Automação e processamento de medições em pinças ópticas

Um projeto de Iniciação Científica

Osvaldo Guilherme Arreche

Universidade Federal do ABC

AUTOMAÇÃO E PROCESSAMENTO DE MEDIÇÕES EM PINÇAS ÓPTICAS

UM PROJETO DE INICIAÇÃO CIENTÍFICA

by

Osvaldo Guilherme Arreche

em cumprimento parcial dos requisitos para

Iniciação Cientifica
PIBIC Edital 01/2014

na Universidade Federal do ABC, Santo André, Brasil.

Orientador: Prof. Dr. Antonio Alvaro Ranha Neves

Uma versão eletrônica deste trabalho está disponível em
http://www.ufabc.edu.br/.

PREFÁCIO

A montagem e calibração de pinças ópticas foi realizada como projeto de IC. Para esta ferramenta ser aplicada em pesquisas de fronteira da biotecnologia e nanotecnologia precisamos de um sistema de controle, processamento e analise de dados. O projeto tem como base o programa LabVIEW (National Instruments, 2014), cuja linguagem computacional foi desenvolvida para o rápido emprego em soluções de comunicações e de automação entre instrumentos. Além disso, foi desenvolvido este manual explicativo que junto com os programas, complementa o desenvolvimento experimental dessa linha de pesquisa. Esse projeto foi realizado no Laboratório de Espectroscopias Ópticas e Eletrônicas. O desenvolvimento dessa pesquisa permitirá acesso a técnicas modernas de caracterização de células orgânicas e de micromanipulação óptica.

Inicialmente foi necessário, com o auxílio do orientador, se familiarizar com linguagem do programa LabVIEW e suas possibilidades. Além disso, foi preciso estudar sobre a instrumentação a ser automatizada como o estágio de translação piezoelétrico e câmera CCD. Em seguida, foram desenvolvidos módulos simples, de acordo com as necessidades de um usuário de pinça óptica. Os programas serviram como base para integrar e construir um único programa mais completo e funcional para o futuro usuário.

Osvaldo Guilherme Arreche
Santo André, junho 2015

SUMÁRIO

1

INTRODUÇÃO

Antes de compreender os programas que foram desenvolvidos para a pinça óptica é preciso assimilar os princípios dos instrumentos utilizados e a linguagem de programação. Entender o funcionamento por trás delas e como a partir disso é possível chegar aos objetivos desejados para no final construir um programa funcional e prático ao usuário. Neste capítulo o leitor encontrará algumas características sobre o hardware e o software utilizado e o índice das funções utilizadas no LabVIEW. Para este projeto, foram utilizados:

- Pinça Óptica (Thorlabs)
- Computador Desktop
- Estágio motor piezelétrico (NanoMax-TS TPZ001, Thorlabs)
- Webcam (HD-C270, Logitech)
- Software LabVIEW (versão 2014)

1.1. FUNCIONAMENTO DA PINÇA ÓPTICA

As pinças ópticas funcionam através de do uso de dois feixes de laser contrários, com isso as forças axiais exercidas fariam que partícula fosse estavelmente presa. Por ser uma técnica não invasiva é um importante objeto de estudo.

Quando o fóton entra em contato com o objeto e é desviado, o objeto tende a recuar na direção da força aplicada. Desse modo, quando há esfera que possui índice de refração diferente do meio externo, os raios

a e **b** não se encontram no foco. Devido a isso, ocorre o desvio dos mesmos, ocasionando recuos nas direções $\vec{F}_a$ e $\vec{F}_b$, que combinados geram recuo na direção $\vec{F}$ 1.1. O recuo de $\vec{F}$ proporciona que o centro da esfera esteja posicionado sempre próximo ao foco do laser.

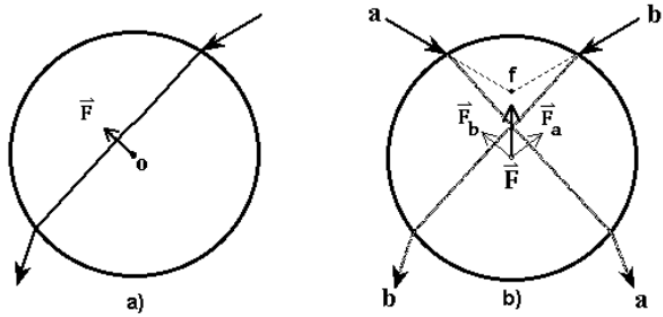


Figure 1.1: A esquerda, o desvio na trajetória (e portanto do momento linear) do fóton faz surgir uma força de reação sobre o objeto na direção de $\vec{F}$. A direita, o desvio dos raios **a** e **b** geram recuos nas direções $\vec{F}_a$ e $\vec{F}_b$ e a combinação de ambos geram o recuo na direção de $\vec{F}$. O recuo $\vec{F}$ faz com que o centro da esfera seja posicionado no foco do laser. Extraído de [1]

Este trabalho é uma continuidade do projeto de iniciação anterior da discente, Ana Maria C. A. F. Bittar [1], cujo trabalho foi montar e calibrar uma pinça óptica. Neste trabalho retomamos a calibração e garantimos a reprodutibilidade das calibrações e economia do tempo utilizando a automação através da programação em LabVIEW.

1.2. CONTROLADOR PIEZOELÉTRICO

O T-Cube Controlador Piezo (TPZ001), 1.2, é um controlador/*driver* compacto de único-canal para fácil controle (automático ou manual) de uma larga extensão de aglomerados piezo e atuadores. Este *driver* é capaz de operar até 150 V a 7,5 mA, desse modo permitindo operar em larguras de banda de até 1KHz, todas as características podem ser vistas na tabela da figura 1.3.

O seu tamanho compacto (60 mm × 60 mm × 47 mm) permite que ele seja posicionado próximo dos sistemas motorizados o que causa conveniência. A montagem é feita com cabos do *driver* com o mínimo de comprimento para fácil manuseio, além de vir com cabo USB Tipo A para o Tipo Mini B (*plug and play*).

Apesar de compacto, ele fornece a capacidade de controle completo do piezo. Para suportar uma grande variedade de dispositivos, a extensão da saída pode ser selecionada para $75V$, $100V$ ou $150V$, no nosso caso para em $75V$ porque é a tensão máxima suportável para o piezelétrico. A resolução do ajuste digitalmente codificado é facilmente alterada para providenciar maior controle de precisão. O controle direto do hardware por saída de alta voltagem pode ser facilitada usando um conector de entrada de baixa voltagem. Um conector de saída de baixa voltagem permite fácil monitoramento da saída HV (Alta voltagem), usando um osciloscópio por exemplo. Como recurso útil, um gerador de funções programável faz esta unidade particularmente conveniente para aplicações piezo de escaneamento.

Além de tudo, usando o T-Cube Controller Hub (TCH002), é possível conectar vários T-Cube Controlador Piezo (TPZ001) via tecnologia USB hub padrão.

Figure 1.2: Controlador Piezo (TPZ001). Extraído de [2].

1.2.1. LARGURA DE BANDA DO DRIVER PIEZO

Sabendo-se da taxa que cada piezo é capaz de mudar o seu comprimento, o torna essencial para várias aplicações de alta velocidade. A largura de banda de um controlador piezo e o conjunto podem ser estimados com os seguintes dados:

- O máximo de corrente que os controladores podem produzir, é de

Especificações	Valores
Piezoelétrico Saída (SMC Macho)	
Voltagem do Drive	0 - 150 V
Corrente do Drive, Max, Contínua	7.5 mA
Estabilidade	100 ppm até 24 hrs (Depois de 30 min Aquecido)
Ruído	<2 mV_{rms}
Capacitância Típica do Piezo	1 - 10 µF
Largura de Banda	1 kHz (1 µF Load, 1 V_{p-p})
Entrada Externa (SMA Fêmea)	0 - 10 V
Saída Monitor (SMA Fêmea)	0 - 10 V
USB Port[a]	Version 1.1 mini
T-Cube Controller Hub Connector	26-Way ERNI
Máxima Frequência para Onda Triangular	~167Hz
Geral	
Alimentação	+15 V @ 220 mA, -15 V @ 50 mA, +5 V @ 350 mA
Dimensão (W x D x H)	60 mm x 60 mm x 47 mm (2.4" x 2.4" x 1.8")
Peso	160 g (5.5 oz)

Figure 1.3: Tabela de especificações do controlador piezo. Extraído de [2].

0,5 A para a série BPC de controladores piezelétricos, que é o *driver* usado nos exemplos abaixo.

