciais em materiais biológicos. No entanto, podem surgir problemas nas imagens, como defeitos e

desalinhamentos, comprometendo a qualidade necessária para análises mais eficazes em métodos computacionais avançados.

2.3.1.1 Caracteríscas em Seccionamento em série

Uma das caracteríscas importantes para o seccionamento em série é a precisão em seções seriais para uma boa reconstrução 3D. Técnicas como registro baseado em inten- sidade, registro elástico, entre outros apresentam correção relativamente incompatível e deformações durante o processo de seccionamento, como dito por [9]. Este métod garante que todas as imagens estejam alinhadas.

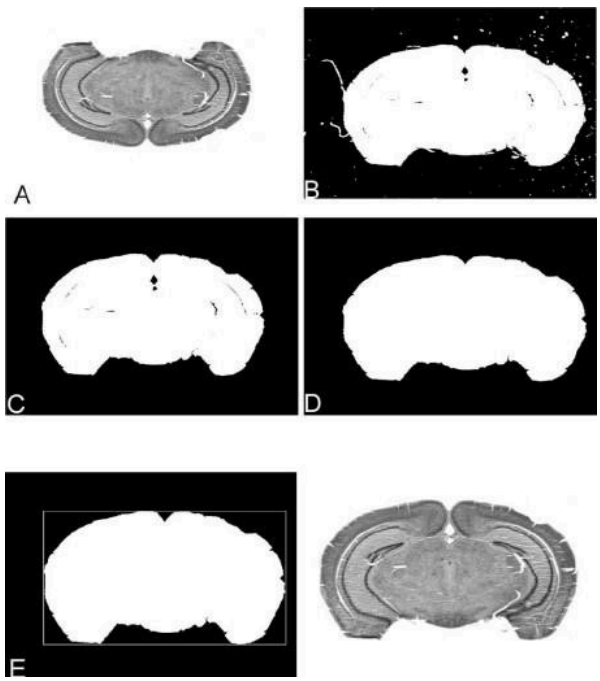

Figura 17: Cerebro de um rato feita com fotos seccionadas. [9]

A figura 17 mostra um procedimento de seccionamento feito por [9]. Métodos de seccionamento devem ser aplicados de certa forma que garantem imagens precisas [37]. A precisão do seccionamento deve ser levado em conta na hora da aplicação para garan- tir uma melhor qualidade na reconstrução tridimensional da microestrutura. Técnicas de corte automáticos, como microtomias e ultamicotomias [38], são usadas para melhor definição das imagens dos cortes realizados, com controle de espessura, diminuindo a possibilidades de erros que podem afetar a seção tirada.

Na figura 18, é apresentado o microtomo, uma das máquinas usadas para realizar seccionamento. Outras técnicas de segmentação automatizadas [**?**], Podem ser utilizadas

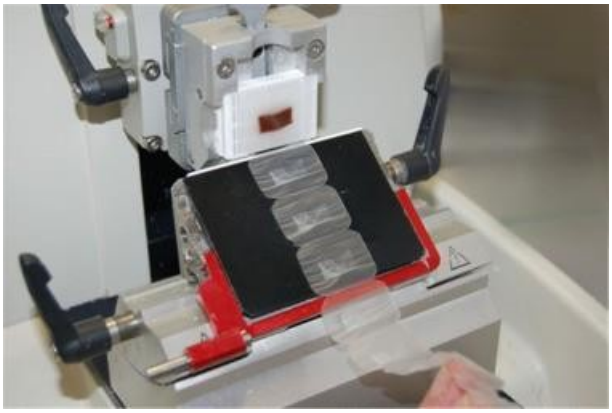

Figura 18: Microtomo [10]

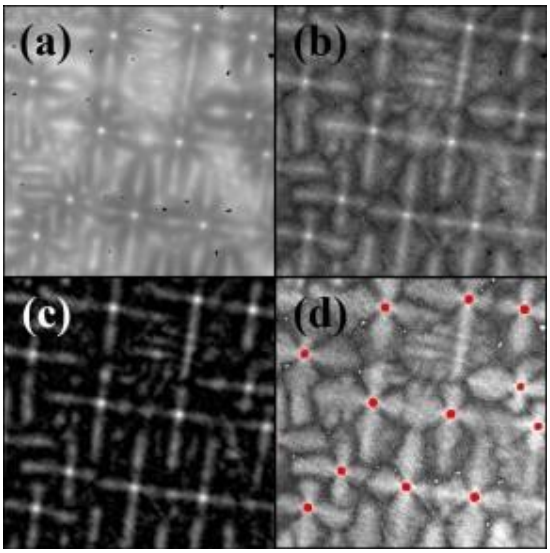

Figura 19: Imagens segmentadas utilizabdo a técnica de simetria em uma turbina[11]

para identificar características em sua estrutura. Métodos que incorporam várias escalas de comprimento e exploram a simetria foram desenvolvidos para melhorar o corte, aumen- tando a qualidade da imagem capturada. A figura 19 apresenta uma técnica desenvolvida para uso em seccionamento [11].

2.3.1.2 Seccionamento em série na reconstrução 3D

Na reconstrução 3D, imagens bidimensionais obtidas por seccionamento em série são montadas para criar uma representação volumétrica do objeto. Esse processo exige aten- ção na correção de distorções e no alinhamento preciso das imagens, utilizando métodos de processamento, como registro por correlação cruzada [13]. O alinhamento preciso das imagens digitais é fundamental para uma representação bem próxima a real ao objeto original.

Vários algoritmos foram desenvolvidos para o avanço das representações mais fieis [12]e com menos distorçãoes na no volume tridimensional criado. Vários métodos criados, como a de interpolação entre as seções, médias, entre outros são usados para suavisar cada processo e que se crie um modelo contínuo. Ourtos métodos usados com base em Machine Learning estão começando a ser estudados para que haja a aprimoração na montagem do seccionamento para gerar a representação tridimensional, com a redução no tempo de processamento e na melhora da qualidade da reconstrução.

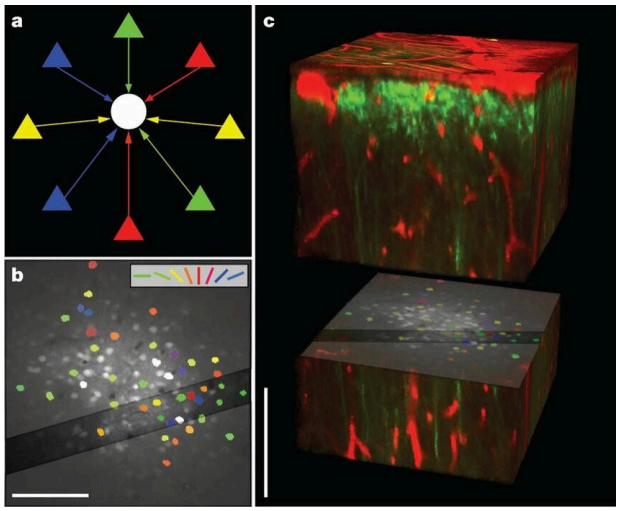

Figura 20: Caracterização funcional de neurônios antes da reconstrução anatômica[12]

Na figura 18 , é mostrada uma reconstrução em série de um cortéx que demonstra a funcionalidade dos neurônios. Uma análise quantitativa de seccionamento serial esta permitindo a representação tridimencional mais detalhada de muitas microestruturas, as- sim fornecendo vários percepções que não são possíveis apenas com seções bidimensionais. Esta técnica pode apresentar uma ampla gama de parâmetros microestruturais [12] é in- dependente da microestrutura da amostra. Na figura 21 é mostrado uma das técnicas usadas para a reconstrução tridimensional, na qual é a técnica de alinhamento elástico.

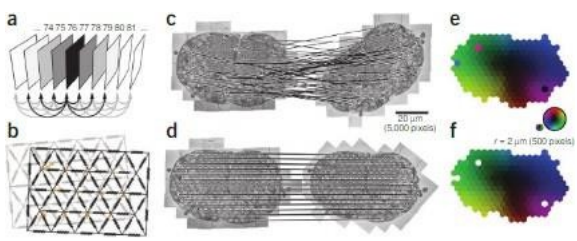

Figura 21: O método de alinhamento elástico [13]

2.3.2 Fundamenos da Reconstrução 3D

A reconstrução tridimensional possui diversas aplicações, incluindo realidade virtual, medicina e engenharia. Esse processo envolve a criação de modelos 3D a partir de dados bidimensionais, como imagens e esboços, e tem avançado significativamente nos últimos anos. A modelagem 3D permite o estudo detalhado de microestruturas, viabilizando investigações aprofundadas e inovadoras em vários campos, dependendo da técnica utili- zada.

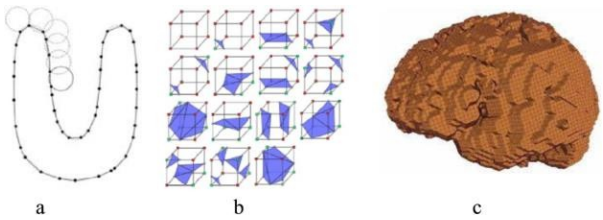

Figura 22: Algoritmo conectando pontos para a formação de um objeto 3D[14]

A figura 22 [14], demonstra bem a técnica sendo aplicada no ramo da medicina usando o método de aplicação de dados para uma melhor eficácia na construção tridimencional feita por eles em seu estudo.

A reconstrução 3D converte dados bidimensionais em um formato tridimensional. Esse processo inclui várias etapas, como aquisição de dados, segmentação, registro de imagens digitais, interpolação e renderização. Se todas as etapas forem realizadas corretamente, obter-se-á um volume tridimensional de alta qualidade.

2.3.3 Aquisição de Dados

Uma das primeiras etapas para esse processo é a aquisição de dados, que depende do tipo de segmento de imagem, exemplos como scanners, câmeras digitais, tomografia com- putadorizada, entre outros. Cada método escolhido tem suas vantagens e desvantagens e pode ocorrer de ser limitada para alguns casos. Um exemplo de um dos métodos é a Fotogrametria, uma técnica antiga e empregada, em muitos casos, na arquitetura. Esta técnica [15] consiste na captura de uma série de fotos sobrepostas, que obtêm algumas pro- priedades requeridas que possibilitam uma reconstrução 3D. O scanning de vários objetos, como casas, pessoas, jóias requer muitos tipos de câmeras diferentes, vários parâmetros de configuração e posição espacial.Uma desvantagem [15]é a forma de informações precisas sobre a manipulação da câmera quando os dados são colhidos, o quee pode torna-se um

trabalho mais desafiador.

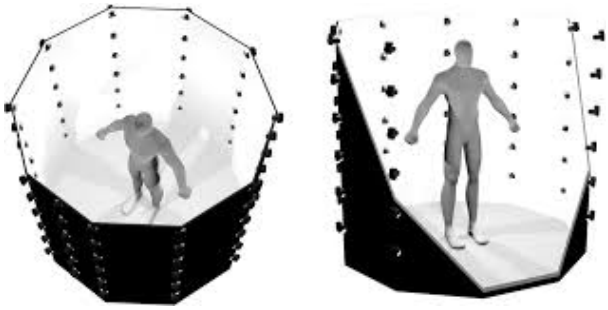

Figura 23: Scanning baseado na técnica de fotogrametria [15]

A coleta de dados deve ser feita com bastante cuidado, para que haja informações bem precisas das imagens geradas e, com isso, ter um objeto 3D bem fiel a realidade, como mostrado na figura 23 [15] [13].

2.3.4 Segmentação

A segmentação é a parte de identificar e isolar as diferentes estruturas de interesse dentro das imagens adquiridas. Esta parte do processo é fundamental para a definição das fronteiras de cada parte do objeto que está em reconstrução. Muitas das técnicas existentes de segmentação envolvem alguns métodos manuais, onde ocorre a delimitação por parte do profissional até podendo ir para métodos automatizados, como em *machine learning* que utiliza algoritmos para tal função.

