UNVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ENGENHARIA ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

LARISSA FERNANDA CUNHA CAVALCANTE DE MELO

DESENVOLVIMENTO DE SOFTWARE EM LABVIEW PARA RENDERIZAR IMAGENS DE TOMOGRAFIA DE COERÊNCIA ÓPTICA EM UM VOLUME TRIDIMENSIONAL COM AUXÍLIO DE FILTROS DO MATLAB

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO - PR
2015

LARISSA FERNANDA CUNHA CAVALCANTE DE MELO

DESENVOLVIMENTO DE SOFTWARE EM LABVIEW PARA RENDERIZAR IMAGENS DE TOMOGRAFIA DE COERÊNCIA ÓPTICA EM UM VOLUME TRIDIMENSIONAL COM AUXÍLIO DE FILTROS DO MATLAB

Proposta de Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Conclusão de Curso, do curso Superior de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica (DAELN) da Universidade Tecnológica Federal do Paraná (UTFPR), como requisito parcial para obtenção do título de Engenheiro Eletrônico.

Orientador: Prof. MSc. Leonardo Costa

CAMPO MOURÃO - PR

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação do Curso de Engenharia Eletrônica

TERMO DE APROVAÇÃO
DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

Desenvolvimento de Software em Labview para Renderizar Imagens de Tomografia de Coerência Óptica em um Volume Tridimensional com Auxílio de Filtros do Matlab

por

Larissa Fernanda Cunha Cavalcante De Melo

Trabalho de Conclusão de Curso apresentado no dia 13 de Julho ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Câmpus Campo Mourão. O Candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO (aprovado, aprovado com restrições ou reprovado).

Prof. Dr. Marcio Rodrigues da Cunha
(UTFPR)

Prof. Msc. Osmar Tormena Junior
(UTFPR)

Prof. Msc. Leonardo Faria Costa
(UTFPR)
Orientador

Campo Mourão, 13 de Julho de 2015.

Dedico aos meus pais.

AGRADECIMENTOS

Agradeço à minha família por todo o suporte, em especial aos meus pais e à minha avó Alzerina por todo o carinho.

Agradeço os meus colegas de classe e amigos que fizeram parte da minha vida durante todos esses anos, tanto de perto quanto de longe.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de graduação-sanduíche nos Estados Unidos.

Agradeço ao prof. Dr. Michael Wang, da Universidade de Miami, pela oportunidade em trabalhar nesse projeto e a Hui Lu, obrigada por toda ajuda que que você me forneceu durante o estágio acadêmico de verão. *(Thanks to prof Dr. Michael Wang, from University of Miami, for the opportunity to work on this project and to Hui Lu, thanks for all the help you have given me during the summer academic training.)*

À Coordenação de Engenharia Eletrônica (COELE) da UTFPR, campus Campo Mourão, por acreditar no potencial do trabalho. Ao prof. Leonardo Costa pela orientação e ajuda prestada durante a elaboração do relatório.

Agradeço também ao coordenador e aos professores da banca examinadora pela atenção e dedicação. E um especial agradecimento ao prof. Nilson Kominek por sempre acreditar no sucesso ao fundar o nosso curso.

“Wyrd bið ful aræd”

O destino é inexorável

(Bernard Cornwell)

RESUMO

MELO, Larissa F C Cavalcante de, **DESENVOLVIMENTO DE SOFTWARE EM LABVIEW PARA RENDERIZAR IMAGENS DE TOMOGRAFIA DE COERÊNCIA ÓPTICA EM UM VOLUME TRIDIMENSIONAL COM AUXÍLIO DE FILTROS DO MATLAB**. 2015. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia Eletrônica. Universidade Tecnológica Federal do Paraná.

Este trabalho propõe o desenvolvimento de um software em LabVIEW para renderizar imagens de tomografia de coerência óptica em um volume tridimensional. A Tomografia de Coerência Óptica (*Optical Coherence Tomography*, OCT) é uma técnica de alta resolução que usa fontes luminosas de baixa potência e suas reflexões para criar imagens. Através da tecnologia OCT é possível gerar imagens de partes internas de órgãos e detectar possíveis anomalias sem a necessidade de cirurgias. Utilizando o LabVIEW como ferramenta de auxílio para manipulação de dados, o projeto consistiu em desenvolver um programa para a leitura e processamento de sequência de imagens OCT do modelo de um olho humano. O programa permitiu tratar a qualidade da imagem com filtros do MATLAB e agrupá-las em pilhas para gerar um volume tridimensional do objeto.

Palavras-chave: LabVIEW, Tomografia de Coerência Óptica (OCT), Processamento de imagens.

ABSTRACT

MELO, Larissa F C Cavalcante de, **LABVIEW SOFTWARE DEVELOPMENT TO RENDER OPTICAL COHERENCE TOMOGRAPHY IMAGES IN A THREE-DIMENSIONAL VOLUME WITH THE SUPPORT OF MATLAB'S FILTER**. 2015. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia Eletrônica. Universidade Tecnológica Federal do Paraná.

This paper proposes the development of a software in LABVIEW to render Optical Coherence Tomography images in a three-dimensional volume. Optical Coherence Tomography (OCT) is a high-resolution technique that uses low power light sources and their reflections to create images. The OCT technology is able to generate images of internal parts of organs and detect possible anomalies without the need for surgery. Using LabVIEW as a tool for data manipulation, the project consisted of developing a software to read and process OCT image stacks of a human eye model. The software produces high quality images with MATLAB filters and it is able to group them into stacks in order to generate a three-dimensional volume of the object.

Palavras-chave: LabVIEW, Optical Coherence Tomography (OCT), Image Processing

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ACRÔNIMOS

OCT – Tomografia de Coerência Óptica (do original *Optical Coherence Tomography*)

LabVIEW – do original *Laboratory Virtual Instrument Engineering Workbench*

GPU – Unidades Gráficas de Processamento (do original *Graphics Processing Unit*)

SLD – Diodo superluminoso (do original *Superluminescent Diode*)

VI – Instrumentos virtuais (do original *Virtual Instruments*)

MATLAB – Laboratório de Matrizes (do original *MATrix LABoratory*)

SUMÁRIO

1. INTRODUÇÃO

A Tomografia de Coerência Óptica (OCT) é uma técnica de alta resolução que usa fontes luminosas de baixa potência e suas reflexões para gerar imagens. Através da tecnologia OCT é possível gerar imagens de partes internas de órgãos e detectar possíveis anomalias sem a necessidade de cirurgias. Por não causar danos ao tecido estudado, a técnica é amplamente aplicada na área de medicina, mas com promissoras aplicações em outras áreas tais como a metrologia e a pesquisa forense. Através da OCT é possível detectar doenças como câncer em estágios iniciais (câncer de pele, por exemplo) e outras irregularidades no tecido a ser examinado.

Dados gerados por instrumentos médicos requerem resultados rápido, de fácil entendimento e alta qualidade de visualização. Com a técnica OCT, informações mais precisas e apuradas podem ser obtidas facilmente. Para a manipulação de dados em diversos formatos, inclusive médicos, o LabVIEW se destaca no meio científico e clínico por fornecer maior flexibilidade, facilidade de aprendizagem e manuseio. Desenvolvido pela National Instruments, LabVIEW (acrônimo para *Laboratory Virtual Instrument Engineering Workbench*) é uma plataforma de desenvolvimento que utiliza programação gráfica em linguagem G aplicando o conceito de fluxo de dados na execução dos programas. As funções do código são chamadas de instrumentos virtuais (VI) e muitas delas são polimorfas, ou seja, adaptam-se a diversos tipos de elementos. As interfaces gráficas podem ser facilmente construídas pelo programador sem a necessidade de escrever qualquer linha de código, permitindo maior facilidade de utilização e leitura.

Partindo desse contexto, em 2011 um grupo de cientistas no Japão [1] desenvolveram o primeiro aparelho OCT tridimensional em tempo real usando LabVIEW para o processamento e visualização de dados. O projeto tinha elevada eficiência ao utilizar unidades gráficas de processamento (GPU) específicas para maior velocidade e precisão de resposta, no qual o equipamento se tornou rápido o suficiente para eliminar quaisquer artefatos de movimentos. Com o uso de processadores especiais e uma apurada arquitetura, o projeto desenvolvido pelo Dr. Ohbayashi e sua equipe [1] era capaz de realizar o a varredura e a visualização tridimensional das partes internas de qualquer tecido em tempo real.

Em 2013, alunos de doutorado em engenharia elétrica em parceria com os alunos de medicina na Universidade de Miami se basearam no projeto do Dr. Ohbayashi [1] e

desenvolveram uma máquina de OCT para fins acadêmicos [2]. Através do aparelho desenvolvido era possível escanear olhos tanto de moldes quanto de humanos, com boa visibilidade e relativamente baixo ruído. As sequências de imagens geradas representavam o olho em corte bidimensional cujo formato era alterado de acordo com a posição da lente do aparelho. Em seguida, para gerar o modelo volumétrico, as imagens eram transferidas para um software especial chamado Amira 3D. Através do software Amira 3D, produzido pela FEI Visualization Sciences Group, era possível posicionar as imagens ao longo de um eixo vertical e extrair o volume tridimensional do objeto escolhido.

Embora o programa AMIRA fosse capaz de extrair a imagem volumétrica, o software possui uma licença de testes restrita por um mês que inviabiliza o teste contínuo para pesquisas. Em comparação, o LabVIEW possui uma licença especial de seis meses para estudantes e uma vasta documentação online. Dessa forma, o projeto proposto consiste em desenvolver um código em LabVIEW com interface amigável para leitura de sequência de imagens OCT. Esse mesmo código será responsável pela filtragem de ruídos em cada imagem e a consequente melhoria na qualidade de visualização. Como último passo, o código irá produzir um volume tridimensional a ser exibido no visor gráfico do próprio LabVIEW de forma que o usuário possa interagir com o dado criado.

1.1. Problema

Apesar do software Amira 3D atender as expectativas e gerar imagens tridimensionais de alta qualidade, o programa requer uma licença de alto custo. Esse valor elevado cria um empecilho para o desenvolvimento do projeto e, consequentemente, interrompe o avanço do estudo. Embora a empresa forneça acesso para testes, eles são limitados a menos de um mês.

Um outro fator de contratempo está no desempenho de um projeto rotulado como tempo real. Para desenvolver um projeto em tempo real é fundamental utilizar computadores de elevado potencial de processamento, principalmente quando esses equipamentos precisam lidar com imagens em alta-resolução. Por conta disso, o desenvolvimento do projeto em tempo real se torna inviável do ponto de vista financeiro. Em adição, também é importante aprender o desenvolvimento de algoritmos em arquitetura-paralela, o que causaria o consumo de mais tempo para a pesquisa e realização do projeto.

Apesar das imagens coletadas em laboratório possuírem relativamente boa qualidade,

muitos ruídos externos foram captados e, dessa forma, se torna necessário fazer uma prévia manipulação desses dados para redução dos ruídos de cada imagem.

Considerando os fatores externos e internos desses impasses, o problema a ser resolvido consiste em desenvolver o próprio programa em LabVIEW que faça todo o processamento visual das imagens coletadas de forma eficiente e econômica utilizando baixos recursos computacionais.

1.2. Justificativa

Nos últimos anos observou-se um grande avanço tecnológico na área médica e clínica. Entre eles, a tecnologia OCT se tornou um grande aliado científico para a análise de diversos tipos de dados. Considerando esse contexto no qual a aplicação da tecnologia OCT se encontra, é importante que ela se torne uma ferramenta de baixo custo que possa atingir diversas camadas sociais.

O desenvolvimento de uma ferramenta de baixo custo para a visualização de imagens de tomografia óptica não requer processadores específicos de alto desempenho de forma que máquinas simples podem executar as tarefas da rotina e exibir resultados com boa visualização.

O programa LabVIEW funciona como um intermediador entre o usuário e o processamento dos dados. Com uma interface gráfica intuitiva e de fácil utilização, o usuário é capaz de manipular diversos tipos de dados e informações. Além disso, a parte de processamento visual pode ser entendida facilmente graças à compreensível disposição gráfica dos instrumentos virtuais e seu método de execução em fluxo.

O projeto possui relevância na área técnica por agregar conceitos físicos básicos da óptica com áreas da computação e bioengenharia. Usando um baixo processamento computacional e softwares de fácil acesso e manuseio é possível obter dados tridimensionais de maneira rápida e eficiente.

1.3. Objetivo

1.3.1. Objetivo geral

O principal objetivo do projeto é criar um programa em LabVIEW que não exija alto poder de processamento e que seja capaz de processar imagens geradas a partir de um dispositivo

óptico. Dessa forma é possível explorar os recursos oferecidos pelo software e aplicá-los em diversos projetos na área de bioengenharia.

1.3.2. Objetivos específicos

Dentre os objetivos específicos pode-se analisar a flexibilidade do software e a compatibilidade e funcionamento com outros softwares simultaneamente. O LabVIEW possui integração com o MATLAB através do instrumento virtual chamado "*mathscript*". Como o MATLAB possui filtros mais robustos e de melhor detecção, é possível avaliar e comparar a qualidade final da imagem proporcionada por ambos os programas de forma a determinar o filtro que melhor se aplica na resolução do problema.

1.4. Metodologia

A fim de direcionar o projeto, o trabalho foi dividido em etapas específicas de forma a obter resultados satisfatórios. A primeira parte do projeto consistiu na pesquisa bibliográfica sobre o assunto e a coleta de informações que serviram de suporte à viabilidade do projeto. Nessa etapa foi possível estudar métodos alternativos para a resolução do problema envolvendo outros softwares.

A segunda etapa do projeto consistiu em fazer um estudo a respeito do software LabVIEW para o aprendizado técnico, tais como funcionamento, integração com outros aplicativos e poder de processamento. Também foi realizado estudos de algoritmos em MATLAB.

Em seguida foram realizados testes e simulações com os filtros, tanto do MATLAB quanto do LabVIEW, para identificar os melhores resultados. Em ambos os softwares, foi feito uma análise comparativa entre os resultados gerados por cada filtro.

A quarta etapa compreende o estágio final do projeto com as documentações e conclusões finais.

1.5. Estrutura do trabalho

O trabalho foi dividido em seis capítulos com a estrutura descrita de acordo com os seguintes capítulos.

- Capítulo 1: O primeiro capítulo apresenta a introdução do tema, problema, justificativa, objetivos e a estrutura metodológica utilizada para a execução do projeto.
- Capítulo 2: Apresenta a revisão bibliográfica com uma introdução acerca do funcionamento da tomografia de coerência óptica.
- Capítulo 3: Apresenta uma visão geral sobre softwares existentes em LabVIEW
- Capítulo 4: Mostra os filtros disponíveis em LabVIEW e MATLAB.
- Capítulo 5: Apresentação dos testes e resultados obtidos.
- Capítulo 6: Discussões e conclusão.

2. FUNDAMENTAÇÃO TEÓRICA SOBRE OCT

2.1. Introdução a Tomografia de Coerência Óptica

A Tomografia de Coerência Óptica (OCT) permite gerar imagens em corte de objetos tridimensionais de forma não-invasiva ao medir reflexões ópticas. O sistema OCT executa múltiplas varreduras longitudinais em uma série de locais laterais para fornecer um mapa bidimensional dos lugares das reflexões da amostra [3]. Devido a isso, a tecnologia OCT pode escanear estruturas biológicas internas. Inicialmente a técnica de OCT foi empregada na área de oftalmologia [4] e por conta dessa vantagem a técnica possui a principal aplicação na área médica. Entretanto, ela também é passível de estudo e aplicações nas mais diversas áreas.

O dispositivo de tomografia de coerência óptica é similar ao mecanismo de ultrassom no modo B [5] e funciona a partir de uma referência baseando-se no princípio de interferometria de baixa coerência óptica onde é capaz de mensurar o tempo de demora de resposta da luz refletida nas diferentes estruturas. A saída de um diodo superluminoso (SLD) é acoplada em uma única fibra e é dividida meio-a-meio, no acoplador, entre a amostra e o braço de referência (onde se localiza a luz refletida) como mostra a Figura 1. As reflexões são combinadas e detectadas pelo fotodiodo. A modulação da intensidade do sinal de saída é detectada pelo fotodetector quando o atraso entre a referência e a amostra quase se correspondem. As amplitudes do sinal e o atraso das reflexões da amostra são medidas ao escanear a posição do espelho de referência e simultaneamente gravam a amplitude do sinal do interferômetro. O interferômetro detecta e analisa a interferência em diferentes profundidades da amostra.

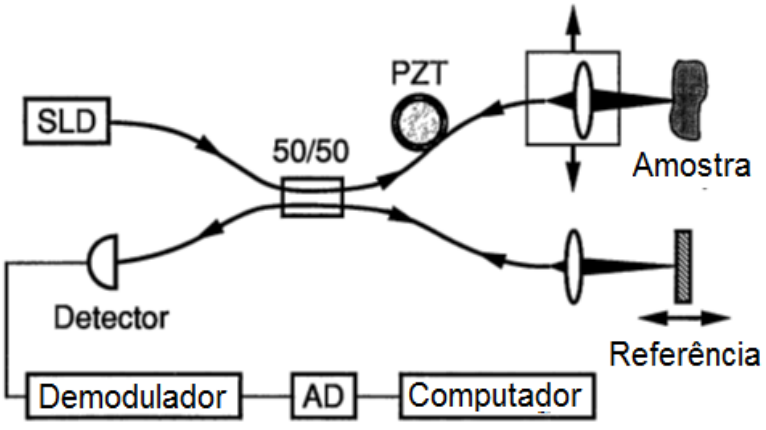


Figura 1 - Esquemático de um dispositivo OCT
Fonte: Adaptado de Huang (1991, p. 1178-1181)

Com a tecnologia OCT é possível medir tecidos biológicos com resolução 20 vezes maior que o ultrassom [5] e detectar precocemente anomalias internas em um tecido ou órgão de forma a facilitar o tratamento da doença. A imagem bidimensional é gerada em tempo real e pode ser visualizada em um monitor.

2.2. Projetos na área

Embora a técnica OCT seja fortemente empregada na área da medicina tradicional, ela tem um futuro promissor na área de engenharia forense com a detecção de impressão digital e no campo de metrologia por ser capaz de fornecer medidas precisas de materiais de espessuras muito finas.