- A capacidade de carga do piezo, quanto maior a capacitância, mais lento o sistema.
- O sinal desejado de amplitude (V), que determina a largura de extensão do piezo.
- O máximo absoluto da largura de banda do driver, que é independente da carga sendo levada. [2]

Para conduzir a saída do capacitor, corrente é necessária para carregá-lo e descarregá-lo. A mudança de carga, dV/dt, é chamada de taxa de variação. Quanto maior a capacitância, mais corrente será necessária.

$$\text{Slew rate} = \frac{dV}{dt} = \frac{I_{max}}{C} \tag{1.1}$$

Então, por exemplo, para um aglomerado de 100 μm, tendo capacitância de 20 μF, sendo conduzido por um controle piezelétrico da série BPC de corrente máxima 0,5 A, a taxa de variação é dada por:

$$\text{Slew rate} = \frac{0,5A}{20uF} = 25V/ms \tag{1.2}$$

Assim, para voltagens instantâneas que variam de 0 V até 75 V, levaria apenas 3 ms para a voltagem de saída atingir 75 V. Observação: Para estes cálculos, é assumido que o máximo absoluto da largura de banda

do *driver* é muito maior que a largura de banda calculada, e assim, a largura de banda não é um fator limitante. Também observe que estes cálculos só se aplicam para os sistemas *open-loop* (corresponde ao sistema atualmente em uso). No modo *closed-loop*, a resposta lenta do *feedback* do *loop* coloca outro limite para a largura de banda.

O sistema *open-loop* é um tipo de controlador que computa o valor entrada no sistema usando somente o estado atual e o modelo do sistema. Uma característica desse controlador é que ele não possui *feedback* para determinar se a saída alcançou o valor de entrada, por causa disso muitas vezes o deslocamento em voltagem não corresponde exatamente ao deslocamento real. No caso o sistema *open-loop* do piezo é dado em volts. Por outro lado, o sistema *closed-loop* possui *feedback* informando se o valor de saída foi defasado, e portando é possível fazer correções a partir disso para qque o deslocamento seja exatamente aquele desejado. O valor é dado diretamente em mícrons.

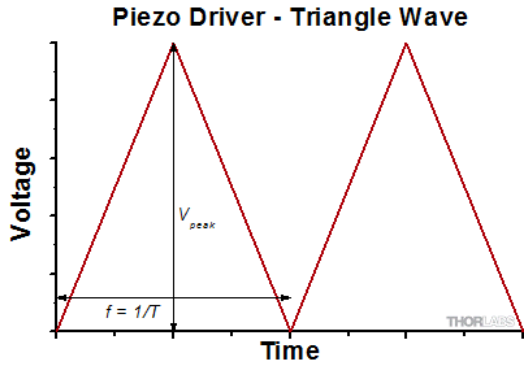


Figure 1.4: Onda triangular. Extraído de [2].

1.2.2. SINAL DE ONDA TRIANGULAR E FREQUÊNCIA MÁXIMA

Para um atuador piezelétrico conduzido por uma onda triangular de voltagem máxima V_{peak} e voltagem mínima de 0 V, a taxa de variação é o declive:

$$\frac{I_{max}}{C} = \frac{2V_{peak}}{T} \tag{1.3}$$

Sendo $f = \frac{1}{T}$:

$$f_{max} = \frac{I_{max}}{2V_{peak}C} = \frac{0,5A}{2(20uF)(75V)} \approx 167Hz \tag{1.4}$$

1.2.3. FUNCIONAMENTO DO CONTROLADOR PIEZOELÉTRICO

As propriedades piezoelétricas dos materiais são devidas à natureza de sua estrutura cristalina. De modo que, se um cristal não possui um centro de simetria que possibilite uma inversão de posição de seus átomos, propriedades elétricas interessantes podem se manifestar. O que caracteriza os piezoelétricos é sua capacidade de ficarem polarizados quando ocorre uma deformação homogênea. O que ocorre então é que o material manifestará um campo elétrico interno sob a ação de forças que o deformam 1.5.

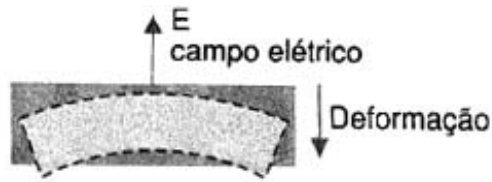


Figure 1.5: Ilustração de uma deformação do material piezo [3].

Da mesma forma, o efeito inverso, denominado eletrostrição, ocorre com este material: quando o submetemos a um campo elétrico ele se deforma. As variações das dimensões do material com a ação do campo elétrico mudam muito, e também depende da orientação do eixo de simetria do material.

Isso significa que aplicando um campo elétrico através de placas fixadas na superfície de um cristal piezoelétrico pode-se obter diversos tipos de deformações **??**.

1.3. FUNCIONAMENTO DA CÂMERA CCD

Uma câmera digital possui um sensor que converte luz em cargas elétricas. A imagem do sensor empregado pela maioria das câmeras digitais é o CCD (*Charge Coupled Device*), dispositivo de acoplamento de carga. Essa tecnologia funciona como conversor de luz em elétrons, pode-se pensar nela como uma matriz de milhares ou milhões de pequenas células solares [4].

Quando fótons de luz atingem a região, pares de elétrons e buracos são criados devido ao efeito fotoelétrico. Os elétrons são atraídos pelo potencial da porta, criando uma carga elétrica na região, que será tanto maior quanto maior a intensidade da luz incidente.

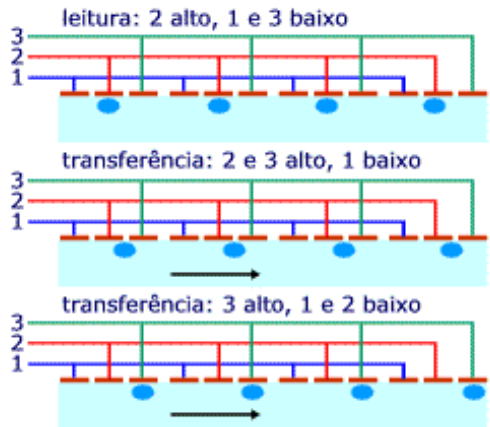


Figure 1.6: Funcionamento de uma CCD. Extraído de [5]

Entretanto, não há como ler diretamente o valor da carga em cada pixel. O potencial da porta é fixo, mantido apenas para segurar a carga. A figura 1.6 mostra o processo que permite a leitura: na realidade, cada pixel tem duas portas auxiliares. A aplicação sucessiva de potenciais diferentes move as cargas ao longo de uma linha ou coluna. Com o uso de uma linha auxiliar na parte inferior, a imagem pode ser movida linha a linha pela aplicação simultânea em cada coluna dos potenciais sucessivos. No canto inferior direito, um eletrodo recebe a informação sequencial de cada linha da imagem.

Uma vez que o sensor converte a luz em elétrons, o valor(carga acumulada) de cada célula é transformada em imagem. Uma CCD transporta carga através do chip e começa a leitura através da ponta da matriz. Um conversor analógico para digital então transforma cada valor dos pixels em valores digitais, por intermédio da carga acumulada em cada ponto da matriz.

1.4. LABVIEW

O LabVIEW (do qual a Universidade Federal do ABC já tem licença de uso) é o software base da plataforma de projeto da National Instruments, assim sendo útil ao nosso propósito para o desenvolvimento de um sistema de automação, medição e controle para a pinça óptica. Ele integra as ferramentas necessárias de que necessitamos para desenvolver uma ampla gama de aplicações em um tempo significativamente menor, o LabVIEW é um ambiente de desenvolvimento voltado à resolução de problemas, produtividade acelerada e inovação contínua, além da redução de insumos, mão de obra e consumo de energia.

Um sistema de teste automatizado típico com vários instrumentos conecta instrumentos de bancadas tradicionais a um PC para aumentar a eficiência e tornar mais fácil para analisar dados e gerar relatórios. Com o software NI LabVIEW, é possível adquirir dados de qualquer instrumento de bancada, por qualquer barramento, além de ter bibliotecas abrangentes para o processamento de sinais e a visualização dos dados.

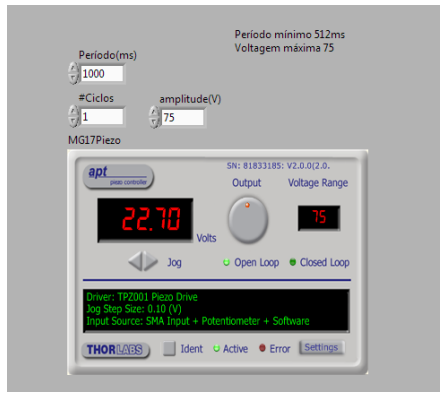


Figure 1.7: Exemplo de uma interface do painel de controle

O desenvolvimento do sistema de controle que gerencia todos os equipamentos e adquire os dados será feito baseado nas necessidades de um usuário padrão cuja interface gráfica ira fornecer todos os parâmetros e variáveis experimentais disponíveis. A comunicação entre o PC e a instrumentação de fornecedores variados será via GPIB ou USB/serial. Cada parte do experimento envolve instrumentos bem

distintos e complexidade do desenvolvimento do software.