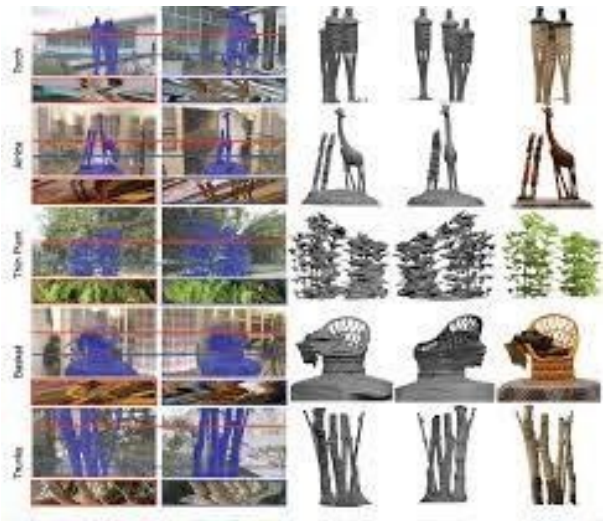

Figura 24: Reconstrução 3D feita a partir de segmentação manual [16]

A figura 24 há uma reconstrução feita a partir do método de segmentação [16],

utilizando-se de algoritmos que aproveitam uma dosagem alta de espaço angular de cam- pos de luz densos a favor da segmentação eficiente de objetos em primeiro plano, com fundos desordenados, assim, melhorando a qualidade da reconstrução volumétrica. Um exemplo de segmentação automatizada é o uso de um método que reconstroí. superfícies tridimensionais a partir de dados de nuvens de pontos incompletos , onde preserva-se a característica e o detalhe geométrico do objeto e não exigindo muita intervenção humana [39]. Há uma rede neural que vai se integrar a um mecanismo de atenção de borda para a melhora na reconstrução, principalmente nas áreas de contorno, que são mais críticas. Este método proposto [39] tem diferenciação por uma preservação maior das bordas e no fornecimento de resultados bem fieis aos reais, o que se destaca de outros métodos propostos. Essas são algumas técnicas apresentadas em recentes psquisas na área.

2.3.5 Registro de Imagens

O registro de imagem é um processo comum na reconstrução tridimensional, principal- mente em áreas da medicina e em pesquisas biológicas. Esta parte envolve o alinhamento das imagens em um único modelo 3D. Este processo é muito importante nas reconstru- ções anatômicas, nas intervenções guiadas por imagens, entre outros. Esta etapa é bem desafiadora, pois envolve uma coleta grande de dados, pode ocorrer pequenas variações no alinhamento, se o processo não for bem feito, o que pode resultar em reconstruções artificíais o imprecisas. Na figura 25 é mostrado uma da técnicas utilizadas de registros não rígidos [17] que usa tranformações rígidas combinadas resolvendo deformações não rígidas, descontinuidade de tecidos e alterações na escala da reconstrução, que parte de seções seriais.

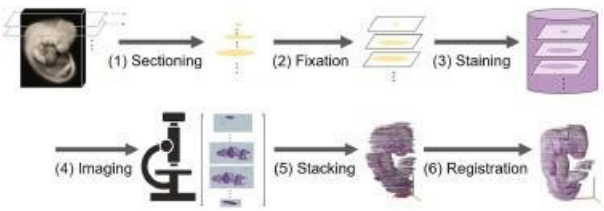

Figura 25: Cada processo de reconstrução 3D a partir de seções seriadas: (1) cortar as amostras em seções finas; (2) fixar as seções em uma lâmina de vidro; (3) corar as seções;
(4) fazer a imagem das seções usando microscopia; (5) empilhar as imagens para gerar um resultado não suave devido às distorções durante os processos (1) a (4); (6) alinhar as imagens usando registro para gerar um resultado de reconstrução 3D suave. O resultado do método proposto é mostrado.[17]

Um método de registro é o registro de imagem difeomórfica, que utiliza algoritmos para oferecer soluções robustas de registro de imagens 2D e 3D. Esse método resolve o problema de registro não linear por meio de otimização local e métodos de *MultiGrid*, assegurando uma codificação precisa [40]. Por ser totalmente automático, apresenta alta precisão. Além disso, a combinação de diferentes modalidades de imagem e o uso de técnicas avançadas, como aprendizado de regressão e aceleração de GPU, aumentam ainda mais a precisão e a eficiência no registro de imagens 3D.

2.3.6 Interpolação

A interpolação é o processo de gerar dados intermediários para preencher lacunas entre seções adquiridas, resultando em um volume contínuo. Esse processo é crucial para suavizar as imagens e corrigir espaços entre elas. Existem vários métodos disponíveis, como:

- Um dos métodos conhecidos no meio acadêmico é o de interpolação bilinear [33].A interpolação bilinear é uma técnica que considera os valores dos quatro pixels mais pró- ximos ao ponto de interesse em uma imagem 2D para calcular um novo valor de pixel;

- Outro método também usado, segundo é o da interporlação trilinear [41] onde são utlilizados quatro pontos, em uma malha de dois, para a interpolação de valores;

- Há também a interpolação spline [18], cujas as funções são mais fáceis para a interpolação contínia de valores, usada para superfícies mais complexas, um exemplo de desenvolvimento da interpolação spline é representado pela figura 26:

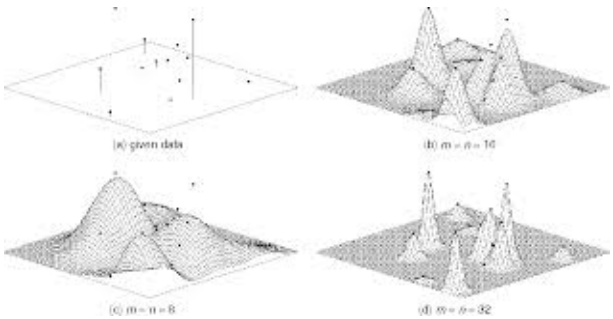

Figura 26: Exemplos de aproximação B-spline sob diferentes resoluções de rede de controle [18]

- Outra interpolação é por vizinhaça natural [42],que é calculado os pontos próximos um do outro, para o cálculo da interpolação.

Existem outros tipos de métodos de preenchimento das partes vazias, porém não serão abordadas nesta pesquisa.

2.3.7 Renderização

A parte da renderização é onde pode-se ver, através de um software, o objeto 3D reconstruído. Essa é a parte final do processo.Técnicas de renderização incluem a rende- rização por volume, onde se vê a estrutura formada internamente e a de surperfície, que permite ver a rescontrução externamente, A escolha do tipo de programa a ser utilizado depende do objetivo do estudo [43]. Hoje, vê-se muitos desafios no campo da reconstrução 3D, com pesquisas com *deep learning*, *machine learning* para melhorar o tempo de pro- cessamento e precisão dos algoritmos e [43] com a computação em nuvem e novas GPUs têm-se permitido o maior números de dados possíveis para a recriação tridimensional.

Capítulo 3

Metodologia

Este capítulo aborda a construção da pesquisa, incluindo os métodos, ferramentas e equações matemáticas utilizadas. Optou-se pelo método de seccionamento em série para a coleta de dados das imagens 2D, com apoio de capturas fotográficas e ressonância magnética. Para o desenvolvimento do código, foi escolhida a linguagem Python [23], com o uso de bibliotecas gráficas como OpenCv [20] e Tthinker [21].

3.1 Seccionamento em Série

O seccionamento em série foi realizado em imagens reais usando técnicas para ob- ter dados bidimensionais, os quais foram utilizados para reconstrução tridimensional. As imagens foram empilhadas cuidadosamente, garantindo que todas tivessem a mesma reso- lução, de modo que cada *pixel* estivesse alinhado, resultando em uma imagem 3D uniforme e sem deformações. O programa não aceita imagens com resoluções diferentes. Para esse procedimento, foram utilizadas uma câmera digital e ressonância magnética. Além disso, o código foi testado com imagens de um cérebro humano, obtidas por ressonância magné- tica, que totalizaram 176 imagens. A ressonância magnética utiliza um campo magnético forte e pulsos de radiofrequência, com um sistema de detecção, para gerar imagens de alta resolução dos órgãos humanos.

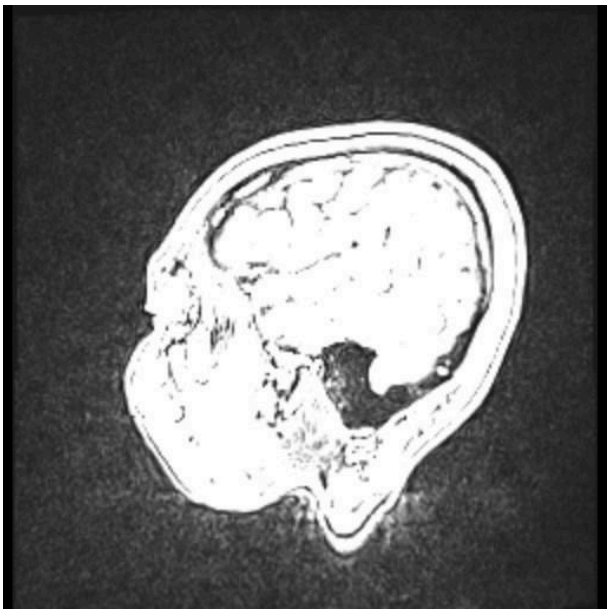

Figura 27: Exemplo de uma foto de RM usada nesta pesquisa

3.2 Desenvolvimento do Código

O objetivo do desenvolvimento deste código foi melhorar a interface gráfica e o có- digo de uma construção de uma mesma pesquisa nesta linha, feita em C++ [1], feita anteriormente e que um de seus futuros trabalhos é a melhoria em seu código e interface.

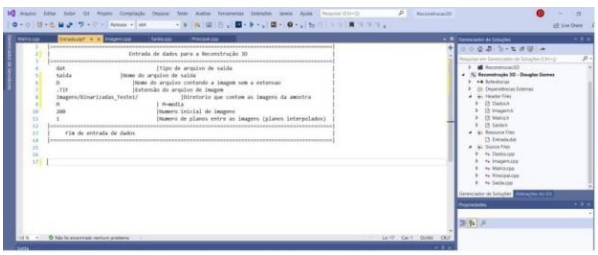

Figura 28: Interface feita em C++

- O código foi desenvolvido usando a Linguagem de Programação Python 3.12.5 [23], na qual é uma das linguagens mais utilizadas, hoje em dia, na área de ciências de dados, aplicações em web, desenvolvimento de softwares, entre outros;
- OpenCV[20] é uma biblioteca para o uso de processamento de imagens computacio- nais para a reestruturação tridimensional, sendo uma biblioteca de uso livre, desenvolvida pela Intel em 1999 [3];

- Tkinker [21]é uma biblioteca para criação de interfaces gráficas básicas, onde a importação desta biblioteca possibilita a construção de janelas, botãos e etc;

- Construção dos modulos.py para, usando as bibliotecas citadas acima, além de outras para o cálculo de matriz, vértices, de construção de preenchimento entre cada seção, entre outros. Essa estrutura é usada para a escrita de um programa extenso, pois facilita no entendimento na escrita. Estes módulos se conversam para a execução do programa;

- Houve a construção do código sem a estrutura "Classe"para a comparação de resul- tados em tempo de processamento e consumo de memória RAM.

3.3 Estruturação do Código

Nesta parte conterá a metodologia usada para a criação de cada módulo e como o programa conversa entre si para o seu funcionamento.A estrutura do código é mostrada pelo fluxograma:

Figura 29: Fluxograma da estrutura do código

3.3.1 FormatArq.py

Este módulo é dedicado para criação de cada cabeçalho para a criação de formatos de saída .dat e .vtk para o uso em softwares de leitura desses tipos de arquivo que convertem em imagens gráficas.

O primeiro cabeçalho é para criação de um arquivo básico .vtk. Este cabeçalho é defi- nido por 'cabecalhoVTK(tamanho)'. No código construído, ele tem como funcionamento a definição da saída .VTK que será gerada depois da execução do programa. Nesta parte contém a informações de dados, tipo do arquivo, seu título e o seu tamanho.

O segundo cabeçalho, 'cabecalhoVTK2()', é definido para quantas faces e conexão com seus vértices, o responsável pelas informações do polígono que comporá a reconstrução.