Como os tradicionais métodos biométricos usados na segurança e identificação de pessoas são passíveis de falhas e fraudes, o emprego da tecnologia OCT no reconhecimento do indivíduo se torna uma alternativa para esses métodos por extrair características específicas internas do corpo humano. Em [6] é descrito a aplicação da tecnologia OCT como ferramenta de reconhecimento de impressão digital de duas e três dimensões, onde camadas adicionais abaixo da pele também podem ser identificadas e armazenadas. Com essa técnica, impressões digitais artificiais, feitas com diversos tipos de materiais, podem ser detectadas de forma a evitar possíveis fraudes.

Na área de bioengenharia, foi no Japão que um grupo de cientistas desenvolveram o primeiro aparelho OCT tridimensional em tempo real usando LabVIEW para o processamento e visualização de dados [1]. O projeto utilizava unidades gráficas de processamento (GPU) específicas para maior velocidade e precisão de resposta. O dispositivo projetado também era capaz de realizar 700.000 FFTs a cada segundo. Além disso, o equipamento era rápido o suficiente para eliminar quaisquer artefatos de movimentos. Levando em consideração um olho humano como exemplo, para inspecioná-lo só é necessário um apoio para a cabeça de modo que o movimento não cause distúrbios às imagens. Entretanto, em endoscopias OCT, tais como do sistema digestório e respiratório, o tecido a ser escaneado não pode ser fixado e depende de um dispositivo ultra veloz que seja capaz de eliminar esses artefatos de movimentos. Com o uso de processadores especiais e uma apurada arquitetura, o projeto desenvolvido pelo Dr. Ohbayashi e sua equipe era capaz de realizar a varredura e a visualização tridimensional em tempo real. No projeto final também era possível escolher até três modos de visualização: visualização contínua da imagem em 3D renderizada como mostra a Figura 2, visualização

contínua da imagem 2D ao longo dos eixos de um cubo tridimensional e visualização contínua e atualizada de todas as imagem adquiridas em modo B [1].

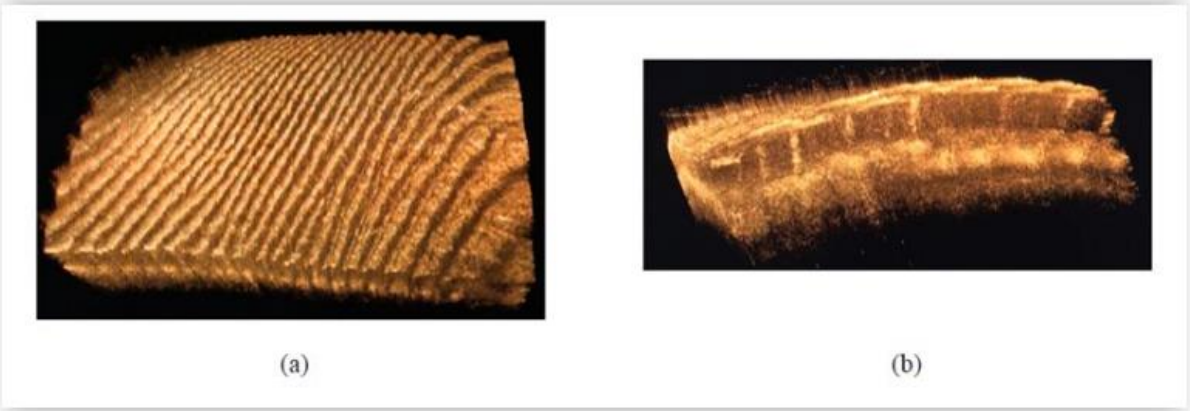


Figura 2 - Imagem em tempo real renderizada da pele de um (a) dedo mostrando diretamente a impressão digital e o (b) corte na direção lateral onde pode-se observar as glândulas sudoríparas.
Fonte: Ohbayashi. 2014.

3. INTRODUÇÃO AO LABVIEW

3.1. Visão geral sobre o LabVIEW

O LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) é um ambiente de desenvolvimento gráfico desenvolvido pela National Instruments para criar aplicações que possam interagir com diversos tipos de dados a partir de ligações de fios virtuais [7]. Essa linguagem é comumente usada para a aquisição de dados, processamento de sinais e controle de hardware. A programação em linguagem G é o centro do LabVIEW e é feita em fluxo de dados. As funções são chamadas de instrumentos virtuais (VI) pois seus ícones imitam instrumentos físicos de medição, tais como osciloscópios e multímetros [8]. O código é processado por um compilador garantindo um desempenho similar às linguagens de programação de alto-nível.

O ambiente de desenvolvimento do software consiste de duas janelas na inicialização: painel frontal (“*Front Panel”*) e diagrama de blocos (“*Block Diagram”*), conforme pode ser observado na Figura 3. No painel frontal encontram-se controles e indicadores (entrada/saída e displays) enquanto que no diagrama de blocos é onde se localiza os terminais correspondentes do painel frontal com as funções (VIs), ou seja, o código e o funcionamento do programa.

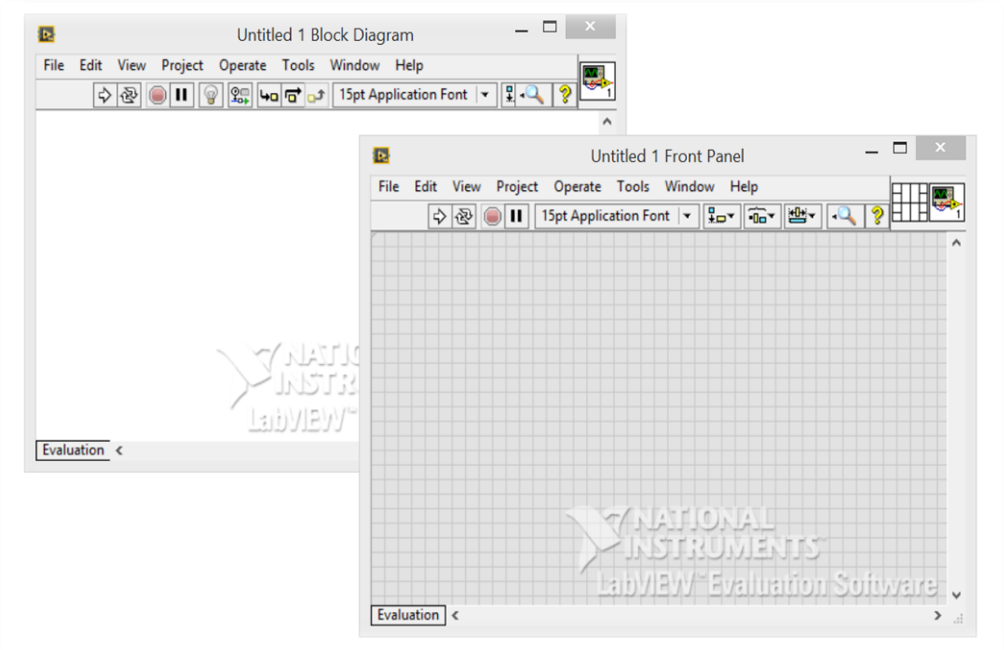


Figura 3 – Imagem de um painel frontal e de um diagrama de blocos do LabVIEW.
Fonte: Autoria própria

Quando existem tarefas repetitivas ou com grande quantidade de blocos, o LabVIEW permite criar subVIs que permitem simplificar e organizar o ambiente de programação. Dessa forma o código é encapsulado tornando o ambiente visual mais limpo e fácil de programar.

3.2. Vantagens de usar o LabVIEW

Segundo [7], a principal vantagem da abordagem gráfica é que o principal foco está no dado em si, e não no processo de manipulação do mesmo. Dessa forma, grande parte da complexidade da programação, como alocação de memória e sintaxe da linguagem, são eliminados de forma a simplificar o problema.

Também é relatado que novos programadores possuem uma curva de aprendizagem mais curta com a linguagem G do que com outras linguagens devido a facilidade em representar o código para gráficos de fluxo e outras representações visuais familiares. Programadores mais experientes também podem adquirir maior produtividade ao trabalhar com níveis mais elevados de abstração enquanto trabalham com programação orientada a objeto e encapsulamento, por exemplo. Além disso, o software apresenta uma interface intuitiva e de fácil utilização.

O LabVIEW também possui diversas bibliotecas de processamento e análise de sinais, controle de algoritmos, comunicação, entrada e saída de arquivos e conectividade que permitem maior facilidade para solucionar problemas. Além disso, o software fornece liberdade para o usuário escolher entre a facilidade de uso e o baixo nível de flexibilidade.

3.3. Softwares existentes

Através do site da National Instruments é possível ter acesso a uma vasta opção de aplicativos desenvolvidos pelos usuários para solucionar os mais diversos problemas. Também dentro do software há diversos exemplos de demonstrações que podem ser alteradas e adaptadas de acordo com o programador.

Entre os aplicativos disponíveis para download, há o "*Biomedical Toolkit*" que é um kit de ferramentas para uso na área biomédica. Nesse kit existe o software "*3D Reconstructor*", conforme mostra a Figura 4, e no qual através dele é possível fazer o upload de imagens médicas para reconstrução tridimensional e análise de biosinais. A Figura 5 mostra três exemplos das imagens utilizadas como exemplo para a execução da renderização e visualização do volume

tridimensional. Essas figuras se encontram dentro de uma pasta específica do programa e podem ser acessadas pelo diretório. É importante comentar que essas figuras possuem uma perceptível alta nitidez e delimitação dos tons de acordo com a profundidade. A Figura 6 mostra o poder de renderização do programa. Comparativamente, as imagens geradas em laboratório pelo OCT possuem demasiada quantidade de ruídos. E esses ruídos prejudicam a renderização gerando como resultado uma imagem de baixíssima qualidade

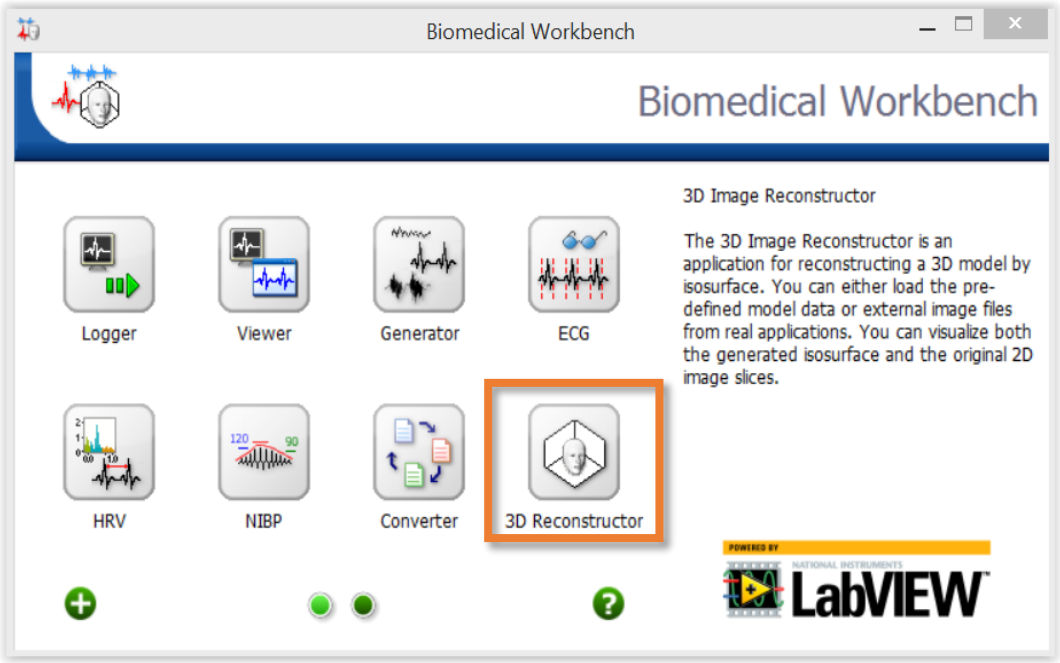


Figura 4 – Área de trabalho do kit biomédico do Labview com o atalho para o 3D Reconstructor.

Fonte: Autoria própria

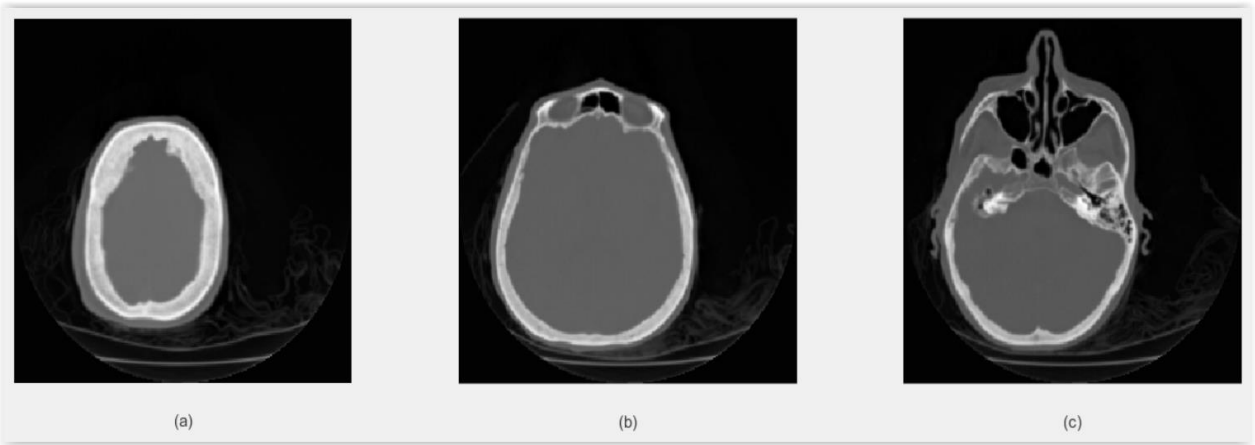


Figura 5 – Imagens OCT disponibilizadas no diretório do programa "*3D reconstructor*"

Fonte: Autoria própria

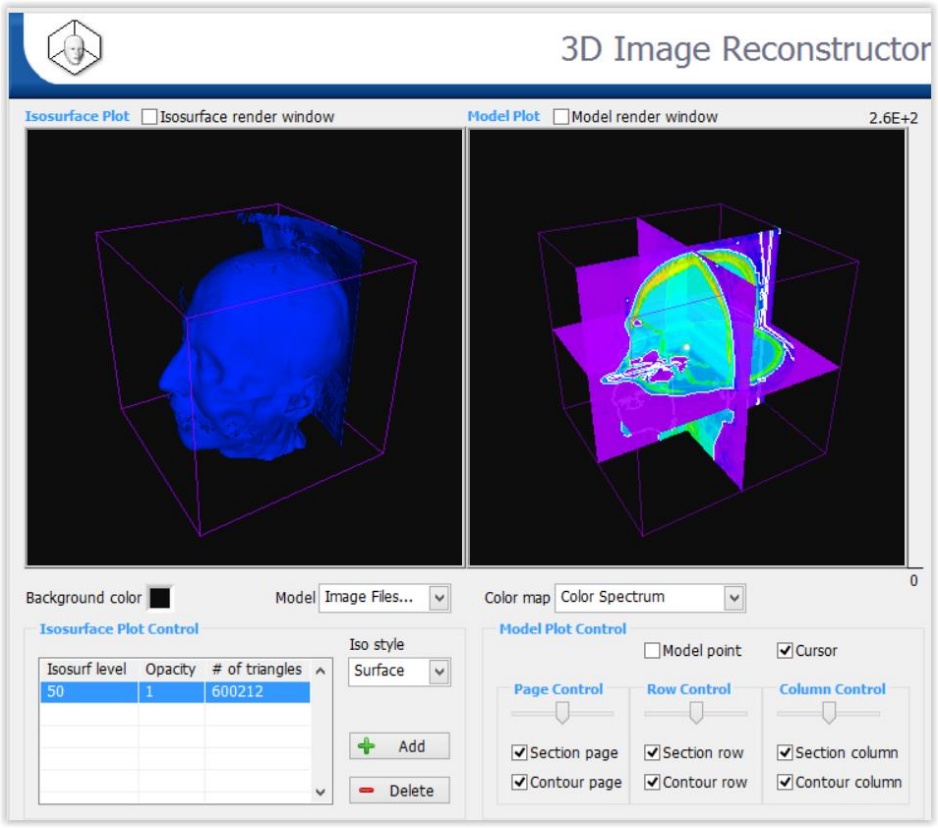


Figura 6 – Interface do aplicativo reconstrutor de imagem 3D dentro do "*Biomedical Toolkit*" com as imagens de demonstração do próprio programa.
Fonte: Autoria própria

Através de um dos fóruns de discussão sobre plotagem gráfica, foi possível descobrir que o LabVIEW já possui uma função específica para a alocação de pontos em suas respectivas dimensões e a consequente geração gráfica dos dados. Essa função chama-se Cwgraph3D, onde o ícone da VI pode ser visto na Figura 7, e ela é responsável pelo controle gráfico tridimensional das informações.

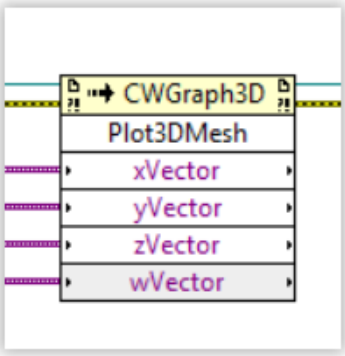


Figura 7 – Ícone de VI responsável por extrair os vetores dimensionais de uma imagem.
Fonte: Autoria própria

4. FILTROS EM LABVIEW E MATLAB

4.1. Filtro passa-baixa do LabVIEW

O LabVIEW possui uma paleta de filtros adaptativos dentro da seção "*Vision and Motion*". Entre eles, o filtro "IMAQ *LowPas*", mostrado na Figura 8, é uma função programada para tratar ruídos de imagens utilizando o conceito de um filtro passa-baixa. Esse filtro calcula a variação inter-pixel entre o pixel processado e o da vizinhança [9]. Por conta disso, ele permite ajustar o valor médio do pixel calculado a partir dos outros ao redor, para o caso de o pixel que está sendo processado possuir uma variação maior que o percentual determinado. Além disso, o "IMAQ *LowPas*" permite o usuário controlar o tamanho da matriz de cálculo e ajustar os parâmetros de tolerância.