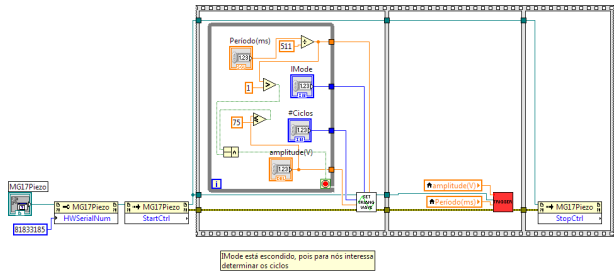


Figure 1.8: Exemplo de uma interface do diagrama de blocos

Os programas dessa linguagem são chamados de Instrumentos Virtuais ou VIs, que são representados em blocos no Diagrama de Blocos (código gráfico) figura 1.8 e podem possuir uma interface correspondente no Painel Frontal (visualização do usuário), figura 1.7. Esses programas são compilados e podem ser comparados a linguagens de programação de alto nível.

Uma das vantagens do LabVIEW é poder usar os Instrumentos Virtuais (VIs) como subVIs, dessa maneira é perfeitamente funcional compilar uma VI sozinha, ou usá-la como subVI em uma outra VI criando programas mais complexos.

1.4.1. ÍNDICE DA LINGUAGEM LABVIEW USADA

Abaixo elencamos alguns comandos utilizados no desenvolvimento deste projeto, junto com uma breve descrição.

Boolean:

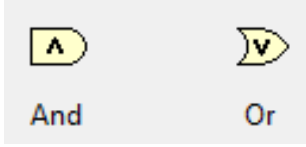


Figure 1.9: Operadores booleanos.

- **And**: Transmite uma condição verdadeira se os dois argumentos recebidos são verdadeiros.
- **Or**: Transmite uma condição verdadeira se um dos dois argumentos recebidos são verdadeiros.

Cluster, Class and Variant:

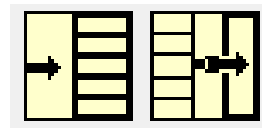


Figure 1.10: Bundle e Unbundle

- **Bundle**: Monta um cluster de elementos individuais.
- **Unbundle**: Separa os elementos indiduais de um cluster

Comparison:

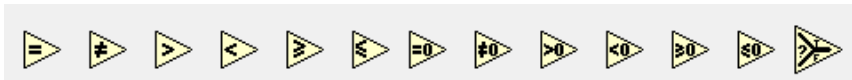


Figure 1.11: Da esquerda para a direita respectivamente: Equal ?; Not Equal ?; Greater ?; Less ?; Greater or Equal ?; Less or Equal ?; Equal to 0 ?; Not Equal to 0?; Greater than 0 ?; Less than 0 ?; Greater or Equal to 0?; Less or Equal to 0 ?; Select.

- **Select**: Analisa se é verdadeiro ou falso o que se liga a ele, caso seja verdadeiro, o valor passado é que está conectado em cima ao lado de "T", no caso falso o valor é o que se conecta a "F", no lado de baixo.
- **Greater or Equal ?; Less or Equal ?; Greater ?; Less ?; Equal ?; Not Equal ?; Greater or Equal to 0?; Less or Equal to 0 ?; Greater than 0 ?; Less than 0 ?; Equal to 0 ?; Not Equal to 0?**: Funções comparativas entre dois números (maior, igual, maior ou igual, menor ou igual, igual, diferente).

Connectivity (ActiveX,Microsoft):

Figure 1.12: Property Node e Invoke Node.

- **Property Node**: Lê ou escreve as propriedades de uma referência.

- **StartCtrl**: Aciona a comunicação com o controlador piezo criando as referências necessárias.
- **HWSerialNumber**: Identifica o controlador piezo pelo número de série.
- **StopCtrl**: Encerra a comunicação com o controlador piezo.
- **Invoke Node**: Invoca um método ou ação em uma referência. igual, diferente).
- **SetVoltOutput**: Métodos que escreve um valor em Volts para o controlador piezo.
- **GetVoltOutput**: Método que lê o valor em Volts da posição atual do controlador piezo.
- **SetOutputLUTValue**: Este método permite definir uma sequência de voltagem para a saída que é dada por amostras ou "pontos" de intervalos adjacentes. O usuário define o valor de cada ponto ou amostra criando podendo criar uma onda pré-definida. A resolução é de 512 pontos (0 a 75 V). A sigla `LUT` corresponte a `Look Up Table`.
- **SetOutputLutParams**: Neste método se define alguns parâmetros para se poder gerar uma onda.
- **StartOutputLUT**: É o método que inicia o processo de geração de onda.

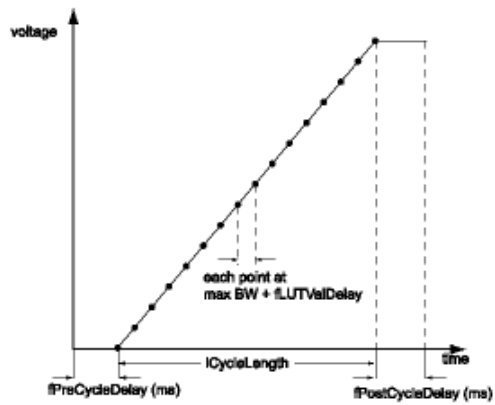


Figure 1.13: Gráfico representando o `SetOutputLUTParams`.

A figura 1.13 auxilia o entendimento desse método, ele mostra um

ciclo. `IMode` significa se o programa vai trabalhar com ciclos fixos (2) ou se serão ciclos contínuos (1), neste caso está definido como 2. `ICycleLength` é tamanho de cada ciclo, como a resolução é de 512 pontos, temos 511 "espaços". `INumCycles` é o número de ciclos (caso o `IMode` seja 2). `fLUTValDelay` é o tempo em ms entre um ponto e outro do ciclo. `fPreCycleDelay` é tempo de espera em ms antes de cada ciclo, neste caso definido como 0. `fPostCycleDelay` é tempo de espera em ms após cada ciclo, neste caso definido como 0.

Imaq:

Figure 1.14: Da esquerda para a direita respectivamente: Start Acquisition; Imaqdx Get Image; Imaqdx Open Camera; Stop Acquisition; Unconfigure Acquisition; Imaqdx Close Camera; Configure Acquisition;Imaq Cast Image; Imaq Dispose; Imaq Multiply; Imaq Subtract; Imaq Add; Imaq Divide; Vision Acquisition Image Logging;Imaq Extract; Imaq Create; Imaq ArrayToImage VI; Imaq ComplexImageToArray VI; Imaq FFT VI; Calculate Frames per Second; Imaq Complex Truncate VI; Imaq ReadFile; Imaq Copy; Imaq Threshold; Imaq ComplexFlipFrequencies; Imaq Overlay Text; Imaq Overlay Line; Imaq Light meter Rectangle; Imaq Setup Learn Pattern 2; Imaq Learn Pattern 2; Imaq Setup Match Pattern 2; Imaq Match Pattern 2

- **Imaqdx Open Camera**: Inicia comunicação com câmera, carrega seus arquivos de configuração e cria uma referência única para esta câmera.
- **Configure Acquisition**: Configura a câmera em baixo-nível quanto ao modo de aquisição e o número de buffers. Especifica-se o tipo de aquisição com os parâmetros `Continuous` e `Number of buffers`.
  - **Snap**: Continuous = 0; Buffer Count = 1
  - **Sequence**: Continuous = 0; Buffer Count > 1
  - **Grab**: Continuous = 1; Buffer Count > 1
- **Start Acquisition**: Inicia a aquisição.
- **Imaq Create**: Cria uma memória temporária para a imagem.
- **Imaqdx Get Image**: Adquire o frame especificado para a imagem de saída. É possível escolher o modo do número de buffer, o valor padrão é Next, que especifica que o driver vai esperar

pelo próximo buffer adquirido. Outro valor possível é Last, que especifica que o driver não vai esperar pelo próximo buffer, mas vai retornar o último buffer adquirido, e o Buffer Number, que especifica um número exato de buffers para retornar.

- **Calculate Frames per Second**: Calcula a taxa de frames dentro de um loop.
- **Imaq Cast Image**: Converte uma imagem para outro tipo especificado (como 8bits ou 16bits em tons de cinza).
- **Stop Acquisition**: Encerra a aquisição de imagens.
- **Unconfigure Acquisition**: Desconfigura a aquisição previamente configurada.
- **Imaqdx Close Camera**: Libera os recursos usados para a aquisição e encerra a sessão de câmera específica.
- **Imaq Dispose**: Destrói a imagem e libera o espaço usado pelo Imaq Create.
- **Vision Acquisition Image Logging**: Salva uma imagem, para isso é preciso alguns parâmetros como onde o arquivo vai ser salvo ("Image Logging Path"), seu nome ("Image Logging Prefix") e um cluster (formato da imagem, qualidade, compressão, perca, codec, taxa e máximo de frames) formado através da função "Bundle".
- **Imaq Add**: Soma duas imagens ou uma imagem a uma constante.
- **Imaq Subtract**: Subtrai duas imagens ou uma imagem a uma constante.
- **Imaq Divide**: Divide duas imagens ou uma imagem a uma constante.
- **Imaq Multiply**: Multiplica duas imagens ou uma imagem a uma constante.
- **Imaq Copy**: Copia uma imagem, caso queira preservar a original para algum processamento.
- **Imaq BCGlookup**: Altera os parâmetros de brilho, contraste e o fator de correção gamma da imagem que aumenta o contraste para pixels de valores altos e diminui para o de valores baixos ou vice-versa.
- **Imaq Threshold**: Atribui valores de 0 ou 1 para todos os pixels a partir de um valor de corte delimitado pelo usuário.
- **Imaq ComplexFlipFrequencies**: Inverte as altas frequências espaciais com as baixas.
- **Imaq Overlay Text**: Permite escrever um texto em uma imagem.