O 'cabecalhoVTK3(tamanho) é feito para receber os dados dos pontos, indicando os tipos de pontos que serão usados. No 'cabecalhoVTK4(tamanho) indica a string

para a tabela de pesquisa para a mapeação do valor numérico que compõem cada cor. No

'rgbVTK(r, g, b) converte cada cor para números binários (0 ou 1), desde o valor RGB 0 ao 255. Retorna a string no formato .vtk, como os valores de suas cores. O último cabeçalho é relacionado ao .dat, o 'cabecalhoDAT(i, j, k) gera o formato para Tecplot, onde se compõem as variáveis x, y, z e a intensidade e cada cor formada. Foi feita as definições de alinhamento, para que cada casa decimal que seja diferente fique em alinhamento.

3.3.2 Modulos.py

Esta estrutura de código é responsável pela importação das principais bibliotecas que serão utilizadas para o funcionamento do programa como um todo. Algumas bibliotecas usadas são:

- 'from sys import exit': esta biblioteca é responsável pela parada do processamento do programa caso ocorra um erro grave, crítico;
- 'from tkinter import * ': incorporação da biblioteca, usando o comando de instalação pip install tK. Esta biblioteca é responsável pela interface gráfica que será gerada na tela, como as janelas e os botões, que serão importantes para a entrada dos dados externos que serão recebidos;
- 'import cv2': essa é a biblioteca onde o OpenCV é aplicado. Para o uso desta biblioteca, é preciso baixá-la pelo site da própria empresa fornecedora, descompactar os arquivos baixados e instalá-la no computador. Importando a cv2 para o programa, ela vai ser responsável pelo processamento das imagens digitais;
- Ocorre, neste módulo, a importação dos módulos FormatArq e Class para a interação de cada código, que consiste na manipulação dos formatos de saídas e de classes usadas para o processamento de imagens que serão exportadas para serem manipuladas para a saída escrita em FormatArq.py.

3.3.3 Class.py

Este módulo é responsável pela leitura de cada imagem processada, na qual elas passarão uma vez por ela, que é composta pela lista 'mat', o formato em matriz das imagens. É uma classe 'read', definida para a leitura das informações que serão dadas pelo usuário. Antes da definição da classe, este módulo importa as bibliotecas que foram incorporadas em 'Modulos.py'. Elas são necessárias para o funcionamento da classe 'read'.

No primeiro código feito para este teste não contou com esta estrutura.

Esta é uma classe cuja tem uma fuma função única, porém ela é parte do funciona- mento do main.py, na qual é chamada para produzir uma matriz de imagens com suas respectivas informações. São recebidos quatro parâmetros, o *”numimgs”*, "endereço"e "exten", que correspondem ao número "z"de imagens, onde serão armazenadas imagens do nome do diretório, o nome antes de sua numeração e, posteriormente, o nome após a numeração.

Depois deste passo, é utilizada uma sequência de estruturas de ’for’, que é usada para coletar as informações de cada vetor, denominados ’mat’,’vet’ e ’pixel’. O vetor ’pixel’ armazena informações de cada pixel analisado em determinado número de’loop’, o vetor ’vet’ armazena esses dados de pixels de cada imagem analisada. O vetor ’mat’ armazena cada vetor ’vet’ à medida que cada imagem passa por ele. Os ’loops’ são realizados utilizando os ’img.shape[0]’ e ’img.shape[1]’, no qual permite saber a dimensão de cada imagem para definir quando os loops possam ser realizados. Para não ocorrer erros no programa, é utilizada a condição de contorno com ’if’, porque o ’img.shape[]’ se estende ao número máximo de dimensão durante o tempo que ’for’ trabalha de zero até o máximo subtraído das imagens −1.

O código loop de ‘for‘ para coletar, tridimensionalmente, os pixels das imagens, primeiro em relação ao eixo Z (quantidade de imagens), depois pelo eixo Y (repre- senta a variação vertical dentro do plano da imagem) e, ao longo do eixo X (varia- ção horizontal de cada plano da imagem). Cada imagem é lida pela biblioteca opencv ‘cv2.imread(filename)‘,na qual ‘filename‘ representa o caminho completo da imagem, in- cluindo o diretório, nome base, índice (correspondente ao eixo Z) e extensão, e o ’filename’ é o nome da imagem a ser analisada.

Cada leitura da imagem pela função ‘imread‘ retorna um array, que conterá cada intensidade da cor no formato RGB (azul, verde, vermelho) para cada pixel. A função ‘np.array()’, que utiliza a biblioteca NumPy, organizará essas informações em uma matriz 2D, onde cada elemento representa um vetor com as três componentes de cor. Depois essas intensidades serão combinadas para calcular o valor de intensidade do pixel, onde armazenará as informações de cor em um único valor numérico.

O método ‘append‘, um método nativo da linguagem Python, é utilizado para adici- onar novos elementos ao vetor, criando um espaço novo para receber dados.

3.3.4 main.py

Nesta seção será mostrado como foi realizada a parte do código principal, como a sua interface e as interpolações feitas em cada parte do preenchimento tridimensional.

3.3.4.1 Interface Gráfica

A interface gráfica foi desenvolvida para a interação com o usuário para o armazena- mento de arquivos dados que o próprio fornecerá, para que depois gere a saída requerida. Nesta parte foi utilizada a biblioteca 'tkinter', na qual proporciona os recursos para a cons- trução de janelas e botões. Essa biblioteca teve um acesso diferente, para poder acessar todos os métodos como se fossem funções do código principal, feito assim para economizar memória, o 'Try' para acessar e executar o 'tkinter', senão segue para o caminho 'except' que informará um erro com o uso da biblioteca.

A continuação da escrita do código é a criação de uma simples janela que tem instância no objeto 'janela', onde 'tk()' , representando a biblioteca tkinter, que foi utilizado pelo objeto 'janela'. Em seguida é definida as dimensões dessa janela, o título, o local aonde ela irá ficar e utilizou-se uma forma de plano de fundo com cor que envolverá toda ela, como 'Frame()' e 'Label()'.

Código 3.1: Código da janela

```
1  from Modulos import *
2
3  # cores
4  preto     =
5  branco  =
6  cinza = "#7 F7F7F "
7  azulEscuro Alto  = "# 092642 "
8  azulEscuro Medio  =  "# 174276 "
9  azulEscureo Baixo = "#235 A9F "
10 azul = "# 007804 "
11 azulClaro Baixo = "#0078 D4 "
12 azulClaro Medio  =  "#80 B9EE "
13 azulClaro Alto = "# ADD8FF "
14
15 # Configurando a janela
16 janela  = Tk ()
```

```
17 janela . title (" R e c o n s t r u              o 3 D")
18 janela . geometry (" 600 x394 ")
19 janela . config ( background =" gray ")
20 janela . wm_maxsize ( width =600 ,          height =396)
21 janela . wm_minsize ( width =600 ,          height =396)
22
23 # A p a r n c i a da janela
24 frame_tela = Frame ( janela , width =600 , height =60 , background =" gray ")
```

Depois foi definida a ’stringVar()’ e a ’IntVar()’ para usar textos e números inteiros, respectivamente. E com ’set()’, foram pré-estabelecidos valores de exemplo para parâme- tros de entrada.

Código 3.2: Implementação Python do Método Original

```
frame_marc  = Frame ()
frame_marc . place ( x=62 , y =229)

# criacao de um frame para simplificar o posicionamento das o pe s para escolha do
frame_marc2  = Frame ()
frame_marc2 . place (x=200 ,  y =43)

# V a r i v e i s  e input
tipo_arq_saida  =  String Var ()
""" ex: . dat . vtk """

nome_arq_saida = String Var ()
""" Nome  do  arquivo  de  s a  d a  sem  o  tipo  do  arquivo """

name = String Var ()
""" Nome  do arquivo  antes  do  n m e r o """

ext  =  String Var ()
""" Nome do arquivo          a p  s  o  n m e r o """

Endereco_Img  =  String Var ()
""" Local  da  pasta  que  c o n t m  as  imagens """

metodo_ = String Var ()
""" M  t o d o  escolhido  para  trabalhar  as  imagens """

Num Img Total = IntVar ()
""" N m e r o de imagens  originais """

plan_entre_img  =  IntVar ()
""" N m e r o de imagens interpoladas , ou seja , criadas pelo programa """
```

Para o usuário inserir as suas informações será pelos retângulos que aparecerão na interface e com texto editável,que será criada pela função 'entrada()' e com botões de marcação de suas escolhas, que será criadas pelas funções 'opções()' e 'opções-tipo de

arquivo()'. A classe de entrada 'Entry()' edita e posiciona cada retângulo, enquanto a classe 'CheckButtom()' definirá as opções de valores correspondentes a cada opção.

Código 3.3: Valores pré estabelecidos

```
1
2
3  # P r      - estabelece valores para testes ou e x e m p l i f i c a                    o
4  tipo_arq_saida . set(". vtk ")
5  nome_arq_saida . set(" Micro ")
6  name . set(" Sem  Titulo -")
7  ext. set(" ( Copy ). tif")
8  Endereco_Img . set(" Imagens/ Imagens 2 /")
9  m etod o_. set("o")
10 Num Img Total. set(" 2 ")
11 plan_entre_img . set("4 ")
```

No algoritmo a feito, foi utilizado balões que recebem o texto que criará o usuário o que deve ser feito na interface, utilizando a função 'BalaoEntrada()', a 'disty' que receberá qual a distância em relação ao topo da janela e 'comp' que indicará o tamanho do balão na interface horizontalmente.

No final da criação do código, apresenta botões com coloração verde e vermelha, o primeiro usando o comando de 'executar()', que irá começar o processo de reconstrução tridimensional, utilizando os dados que o usuário informou, e o seguexit(), quando o botão de sair é selecionado 'EXIT', o que encerrará todo o processo da ferramenta 'tkinter', incluindo a janela e a execução do programa. Também foi utilizado o 'janela.mainloop()', na qual é utilizado o loop infinito para que a janela não desapareça na tela enquanto não houver o clique no botão 'EXIT'.

3.3.4.2 Métodos de Execução

A uma função atrelada ao botão 'Executar'. Ao utilizar essa função, iniciam-se as variáveis a serem utilizadas para a criação do arquivo de reconstrução, 'r','g','b','x','y','z' são as variáveis, onde 'r','g','b' são as variáveis de cor, 'x','y' são variáveis que demonstram cada posição de cada pixel e 'z' mostra qual imagem está interagindo com o programa.

Porém, o programa verifica se as dimensões das imagens são iguais usando o 'img.size' que retornará 1 caso não for acessada. Para não seguir com o erro é usada 'sys.exit()', então a variável 'z' é zerada e reutilizada durante outras etapas do código.Para essa

análise, é utilizada uma estrutura de repetição ‘for‘ que, por meio da variável ‘filename‘, acessa a imagem correspondente ao valor de ‘z‘ e armazena suas informações no vetor ‘vet-de-img‘. Em seguida, são executadas duas estruturas condicionais ‘if‘,o primeiro é demonstrado para na qual se ocorrer algum erro, como endereço ou nome da extenção não identificadas,o programa é abortado, enquanto a segunda verifica se a dimensão da imagem atual é igual à da imagem anterior, utilizando o valor da variável ‘img.shape‘, armazenada na posição 4 do vetor de informações das imagens.

Em seguida é utilizada a estrutura ’mat.case’, no qual define em qual caso e que tipo de sáida de dados o usuário selecionou para a execução do programa.Esta parte avalia o que o usuário informou para a variável ’metodo-’ usando a função ’get()’, no qual retorna o valor da variável ’metodo-get()’.