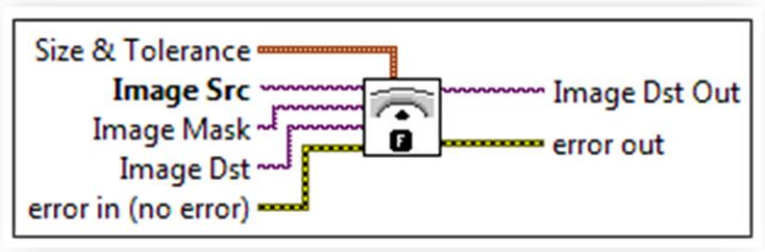


Figura 8 – Ícone do filtro passa-baixa do LabVIEW com as entradas de dados e suas respectivas funções. Fonte: Autoria própria

4.2. Filtros do MATLAB

No programa MATLAB é possível ter acesso a diversas rotinas pré-programadas de filtros para processamento de imagens. Entre eles, alguns filtros são responsáveis pela conversão dimensional da imagem e outros reduzem a quantidade de ruídos no qual, consequentemente, causam significativa melhora qualitativa.

Com base em um limiar fixo, o filtro "im2bw" é capaz de modificar a imagem para uma imagem do tipo binária. Esse filtro converte a imagem de entrada para um formato em escala de cinza e então para binária [10] conforme pode ser visto na Figura 9.

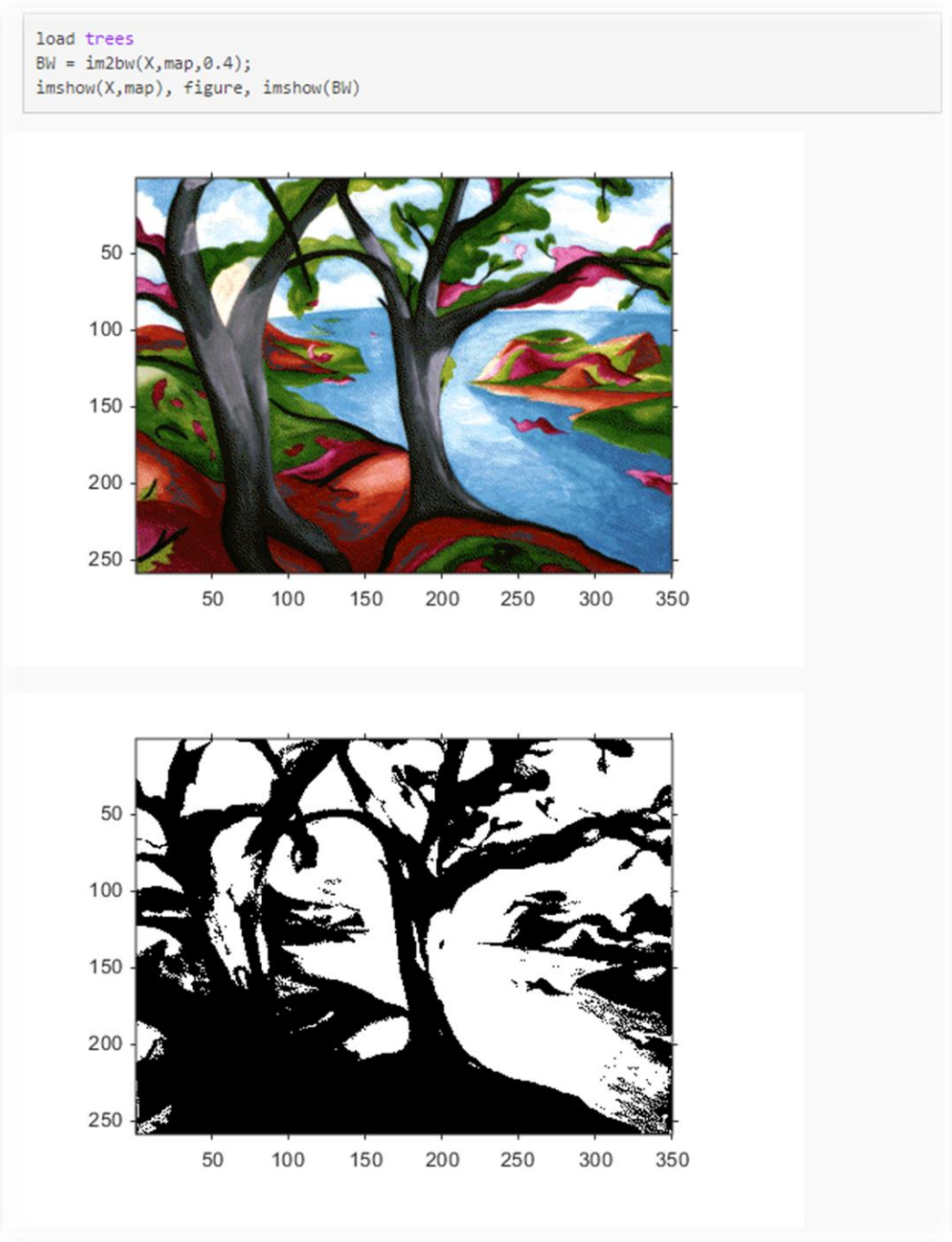


Figura 9 – Exemplo de uso do filtro "im2bw" e o resultado das imagens.
Fonte: Documentation Center, Mathworks.

O filtro não-linear "medfilt2" realiza a filtragem mediana de uma matriz de duas dimensões [11]. Cada pixel de saída contém o valor médio dos pixels ao redor correspondente à imagem de entrada fazendo com que reduza a presença de sinais indesejados e preserve o contorno, como mostra a Figura 10.

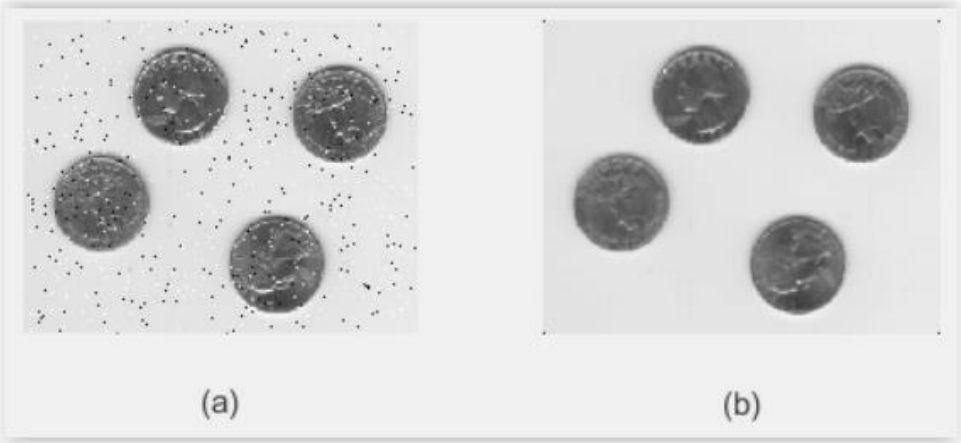


Figura 10 – Exemplo de uso do filtro "medfilt2" com a imagem (a) contendo ruídos e o (b) resultado após o processo de filtragem
Fonte: Documentation Center, Mathworks.

De maneira similar, o filtro passa-baixa "wiener2" é um outro exemplo de filtro adaptativo de duas dimensões para a remoção de ruídos. Ele usa um método inteligente de manipulação de pixels baseando-se em estatísticas estimadas a partir da "vizinhança" local de cada pixel [12]. O resultado final pode ser observado na Figura 11.

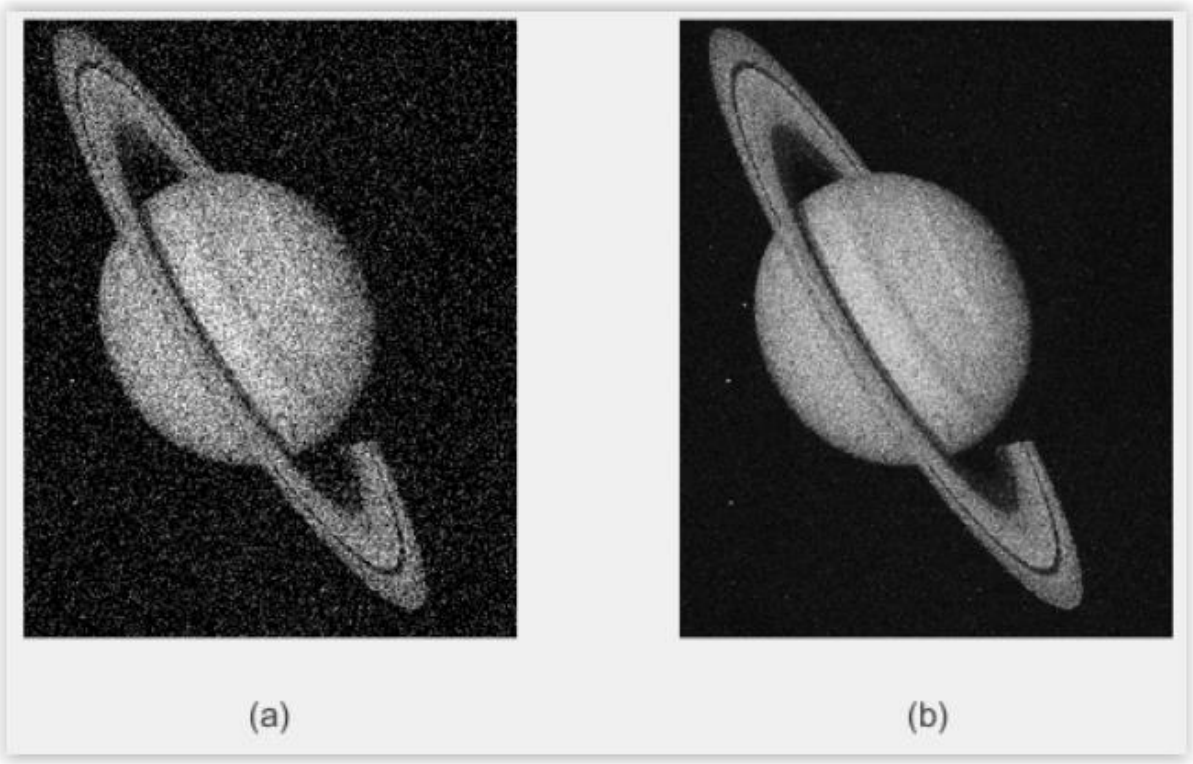


Figura 11 – Exemplo de uso do filtro "wiener2" com a imagem (a) contendo ruídos e o (b) resultado após o processo de filtragem.
Fonte: Documentation Center, Mathworks.

4.2.1. Filtros do MATLAB e equivalência em linguagem C

A sintaxe do MATLAB é muito similar com a linguagem C, com o diferencial de que o MATLAB é mais flexível com o tipo de dado de entrada. Com o intuito de testar a velocidade de processamento do algoritmo do filtro e a qualidade do resultado, o algoritmo foi modificado para a linguagem C. A conversão consiste na adaptação das variáveis para o problema a ser solucionado. Essas variáveis e algumas estruturas podem ser removidas e substituídas por determinados tipos de dados de entrada. Os três filtros foram readaptados e podem ser visto no Apêndice A, Apêndice B e Apêndice C. O objetivo desse passo é criar subVIs para cada filtro de forma que o código esteja encapsulado dentro do programa e que possua uma rápida execução.

5. TESTES E RESULTADOS

5.1. Teste dos filtros do MATLAB

As imagens utilizadas foram gentilmente cedidas pelo prof. Dr. Michael Wang do laboratório de óptica no departamento de engenharia elétrica da Universidade de Miami (*University of Miami*). Juntamente com uma equipe de alunos de doutorado, o grupo tinha por objetivo obter resultados similares ao projeto do prof. Obayashi [1]. As imagens foram coletadas pelo equipamento desenvolvido em [2] e o pacote que continha as melhores amostras possuía 73 sequências de imagens, parcialmente demonstrada na Figura 12. A primeira parte dos testes consistiu na detecção do filtro que melhor se adaptava para solucionar o problema e definir os parâmetros a serem utilizados quando requisitados.

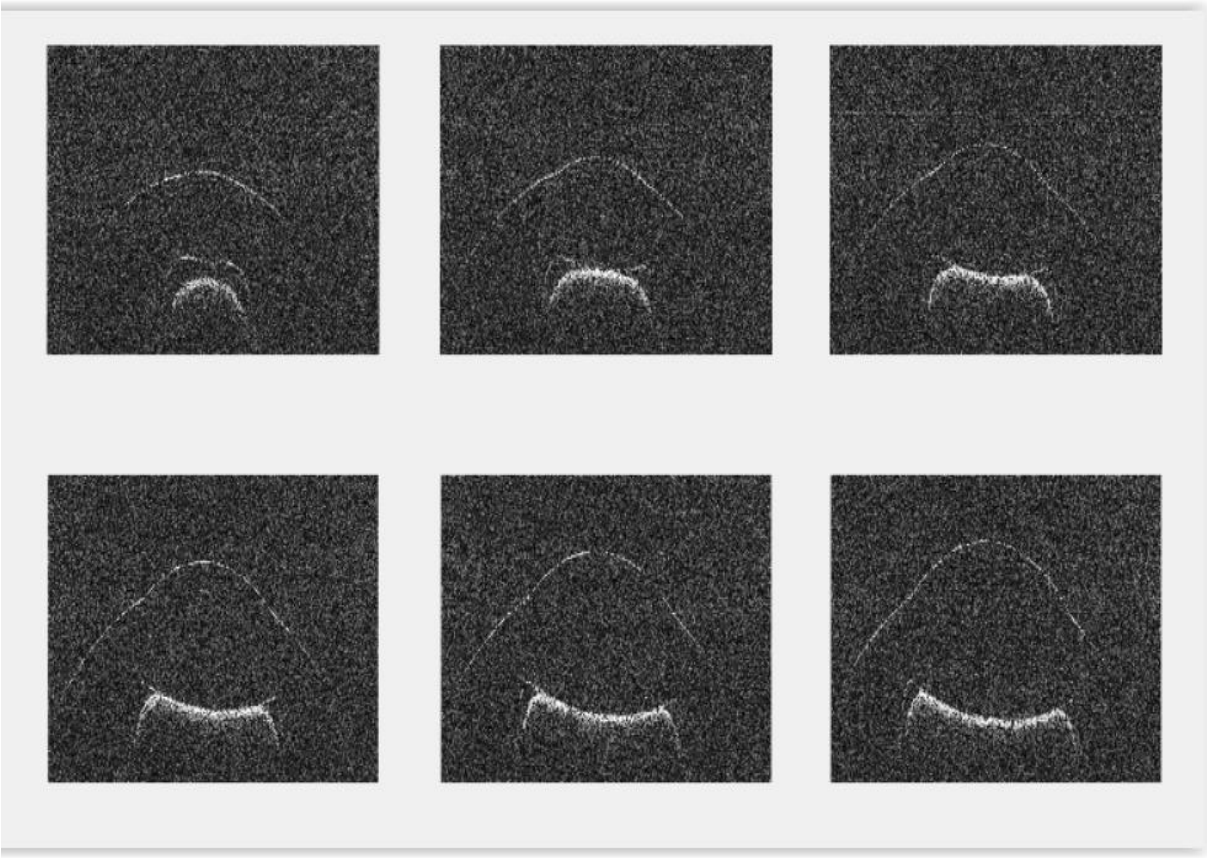


Figura 12 – Exemplo de 6 imagens em sequência de um modelo de olho humano escaneado utilizando um equipamento OCT.
Fonte: Lui (2014).

Utilizando os filtros do MATLAB anteriormente descritos na seção 4.2, é possível perceber o comportamento individual dos filtros sobre a mesma imagem. Como as imagens eram tridimensionais (pois continham valores de intensidade RGB), foi-se necessário alterá-la para o tipo binário utilizando o filtro “im2bw” como exibido na Figura 13 ou para tons de cinza através da função “rgb2gray”.

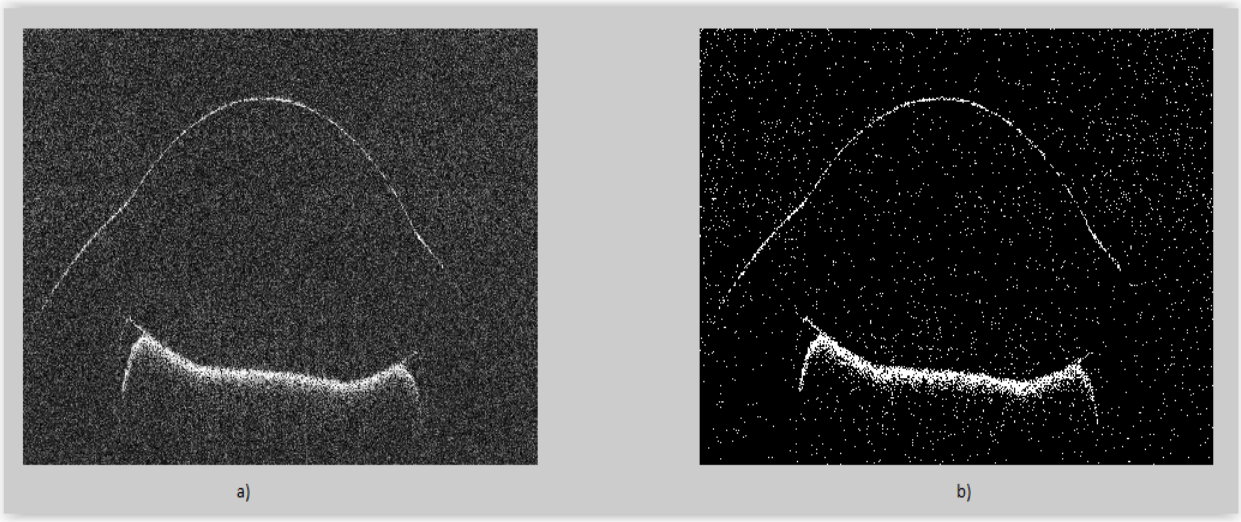


Figura 13 – Imagem (a) original de um modelo de olho humano escaneado por OCT e a mesma imagem (b) alterada usando o filtro “im2bw” do MATLAB.
Fonte: Adaptado de Lui (2014).