- **Imaq Overlay Line**:Permite desenhar uma linha eem uma imagem.
- **Imaq Light Meter Rectangle**:Retorna para o usuário, entre outras funções, o valor médio da intensidade dos pixels em uma determinada área.
- **Imaq Extract**: Extrai parte de uma imagem dado um retângulo.
- **Imaq Setup Learning Pattern 2**: Define os parâmetros no aprendizado de um padrão de imagem
- **Imaq Learn Pattern 2**: Cria uma descrição da imagem a ser procurada durante a fase de matching.
- **Imaq Setup Match Pattern 2**: Define os padrões usados durante a busca da imagem desejada.
- **Imaq Match Pattern 2**: Faz a busca do padrão da imagem na imagem desejada.
- **Imaq FFT VI**: Essa VI cria uma imagem complexa que agrupa altas frequências no centro,enquanto baixas frequências ficam localizadas nas bordas.
- **Imaq Complex Truncate VI**: Trunca as frequências de uma imagem complexa.
- **Imaq ArrayToImage VI**: Cria uma imagem de um array 2D.
- **Imaq Convert Rectangle to ROI**: Converte as coordenadas de um retângulo em uma região de interesse na imagem.
- **Imaq Convert ROI to Rectangle**: Converte uma região de interesse na imagem nas coordenadas de um retângulo.
- **Imaq ComplexImageToArray VI**:Extrai os pixels de uma imagem complexa em um array 2D complexo.

Numeric:

Figure 1.15: Da esquerda para a direita respectivamente: Add; Subtract; Multiply; Divide; Quotient Remainder; Absolute Value; Negate; Random Number.

- **Add**: Soma números.
- **Subtract**: Subtrai números.
- **Divide**: Divide Números.

- **Multiply**: Multiplica Números.
- **Negate**: Inverte o sinal do número.
- **Quotient and Remainder**: Retorna o quociente e o resto de uma divisão.
- **Random Number**: Gera números aleatórios de 0 a 1.
- **Absolute Value**: Retorna o módulo do número.

Structures:

Figure 1.16: Da esquerda para a direita respectivamente: For Loop; While Loop; Case Structure; Flat Sequence; Event Structure.

- **While Loop**: Executa repetidamente a programação que está em seu interior a cada iteração até que seja satisfeita sua condição de parada (Como um botão “Stop”).
- **Shift Register**: É um opcional que pode ser integrado ao While Loop ou For Loop, ele retorna o valor ligado a ele no fim de uma iteração para poder ser usado na iteração seguinte e assim por diante, e antes da primeira iteração também é possível definir um valor de entrada.
- **For Loop**: Executa repetidamente a programação que está em seu interior a cada iteração até que seja atingido um determinado número de iterações pré-estabelecido pelo usuário.
- **Case Structure**: Executa a programação que está em seu interior dependendo das condições, se for verdadeira executa um código, se for falsa executa outro.
- **Flat Sequence**: Executa a programação que está em suas divisões ordenadamente da esquerda para a direita como sequências.
- **Event Structure**: Aguarda até que ocorra algum evento, depois executa a tarefa apropriada.

Timming:

- **Wait (ms)**: Causa uma espera proposital onde for colocado no programa em milisegundos.

Figure 1.17: Wait (ms).

2

PROGRAMAS DESENVOLVIDOS

Este capítulo é sobre os módulos desenvolvidos para o programa final, cada um possui uma função diferente que são divididas entre rotineiras e de calibração. Algumas delas, são de autofocalização, escaneamento, calibração da constante elástica e calibração da câmera CCD, todas feitas em LabVIEW, cada seção é sobre um módulo e apresenta como foi construído. Os programas serão apresentados primeiramente aqueles que controlam exclusivamente o estagio piezoelétrico, depois apenas a câmera CCD, e por fim a combinação destes.

2.1. CONTROLE DO ESTÁGIO PIEZOELÉTRICO

2.1.1. POSICIONAMENTO DO ESTÁGIO PIEZOELÉTRICO (GOPIEZO.VI)

Os blocos `MG17Piezo` representam o motor piezo por meio de um controle `ActiveX`, e se ligam o `OPEN PIEZO` como entradas. O `OPEN PIEZO` inicia a comunicação com o motor piezo. Dentro do `MOVE PIEZO` é onde está os métodos que fazem o motor piezo se mover. O `CLOSE PIEZO`, encerra as comunicações.

Os blocos X, Y e Z, da figura 2.1, são entradas para o usuário inserir a posição desejada em Volts (0 à 75). Já os `pfVoltage` X, Y e Z são saídas que indicam a posição em que realmente se encontra o motor piezo, Uma vez que a posição real e aquela informada não são idênticas, mas próximas quando se opera em *open-loop*. Este ultimo blocos se ligam ao `MOVE PIEZO`.

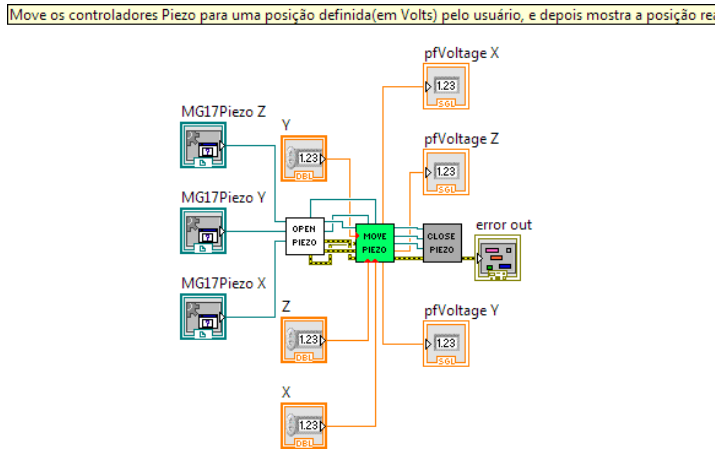


Figure 2.1: Diagrama de blocos do módulo do `GoPiezo`, composto por três subVIs principais, a primeira,`OPEN PIEZO`, `MOVE PIEZO` e `CLOSE PIEZO`

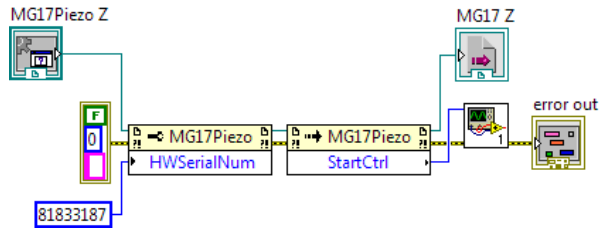


Figure 2.2: VI `Open Piezo`, é composto por um `Property Node` cuja função é identificar o controlador piezo pelo número de série. Também há um `Invoke Node` cuja função é habilitar um método, no caso o `StartCtrl` para iniciar a comunicação[6].

Na primeira sequência da figura 2.3 há o método `SetVoltOutput` para escolher a voltagem desejada, na segunda existe um `Wait` para aguardar 150 ms até a próxima ação. Na terceira sequência há o método `GetVoltOutput` para obter a posição em voltagem do piezo. Para encerrar o módulo, basta ligar as referências ao `CLOSE PIEZO`.

2.1.2. PROGRAMA DE VARREDURA COM VELOCIDADE CONSTANTE (TRIANGULARWAVE.VI)

Na primeira sequência da figura 2.6 há um `While Loop` cuja condição de parada é a voltagem inserida ser menor que 75 V, e a divisão do período por 511 espaços (a resolução é de 512 pontos) ser maior que 1 ms que é o limite do valor de saída para a `LUT` pela especificação

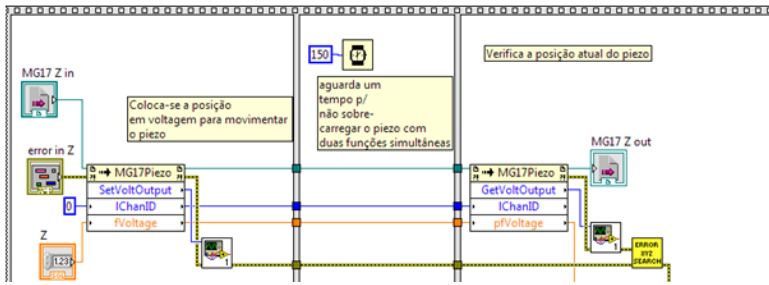


Figure 2.3: VI `Move Piezo` é composto por uma `Flat Sequence Structure` que sequencialmente executa o que está em seu interior da esquerda para a direita.