Código 3.4: Implementação Python do Método Original

```
1
2
3   case  "0 " | "o":              # Caso Original
4                       print(" Dado  m t o d o  original")
5                       linhas , colunas , cores  = img. shape                    # O b t m d i m e n s e s
6                       print(" Processando          imagem ...")
7
8                       #  Cria  arquivo  de           s a  d a
9                       nome_arquivo = f" Exportacao /{ nome_arq_saida . get()}{
                            tipo_arq_saida . get()}"
10                      with  open ( nome_arquivo , "w")  as meu_arquivo :
11
12                          # Escreve c a b e a l h o Tecplot
13                          cabecalho  =  formatacao . cabecalho DAT (
14                                  colunas ,
15                                  linhas ,
16                                  Num Img Total. get()
17                          )
18                          meu_arquivo . write ( str( cabecalho ))
```

- 'Case O': Utiliza as imagens originais para serem feito a reconstrução tridimensional sem qualquer cálculo;

- 'Case V': Insere imagens vazias, no qual não contémm nenhuma informação, entre as imagens originais para criar uma separação en cada plano, se houver necessidade;

- 'Case R': Reutiliza a última imagen lida original para o preenchimento entre as partes vazias das imagens, replicando os dados nas partes que não há informação. Para usar o método de repetição de planos, é usado contadores no código main.py, para ocorrer a repetição da cor do plano inferior até o plano superior. Os contadores indentificam o término de cada plano e depois o segundo é usado para repetir o estado do plano anterior [1];

- 'Case 'M': cálcula a média a cada par de imagens consecutivas, utilizando algum método para a interpolação, criando uma transição entre as imagens que preservem a sua continuidade na reconstrção 3D.

Para a obtenção desse valor basta somar os valores de R, os valores de G e os valores de B e fazer a divizão de cada um por dois. O resultado é uma nova cor que será a

média entre as duas cores da imagem, demonstrada na figura 32. Para fazer essa técnica, o primeiro passo é empilhar as imagens, que são os dados informados pelo usuário, As partes vazias [1] passam a ser preenchidos utilizando o valor da equação 4.1, que é a taxa de variação de uma função, pois so há um plano intermediário entre as duas imagens

$$T_{V_{canal}} = \frac{f_{canal}(x_k, y_k, z_{k+1}) - f_{canal}(x_k, y_k, z_k)}{k+1 - z_k} \quad (3.1)$$

,onde $f(x, y, z)$ é uma função de três variáveis que representam R,G,B respectivamente. A variável *canal* deve ser preenchida por R,G e B separadamente, para cada um a taxa de variação é calculada para diferentes valores de k, onde $k = 1, 2, \ldots, n - 1$ e n é o número de planos que entrarão. A taxa 'TV' mede a distância entre valores de RGB [1], dependendo da quantidade de planos a serem construídos para a reconstrução tridimensional.

R	G	B	Cor
255	255	255	
213	213	213	
170	170	170	
128	128	128	
85	85	85	
43	43	43	
0	0	0	

(a) Branco e preto.

R	G	B	Cor
127	127	127	
129	106	109	
130	85	92	
132	64	74	
133	42	56	
135	21	39	
136	0	21	

(b) Cinza e vinho.

R	G	B	Cor
237	28	36	
240	64	30	
243	99	24	
246	135	18	
249	171	12	
252	206	6	
255	242	0	

(c) Vermelho e amarelo.

Figura 30: Tabela de média de cores: a) Branco e Preto; b) Cinza e Vinho; c) Vermelho e Amarelo [1]

As coordenadas x e y são fixadas ao longo do plano, enquanto z demonstra o plano específico, variando conforme o algoritmo repassa os planos principais. A diferença entre z_{k+1} e z_k é sempre o número de planos intermedários mais um [1].

3.3.4.3 Tipos de Métodos de Saidas

Para esta parte, o arquivo main.py usa o cabeçalho do FormatArq.py para gerar os tipos de saídas das informações processadas, o que depois irá para a redenrização.

- Saída '.dat' utilizando o método original: Esta parte do código é descrito o desen- volvimento do código para ler imagens e gerar um arquivo '.txt' contendo as informações das imagens. Para isso, foram usadas as bibliotecas Numpy e OpenCV, que facilitam o processamento das imagens. O código lê as dimensões da imagem, como linhas, colunas e cores, e escreve as informações no arquivo;

- Saída ’.dat’ utilizando o método vazio: Aqui são inseridas algumas imagens vazias entre as imagens originais. O código é parecido com o modelo do método original, com modificações para controlar a criação dessas imagens vazias. Variáveis auxiliares, como ’vazio’, ’cont’ e ’aux3’, ajudam a gerenciar a inserção das imagens;

- Saída ’.dat’ utilizando o método de repetição: Esta parte do código insere imagens repetidas entre as imagens originais em vez de imagens vazias. A lógica é parecida com a do método anterior, mas neste caso, a imagem anterior é repetida nos espaços entre as imagens originais. Variáveis como ’aux4’ e ’Repet’ são usadas para controlar o processo de repetição;

- Saída ’.dat’ utilizando o método da média: Neste código a imagem interpolada entre duas imagens originais, onde foram calculadas como a média das cores dos pixels dessas imagens, utilizando a equação 4.1. O processo envolve cálculos de interpolação linear para as cores (RGB), criando gradualmente as imagens intermediárias;

- Saída ’.vtk’ utilizando o método original: A estrutura do código é parecida com o da saída ’.dat’, porém o formato de saída do arquivo é diferente. No formato ’.vtk’, além das informações de cor, são incluídas as coordenadas dos pixels no espaço 3D. O código gera as coordenadas (x, y, z) e as escreve no arquivo junto com as informações de cor;

- Saída ’.vtk’ utilizando o método vazio: O mesmo processo da saída ’.dat’, onde são inseridas imagens vazias entre as originais. As coordenadas 3D dos pixels são geradas e escritas no formato ’.vtk’, e as imagens vazias têm cores e intensidades zeradas;

- Saída ’.vtk’ utilizando o método de repetição: O mesmo processo da saída ’.dat’, onde são inseridas cópias das imagens originais entre elas, repetindo as coordenadas e as cores dos pixels no arquivo ’.vtk’. As variáveis ’aux3’ e ’aux4’ controlam a alternância entre imagens originais e repetidas;

- Saída ’.vtk’ utilizando o método da média Semelhante ao método da média na saída ’.dat’, onde são calculadas as imagens intermediárias como médias das imagens originais adjacentes, utilizando a equação 4.1. As coordenadas 3D são mantidas, e as cores interpoladas são calculadas para cada pixel.

3.3.5 Como funciona o programa

Main.py, sendo o programa principal, recebe as informações das imagens utilizando a interface gráfica, que aparecerá para que o usuário coloque seus dados de

imagens

para serem feitas a reconstrução 3D , elas passam pela class.py, para serem lidas todas as imagens e organizar todas as informações e uma matriz, que será utilizada em algum ’case’ quando chamada.O modulos.py é chamada na main.py para importação das bibliotecas que serão utilizadas no ’case’ que foi escolhido pelo usuário e o formatArq.py interage com a main.py para a chamada do cabeçalho de cada tipo de saída que o usuário escolheu para gerar o arquivo saída.

3.4 Renderização

Para que haja uma leitura de um arquivo de saída ’.dat’ ou ’.vtk’, preciza-se de um programa que possa ler e demonstrar se o código realmente gerou uma reconstrução tri- dimensional. Para isso, foi escolhido o programa ParaView, que demonstrará o resultado do código feito.

Figura 31: Logo do Paraview

O computador utilizado tem as seguintes configurações:

O Positivo Motion Q4128C-S.

- Processador: Intel Celeron N3350;
- Memória RAM: 4 GB;
- Armazenamento: 128 GB HDD;
- Tela: 14 polegadas, resolução HD (1366 x 768);
- Sistema Operacional: Windows 11;
- Conectividade: Wi-Fi 802.11 b/g/n, Bluetooth;
- Portas: USB 3.0, USB 2.0, HDMI, leitor de cartões SD;
- Bateria: Autonomia variável, geralmente em torno de 4 a 6 horas dependendo do

uso.

Computador do Laboratório de Materiais possue as seguintes configurações:

- : NMM04
- : Processsador: Intel(R) Core (TM) i7=3770k CPU @3,50 Ghz;
- : Memória Ram: 32,0 GB;
- : Sistema: Windows 10 Pro 64bits.

Capítulo 4

Resultados e Discussões

Neste capítulos desmonstrá-se os resultados das pequisas quanto as análises feitas durante este período.

4.1 A Interface Gráfica

Para a utilização do programa, deverá conter o programa VScode instalado no com- putador para a inicialização do processo. Este programa pode ser baixado pela loja da Microsoft. O OpenCV também precesirá ser baixado e instalado no computador para o funcionamento e instalação da biblioteca no código feito, ele pode ser achado e baixado pelo próprio site do programa. A utilização do programa de linguagem Python e indispen- sável e por isso deverá ser instalado também. Para a renderização da saída do programa, o Paraview foi instalado neste computador por via de download feito do próprio site da fabricante.

Primeiro precisa-se criar um ambiente virtual, para isso é necessário a abertura do terminal no VScode e digitar: python m -venv e nome do ambiente (pode ser o que desejar, desde que não contenha acento ou caractere especial). Depois deste passo, aparecerá a sua pasta, com o nome que você a deu e incluindo outras pastas nela (Lib,Include,Script) e um arquivo criado chamado ’pyvenv.cfg’. Eles são necessários para que o programa rode completamente sem erros. Assim o ambiente está criado.

Agora, abra o terminal do cmd no vscode mesmo, em outras palavras, clique na setinha para baixo ao lado do "+"que fica ao lado do "powershell"e clique em "Command Prompt". Em seguida, verifique se está aberto na mesma pasta, caso contrário você precisará abri-lá e digitando .-do-ambiente.No terminal vai mostrar a mensagem do local

que você está acessando entre parênteses o nome do ambiente de trabalho. Ex: (nome- do-ambiente) C:4 Se aparecer algo semelhante ao exemplo com o ambiente de trabalho entre parênteses significa que o ambiente de trabalho foi criado com sucesso.

Depois da criação do ambiente, precisará ter as instalações das bibliotecas para o funcionamento do processo. Utilize o terminal do Powershell para a sua instalação. As bibliotecas são: pip install tk, pip install opencv-python e pip install numpy. Desde que se siga estes passos, o programa está pronto para funcionar.

Ao rodar o programa no VScode, caso não ocorra erros, uma janela irá ser aberta em seu computador

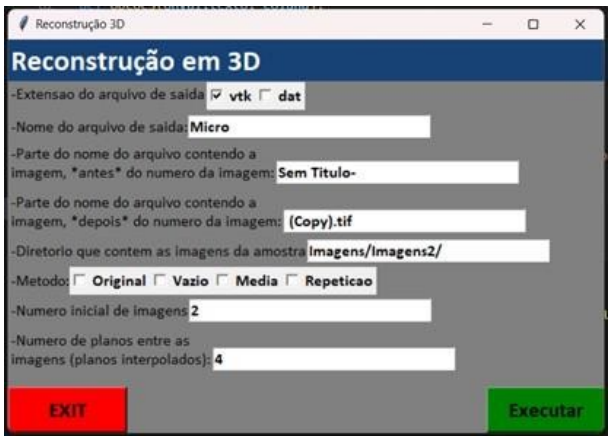

Figura 32: Janela para inserir as informações

Na figura 32, ela já vem com um padrão de preenchimento para guiar o usuário em qual lugar se deve inserir as suas informações. Escolha-se um tipo de arquivo (.dat ou .vtk), o nome de saída do programa, o nome das imagens que constam no diretório (nomes devem está padronizados, sem parênteses nos números e a contagem das imagens começando com 0, exemplo R-0). O tipo de extensão que as imagens possuem (.jpg,.tiff,.png) o diretório que contém esta imagem, o tipo de interpolação a ser escolhido pelo usuário, o número de fotos que contém no diretório e o número de planos a serem contidos na interpolação.

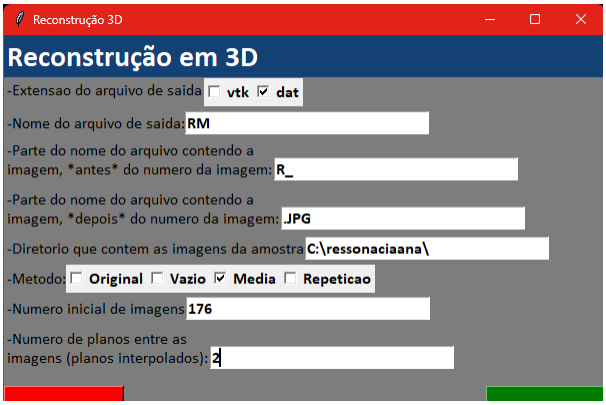

Figura 33: Janela preechida com as informações dadas para o usuário

De acordo com a figura 33, ocorreu a inserção de dados pelo usuário para o programa, informando o tipo de arquivo de saída que ele quer,o seu diretório de imagens e o modo que ele escolheu para que o processo de reconstrução 3D ocorra.