Para observar o efeito dos filtros, foi utilizado primeiramente o comando “rgb2gray” que é responsável por converter a imagem RGB para tons de cinza. Em seguida foi feito o teste utilizando o filtro “medfilt2”. Apesar de possuir a opção para escolher os parâmetros de matrizes que serão usadas para o cálculo, o código sem os parâmetros apresentou uma boa qualidade gerando uma imagem mais suavizada e com contornos nítidos que podem ser observados na Figura 14.

No teste do filtro “wiener2” foi preciso escrever um script para testar as combinações que gerassem melhores resultados. Para não consumir muita memória, foi pré-determinado que seriam testadas inicialmente matrizes de dimensões iguais (2x2, 3x3, ...). Após diversos testes e comparações, a matriz que melhor apresentou resultados era a de 3x3 por deixar a imagem mais suave, onde pode-se observar na Figura 15.

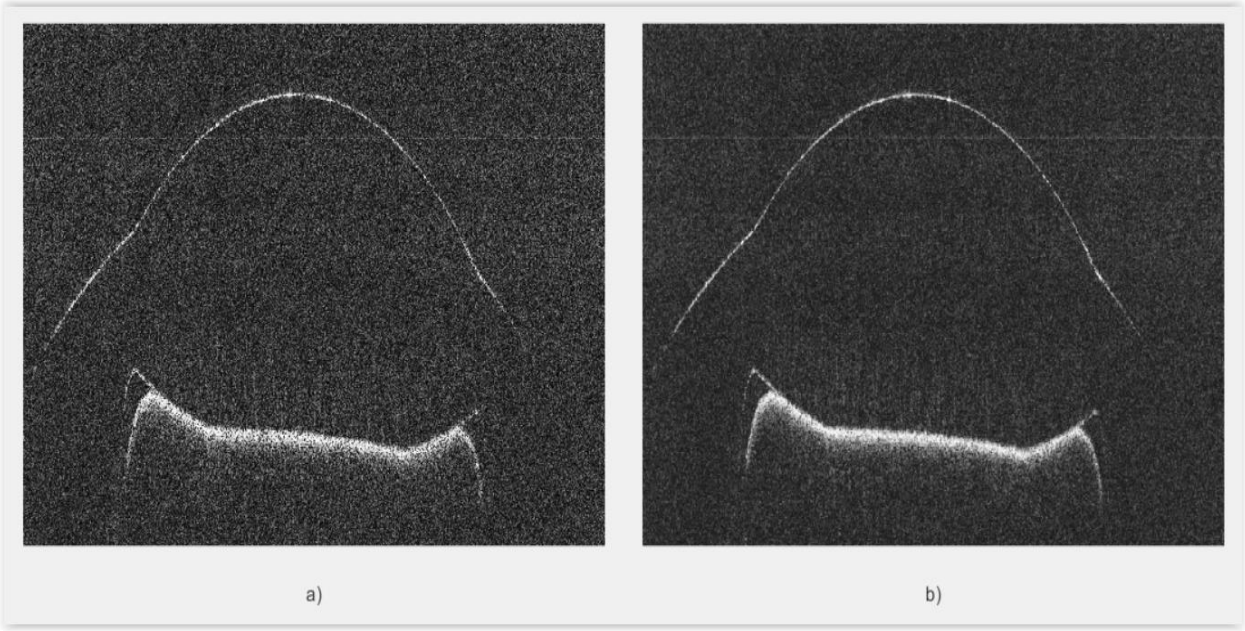


Figura 14 – Imagem (a) original de um molde de olho humano e (b) após o uso do filtro “medfilt2”.
Fonte: Adaptado de Lui (2014).

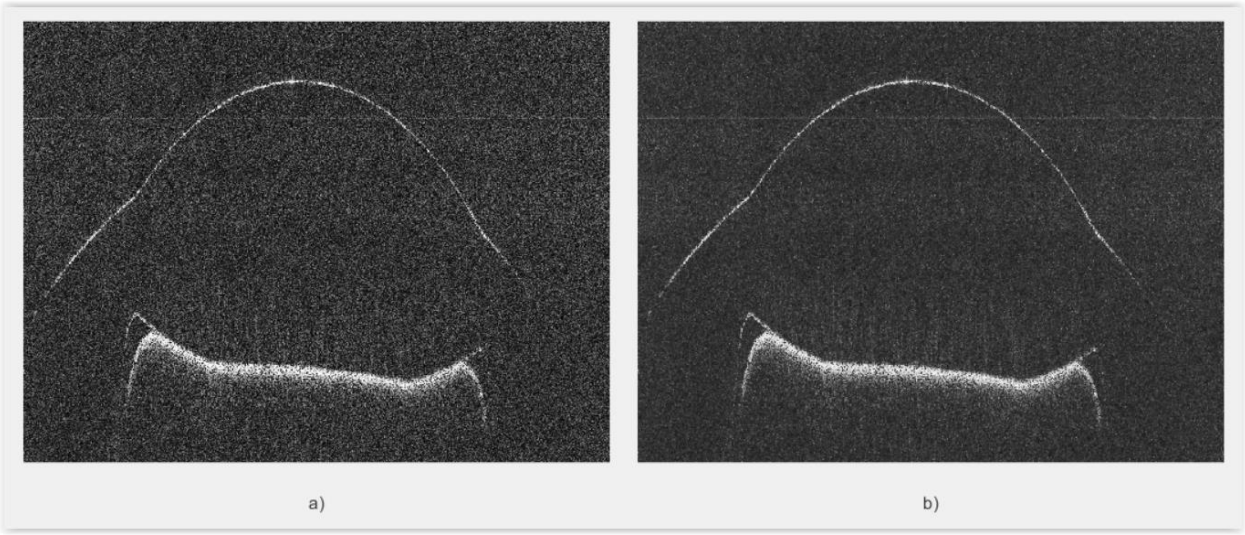


Figura 15 – Imagem (a) original de um molde de olho humano e (b) após o uso do filtro “wiener2” com a matriz [3 3] como parâmetro de entrada.
Fonte: Adaptado de Lui (2014).

As imagens foram armazenadas para serem testadas no LabVIEW ao final do projeto e serem intercombinadas para verificar a melhor ordem de combinação dos filtros a ser aplicado a toda sequência de imagens.

5.2. Teste do filtro em LabVIEW

Com o intuito de testar o filtro em LabVIEW, foi feito um pequeno código para leitura da imagem, manipulação e exibição do resultado. Dentro desse código, o tamanho da imagem também era reduzido para poupar espaço e tempo de processamento.

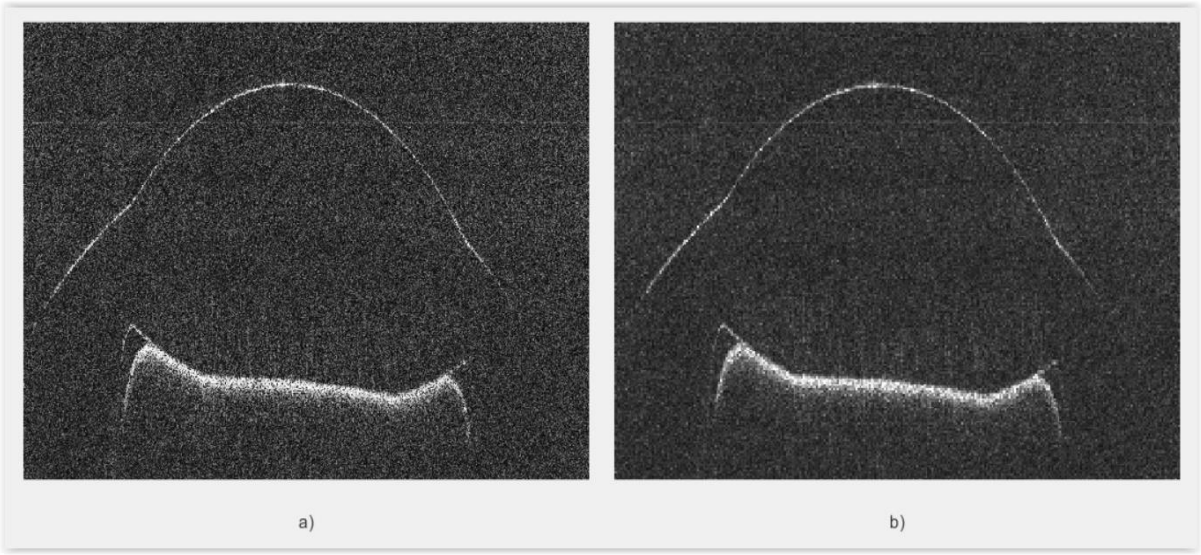


Figura 16 – Imagem (a) original de um molde de olho humano e (b) após o uso do filtro passa-baixa "IMAQ Low Pass" do LabVIEW.
Fonte: Adaptado de Lui (2014).

O filtro em LabVIEW (resultado na Figura 16) também fornece a opção de determinar o parâmetro da matriz que pode ser determinado pelo painel frontal da interface. A Figura 17 mostra a interface gráfica intuitiva onde pode-se ver dois displays gráficos que exibem a imagem original (visor "*image*") e a imagem reduzida e filtrada (visor "*image 2*"). No lado esquerdo, há painéis (melhor visualizado na Figura 18) para a inserção da quantidade de imagens a serem abertas, do caminho da pasta contendo as figuras, o nome, a extensão com o tipo e a pasta de destino onde será salvo as novas imagens.

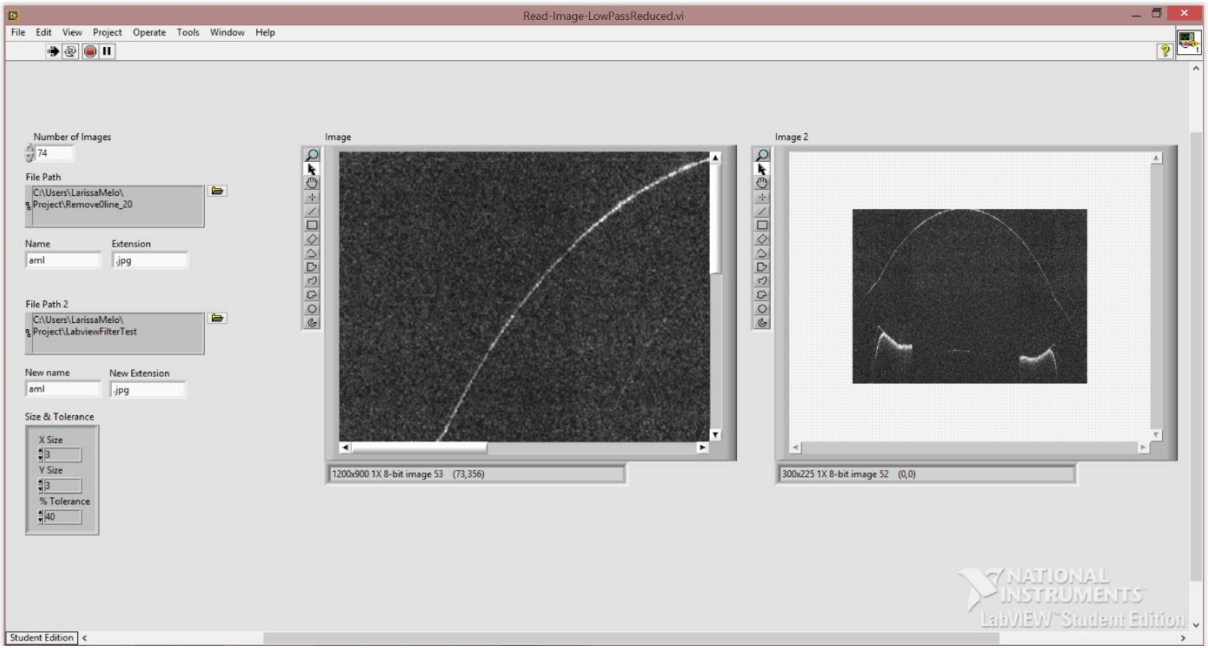


Figura 17 – Interface gráfica em execução do software para leitura e aplicação do filtro passa-baixa em LabVIEW.
Fonte: Autoria própria.

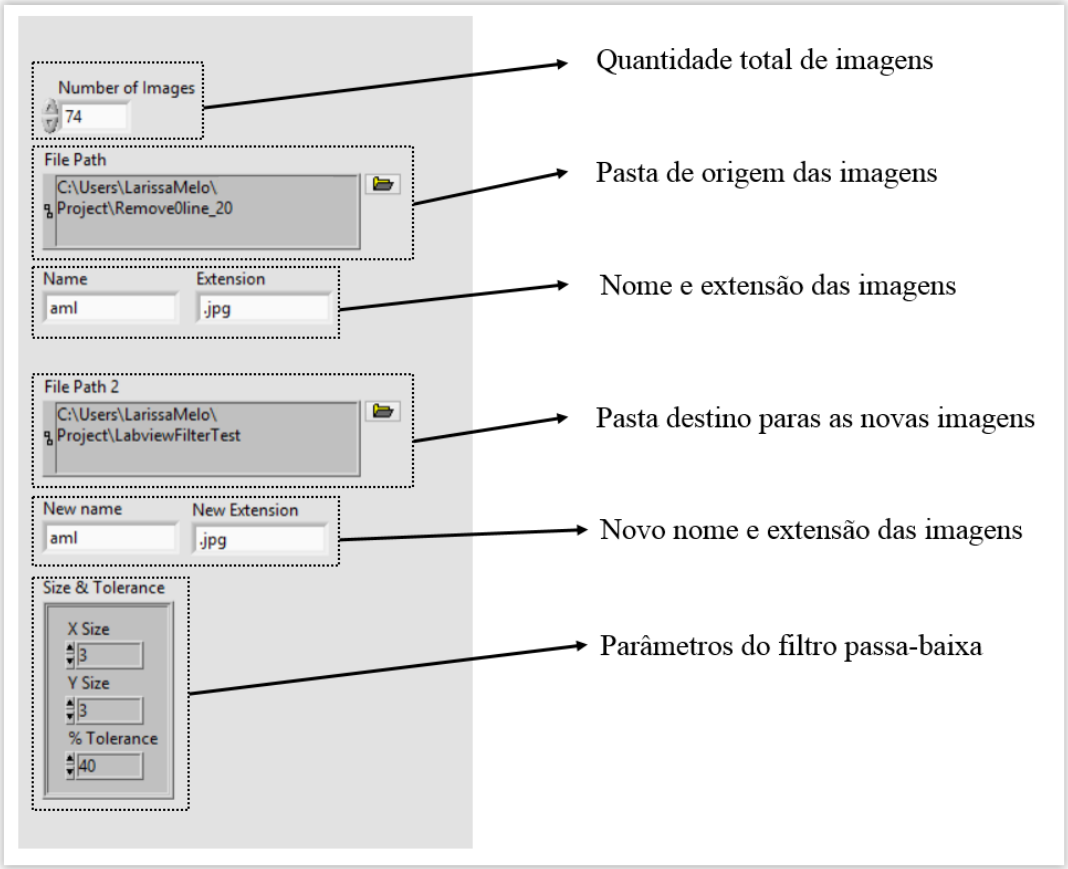


Figura 18 – Detalhe do painel comentado com as respectivas funções.
Fonte: Autoria própria

As imagens devem ser salvas com nomes iguais e com um número como sufixo, sendo enumeradas diferentemente, por exemplo “aml24”, “aml25”, etc. No diagrama de blocos, o programa funciona dentro de um laço por onde percorre as imagens pelo nome e numeração. O diagrama de blocos está disponibilizado em totalidade no Apêndice D, enquanto que a seguir será descrito os processos da execução parcialmente.

A imagem da Figura 19 mostra a primeira parte do programa em detalhe onde compreende a leitura inicial dos dados. Nessa imagem, o processo foi dividido em 4 etapas sequenciais. A parte delimitada em 1 corresponde ao laço de execução com o número de vezes que o programa tem que percorrer a pasta até coletar todas as imagens necessárias. A etapa 2 equivale ao processo de indicação do caminho dos arquivos. O bloco “*File Path*” é um ponto de entrada de dados com o local do diretório a ser aberto. No bloco correspondente ao painel frontal o usuário cola o *path* da pasta que contém todas as imagens. Os blocos “*Name*” e “*Extension*” são concatenados para formar o nome do arquivo e a extensão com o tipo de imagem. No bloco 3, encontra-se o sufixo numérico de cada figura. Conforme o valor de “i” vai sendo incrementado, esses valores são transformados do tipo numérico para dado do tipo *string* no qual será adicionado novamente na etapa 2 e formará o nome completo dos arquivos a serem manipulados sequencialmente. A etapa 4 corresponde a leitura dos dados em si. O bloco “IMAQ” é responsável por criar um local temporário de memória para as imagens abertas. O bloco ao lado lê as imagens no formato pré-identificado. Por fim, na etapa 5 ocorre o envio dos dados para um vetor (bloco “*Image to Array*”) que são enviados para o bloco “*image*” exibir as imagens originais no display.