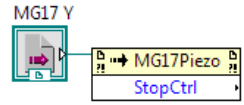


Figure 2.4: VI `Close Piezo` é composto por um método `StopCtrl` para encerrar a comunicação com o motor piezo.

da Thorlabs para que possa entrar na subVI `SET TRIANGULAR WAVE`. Na segunda sequência há outra subVI `TRIGGER` que inicia a ação do programa e na terceira há o fechamento das comunicações. `IMode` é modo operacional, no caso ajustado para ciclos pré-determinados da entrada `Ciclos`.

Na subVI `SET TRIANGULAR WAVE` há novamente três sequências da `Flat`
`Sequence Structure`. Na primeira há o método `GetOutputVoltage` para adquirir a voltagem em que se encontra os controladores piezo. Na segunda é onde são definidos os valores da onda, podem ser definidos até 512 valores.

Mais detalhadamente, se multiplica a amplitude por 2 (isso porque estamos fazendo com que ela complete o ciclo em um período), e divide-se pelo número de pontos (512), assim se obtém o tamanho de cada incremento que será multiplicado por cada iteração (0, 1, 2, 3 ...) e será somado ao valor da voltagem inicial. A partir dessa parte se usa o `Select` (*Functions*> *Programming*>*Comparison*, esse comando analisa se o que se liga ao `Select` é verdadeiro ou falso, caso seja verdadeiro, o valor passado é que está conectado em cima ao lado de "T", no caso falso o valor é o que se conecta a "F", no lado de baixo. Caso a

Figure 2.5: Fluxograma da VI `Onda Triangular`.

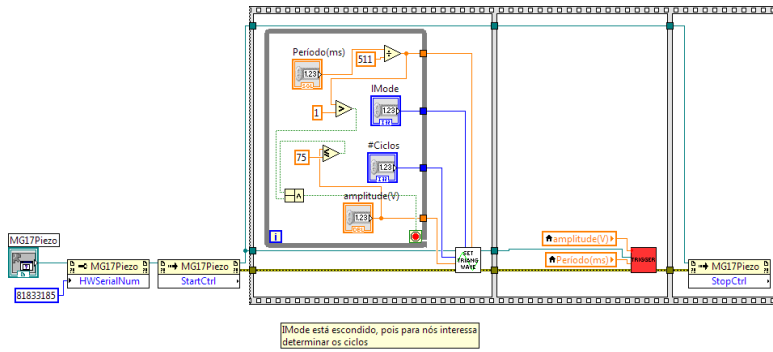


Figure 2.6: Diagrama em blocos da VI `Onda Triangular`. Antes de adentrar na `Flat Sequence Stucture` há a abertura da comunicação como mostrado no `Go Piezo` 2.1

soma do valor inicial de voltagem com o incremento multiplicado pelo número de iterações seja maior ou igual ao valor inicial de voltagem, então se passa o valor da amplitude multiplicada por 2 menos essa soma, caso contrário se passa o valor da própria soma. Então se tem mais um `Select` para verificar se o valor é menor que 0, caso seja, é usado o `Negate` (*Functions>Programming>Numeric*) que faz com que os números positivos fiquem negativos e vice-versa. Caso não seja, é passado o mesmo valor, veja figura 2.7.

Na terceira sequência há o método `SetOutputLUTParams` onde define-se os ciclos pré-determinados e quantos são, o período entre cada ponto definido anteriormente e o período entre ciclos. Na segunda sequência principal, está a subVI `TRIGGER` como indicado na figura 2.8.

Após todos os parâmetros estarem definidos se usa o método `StartOutputLUT` que engatilhará o processo, para isso foi criado uma `Flat Sequence Structure` com duas divisões, como visto acima o método se enquadra na primeira enquanto na segunda há mais uma subVI, `Motor Moving?`, que verifica se o piezo já parou de se mover.

A subVI `MotorMoving?` (figura 2.9) é estruturada com um `While`

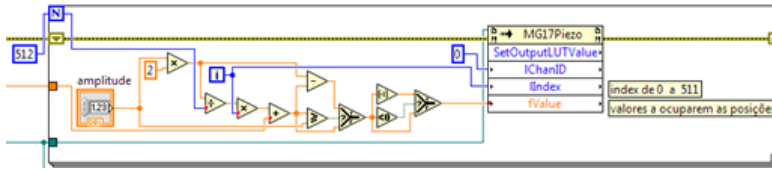


Figure 2.7: Definição dos valores de onda.

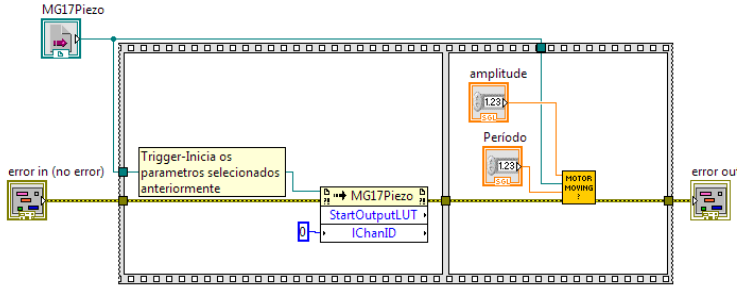


Figure 2.8: Diagrama de blocos da subVI `TRIGGER`.

`Loop` cuja condição de parada é o piezo estar parado, para essa condição tomamos como parado se na próxima iteração a "movimentação" do piezo ser menor do que o incremento. Sendo que entre as iterações existe um intervalo de 200 ms para evitar sobrecargas de iterações visto que não é necessário, isso é feito com o comando `Wait`. Aqui aparece novamente uma `Flat Sequence Structure` com quatro divisões. Na primeira tem-se o método `GetOutput Voltage` para obter a posição atual em volts, em seguida, na segunda divisão, é necessário esperar um tempo razoável (duas vezes o período) para aguardar a execução do incremento para obter a nova posição (terceira divisão). Já na quarta divisão é onde se verifica a veracidade da condição, se pega o módulo `Absolute Value` da diferença das duas posições e se ele for menor que o tamanho do incremento (amplitude total dividido pela resolução de 512 pontos), satisfaz a condição do `While Loop`. Na terceira sequência principal, se encerra a comunicação com o piezo.

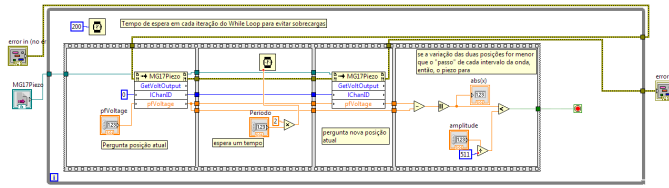


Figure 2.9: Diagrama de blocos da subVI `MotorMoving?`.

2.2. CONTROLE DA CÂMERA

2.2.1. PROGRAMA ELIMINAÇÃO DE IMAGEM DE FUNDO (SUBTRACTBACKGROUND.VI)

Para este programa são usados vários `Imaq Create` porque é necessário reservar memória para as operações com imagens que as bibliotecas `Imaq` fazem diretamente por meio de operadores.

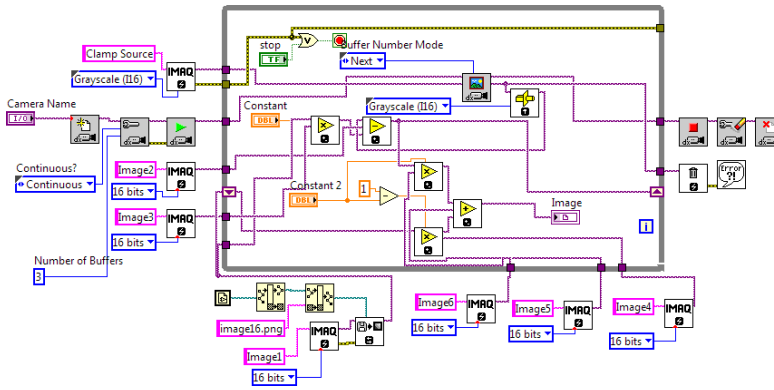


Figure 2.10: Eliminação do plano de fundo.