4.2 Execução do Programa

Depois, com o programa de posse das informações, ele fará a leitura das imagens, sua verificação e seu 'case' escolhido para a reconstrução tridimensional. Essa reconstrução está sendo testada com o método de seccionamento RM.

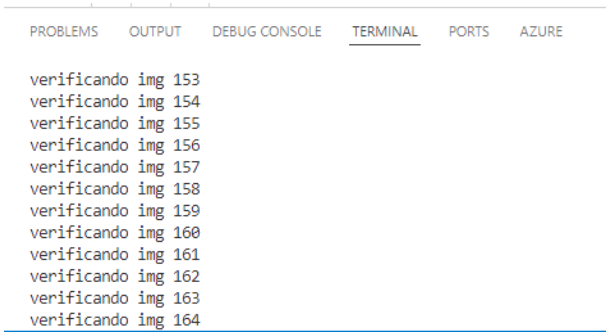

Figura 34: Verificação das imagens sendo feita pelo programa

Essa parte, mostrada pela figura 34, mostra a verificação feita de cada dado, pois o programa depende que todas as imagens tenham a mesma dimensão para que ele possa fazer a reconstrução, pois cada pixel deve estar alinhado com cada imagem,e em dimensões diferentes de fotos isso não será possível.

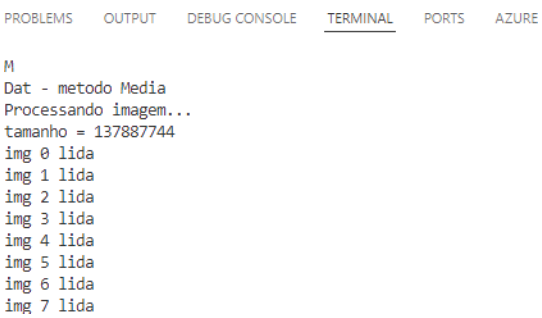

Figura 35: Leitura das imagens

Após a verificação, o terminal exibirá o tipo e o tamanho da saída, além do método escolhido. Como indicado na figura 35, neste momento são lidas as informações de cada imagem, que serão armazenadas em vetores, gerando a ’mat’, que será convertida no ’case’ média .dat.

Ao final do processo, o terminal mostrará uma mensagem de saída concluída, o tempo de processamento e o nome do arquivo gerado. Este arquivo será salvo na pasta Exporta- ção do código, onde será utilizado para a renderização final. A figura 36 ilustra o resultado do processamento do arquivo, e o apêndice A apresentará a demonstração de cada etapa da renderização das imagens tridimensionais.

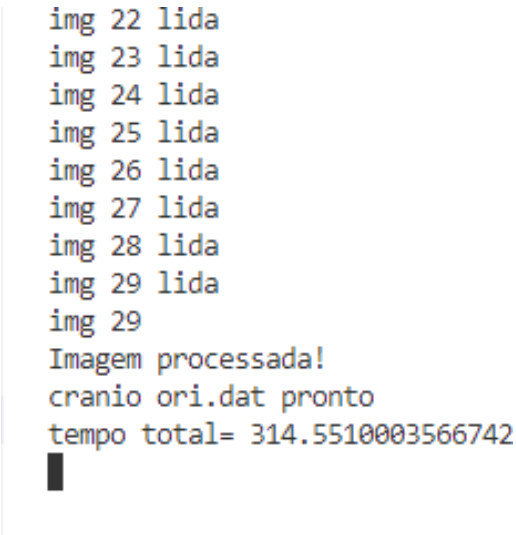

Figura 36: Tempo de processamento e nome do arquivo de saída

4.3 Comparação de Método de Reconstrução

Para fins de mostrar a diferença entre o método de reconstrução tridimensional feito pelo código apresentado, usaremos como exemplo uma figura reconstruída via scannea- mento, apresentada abaixo:

Figura 37: Figura 3D do brinquedo quebra cabeça

A figura 37 mostra uma reconstrução tridimensional de um quebra-cabeça feita por método de scanneamento, esse método é eficaz até certo ponto, pois ele apresenta uma limitação que difere do método de reconstrução usado nesta pesquisa. O objeto não é reconstruído por total, pois este método não consegue captar a parte traseira do objeto esta causando a falta desta parte [15]. A figura 38 mostra esta limitação:

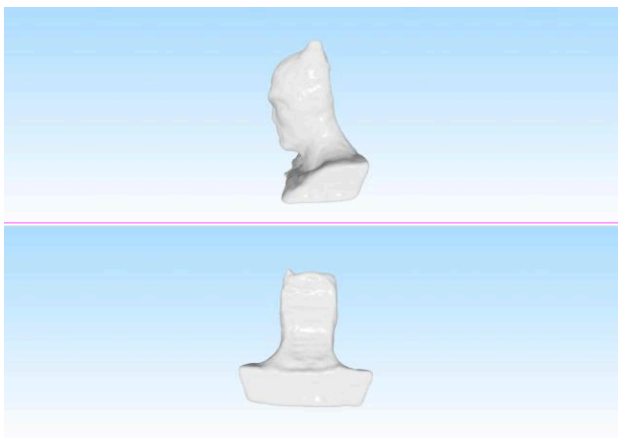

Figura 38: Partes não reconstruída pelo método de scanneamento

Na pesquisa feita, consiguiu-se um resultado melhor que o de scanneamento [15], pois o código construído conseguiu reestruturar tridimensionalmente o objeto usado na pesquisa mais aproximado em todas as partes das seções usadas.

Figura 39: Imagens reconstruídas usando o código da pesquisa

A figura 39 mostra a reconstrução usando imagens de RM com o método da interpolação por repetição [1]. Nestas imagens é possível ver que o objeto é reconstruído por inteiro, o que difere de métodos como o de scanneamento que foi usado anteriormente para este mesmo fim e que pode-se notar uma vantagem neste método de reconstrução via código em linguagem Python em comparação a outros estudos feitos.

4.4 Comparação entre os tipos de interpolações

Durante a pesquisa foi-se usado quatro diferentes tipos de interpolações mencionados no cápitulo 3 deste mesmo documento. Todos eles estão escritos no código e possuem a finalidade de escolha do método a ser usado para que se faça a reconstrução de secciona- mentos em série de acordo com o interesse de cada estudo que será feito.

4.4.1 Interpolação Original

Neste tipo de interpolação, as figuras são empilhadas em ordem e são usadas sem ne- nhuma interferência para o preenchimento entre seus vãos, então ela é usada sem nenhum dado praticamente em cada parte de suas áreas que ficavam livres.

Figura 40: Ângulo visto de frente da parte original

De acordo com a figura 40, a parte do rosto fica de modo achatado, o que demonstra que suas figuras foram praticamente empilhadas sem nenhum dado ou equação para o preenchimento de suas partes para que houvesse uma suavização em sua forma. Em outros ângulos, é possível verificar essa mesma particularidade.

Figura 41: Ângulo de lado do método vazio

Pelo ângulo de lado, mostrado pela figura 41, não aparece tanto esse achatamento da imagem tridimensional pelo método vazio. Ele é bem mais percebido em outros ângulos de sua figura, como na figura 42 :

Figura 42: Ângulo da parte de cima da cabeça

Estas imagens não mostram tantos defeitos em sua concepção como em outros cálculos de interpolações, porém é bem notório que há alguns ruídos em partes de suas áreas e também é percebido as camadas das imagens seccionadas usadas nesta pesquisa.

4.4.2 Interpolação Vazio

Neste tipo de interpolação que insere dados, se necessário, entre cada seccionamento, para que haja um preenchimento em cada separação. Esses dados não contêm valores.

Na figura 43 demonstra a parte da frente do empilhamento de imagens usando este tipo de interpolação:

Figura 43: Ângulo de frente do rosto em método vazio

Na parte frontal do objeto gerado, é possível visualizar os vazios em cada fatia de imagem utilizada na montagem tridimensional, sem que ocorra o achatamento da imagem em relação à interpolação original.

Figura 44: Ângulo de cima da reconstrução usando interpolação vazio

De acordo com a figura 44 fica mais evidente as imagens usadas para montagem tridimensional da pesquisa, porém não aparece tanto ruído em sua reconstrução em relação

a outras formas de interpolação. Esses ruídos são referentes aos pontos que aparecem em sua forma tridimensional.

4.4.3 Interpolação Média

Esse tipo de interpolação usa a equação 3.1 demonstrada no cápitulo 3 [1], Esta equação é usada para fazer a média a cada dois planos que entram no processamento. Esta equação faz a média de cada ponto da imagem usando o método de cor RGB para que a continuidade da imagem seja respeitada e bem fidelizada. A figura 45 demonstra este tipo de interpolação aplicada no código para o preenchimento de cada parte do seccionamento.

Figura 45: Ângulo da reconstrução tridimensional da interpolação média

A interface mostrada na figura 46 apresenta um preenchimento superior, demons- trando maior fidelidade ao tamanho do rosto na pesquisa utilizando RM, embora haja ruídos na região inferior do pescoço (pontos visíveis na figura). Esse método é eficaz para aproximar o tamanho real do rosto, ao contrário da interpolação original, que resulta em uma reconstrução achatada e distorcida, deixando buracos característicos da interpolação vazia.

Figura 46: Ângulo visto de cima da reconstrução da interpolação média

Apesar da suavização obtida com a interpolação média para preencher cada seção, ainda são visíveis as partes das figuras usadas no empilhamento. No entanto, o pre- enchimento das seções é adequado, proporcionando uma noção realista do tamanho do objeto.

4.4.4 Interpolação Repetição

A interpolação por repetição visa utilizar dados da imagem anterior, repetindo cada ponto na imagem seguinte para preencher as lacunas deixadas pelo seccionamento em série. No entanto, esses dados não apresentam um valor médio de cores como na interpolação média, mas sim informações repetidas de dados utilizados anteriormente, sem um valor específico. A figura 48 ilustra um exemplo dessa aplicação.

Figura 47: Ângulo da parte da frente do rosto em interpolação repetição

A reconstrução tem um tamanho aproximado ao de um rosto real, mas sua definição é inferior. Na interpolação média, a definição da reconstrução tridimensional é superior. É possível perceber partes das imagens seccionadas utilizadas pelo código na compila- ção. Isso acontece porque a repetição utiliza dados anteriores de uma imagem, o que não proporciona suavização. Em contraste, a interpolação média realiza cálculos de preenchi- mento em três direções, garantindo uma continuidade maior entre as imagens, resultando em uma definição superior.

Figura 48: Ângulo de cima da cabeça da parte interpolada repetição

A figura 48 demonstra que há umas partes seccionadas, porém não há tanta percepção de separação como na interpolação vazia, porém há uma continuidade parecida com uma interpolação média.

No final, cada interpolação tem suas vantagens e desvantagens em relação a seu uso em cada tipo de preenchimento de partes vazias deixadas pelos tipos usados pelo método de seccionamento em série escolhido para fins de estudo. Usando configurações para melhor refinamento da imagem, obtém-se este resultado na figura 49:

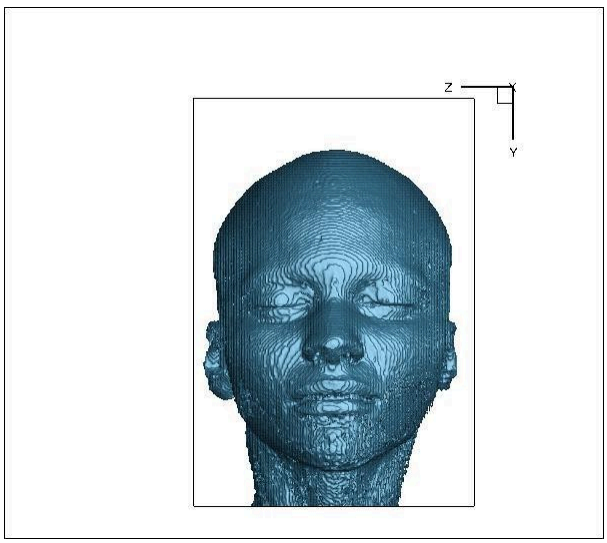

Figura 49: Figura refinada

4.5 Comparação entre códigos

Para esta pesquisa, foram utilizados dois tipos de código para a reconstrução. O primeiro realiza o processamento direto das imagens, enviando cada uma diretamente para a reconstrução. O segundo código utiliza a parte 'Classe', economizando memória RAM ao armazenar os dados da imagem processada em 'mat', que depois são enviados para a reconstrução. Nesta seção, será apresentada uma comparação entre uma das interpolações utilizadas para comparar os códigos em relação à reconstrução, enquanto a segunda seção abordará o tempo de processamento de cada código.