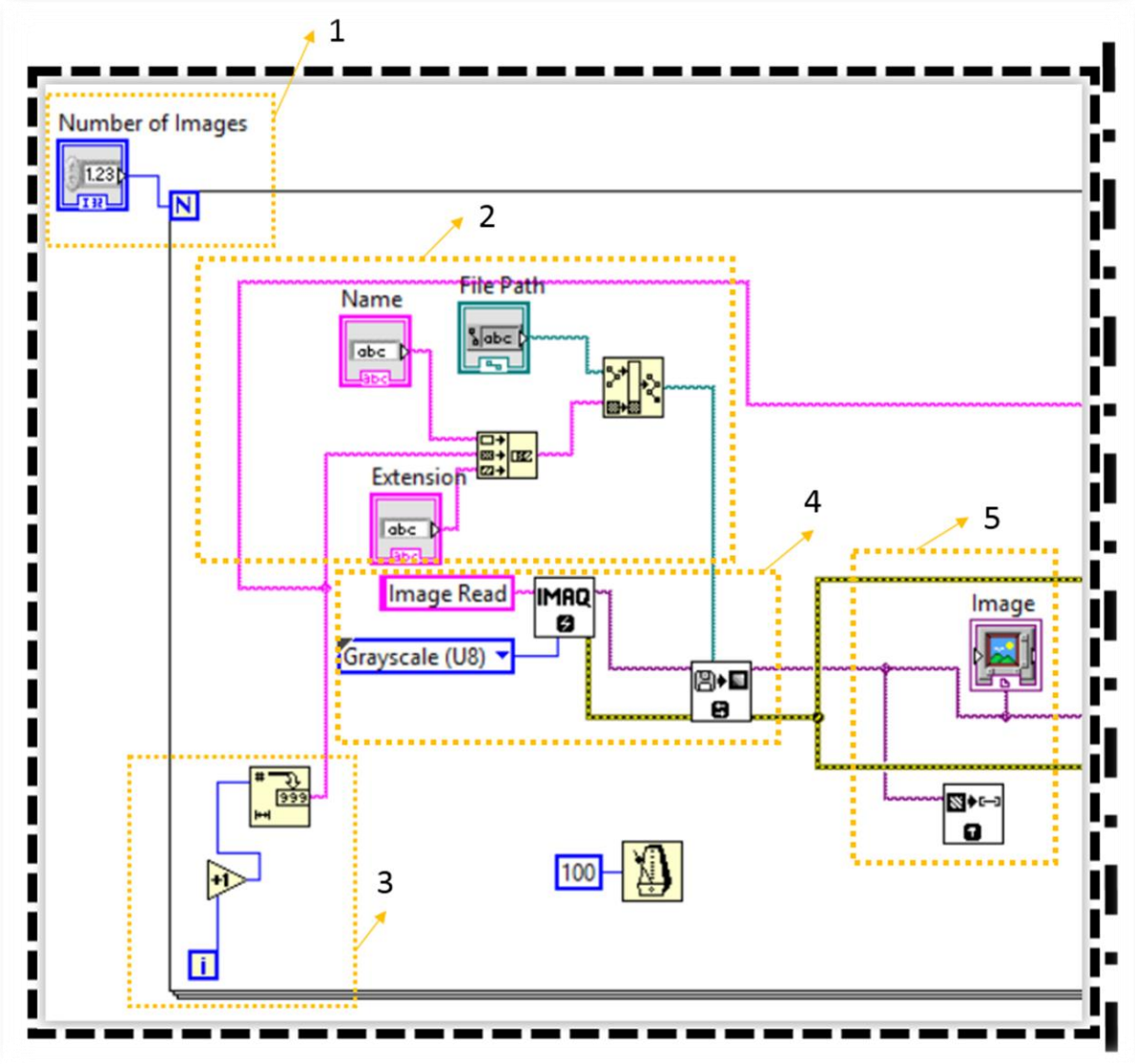


Figura 19 – Código parcial da leitura e exibição de imagens em LabVIEW.
Fonte: Autoria própria.

Após essa primeira parte, começa a parte de processamento e filtragem das imagens. A Figura 20 mostra os passos enumerados em continuação com a figura anterior para manter a ideia de execução em fluxo de dados do programa. Na etapa 6 ocorre a filtragem de cada imagem e o bloco "*Size & Tolerance*" é uma VI de entrada no qual o usuário pode alterar os parâmetros correspondentes ao tamanho da matriz e o percentual de tolerância no painel frontal. O bloco localizado abaixo é o filtro passa-baixa. Em seguida, as imagens são reduzidas ao tamanho 225x300 (equivalente a 1/3 do tamanho da imagem original) no ícone do módulo "IMAQ *Resample*" e são armazenadas em um vetor temporário para serem exibidas no display "*Image 2*". No passo 7 ocorre o processamento para alocar os novos nomes e extensões no novo diretório descrito em "*File Path 2*" de modo similar ao passo 2 da Figura 19. No oitavo e último

passo, ocorre o armazenamento das imagens no novo diretório com seus respectivos nomes descritos no passo anterior.

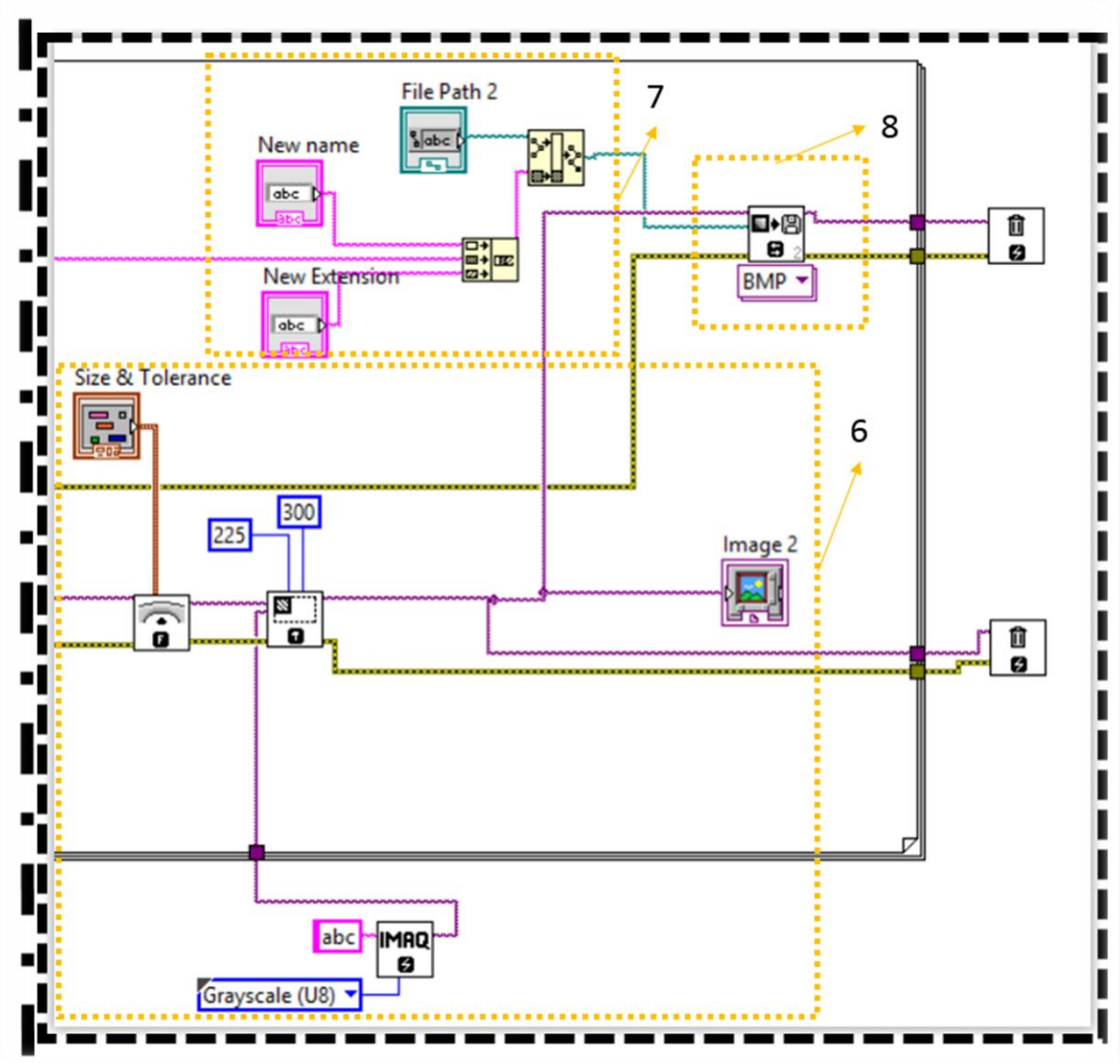


Figura 20 – Código parcial da filtragem e armazenamento de imagens em LabVIEW.
Fonte: Autoria própria.

5.3. Código em LabVIEW

O desenvolvimento do código em LabVIEW se baseou no princípio de plotagem de um gráfico de 4 dimensões onde a quarta dimensão representa a intensidade. No fórum da *National Instruments Comunity* foi postado um exemplo de um display gráfico para um vetor de quatro dimensões [13] e o princípio de rearranjo de dados foi utilizado como referência no

desenvolvimento do código proposto. No Anexo A encontra-se o painel frontal do programa, enquanto que no Anexo B está o diagrama de blocos por completo. O programa desenvolvido em [13] utiliza três dimensões de valores numéricos aleatórios como dados de entrada a serem plotados e uma quarta dimensão para representar a intensidade dos pixels, na qual é simbolizada pelas cores do gráfico.

A Figura 21 indica a parte do código que serviu de referência à programação do projeto. A parte interna da linha tracejada delimita a parte de processamento gráfico para a plotagem da informação tais como os blocos de ajuste das propriedades de plotagem (ex. transparência e estilo), conversão de cores e seus respectivos mapas de cores. A parte fora da linha tracejada compreende a entrada dos dados a serem utilizados e adaptados ao problema proposto.

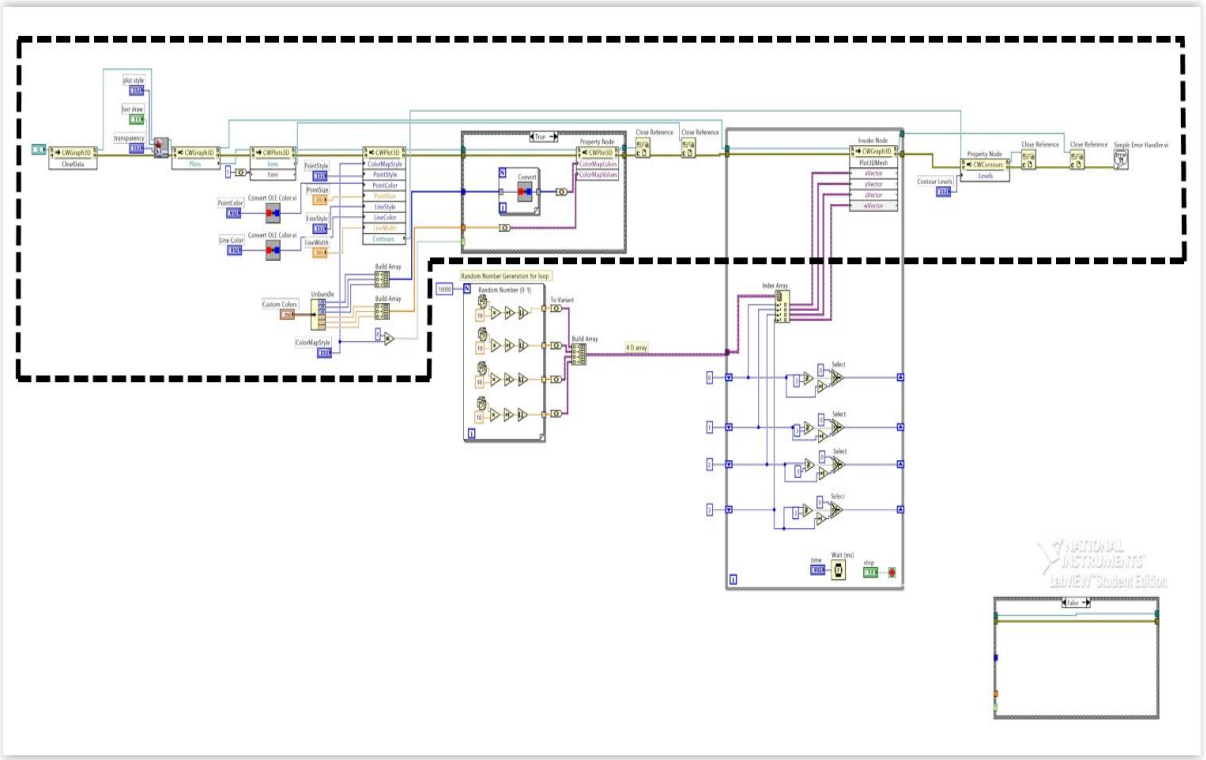


Figura 21 – Parte utilizada como referência do diagrama de blocos de um display gráfico de 4 dimensões. Fonte: Graphically displaying 4D array, 2011.

A primeira parte adicionada foi o código referente a leitura de imagens que está detalhado no Apêndice D. Com essa primeira parte funcionando, o programa já era capaz de abrir, filtrar e exibir as imagens dentro do mesmo código. Todo o processo de inserção de código era feito por partes, testado para verificar o funcionamento e salvo. O fluxograma do processo pode ser melhor visualizado na Figura 22. Nela pode-se observar que enquanto o código funcionasse mas ainda não resolvesse o problema final para gerar o modelo

tridimensional, era necessário voltar ao passo inicial para adicionar mais uma etapa e continuar o processo. Quando o programa era reprovado no teste era necessário retornar às partes exibidas com erro e corrigi-las.

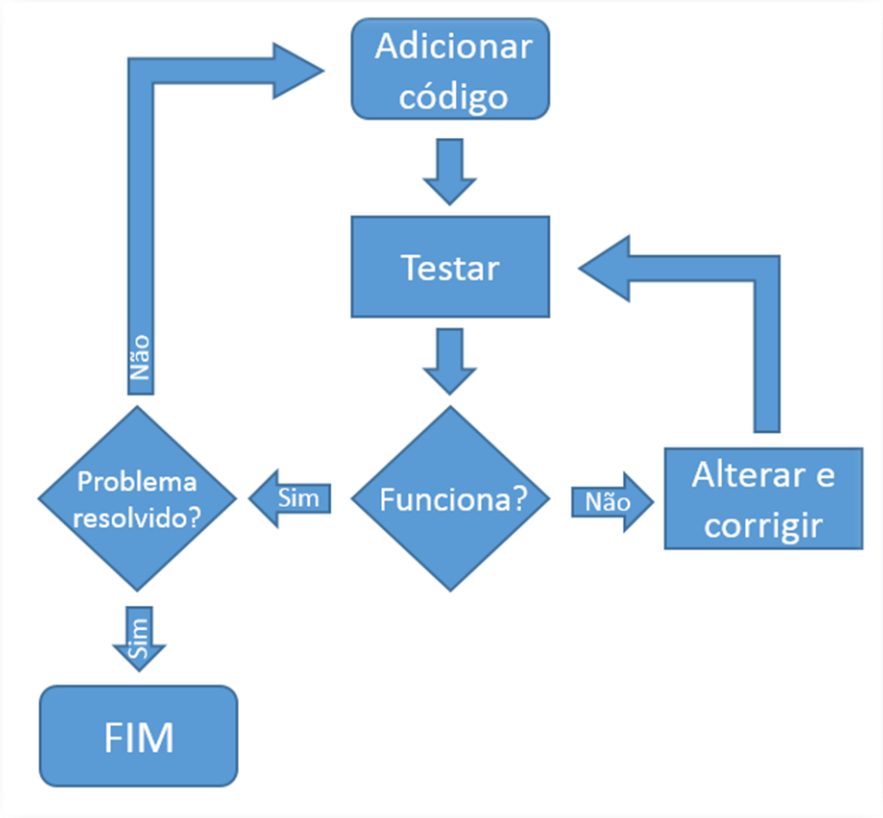


Figura 22 – Fluxograma da programação do código em LabVIEW.
Fonte: Autoria própria.

A lógica para o processamento das imagens corresponde a varredura da posição dos pixels de cada imagem e a alocação nos eixos X e Z. O vetor Y é responsável por armazenar a posição da imagem para montar a pilha no gráfico enquanto que a intensidade era armazenada em W. Uma estrutura sequencial foi utilizada para manter a certeza de que um subdiagrama seria executado após o outro e forçar a sequência das ações. Para essa parte do programa, o código vasculha as coordenadas de uma imagem, compara se os valores de intensidade do pixel dessas imagens estão dentro dos limites pré-estabelecidos empiricamente e os armazena nos vetores.

Como era necessário detectar os pontos que estivessem mais claros, as intensidades mais importantes eram aquelas que possuíssem valores mais elevado. Esses valores foram estimados com base no histograma da Figura 23 de forma a reduzir o tempo computacional comparado se fosse utilizar todos os pontos da figura. Através do histograma pode-se perceber que as partes

de concentração de pontos claros encontram-se na faixa de intensidade 100 a 256. Por conta disso, foram criados três grupos de valores de intensidades nos quais os parâmetros de delimitação a serem buscados estivessem entre 90 e 105; 200 e 256 e iguais a 170. As posições dos pixels que satisfizessem essas condições eram armazenadas em três vetores diferentes que correspondessem as coordenadas X e Z, enquanto que os valores de intensidade em si eram armazenados em W e o número da imagem analisada era armazenada diretamente em Y. No final de todas as iterações, os vetores eram concatenados em um vetor único do mesmo tipo, por exemplo, os vetores "X Vector 1", "X Vector 2" e "X Vector 3" foram armazenados no vetor "X Vector".

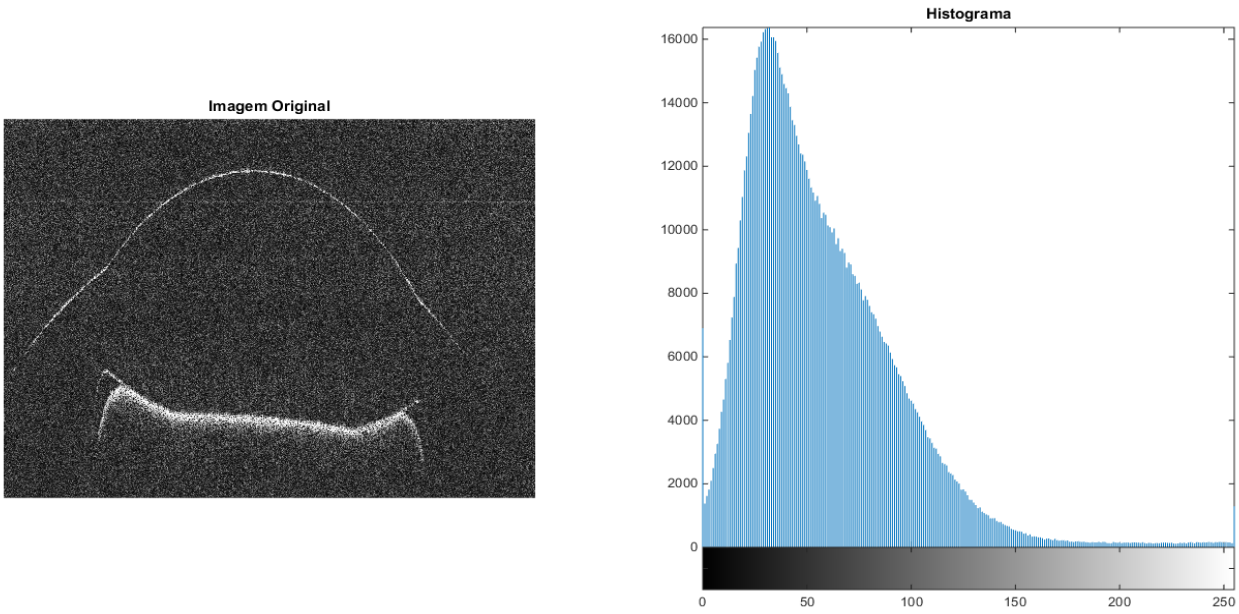


Figura 23 – Imagem original e histograma feitos em MATLAB para detecção de pontos de interesse da imagem.