Primeiramente as comunicações são iniciadas com as bibliotecas `Imaq`, o `Imaqdx Open Camera` é conectado ao `Configure Acquisition` (modo `continuous` e com 3 `Buffers`) que por sua vez é ligado ao `Start Aquisition` (gatilho) e também se reserva um espaço de memória com o `Imaq Create` para o futuro vídeo. O plano de fundo é carregado pelo `Imaq ReadFile`. Dentro do `While Loop` é onde se ocorre as operações com as imagens, sendo a condição de parada algum erro ou o botão `STOP`. No fim as comunicações são encerradas com `Stop Acquisition`, `Unconfigure Acquisition` e `Imaqdx Close Camera`,

além do `Imaq Dispose` que libera a memória previamente alocada.

Os espaços de memórias estão na forma `Grayscale I16`, ou seja, em `16 bits` de tons de cinza com sinal e da mesma maneira todas as imagens devem estar neste formato para que as operações possam ocorrer. Para transformar os frames do vídeo obtido para este formato, é usado o `Imaq Cast Image`.

O plano de fundo é multiplicado com o `Imaq Multiply` por uma constante de 0 até 1 (pode ser encarado como uma porcentagem da imagem), o resultado vai subtrair a imagem obtida pela câmera com `Imaq Subtract`. Essa nova imagem é então passada para a próxima iteração por meio do `Shift Register` e também é multiplicada por outra constante e somada com a imagem da iteração anterior multiplicada por 1 menos a constante (com o intuito de obter melhores ajustes para visualização) [7]. No final, a imagem sem o fundo é exibida para o usuário, e as comunicações `Imaq` são encerradas.

2.2.2. PROGRAMA CALIBRAÇÃO ESPACIAL (SPACIALCALIBRATION.VI)

Este programa tem por objetivo encontrar a relação entre a dimensão de um pixel capturado em uma imagem da CCD, em dimensão de mícrons. Para isto, se utiliza de uma grade de difração (com período de 1 μm), utilizada na iniciação cientifica anterior [1], mas desta vez empregando o método de transformada de Fourier.

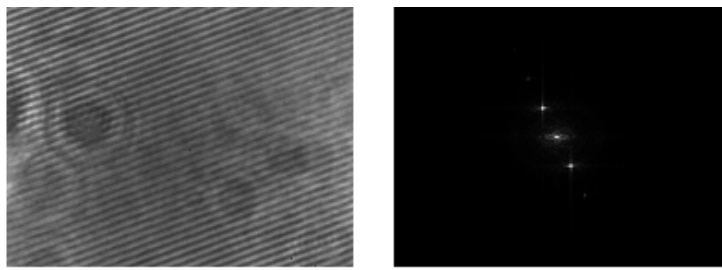


Figure 2.11: A esquerda a imagem de uma grade de difração, a direita a sua respectiva transformada de Fourier. Note os pontos brancos, que indicam as frequências espaciais (com relação ao ponto central) mais frequentes contida na imagem. A orientação dos pontos correspondem a orientação da grade na imagem original.

Primeiramente a imagem da grade, `8bits` grayscale, é carregada no LabVIEW por meio do `Imaq ReadFile`, um `Property Node` ajusta a

opção de aumento (`zoom to fit`). Em seguida é transformada em uma imagem complexa pelo `Imaq FFT`, para isso são alocados dois espaços de memória com o `Imaq Create`. Logo após, com um `Property Node` é possível obter as dimensões da imagem como mostra a figura 2.12.

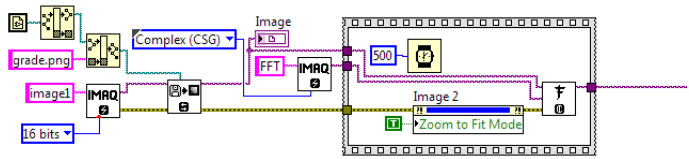


Figure 2.12: Diagrama da Calibração Espacial parte 1.

Em sua segunda parte, conforme 2.13, há um `Property Node` para a função de `zoom`, depois é utilizado o `unbundle` para separar do `cluster` a dimensão em pixels da altura e largura da imagem obtidos por um novo `Property Node` e ambos são divididos por 2 e então enviados a um outro `Property Node` da imagem que centraliza a figura.

Então é usada a função `Imaq Complex Flip Frequency` que transpõe as componentes frequenciais de uma imagem complexa, invertendo as de frequências altas com as baixas (representação padrão de uma imagem no espaço de Fourier). Feito isso é usado o Imaq Complex Image to Array para transformar essa imagem em uma matriz e obter seu módulo com o `absolute value`. Em seguida, essa matriz é dividida pela área da imagem multiplicada por 2 e multiplicada por 256 e inserida na função `In Range and Coerce` que trunca os valores maiores que 255 para 255 e os menores que 0 para 0, então é usada a função array to image para obter uma imagem de 8 bits.

O
Vision Assistant é uma ferramenta do LabVIEW que serve para facilitar certos aspectos da programação. A imagem processada dentro dele ao sair pode ser salva, similar ao programa de `SubtractBackground.vi` e a outra saída é justamente as coordenadas dos 2 pontos mais brilhantes com exceção ao central.

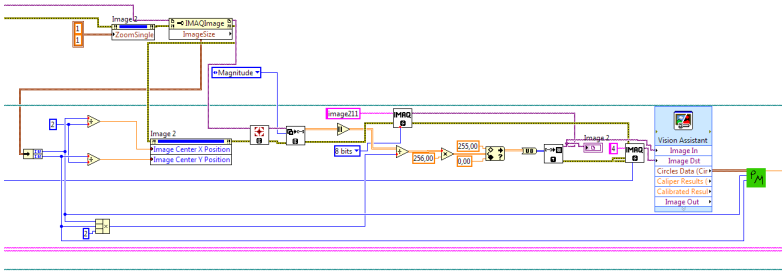


Figure 2.13: Diagrama da Calibração Espacial parte 2.

O *Vision Assistant* também pode ser encarado como uma subVI. Dentro dele, conforme figura 2.14, a entrada é a imagem 8 bits gerada da imagem complexa, essa imagem então é transforma em uma imagem binária pelo `Threshold`, que atribui valores para 0 a todos os pixels até a intensidade de 110 e 1 para os maiores que 110 e menores que 255. Em seguida o `Basic Morphology` dilata os pontos de valor 1, que passam a ter o raio um pouco maior. Após isso, a função `Circle Detection` consegue achar objetos circulares a partir de determinado raio e obter suas coordenadas.

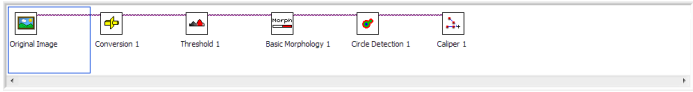


Figure 2.14: Vision Assistant.

Após sair do *vision assistant*, esses valores entram na subVI `PixelMicron`, figura 2.15, onde ocorre o cálculo para obter o valor da calibração:

Podemos considerada a imagem direta como uma matriz de pixels $M \times N$, neste caso $(M \times N)$=(960 × 1280). Seja portanto (u,v) o par de coordenadas no espaço de Fourier. A transformada de Fourier em duas dimensões é dado por,

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)} \tag{2.1}$$

onde o para de coordenadas do espaço direto vão de $(x,y) = (0,0)$ até $(x,y) = (M-1, N-1)$. Já as coordenadas da distância de um dos

pontos brilhantes da imagem de Fourier com relação ao central, esta relacionada a periodicidade da grade de difração. Portanto, obtemos o fator de calibração desejado em pixels/mícrons através ad equação,

$$L = \left(\sqrt{\left(\frac{u}{M}\right)^2 + \left(\frac{v}{N}\right)^2} \right)^{-1} \tag{2.2}$$

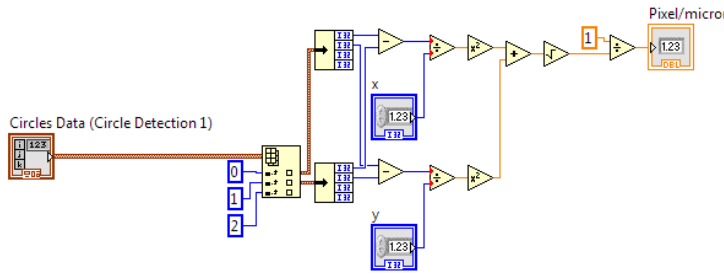


Figure 2.15: Diagrama bloco do calculo do fator de calibração de acordo com a equação 2.2.

2.2.3. AUTOFOCALIZAÇÃO (AUTOFOCUSVARIANCE.VI)

O objetivo deste programa é auxiliar o usuário obter determinada área da imagem focalizada com a maior precisão possível.

Primeiramente, pela figura 2.16, as comunicações são iniciadas com as bibliotecas `Imaq`, o `Imaqdx Open Camera` é conectado ao `Configure Acquisition` (modo `continuous` e com 3 `Buffers`) que por sua vez é ligado ao `Start Aquisition` (gatilho) e também se reserva um espaço de memória com o `Imaq Create` para o futuro vídeo. Além disso, também é iniciado uma `ROI` (Região de Interesse) padrão na imagem com o `Convert Rectangle to ROI`.