4.5.1 Comparação entre Reconstruções

Para a comparação entre as reconstruções feitas em cada código será utilizada a inter- polação média para este fim. Primeiro, a figura 50 demonstra o método de interpolação média com o código sem o uso da 'Classe'.

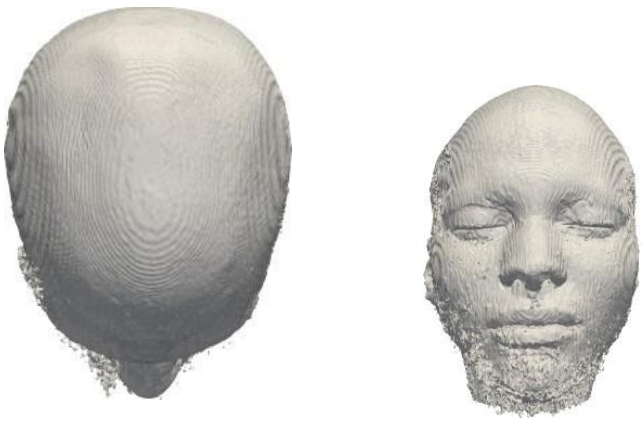

Figura 50: Interpolação Média sem a 'Classe'

Nesta reconstrução ocorre com o processamento de cada imagem passando direto para a reconstrução. As características já foram discutidas na seção anterior sobre tipos de interpolação. Agora será comparada com as imagens reconstruídas usando o código com a inclusão da 'Classe' para a economia de memória RAM.

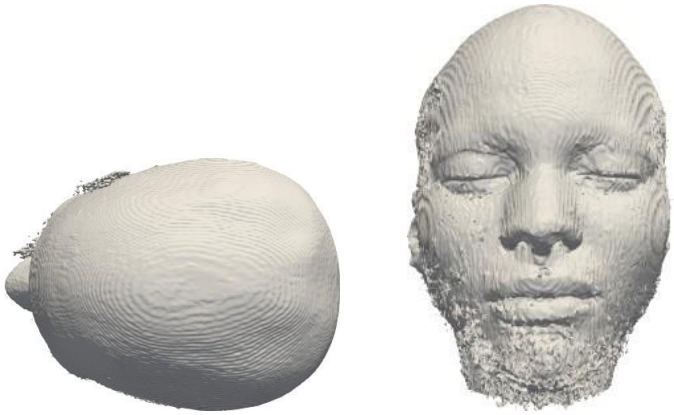

Figura 51: Interpolação Média usando a ’Classe’

De acordo com as imagens acima da figura 51 e usando a comparação com a figura 52 não ocorreu alteração entre as reconstruções usando códigos com alterações em sua estrutura. Tanto o código de processamento de cada imagem direta quanto ao código que contém a ’Classe’ para guardar o processamento de cada imagem em ’mat’ para a economia de memória RAM trazem o mesmo resultado em seu uso.

4.5.2 Tempo de Processamento

Os dois códigos trazem o mesmo resultado, porém são diferentes em tempo de pro- cessamento em relação a cada tipo de computador. Para a reconstrução do rosto inteiro, o computador usado não processou bem o código com a configuração ’Classe’ em sua estrutura. Apesar de haver economia em memória RAM, seu processamento para gerar a saída .dat ou .vtk pede um processamento de geração de saída maior, o que ocorre em falhas na tentativa de reconstrução de números maiores de imagem.

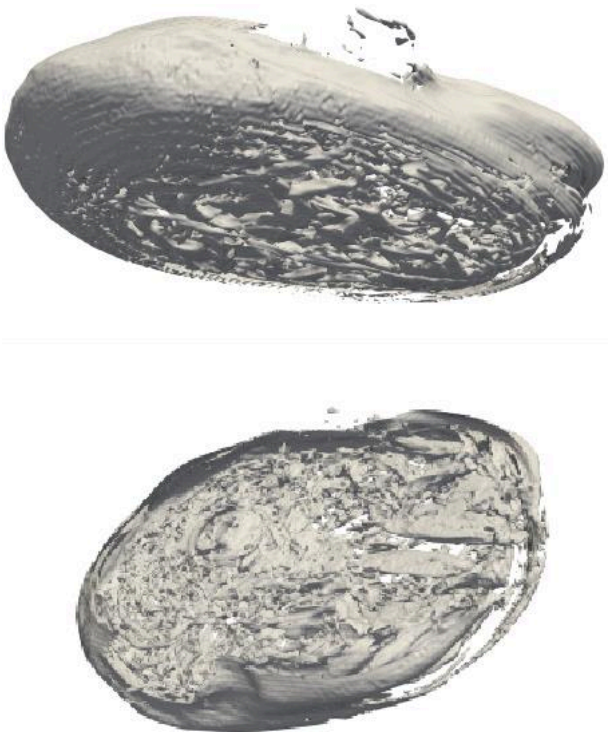

Figura 52: Crânio média

Para a geração desta reconstrução tridimensional de uma parte do crânio, mostrada na figura 52 usando a interpolação média, contendo trinta imagens, foi preciso este tempo de execução de acordo com a figura 54.

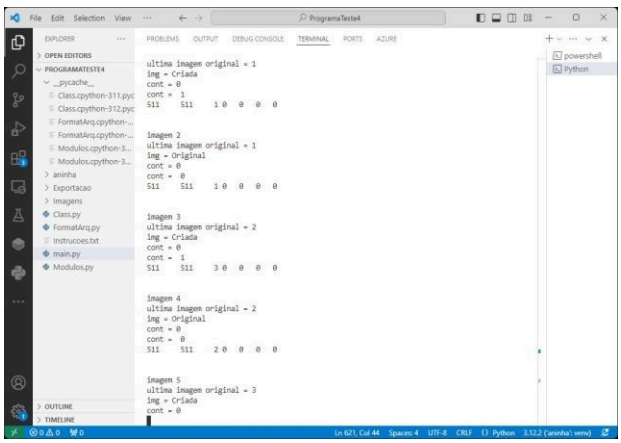

Figura 53: Tempo de processamento da reconstrução do crânio

O tempo de processamento total foi de 1027,65 segundos, o que é muito para um código conseguir fazer uma estrutura de 30 imagens. Quando este mesmo código é usado para números maiores de imagens ocorre esse erro demonstrado pela figura 53.

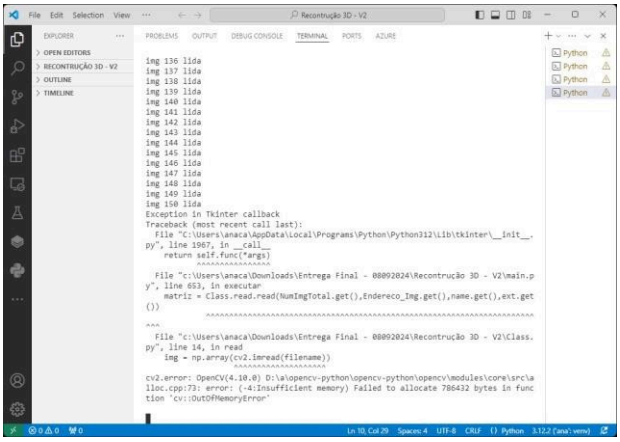

Figura 54: Erro apresentado pelo código

Este erro ocorre porque o código utiliza a estrutura ’Classe’; embora isso economize memória RAM, sua execução é lenta. Em contrapartida, o código sem a estrutura ’Classe’, apesar de consumir mais memória RAM, executa rapidamente. Para a reconstrução 3D de figuras maiores e com mais seções, foi necessário utilizar o computador do laboratório de simulação para testar o código com a estrutura ’Classe’ em busca de melhores resultados.

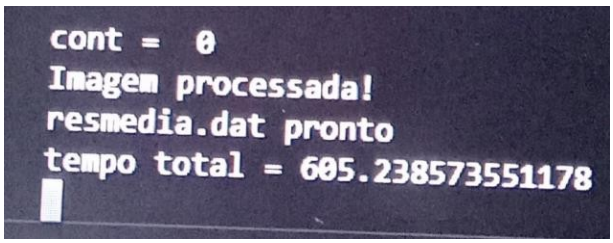

Figura 55: Tempo de processamento, em segundos, em um computador do laboratório da UFF

Em computadores com configurações superiores, a performance é melhor, como ilus- trado na figura 54. Ao comparar com a figura 55, o processamento de 176 imagens em um dos computadores do laboratório universitário foi mais rápido que o de um computador com configurações simples processando 30 imagens. Portanto, conclui-se que o código utilizando a estrutura ’Classe’ tem um tempo de execução maior do que o código sem essa estrutura.

Capítulo 5

Conclusões e Sugestões para Trabalhos Futuros

5.1 Conclusões

A pesquisa visou desenvolver e implementar um método computacional para a reconstrução tridimensional de imagens seccionadas, utilizando interpolação e otimização de código. Os resultados demonstraram que a metodologia é viável e eficaz na conversão de imagens bidimensionais em representações tridimensionais, com aplicações potenciais em áreas como a medicina e biomédica.

A implementação do sistema possibilitou uma análise comparativa de diferentes técni- cas de reconstrução, evidenciando que a estrutura 'Classe' no código economiza memória RAM, embora aumente o tempo de processamento. Em contrapartida, o código sem essa estrutura apresentou execução mais rápida, mas com maior consumo de memória. Esses achados são essenciais para futuras otimizações, permitindo ajustes de eficiência conforme as necessidades de usuários e hardware.

A interface gráfica desenvolvida foi intuitiva e de fácil acesso, facilitando o uso por pesquisadores e estudantes. As interpolações apresentaram desempenhos variados, com prós e contras a serem considerados conforme a aplicação. A reconstrução tridimensi- onal conseguiu capturar adequadamente a forma das imagens processadas, assegurando fidelidade às estruturas analisadas.

A pesquisa alcançou seus objetivos ao validar a técnica aplicada e oferecer uma base para melhorias futuras no sistema. A conversão de imagens de ressonância magnética em modelos tridimensionais é uma contribuição significativa para áreas como planejamento cirúrgico, estudo anatômico e modelagem de estruturas físicas a partir de imagens médicas.

5.1.1 Objetivos Específicos Alcançados

• : Código reconstrói imagens seccionadas em 2D para objetos tridimensionais;

• : Interface gráfica bem interativa, de fácil manuseio e suave em sua concepção;

• : Interpolações funcionam bem para os objetos reconstruídos, tanto em questão da reconstrução tridimensional acontecer quanto na parte de preenchimento de lacunas existentes em sua concepção;

• : O código consegue atender à demanda de reconstruir imagens em RM bidimensio- nais em peças 3D, alcançando o objetivo de utilidade na parte médica e biomédica;

• : o código feito conseguiu ir além na reconstrução tridimensional usando imagens de partes corporais seccionadas em uma imagem de um rosto completo.

5.1.2 Trabalhos Futuros

• : Otimização do código para que seu tempo de execução seja menor;

• : Aprimoramento da interface gráfica para que haja maior facilidade em seu uso (como o uso de mais janelas ou botões que já tragam direto o caminho da pasta de imagens sem precisar de sua escrita);

• : Teste em outros tipos de materiais e fotografias em série (como em tomografias, scanneamento por cada imagem, fotos digitais geradas por câmera,entre outros);

• : Aprofundamento em concepção de estrutura de código para que possa haver outras possibilidades de saídas de dados, como a .obj, para que possa ser usada em impressoras 3D por exemplo, para a aprimoração de estudos e pesquisas acadêmicas;

• : Criar uma linha de código para que possa tirar os ruídos das imagens que serão processadas, porque as imagens usadas neste estudo possui ruídos em partes escuras da figura.