Fonte: Autoria própria.

Pela Figura 24 pode-se visualizar melhor os detalhes de cada estágio de processamento. No passo 1, a linha que está em volta de todos os blocos é uma VI de estrutura sequencial que forçará a execução das partes 2 e 3 primeiramente e depois executará a parte 4. A parte 2 demonstra as restrições de busca. Os blocos da parte 3 são responsáveis pela alocação das informações nos vetores específicos. O mesmo acontece no bloco inferior para quando o pixel é igual ao valor 170. Na parte 4, todos os vetores são concatenados em um *array* para cada tipo conforme explicado no parágrafo anterior. A parte 6 corresponde a fase final onde há o

processamento visual dos pontos detectados para a posterior exibição no display gráfico no painel frontal.

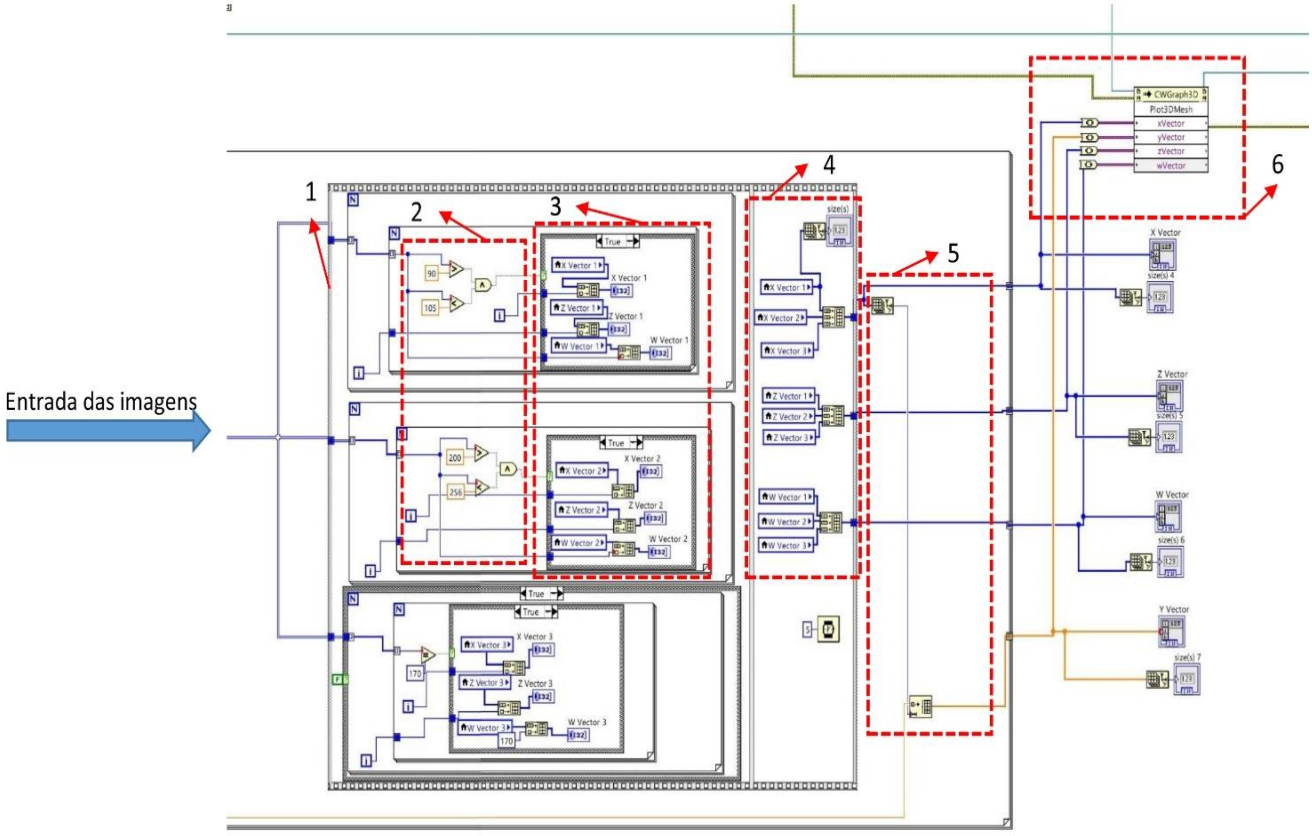


Figura 24 – Detalhe da parte do processamento dos pixels do Código em LabVIEW.
Fonte: Autoria própria.

5.3.1. Integração entre LabVIEW e MATLAB

Como somente o filtro do LabVIEW não era suficiente para melhorar a qualidade das imagens durante o processamento, era necessário que o programa fosse compatível com o MATLAB. O LabVIEW possui VIs específicas para scripts de funções dentro da paleta de funções chamada "*Script Node*", conforme pode ser vista na Figura 25. Entretanto, a ideia inicial consistiu em otimizar o processamento do filtro ao forçar somente o LabVIEW a executar o algoritmo. Com esse passo não seria necessário ter o MATLAB instalado na mesma máquina que executasse o programa.

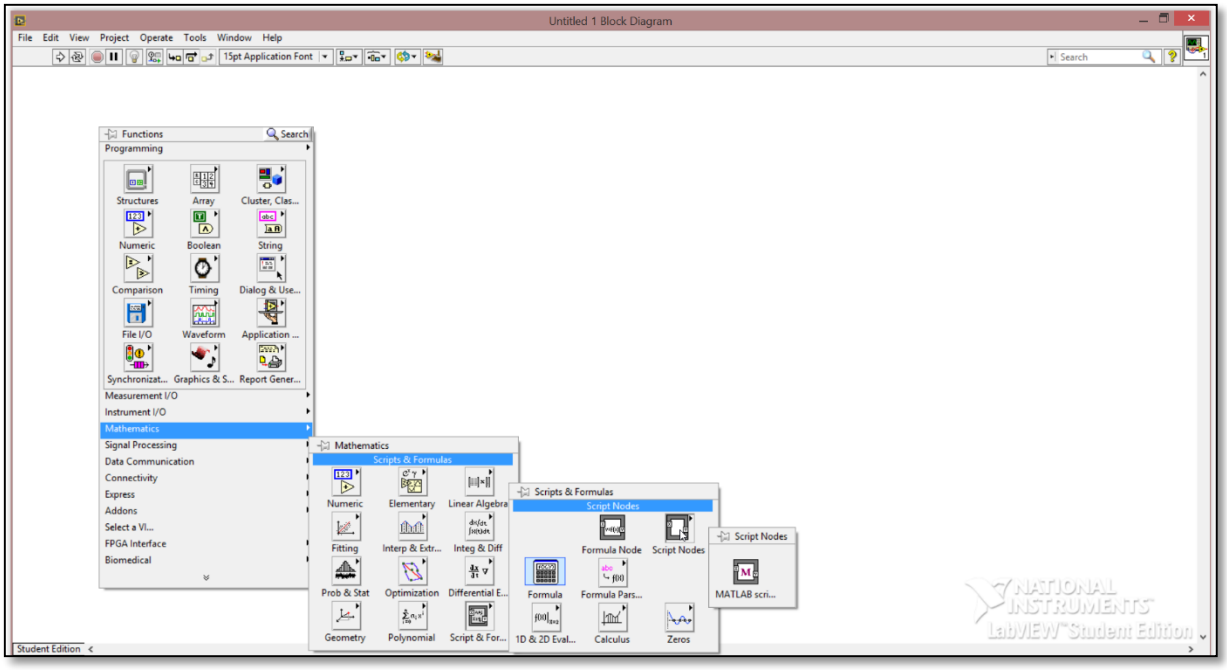


Figura 25 – Paleta de funções de scripts no Labview.
Fonte: Autoria própria.

No primeiro bloco chamado de "*Formula Node*" é possível adicionar códigos em linguagem C enquanto que no segundo bloco do "MATLAB *script*" pode-se inserir códigos na sintaxe do programa. Dessa forma, o código dos filtros em MATLAB foram transformados para linguagem C (vide Apêndice A, Apêndice B e Apêndice C) e inseridos no bloco em branco conforme mostra a Figura 26.

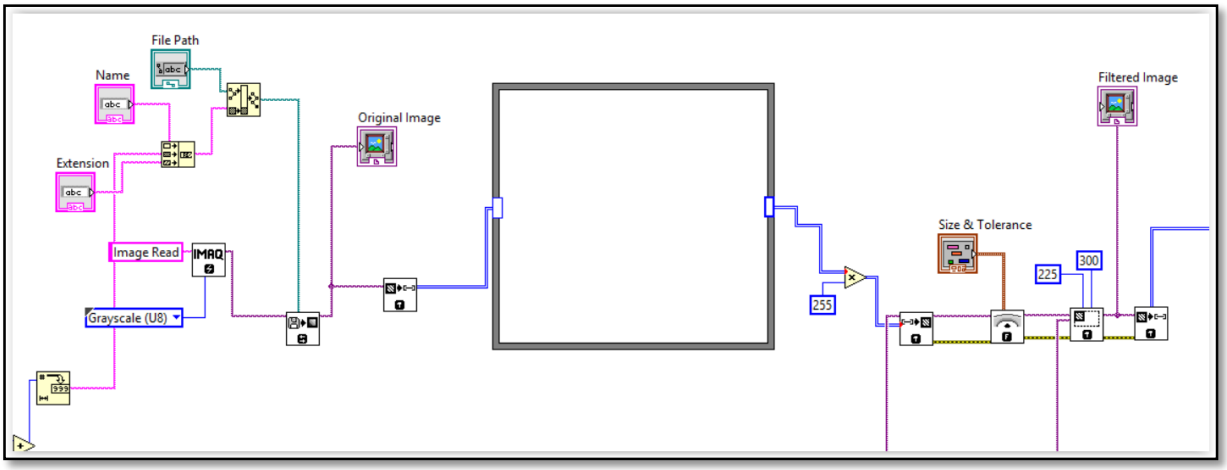


Figura 26 – Parte do código para a inserção de funções em linguagem C.
Fonte: Autoria própria.

Entretanto, houveram falhas de compatibilidade pois a conversão do algoritmo não foi total. Por utilizar funções específicas no algoritmo, a VI não conseguia compilar completamente. Dessa forma, foi necessário trocar o bloco de funções gerais para o específico do MATLAB. Devido a isso, é necessário que o computador que execute o programa também tenha o MATLAB instalado. Caso contrário, a parte com o código dos filtros não será executado. Após diversos testes empíricos com as imagens descritas no capítulo 5.1 determinou-se que os melhores resultados foram obtidos com a sequência de filtros descritos na Figura 27.

```
passo1 = im2bw(Figura);
passo2 = medfilt2(passo1);
passo3 = wiener2(passo2, [3 3]);
```

Figura 27 – Sequência de filtros do MATLAB para obter o melhor resultado.
Fonte: Autoria própria.

Esses filtros tornam mais visíveis os pontos de interesse da figura, gerando resultados mais satisfatórios conforme pode-se analisar na Figura 28, que foi feita no MATLAB. Na Figura 28.b) a imagem original sofreu alteração do primeiro filtro e percebe-se uma elevada quantidade de ruídos ao redor. Na imagem seguinte, o filtro "medfilt2" eliminou quase todo o ruído da imagem, mas a linha do traço ainda possui uma espessura muito fina. Com o último filtro ("wiener2"), as espessuras das linhas externas são engrossadas e realçadas (Figura 28.d).

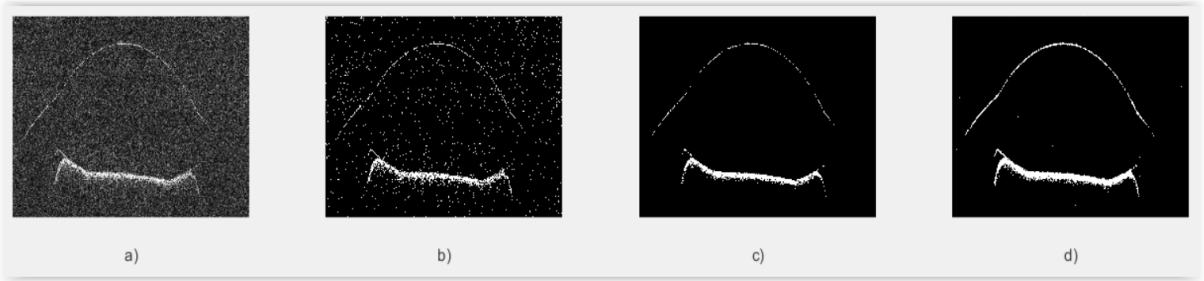


Figura 28 – Imagem a) original e estágios de aplicação do filtro b) "im2bw", c) "medfilt2" e d) "wiener2" em sequência.
Fonte: Adaptado de Lui (2014).

No LabVIEW o *script* teve que ser adaptado para receber as imagens. Essas adaptações foram definidas empiricamente e testadas diretamente no MATLAB. Inicialmente, as imagens tinham que ser convertidas para 8 bits e a saída tinha que ser convertida para o tipo *double*, vide

Figura 29. No final da execução utilizou-se uma VI para converter esse dado em 8 bits novamente com o alcance de 0 a 255. O bloco de operação multiplicadora tem como função elevar o contraste da imagem e melhorar a visualização. Com essas adaptações, o filtro é reconhecido e aplicado devidamente em cada imagem a ser processada.

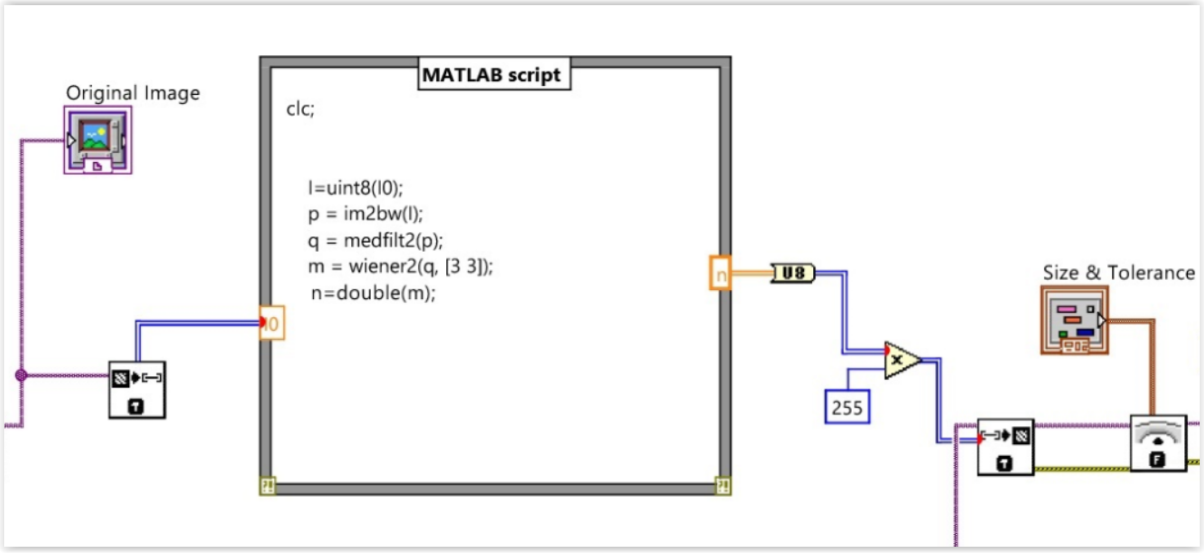


Figura 29 – Detalhe com a VI para a execução do script do MATLAB.
Fonte: Autoria própria.

5.3.2. Interface gráfica

Com o intuito de ser um programa de fácil manuseio, foi criada uma interface gráfica amigável e intuitiva conforme exibida por completo no Apêndice F. Na Figura 30 pode-se observar melhor os detalhes do painel de controle desenvolvido. Na parte 1 encontra-se a configuração inicial para o valor da quantidade de imagens, o nome e o tipo de imagens a ser analisadas. No painel inferior ("*File Path*") está localizado o local onde adiciona-se o diretório para o caminho da pasta que contenha a sequência de imagens. O painel de número 2 refere-se a configuração de parâmetros do filtro passa-baixa do LabVIEW. Dessa forma o usuário tem maior liberdade para alterar esses parâmetros conforme sentir a necessidade. No bloco de número 3 encontra-se os displays gráficos da imagem original (painel superior) e filtrada (painel inferior). Por fim, as imagens são montadas em pilha no eixo cartesiano e exibidas no display marcado como número 4. Nesse mesmo display é possível configurar o percentual de transparência (*"Transparency"*), estilo de plotagem (*"Plot Style"*), se quer que seja um desenho

rápido (*"Fast draw"*) e os níveis de contorno da imagem (*"Contour Levels"*). Os valores-padrão para cada requisito está na Tabela 1.