Dentro do primeiro frame do `Flat Sequence Structure` e do `While Loop`, o `Invoke Node` da imagem `GetLastEvent`, pode criar um booleano verdadeiro toda vez que se desenha uma nova `ROI` na imagem fazendo com que se ative o caso verdadeiro do `Case Structure`,

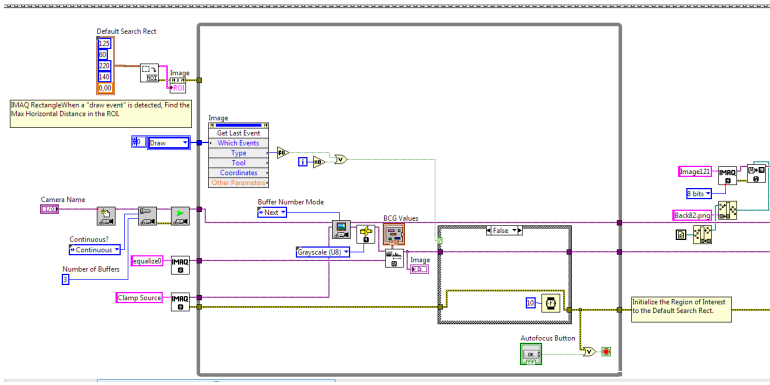


Figure 2.16: Autofocalização parte 1.

onde a `ROI` é transformada nas coordenadas de um retângulo com o `Convert ROI to Rectangle` e é limpa da imagem com o `Clear Overlay`. No caso falso, é ativo apenas um atraso de 10 ms. O vídeo é continuamente capturado com o `Get Image`, já explicado nos programas anteriores e transformado para `8 bits` com o `Cast Image`, e para melhorar os parâmetros de brilho, contraste e correção é usado o `BCGLookup`. Quando o usuário selecionar a `ROI` desejada basta clicar em `BackGround` para ir ao próximo quadro.

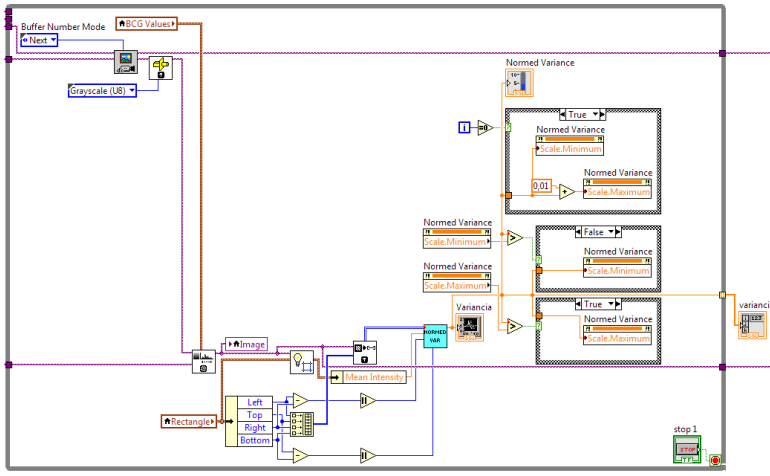


Figure 2.17: Autofocalização parte 2.

Na segunda parte, figura 2.17, que está no segundo quadro da `Flat Sequence Structure` temos a continuação da captura de vídeo como anteriormente. E a obtenção de valores para se calcular a variância normalizada, para isso, através de uma `Local Variable`, o retângulo anterior é passado para `Light Meter` que entre outras funções pode calcular a intensidade média da `ROI`, a `ROI` também é transformada em matriz com o `Image To Array` e passada para a SubVI (`NormedVariance`) assim como a altura e a largura da região de interesse.

O resultado é constantemente mostrado em um `Graph` pelo tempo e em um "termômetro" também chamado de `Normed Variance`, o qual o usuário pode ir verificando a pontuação mais alta no ajuste fino, enquanto tenta deixar o objeto em foco manualmente e quando estiver pronto basta apertar o botão `Stop`. O termômetro se ajusta automaticamente quando se obtém um novo máximo ou mínimo através da seguinte lógica, através de `Property Node`: Na primeira iteração o termômetro atribui o valor atual como o mínimo e o máximo como o mínimo somado com um pequeno incremento. Nas iterações seguintes vai sendo verificado por meio de `Case Structure` se o valor supera o máximo ou o mínimo, e caso for verdade os máximos ou mínimos são trocados.

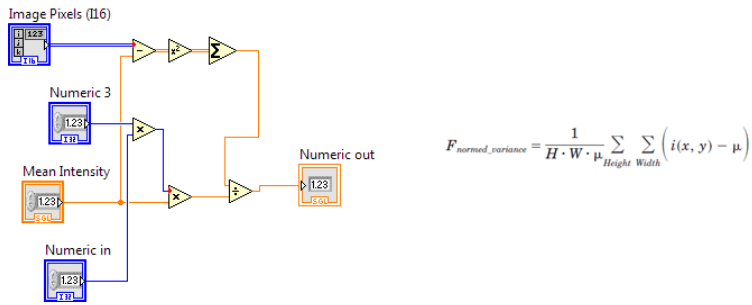


Figure 2.18: A subVI acima representa o algoritmo para calcular a variância normalizada [8].

Na terceira parte, figura 2.19, no terceiro quadro há um `Wait` de 200 ms, e no quarto, a continuação da captura de vídeo. Este pode ser parado a qualquer momento para se encerrar o programa com um segundo `Stop` e então encerra-se as comunicações com a câmera e se exclui a imagem.

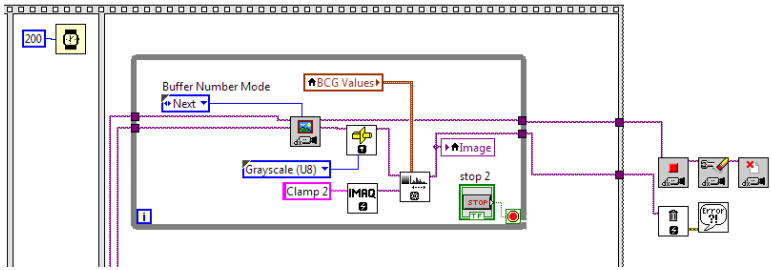


Figure 2.19: Autofocalização parte 3.

2.3. CONTROLE DO ESTÁGIO PIEZOELÉTRICO E DA CÂMERA

2.3.1. PROGRAMA VARREDURA DE PINÇA (2DSCAN.VI)

Esse programa foi inicialmente desenvolvido com finalidade de conseguir aprisionar opticamente vários mitoplastos, para auxiliar na pesquisa do grupo da Prof.ª Iseli Lourenço Nantes. Porem serve para agregar qualquer tipo de micro objetos, usando a armadilha óptica para agrupá-las e estudo posterior. Para tal fim, foi elaborado dois modos de movimentar o estágio piezoelétrico em relação ao feixe laser que fica sempre centralizado. Um de posicionamento aleatório e outro do tipo *raster-scan*, onde estágio se move linha-a-linha.

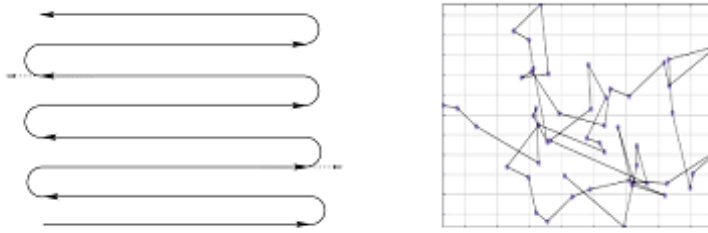


Figure 2.20: Movimentação *raster-scan* e aleatória. Extraído de [9] e [10]

As comunicações são iniciadas com subVI `Open Piezo` criando uma referência para a camera e as bibliotecas `Imaq`, no caso o `Imaqdx Open Camera` que é conectada ao `Configure Acquisition` (configurado no modo `continuous` e com 3 `Buffers`, ou seja, modo de aquisição será

um vídeo) que por sua vez é ligada ao `Start Aquisition` (gatilho para iniciar a captura). Nesta parte inicial também se abre um espaço de memória com o `Imaq Create` para o futuro vídeo e também nesta parte estão os métodos `GetVoltOutput` para X e Y para ter conhecimento da posição inicial.

Após essa parte se entra em um `While Loop` e dentro deste se encontra uma `Case Structure` acionado por um booleano cuja posição *True* é o movimento randômico e *False*, o *raster-scan*.