5.1.2.1 Conclusões finais

O trabalho representa um avanço significativo na reconstrução tridimensional de ima- gens seccionadas, oferecendo um método acessível e eficiente para criar modelos 3D a partir de dados bidimensionais. As técnicas de interpolação utilizadas melhoraram a suavização das transições entre seções, resultando em maior precisão.

Além disso, a pesquisa abriu caminhos para novas investigações e aprimoramentos, tanto na eficiência do código quanto na diversificação de suas aplicações. As sugestões para trabalhos futuros indicam direções promissoras, incluindo melhorias computacionais, testes mais amplos e integração com outras tecnologias. Assim, este estudo contribui para o desenvolvimento de soluções inovadoras e acessíveis na modelagem tridimensional computacional.

Referências

[1] GRACA, A. B. R. de A. et al. Desenvolvimento de ferramenta computacional para criacao de domínios discretos em 3d. *Proceeding Series of the Brazilian Society of Com- putational and Applied Mathematics*, v. 6, n. 2, 2018.

[2] SCURI, A. E. Fundamentos da imagem digital. *Pontifícia Universidade Católica do Rio de Janeiro*, p. 13, 1999.

[3] GOMES, D. S. Aplicação e validação de um código computacional para reconstrução 3d utilizando imagens oriundas do seccionamento em série. 2022.

[4] JIA, Z.-G.; LING, S.-T.; ZHAO, M.-X. Color two-dimensional principal component analysis for face recognition based on quaternion model. In: SPRINGER. *Intelligent Computing Theories and Application: 13th International Conference, ICIC 2017, Li- verpool, UK, August 7-10, 2017, Proceedings, Part I 13*. [S.l.], 2017. p. 177–189.

[5] TAN, J. et al. A robust image representation method against illumination and occlu- sion variations. *Image Vis. Comput.*, v. 112, p. 104212, 2021.

[6] MUHTAR, N. et al. Pattern similarities of vector matrices. *Journal of Physics: Con- ference Series*, v. 1341, 2019.

[7] GRZYBOWSKI, A.; KUPIDURA-MAJEWSKI, K. What is color and how it is per- ceived? *Clinics in dermatology*, v. 37 5, p. 392–401, 2019.

[8] WEBER, K. et al. Rgb marking facilitates multicolor clonal cell tracking. *Nature medicine*, Nature Publishing Group, v. 17, n. 4, p. 504, 2011.

[9] SCHMITT, O. et al. Image registration of sectioned brains. *International Journal of Computer Vision*, v. 73, p. 5–39, 2007.

[10] MCMILLAN, D. B.; HARRIS, R. J. *An atlas of comparative vertebrate histology*. [S.l.]: Academic Press, 2018.

[11] WOODWARD, M. T. e M. Groeber e J. Simmons e A. Rosenberger e C. Extração automatizada de características de microestrutura simétrica em imagens de secciona- mento serial. *Caracterização de materiais*, v. 61, p. 1406–1417, 2010.

[12] BOCK, D. D. et al. Network anatomy and in vivo physiology of visual cortical neu- rons. *Nature*, Nature Publishing Group, v. 471, n. 7337, p. 177–182, 2011.

[13] SAALFELD, S. et al. Elastic volume reconstruction from series of ultra-thin micros- copy sections. *Nature methods*, Nature Publishing Group US New York, v. 9, n. 7, p. 717–720, 2012.

[14] SARMAH, M.; NEELIMA, A.; SINGH, H. R. Survey of methods and principles in three-dimensional reconstruction from two-dimensional medical images. *Visual compu- ting for industry, biomedicine, and art*, Springer, v. 6, n. 1, p. 15, 2023.

[15] ANISIC, D. B. G. e S. Mihic e D. Dragan e Veljko B. Petrovic e Z. Simulação de aquisição de dados 3d baseada em fotogrametria. *International Journal of Simulation Modelling*, 2019.

[16] SORKINE-HORNUNG, K. Y. e A. Sorkine-Hornung e Oliver Wang e O. Segmentação eficiente de objetos 3d a partir de campos de luz densamente amostrados com aplicações para reconstrução 3d. *ACM Transactions on Graphics (TOG)*, v. 35, p. 1 – 15, 2016.

[17]

[18] LEE, S. J.; WOLBERG, G.; SHIN, S. Y. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, v. 3, n. 3,
p. 228–244, 1997.

[19] XU, Y. et al. Vector sparse representation of color image using quaternion matrix analysis. *IEEE Transactions on Image Processing*, v. 24, p. 1315–1329, 2015.

[20] CONTRIBUTORS, O. *OpenCV: Open Source Computer Vision Library*. 2025. Https://opencv.org/.

[21] AL., T. E. O. et. *NumPy: A fundamental package for scientific computing with Python*. 2025. Https://numpy.org/.

[22] MANSSOUR, I. H.; COHEN, M. Introdução à computação gráfica. *RITA*, v. 13, n. 2, p. 43–68, 2006.

[23] FOUNDATION, P. S. *Python: A dynamic object-oriented programming language*. [S.l.], 2025. Accessed: 2025-01-29. Disponível em: <https://www.python.org/>.

[24]

[25]

[26] JACOB, E. L. A medialidade hiperativa da imagem digital. *Latin American Journal of Development*, South Florida Publishing LLC, v. 3, n. 1, p. 147–162, 2021.

[27]

[28] CÂMARA, G.; MEDEIROS, J. S. d. Modelagem de dados em geoprocessamento. *Sistemas de Informação Geográfica: aplicações na agricultura (ED Assad & EE Sano, eds). EMBRAPA, Brasília*, p. 47–66, 2005.

[29] PAL, P. B. Continuity of functions. In: _. *A Physicist's Introduction to Algebraic Structures: Vector Spaces, Groups, Topological Spaces and More*. [S.l.]: Cambridge University Press, 2019. p. 597–606.

[30] KUMAR, T.; VERMA, K. A theory based on conversion of rgb image to gray image. *International Journal of Computer Applications*, v. 7, p. 5–12, 2010.

[31] FAIRCHILD, M. D. *Color Appearance Models*. 3rd. ed. [S.l.]: John Wiley & Sons, 2013. ISBN 9781118653106.

[32] PAYNE, D.; DOLAN, P.; HODOUL, M. Implementing opencolorio v2: A color management framework for motion picture production. In: *ACM SIGGRAPH 2017 Talks*. [s.n.], 2017. p. Article No.: 5. Disponível em: <https://history.siggraph.org/experience/color-management-with-opencolor io-v2- by-payne-dolan-and-hodoul/>.

[33] GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 3. ed. [S.l.]: Pearson Prentice Hall, 2008.

[34] FISHER, P. The pixel: A snare and a delusion. *International Journal of Remote Sensing*, v. 18, p. 679–685, 1997.

[35] BLINN, J. What is a pixel? *IEEE Computer Graphics and Applications*, v. 25, n. 5, p. 82–87, 2005.

[36] PARKER, J. A.; KENYON, R. V.; TROXEL, D. E. Comparison of interpolating methods for image resampling. *IEEE Transactions on Medical Imaging*, IEEE, v. 2, n. 1, p. 31–39, 1983.

[37] SEUNG, S. P. e T. Macrina e N. Kemnitz e M. Castro e B. Nehoran e Z. Jia e JA Bae e E. Mitchell e S. Mu e Eric T. Trautman e S. Saalfeld e Kai Li e S. Pipeline petascale para alinhamento preciso de imagens de microscopia eletrônica de seção serial. *bioRxiv*, 2022.

[38] MICHEVA, K. D.; SMITH, S. J. Array tomography: a new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, Elsevier, v. 55, n. 1, p. 25–36, 2007.

[39] KOMURA, M. W. e Ming Ju e Yuling Fan e Shihui Guo e Minghong Liao e Hui-jun Yang e Dongjian He e T. Reconstrução de superfícies de nuvens de pontos incompletas 3d com segmentação e aprimoramento de recursos. *IEEE Access*, v. 7, p. 15272–15281, 2019.

[40] UM algoritmo rápido de registro de imagem difeomórfica. v. 38.

[41] LEVOY, M. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, v. 8, n. 3, p. 29–37, 1988.

[42] SIBSON, R. A brief description of natural neighbor interpolation. In: *Interpreting Multivariate Data*. [S.l.]: John Wiley & Sons, 1981. p. 21–36.

[43] AL-AZZAWI, A. et al. Auto3dcryomap: An automated particle alignment approach for 3d cryo-em density map reconstruction. *BMC Bioinformatics*, Springer, v. 21, n. 1,
p. 441, 2020.

APÊNDICE A - Reconstruçao tridimensional usando o ParaView

A.1 Passo a passo

Para começar a renderização da saída .dat ou .vtk precisa-se utilizar um programa que leia este tipo de arquivo, um desses programas se chama ParaView. Para começar, abre-se o programa, como mostrado na imagem 56.

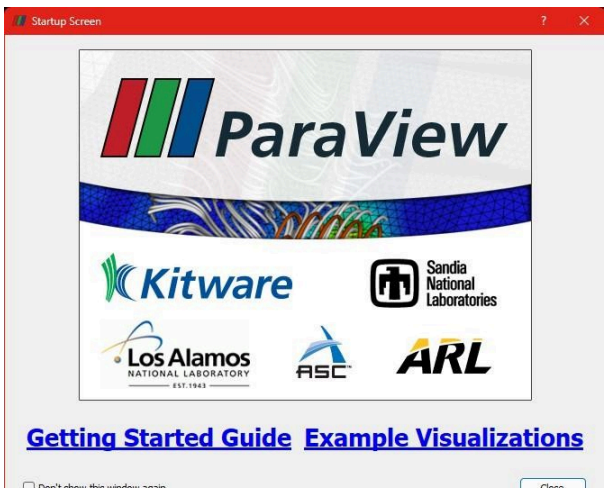

Figura 56: Abertura do ParaView

Depois da abertura do programa, vai-se até a opção *Files* para a abertura das pastas que contém os arquivos processados pelo código feito por esta pesquisa.

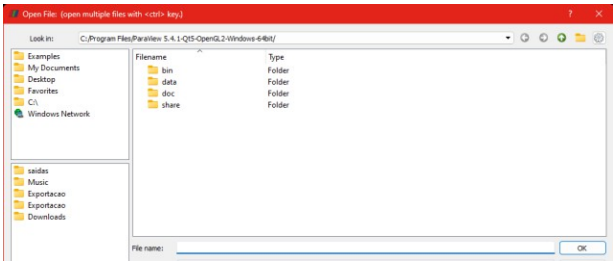

Figura 57: Pastas com os arquivos a serem escolhidos

Seguindo o caminho em que contém o código, vá até a pasta do código feito para a pesquisa, como mostrado na figura 59, e clique na parte exportação. Nesta pasta constam os arquivos de saída gerados pela execução do programa. Achando este arquivo, clique na opção ok para que abra as opções de abertura da saída. Para que ocorra a renderização do código, é escolhido a opção *Tecplot Files*. Depois de escolher esta opção clique em ok, de acordo com a figura 57.

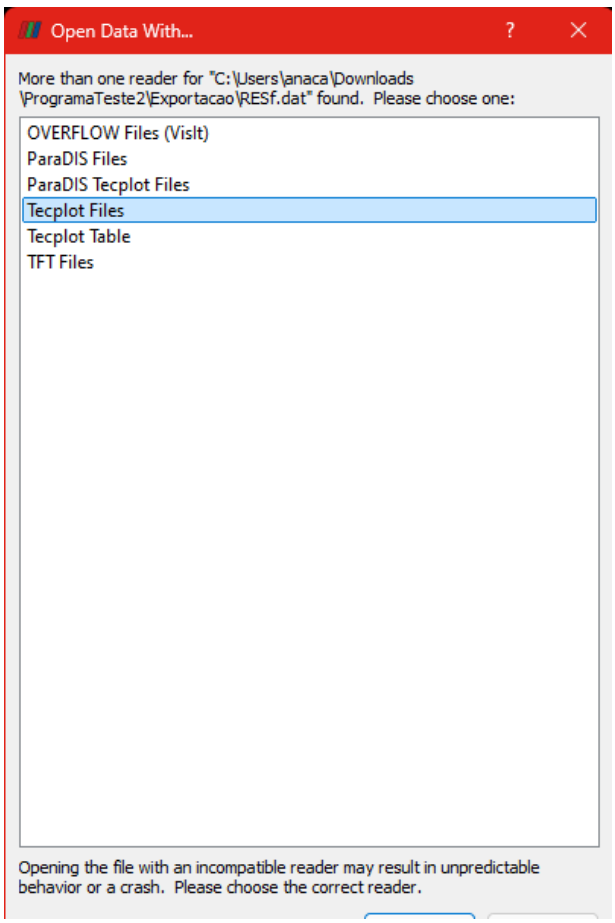

Figura 58: Opções de abertura da saída da execução do código

Depois desse passo, aparecerá na parte esquerda da tela a caixa contendo o nome do arquivo de saída dado, agora clica-se nele e aperta-se o botão *Apply*.