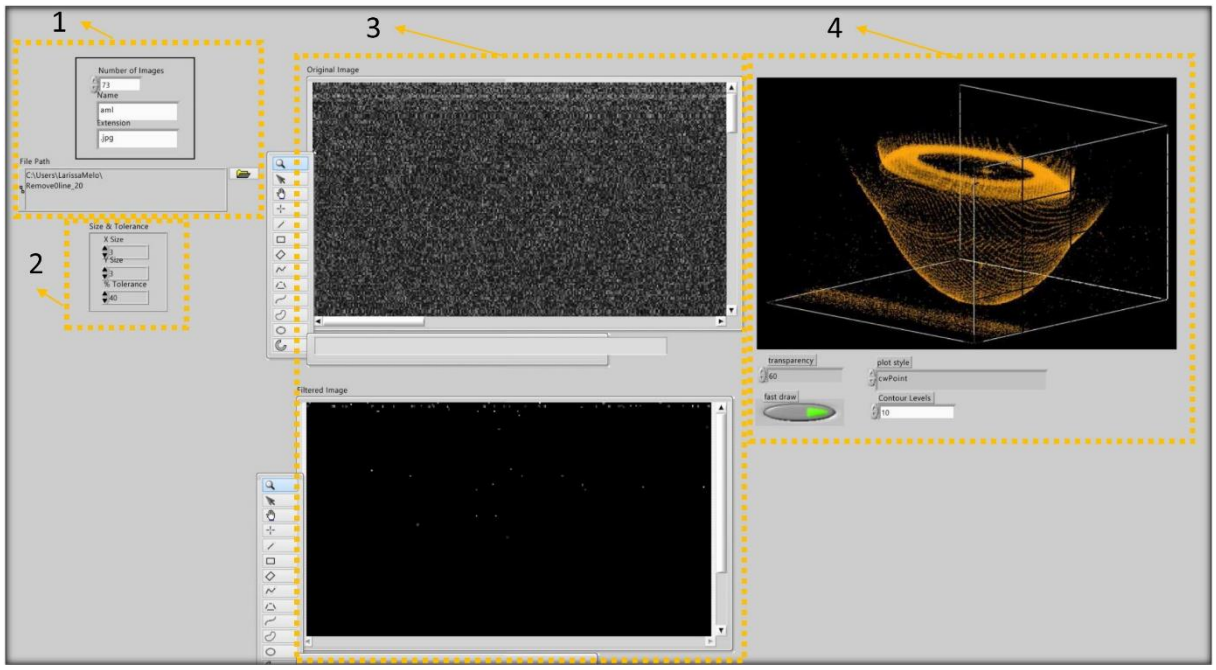


Figura 30 – Detalhe do painel frontal do programa em LabVIEW.
Fonte: Autoria própria.

Tabela 1 – Valores-padrão de parâmetros

Valores-padrão de parâmetros do painel frontal	
Transparência	60%
Desenho Rápido	Sim
Estilo de plotagem	cwPoint
Níveis de contorno	10

Fonte: Autoria própria.

5.4. Resultados

Utilizando as imagens do modelo de um olho humano, após o processamento e filtragem, o display gráfico é capaz de exibir o volume tridimensional do objeto escaneado. Pelas imagens pode-se detectar claramente o formato da córnea e da íris do olho, bem como observar o local da pupila. A Figura 31 exibe a estrutura ocular em um eixo cartesiano vista por cima. Nessa mesma imagem percebe-se alguns ruídos como pontos em amarelo ao redor da imagem principal, mas que não interferem na qualidade da imagem.

Com o gráfico do LabVIEW é possível rotacionar os eixos e observar o volume a partir de diferentes pontos de vistas. Pode-se também modificar as cores de plano de fundo e dos pontos do gráfico de forma que a imagem possa assumir diversas tonalidades e níveis de visibilidade conforme Figura 32.

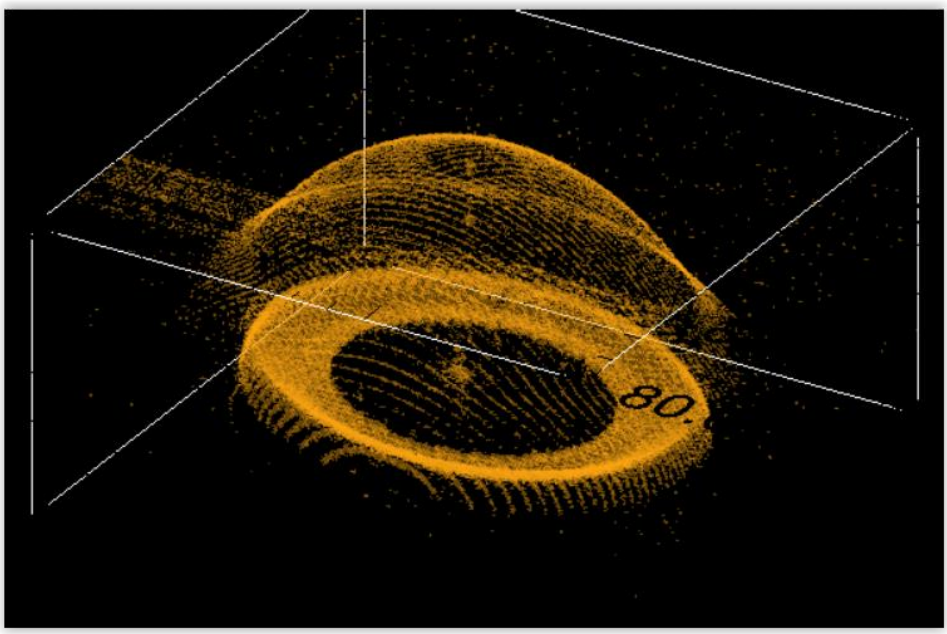


Figura 31 – Volume tridimensional das imagens OCT do modelo de um olho humano no eixo cartesiano renderizado pelo LabVIEW e visto de cima.
Fonte: Autoria própria.

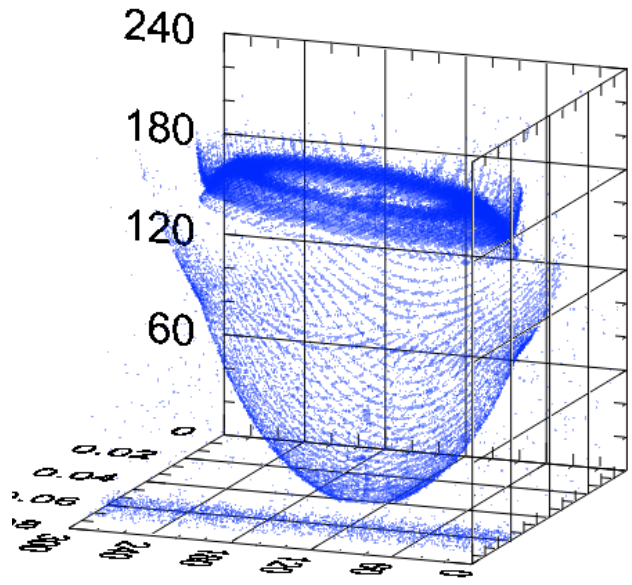


Figura 32 – Imagem renderizada com configurações de cores diferentes.
Fonte: Autoria própria.

Na Figura 33 e na Figura 34 é possível visualizar a imagem renderizada e fora do eixo cartesiano. A Figura 34 demonstra a pupila, responsável por regular a entrada de luz nos olhos; a córnea, responsável por proteger o olho e focar a luz da pupila em direção a retina e a íris, a parte colorida dos olhos. Com esses resultados pode-se confirmar que o programa é capaz de extrair as informações necessárias de cada imagem e criar o volume tridimensional com eficácia.

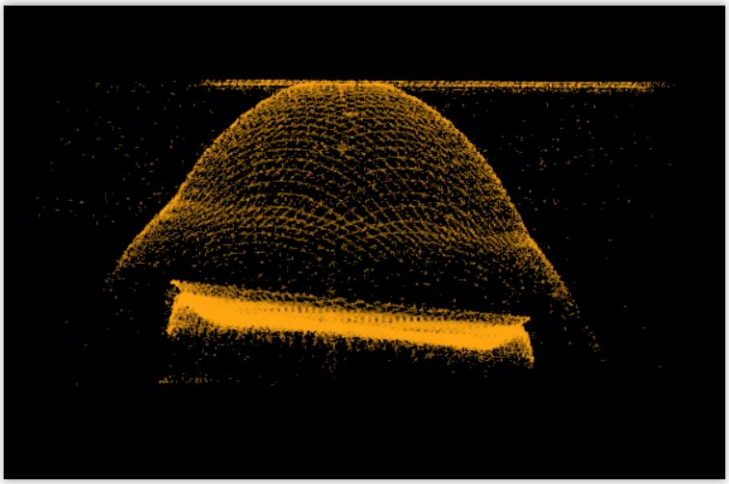


Figura 33 – Modelo do olho humano renderizado, visto pela lateral e fora do eixo cartesiano.
Fonte: Autoria própria.

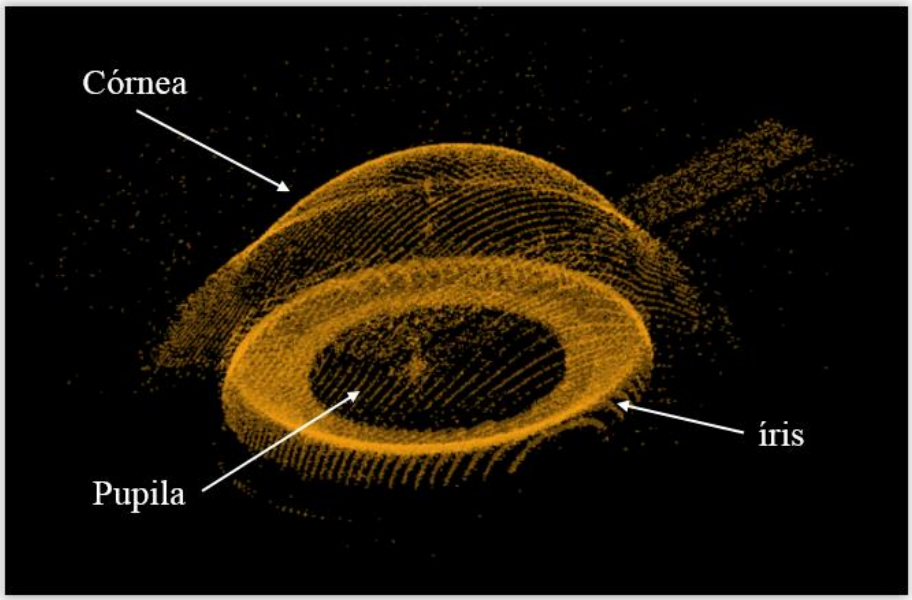


Figura 34 – Modelo de olho humano renderizado no LabVIEW visto por cima, sem o eixo cartesiano e com as definições de cada parte observada.
Fonte: Autoria própria.

6. CONCLUSÃO

Este trabalho propôs o desenvolvimento de uma ferramenta computacional para a manipulação de imagens geradas por tomografia de coerência óptica (técnica OCT). As imagens obtidas eram correspondentes a fatias do modelo de um olho humano e elas deveriam ser renderizadas computacionalmente de forma a reproduzir um volume tridimensional representativo. Essas imagens possuíam uma elevada quantidade de ruídos onde o formato primário da imagem era de difícil detecção pelo computador. Por esse motivo era fundamental que fosse feita a filtragem visual dos dados para melhorar a detecção volumétrica. Utilizando filtros do software MATLAB foi possível realizar testes de aperfeiçoamento de qualidade.

Para o desenvolvimento do programa de processamento foi utilizado o software LabVIEW. Com o LabVIEW foi possível desenvolver um programa responsável por abrir as sequências de imagens, aperfeiçoar a qualidade em paralelo com o MATLAB e processá-las graficamente ao longo de eixos cartesianos a fim de exibir um modelo tridimensional de todas as imagens.

Os testes de verificação de cada filtro, que foram realizados previamente utilizado o MATLAB, foram de fundamental importância para a detecção da melhor ordem de aplicação dos mesmos sobre as imagens. Os filtros utilizados causaram significativos realces na qualidade, reduziram ruídos e forneceram importante auxílio no processamento gráfico final. Além disso, a execução das rotinas não requer o uso de máquinas poderosas para o processamento do projeto.

Para o desenvolvimento do código de processamento em LabVIEW utilizou-se a ideia de varredura de intensidade de pixels de cada imagem e a posterior alocação da posição dos mesmos em vetores específicos. Os vetores armazenavam os três tipos de intensidades escolhidos e as posições dos mesmos. Em seguida, esses vetores foram concatenados em *arrays* que representassem as dimensões gráficas. Utilizando a VI “CWgraph3D” para a alocação dos dados nos respectivos eixos cartesianos, o volume tridimensional foi gerado e exibido no display gráfico diretamente no painel frontal do programa.

A versão final do software possui uma interface simples e auto descritiva. A interface gráfica também foi desenvolvida com o propósito de que o usuário pudesse escolher parâmetros fundamentais com facilidade. No fim, o LabVIEW se mostrou compatível com o MATLAB no qual ambos os softwares eram executados sem afetar significativamente a performance da máquina utilizada.

O resultado gerado no display possui uma excelente qualidade onde pode-se visualizar nitidamente as partes referente à córnea, pupila e íris do olho humano. Além disso, o usuário tem a liberdade de manipular o objeto tridimensional e alterar cores para observar detalhes ocultos.

Este trabalho foi importante para o conhecimento, visto que para a implementação foi possível aprender uma nova e importante linguagem de programação na área de engenharia que é o LabVIEW. O projeto também permitiu conciliar conhecimentos físicos e de computação na área de biomédica. Por conta disso, o projeto forneceu a possibilidade de explorar a diversidade de áreas acadêmicas e ainda assim propor soluções dentro da área do conhecimento de engenharia.

Para um trabalho futuro poderia considerar desenvolver melhores filtros para manter diferentes intensidades e tornar a imagem mais realista. Também pode-se cogitar o incremento de um display gráfico mais dinâmico com a inserção de entradas de controle para o tamanho dos eixos e a adição de melhores opções de plotagem (plotagem do tipo superfície, por exemplo).

REFERÊNCIAS

[1] K. Ohbayashi, D. Choi, H. Hiro-Oka, A. Kubota, T. Ohno, R. Ikeda e K. Shimizu, "Developing the World's First Real-Time 3D OCT Medical Imaging System With LabVIEW and NI FlexRIO," Abril 2011. [Online]. Available: http://sine.ni.com/cs/app/doc/p/id/cs-13387. [Acesso em Maio 2014].

[2] H. Lui, S. Baig, G. i. Jiang e M. R. Wang, "Integrated OCT and Reflectrometry System for Ocular Anterior Segment Imaging and Tear Film Thickness Evaluation," *Optical Engineering,* vol. 53, nº 6, 2014.

[3] D. Huang, E. A. Swanson, C. Lin, J. Schuman, W. Stinson, W. Chang, M. Hee, T. Flotte, K. Gregory, C. A. Puliafito e J. G. Fujimoto, "Optical Coherence Tomography," *Science,* vol. 254, pp. 1178-1181, 22 Nov. 1991.

[4] A. F. Fercher, W. Drexler, C. K. Hitzenberger e T. Lasser, "Optical Coherence Tomography - Principles and application," *Reports on Progress in Physics,* vol. 66, 20 Jan 2003.

[5] A. J. Rodrigues, C. Takimura, P. A. Neto e V. R. Figueiredo, "Tomografia de coerência óptica broncoscópica," *Jornal Brasileiro de Pneumologia,* vol. 38, Março 2012.

[6] S. Chang, Y. Cheng, K. V. Larin, Y. Mao, S. Sherif e C. Flueraru, "Optical coherence tomography used for security and fingerprint-sensing applications," *IET Image Processing,* vol. 2, pp. 48-58, 2008.

[7] National Instruments, "National Instruments," What Is LabVIEW?, 16 Agosto 2013. [Online]. Available: http://www.ni.com/newsletter/51141/en/. [Acesso em 2 Dezembro 2014].

[8] Creative Commons, "LabVIEW wiki," LabVIEW, 7 Abril 2014. [Online]. Available: http://labviewwiki.org/LabVIEW. [Acesso em 2 Dezembro 2014].

[9] "IMAQ LowPass VI," National Instruments, June 2011. [Online]. Available: http://zone.ni.com/reference/en-XX/help/370281P-01/imaqvision/imaq_lowpass/. [Acesso em July 2014].

[10] "Documentation Center," MathWorks, [Online]. Available: http://www.mathworks.com/help/images/ref/im2bw.html. [Acesso em july 2014].

[11] “Documentation Center,” MathWorks, [Online]. Available: http://www.mathworks.com/help/images/ref/medfilt2.html. [Acesso em July 2014].

[12] “Documentation Center,” MathWorks - wiener2, [Online]. Available: http://www.mathworks.com/help/images/ref/wiener2.html.

[13] R. L., “Graphically displaying 4D array,” 8 fevereiro 2011. [Online]. Available: https://decibel.ni.com/content/docs/DOC-15043. [Acesso em Maio 2014].

[14] D.-J. Kroon, “Matlab Central,” 2011. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/21993-viewer3d. [Acesso em July 2014].

[15] A. M. Zysk, F. T. Nguyen, A. L. Oldenburg, D. L. Marks e S. A. Boppart, “Optical coherence tomography: a review of clinical development bench to bedside,” *Journal of Biomedical Optics*, vol. 12(5), September/October 2007.