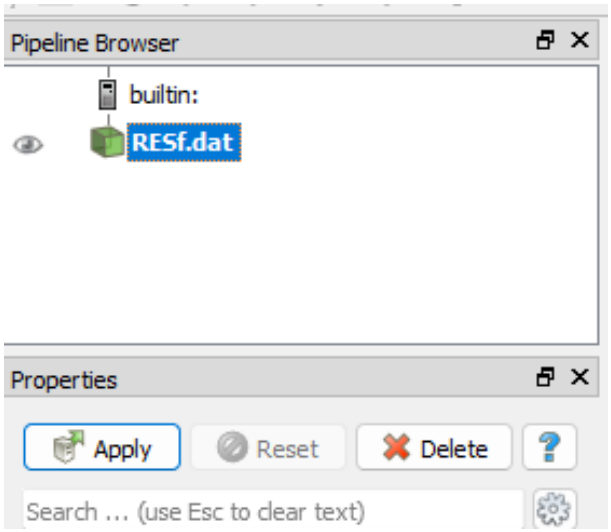

Figura 59: Arquivo saída carregado

Depois do passo mostrado na figura 61, o arquivo terá um tempo para carregar em seu término mostrará uma caixa vazia, onde será mostrada a imagem tridimensional.

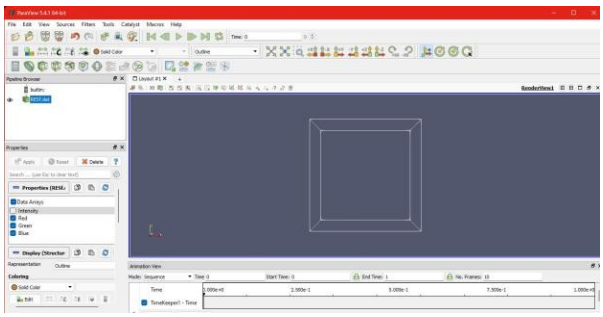

Figura 60: Imagem da caixa carregada depois do processamento da saída

A figura 58 demonstra esta parte do processo. Para que ocorra a reconstrução, deve-se apertar o botão de contorno que aparecerá verde na parte de cima à esquerda da janela, como mostrado na figura 63. clicando neste botão aparecerá uma caixa embaixo da caixa com o nome do arquivo à esquerda da janela.

Figura 61: Botão Contorno

Depois de feito todo este passo, aperta-se o botão *Apply*. O programa processará a renderização feita (esta parte pode ter uma demora, pois depende do tamanho do arquivo de saída gerada pelo código da pesquisa).

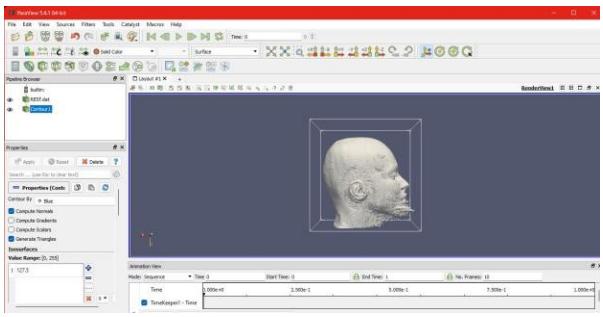

Figura 62: Objeto 3D já processado

Depois do carregamento, o objeto já aparecerá em 3D na tela, de acordo com a figura 59 . Depois deste processo, a renderização está feita.

chapterReconstruçao tridimensional usando o ParaView

.1 Passo a passo

Para começar a renderização da saída .dat ou .vtk precisa-se utilizar um programa que leia este tipo de arquivo, um desses programas se chama ParaView. Para começar,

abre-se o programa, como mostrado na imagem 56.

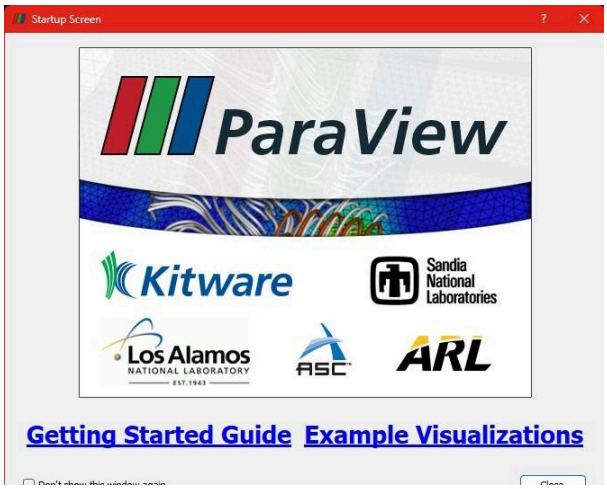

Figura 63: Abertura do ParaView

Depois da abertura do programa, vai-se até a opção *Files* para a abertura das pastas que contém os arquivos processados pelo código feito por esta pesquisa.

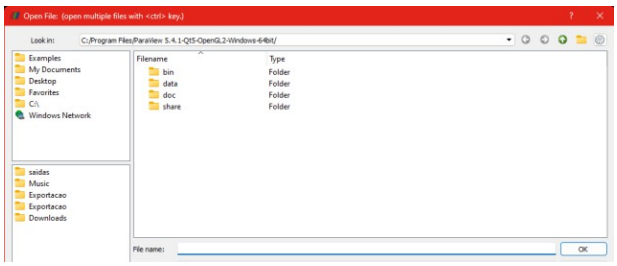

Figura 64: Pastas com os arquivos a serem escolhidos

Seguindo o caminho em que contém o código, vá até a pasta do código feito para a pesquisa, como mostrado na figura 59, e clique na parte exportação. Nesta pasta consta os arquivos de saída gerados pela execução do programa. Achando este arquivo, clique na opção ok para que abra as opções de abertura da saída. Para que ocorra a renderização do código, é escolhido a opção *Tecplot Files*. Depois de escolher esta opção clique em ok, de acordo com a figura 57.

Depois desse passo, aparecerá na parte esquerda da tela a caixa contendo o nome do arquivo de saída dado, agora clica-se nele e aperta-se o botão *Apply*.

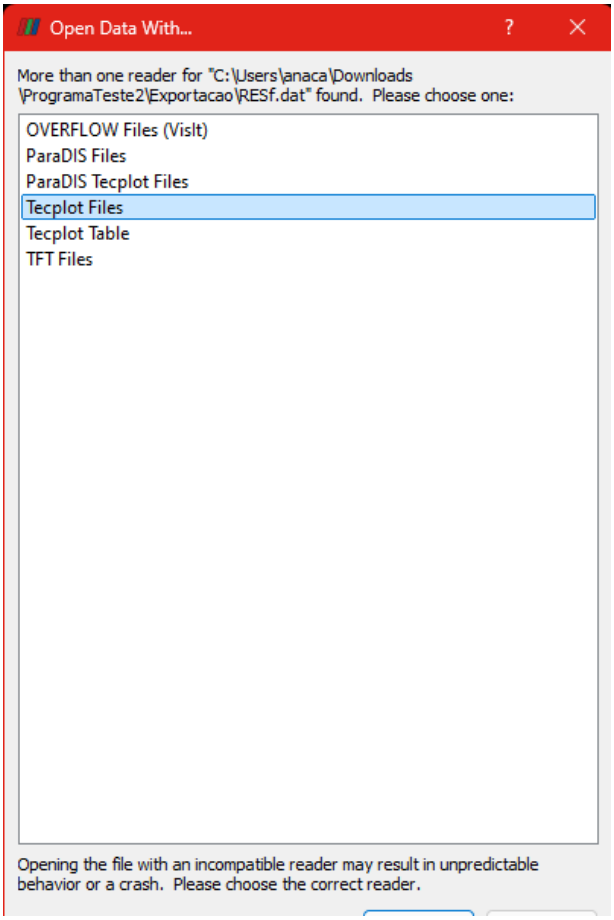

Figura 65: Opções de abertura da saída da execução do código

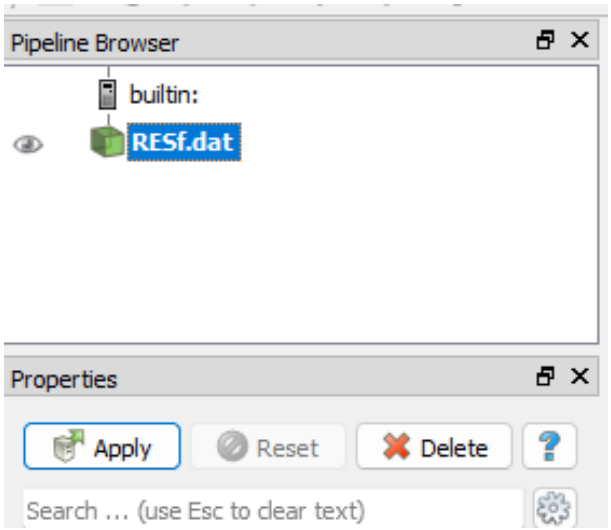

Figura 66: Arquivo saída carregado

Depois do passo mostrado na figura 61, o arquivo terá um tempo para carregar em seu término mostrará uma caixa vazia, onde será mostrada a imagem tridimensional.

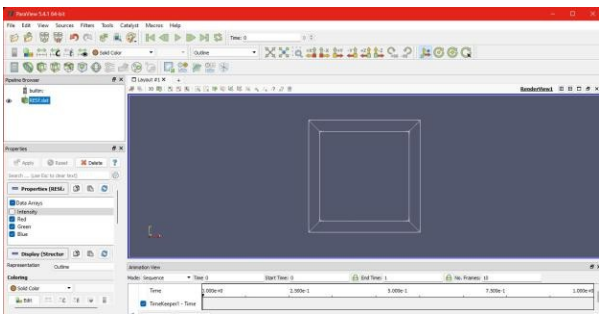

Figura 67: Imagem da caixa carregada depois do processamento da saída

A figura 58 demonstra esta parte do processo. Para que ocorra a reconstrução, deve-se apertar o botão de contorno que aparecerá verde na parte de cima à esquerda da janela, como mostrado na figura 63. clicando neste botão aparecerá uma caixa embaixo da caixa com o nome do arquivo à esquerda da janela.

Figura 68: Botão Contorno

Depois de feito todo este passo, aperta-se o botão *Apply*. O programa processará a renderização feita (esta parte pode ter uma demora, pois depende do tamanho do arquivo de saída gerada pelo código da pesquisa).

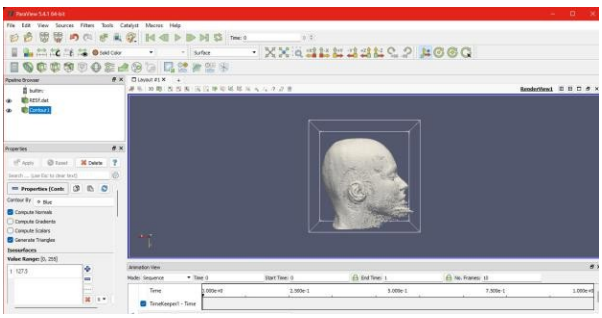

Figura 69: Objeto 3D já processado

Depois do carregamento, o objeto já aparecerá em 3D na tela, de acordo com a figura 59 . Depois deste processo, a renderização está feita.