APÊNDICE

APÊNDICE A – Script do filtro “im2bw”

```
function BW = im2bwModified(I)
map = [];
level = 0.5;
A=I;
if isa(A,'int16')
  A = int16touint16(A);
end

if ndims(A)==3,% RGB is given
  A = rgb2gray(A);
elseif ~isempty(map),% indexed image is given
  A = ind2gray(A,map);
end % nothing to do for intensity image

range = getrangefromclass(A);

if isinteger(A)
  BWp = (A > range(2) *level);

elseif islogical(A)
  %A is already a binary image and does not require thresholding
  warning(message('images:im2bw:binaryInput'))
  BWp = A;
else % double or single
  BWp = (A > level);
end

% Output:
if nargout==0 % Show results
  imshow(BWp);
  return;
end
BW = BWp;
```

APÊNDICE B – Script do filtro “medfilt2”

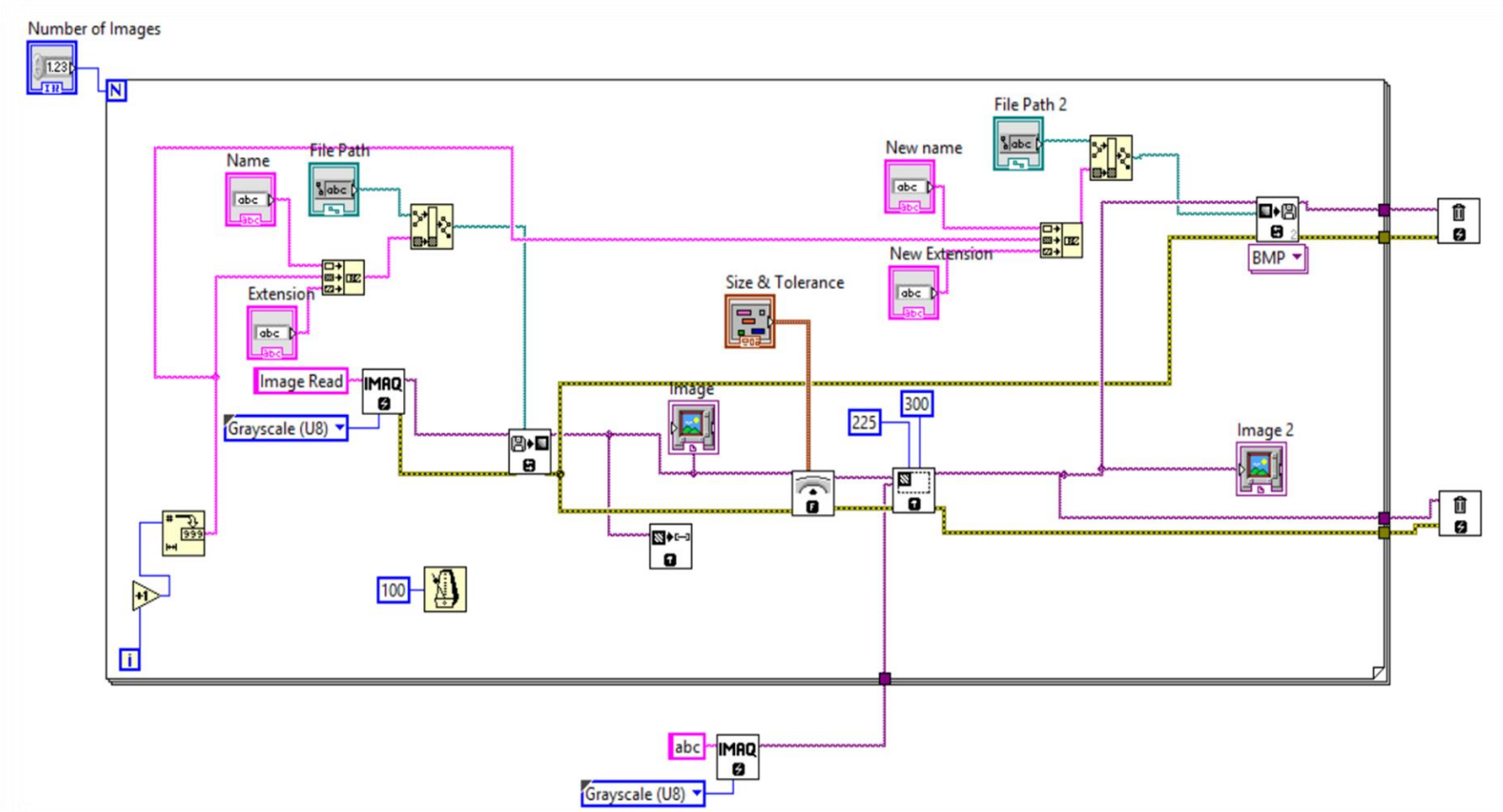
```
function b = medfilt2Modified(a)

mn = [3 3];% default
padopt = 'ones';

if (strcmp(padopt, 'indexed'))
    if (isa(a,'double'))
        padopt = 'ones';
    else
        padopt = 'zeros';
    end
end

domain = ones(mn);
if (rem(prod(mn), 2) == 1)
    tf = hUseIPPL(a, mn);
    if tf
        a = hPadImage(a,domain, padopt);
        b = medianfiltermex(a, [mn(1) mn(2)]);
    else
        order = (prod(mn)+1)/2;
        b = ordfilt2(a, order, domain, padopt);
    end
else
    order1 = prod(mn)/2;
    order2 = order1+1;
    b = ordfilt2(a, order1, domain, padopt);
    b2 = ordfilt2(a, order2, domain, padopt);
    if islogical(b)
        b = b | b2;
    else
        b = imlincomb(0.5, b, 0.5, b2);
    end
end

% ------------------------------------------------------------------------
function tf = hUseIPPL(a, mn)
% switch to IPP iff
% UseIPPL preference is true .AND.
% kernel is  odd .AND.
%      input data type is single .AND. kernel size is == 3x3
% .OR. input data type is (int16 .OR. uint8 .OR. uint16) .AND. kernel size
%      is between 3x3 and 19x19
tf = false;

switch class(a)
    case 'single'
        if all(mn==[3 3])
            tf = true;
        end
    case {'uint16', 'int16', 'uint8'}
        if all(mn >= [3 3]) && all(mn <= [19 19])
            tf = true;
        end
end

tf = tf & iptgetpref('UseIPPL');

% ------------------------------------------------------------------------
function A = hPadImage(A, domain, padopt)
% pad the image suitably -
domainSize = size(domain);
center = floor((domainSize + 1) / 2);
[r,c] = find(domain);
```

```
r = r - center(1);
c = c - center(2);
padSize = [max(abs(r)) max(abs(c))];
if (strcmp(padopt, 'zeros'))
    A = padarray(A, padSize, 0, 'both');
elseif (strcmp(padopt, 'symmetric'))
    A = padarray(A, padSize, 'symmetric', 'both');
else
%   This block should never be reached.
    error(message('images:medfilt2:incorrectPaddingOption'))
end
```

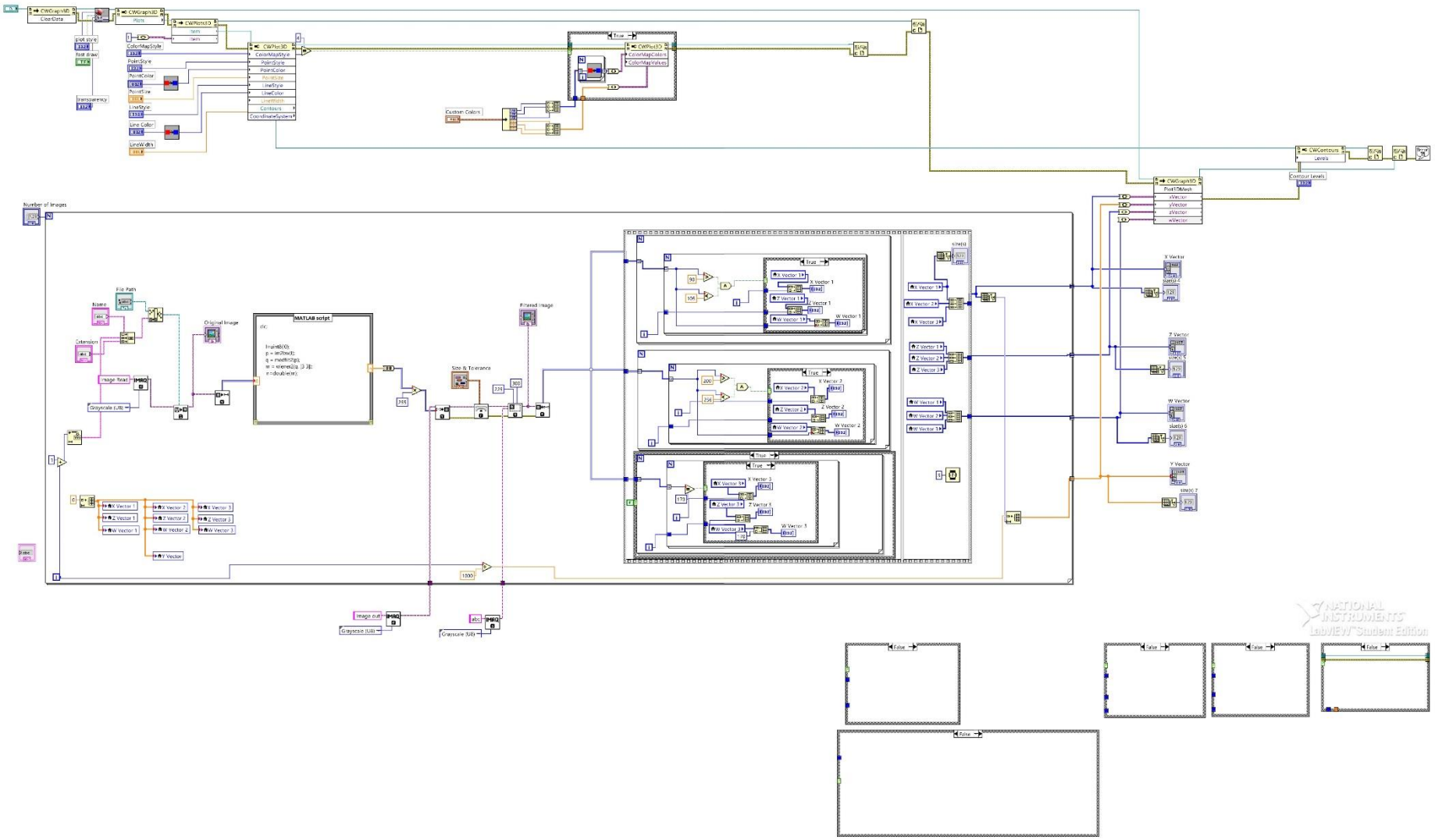
APÊNDICE C – Script do filtro “wiener” modificado

```
function f = wiener2Modified(g,nhood)
noise = [];
nhood=im2double(nhood);
classin = class(g);
classChanged = false;
if ~isa(g, 'double')
  classChanged = true;
  g = im2double(g);
end

% Estimate the local mean of f.
localMean = filter2(ones(nhood), g) / prod(nhood);

% Estimate of the local variance of f.
localVar = filter2(ones(nhood), g.^2) / prod(nhood) - localMean.^2;

% Estimate the noise power if necessary.
if (isempty(noise))
  noise = mean2(localVar);
end

% Compute result
f = g - localMean;
g = localVar - noise;
g = max(g, 0);
localVar = max(localVar, noise);
f = f ./ localVar;
f = f .* g;
f = f + localMean;

if classChanged
  f = changeclass(classin, f);
end

end

function image = changeclass(class, varargin)
%CHANGECLASS will change the storage class of an image.
%   Copyright 1993-2010 The MathWorks, Inc.
%   $Revision: 1.8.4.8 $  $Date: 2011/07/19 23:54:58 $

switch class
 case 'uint8'
  image = im2uint8(varargin{:});
 case 'uint16'
  image = im2uint16(varargin{:});
 case 'double'
  image = im2double(varargin{:});
 case 'single'
  image = im2single(varargin{:});
 case 'int16'
  image = im2int16(varargin{1});
 case 'logical'
  image = varargin{1} ~= 0;
 otherwise
  error(message('images:changeclass:unsupportedIPTClass'));
end
end
```

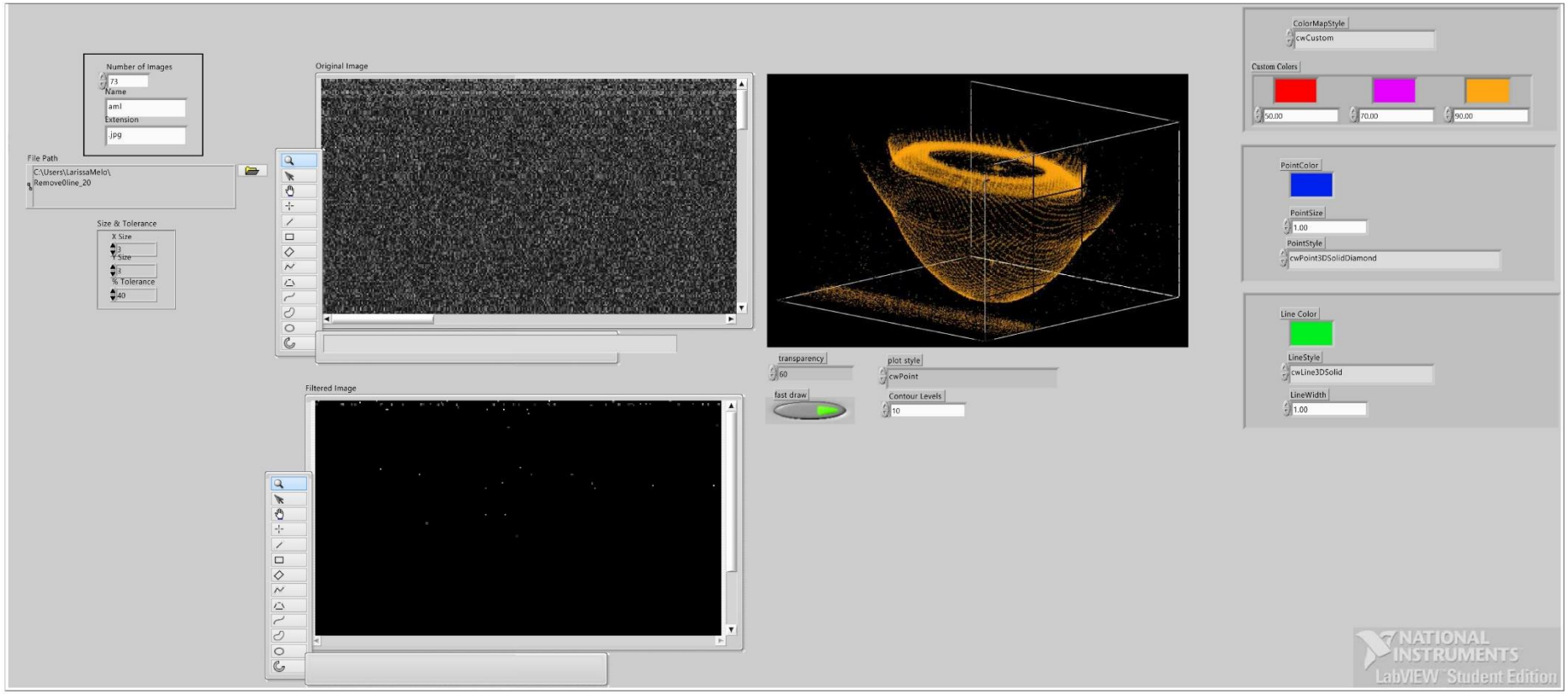
APÊNDICE D – Diagrama de blocos do programa de filtragem em LabVIEW.



APÊNDICE E – Diagrama de blocos completo do programa de manipulação de imagens OCT em LabVIEW

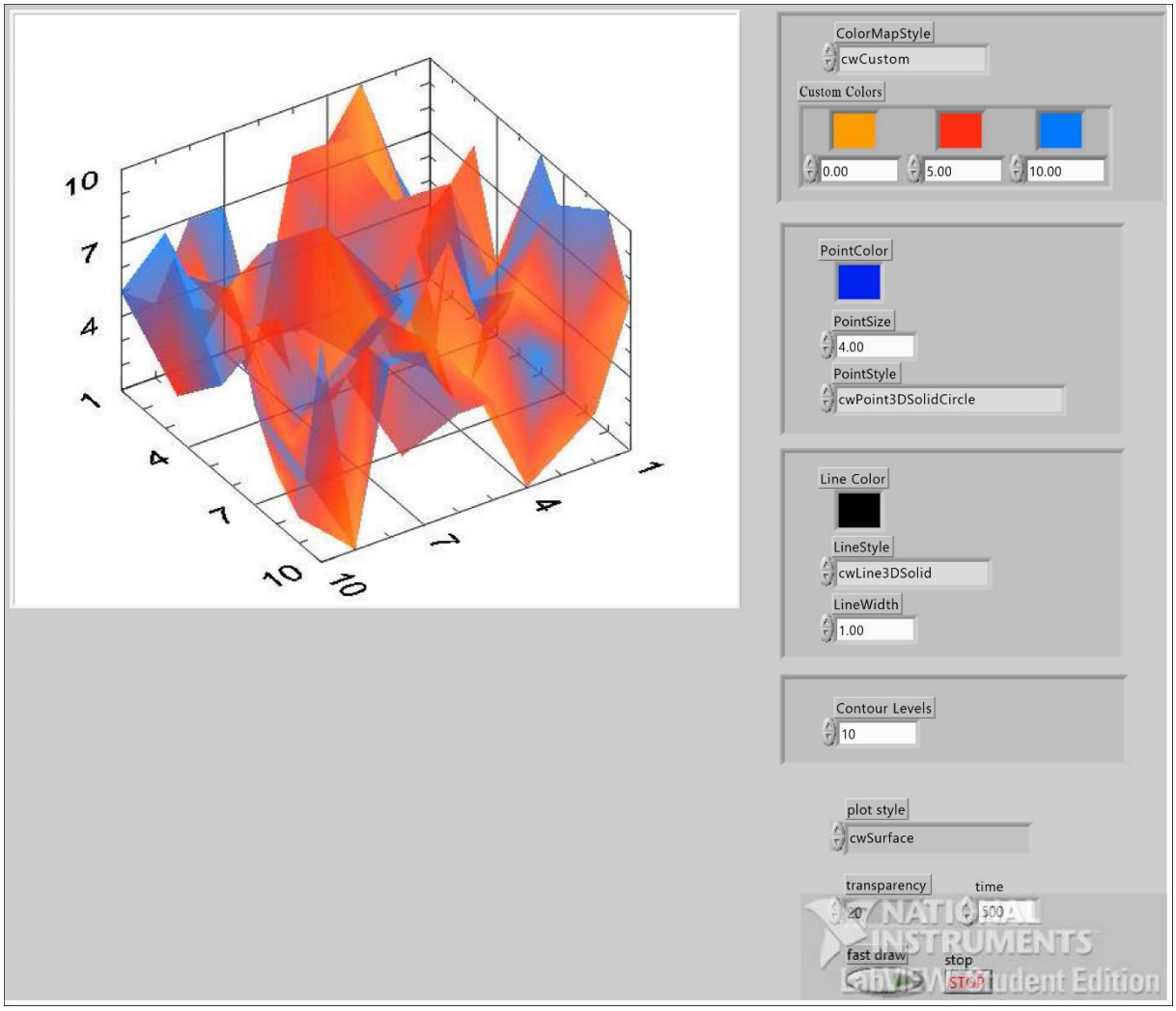


APÊNDICE F – Painel Frontal do programa em LabVIEW exibindo a interface gráfica



ANEXO

ANEXO A – Painel Frontal de um display gráfico de 4 dimensões
Fonte: Graphically displaying 4D array, 2014.



ANEXO B – Diagrama de blocos de um display gráfico de 4 dimensões.
Fonte: Graphically displaying 4D array, 2014.