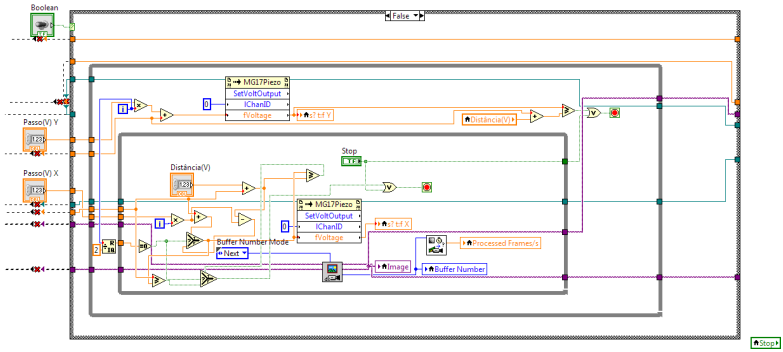


Figure 2.21: Diagrama de Raster-Scan.

No `While Loop` interno da imagem acima há um `Select` que verifica se o resto da divisão das iterações do `loop` mais externo é igual a zero, para isso pode-se utilizar a função `Quotient & Remainder`, caso seja verdade o valor passado para o método `SetOutputVolt` é o **(valor inicial) + (passo X) × (número de iterações)** e caso contrário o valor passado é o **(valor inicial + distância) - (passo X) × (número de iterações)**. Um outro `Select` para verificar se o resto da divisão das iterações do `loop` mais externo é igual a zero, no caso verdadeiro é passado o valor booleano da comparação entre o valor passado do `Select` anterior e do **valor inicial + distância** e caso contrário é feito uma comparação entre o valor inicial e do valor passado do `Select` anterior. Se o valor desse segundo `Select` for verdadeiro, significa que a condição de parada do `While Loop` interno foi satisfeita, sendo a outra o botão `STOP`. No `While Loop` externo, Y funciona semelhante a X, mas para cada iteração em Y ocorre todas em X.

Essa lógica permite uma transição mais suave para a próxima iteração em Y, isso porque desse modo o *raster-scan* se movimenta como

um zigue-zague (caminha-se para a direita em X, quando o resto da divisão for 0, e no final dá-se um passo em Y, então X caminha para esquerda, quando o resto da divisão for 1, e novamente mais um passo em Y e assim por diante).

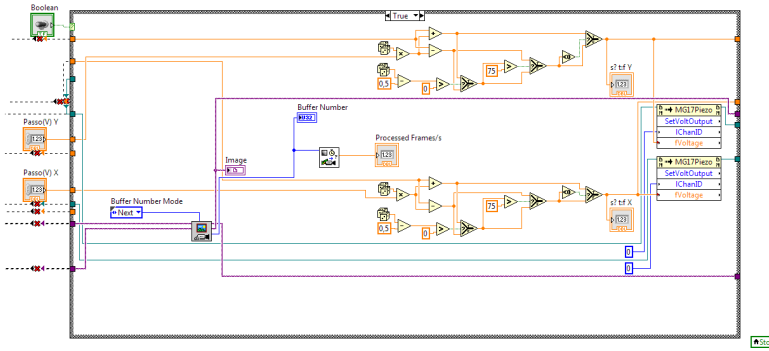


Figure 2.22: Diagrama de Movimento Aleatório.

Na figura 2.22 usa-se o `Random Number` para gerar números aleatórios de 0 a 1 que será multiplicado por um valor determinado pelo usuário, esse resultado será somado ou subtraído do valor original dependendo se um outro valor randômico é maior que 0,5. Para o novo valor é verificado se é maior que 75 V, se for então ocorre a subtração novamente, caso não o mesmo valor é passado sem alteração. Então se verifica se o valor é menor do que 0, se for, a operação de soma ocorre novamente. O valor então é escrito no método `SetOutputVoltage` e a operação fica em ciclos até que se aperte o botão `STOP`. O novo valor de posição vira o novo "valor inicial" por meio do `Shift Register`.

Dentro do `While Loop` se adquire o vídeo frame a frame devido ao `Imaqdx Get Image` com o `Buffer Number Mode` ajustado para `Next` (Neste parâmetro o programa aguarda o frame seguinte) e para se obter a taxa de frames por segundo é utilizado o `Calculate Frames Per Second`.

2.3.2. PROGRAMA OBTENÇÃO DE BACKGROUND (GETBACKGROUND.VI)

Inicialmente neste programa são abertas as comunicações para a captura de vídeo e controladores piezo em X e Y. Em seguida, por intermédio da SubVI `Move Piezo`, X e Y são centralizados na posição

37,5 V. Ao adentrar no `While Loop` principal, posições aleatórias começam a ser enviadas para os controladores piezo exatamente como no módulo `Varredura de Pinça` quando se aciona o movimento aleatório, enquanto isso cada frame do vídeo é somado ao seu anterior até atingir 254 frames ou acionar o botão `STOP` (As imagens são convertidas para 16 bits com sinal em escala de cinza).

Após sair do `While Loop` ao número de iterações é somado mais 1 (para incluir a iteração 0) e com esse número se divide todos os frames somados obtendo uma média que é o plano de fundo.

No fim, as comunicações são encerradas e o plano de fundo é salvo sem compressão e perca como um tipo de `16 bits` com sinal e `8 bits` sem sinal, ambos em escala de cinza.

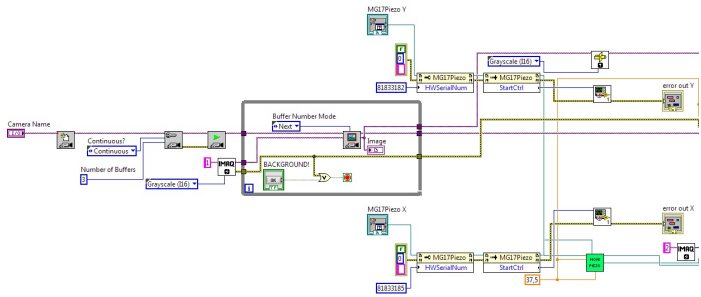


Figure 2.23: Diagrama de blocos da Obtenção de Plano de Fundo parte 1.

Como pode-se ver acima, para abrir comunicação com a webcam é usado o mesmo processo do módulo `Varredura de Pinça`. Até que o usuário aperte o botão `Background!` o vídeo continuará a ser capturado normalmente, só então que se iniciará a comunicação com os controladores piezo em X e em Y, como já mostrado na subVI `Open Piezo`. Nessa parte, o último frame capturado pela câmera é convertido para `Grayscale I16` com `Imaq Cast Type` para ser usado como imagem a ser somada com o primeiro frame no próximo `While Loop`, isso porque para obter o plano de fundo foi-se somando cada frame com seu anterior sucessivamente.

Na parte 2, esse segundo `While Loop` se usa exatamente a movimentação aleatória da `Varredura de Pinça`, os frames capturados são transformados de `RGB U32` para `Grayscale`

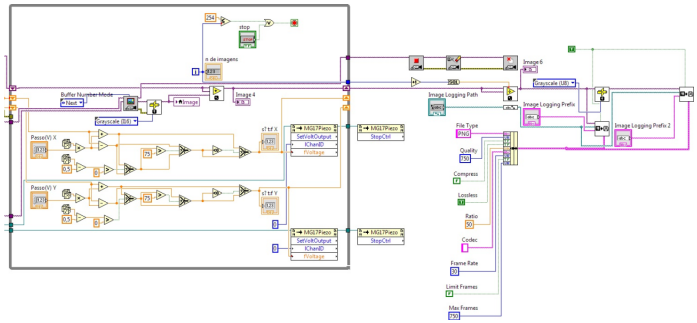


Figure 2.24: Diagrama de blocos da Obtenção de Plano de Fundo parte 2.

`I16`, então são somados com o `Imaq Add` ao frame anterior carregado pelo `Shift Register`. Depois de acionada a condição de parada do `While Loop` (até 254 iterações ou botão `Stop`), as comunicações com a câmera e o controlador piezo são encerradas e também a somatória dos frames é dividido pelo número de iterações criando um plano de fundo. Para salvar o plano de fundo é usado o `Vision Acquisition Image Logging` que precisa de alguns parâmetros como onde o arquivo vai ser salvo (`Image Logging Path`), seu nome (`Image Logging Prefix`) e um `cluster` (formato da imagem, qualidade, compressão, perca, codec, taxa e máximo de frames) formado através da função `Bundle`.

2.3.3. RASTREAMENTO DE UM OBJETO (TRACKING.VI)

O objetivo deste programa é rastrear (determinar a localização quadro-a-quadro) de uma amostra durante a aplicação de uma onda triangular no estagio piezoelétrico. Com estes dados a constante elástica da armadilha óptica pode ser determinada.

Inicialmente, conforme a figura 2.25, o usuário precisa indicar o valor final, inicial e incremento da amplitude da onda triangular, logo em seguida a chave seletora passará o valor do número de série de x ou de y para inicializar o motor piezo desejado. Após isso, há uma verificação para ver se o valor final é maior ou igual a 75V, caso seja, o valor final é substituído por 75V, o mesmo vale caso o valor inicial seja menor que 0, só que será substituído por 0. Nessa mesma etapa, a subtração do valor final pelo inicial é dividida pelo incremento e somado a mais 1 com o intuito de se obter o número de iterações para o `For`

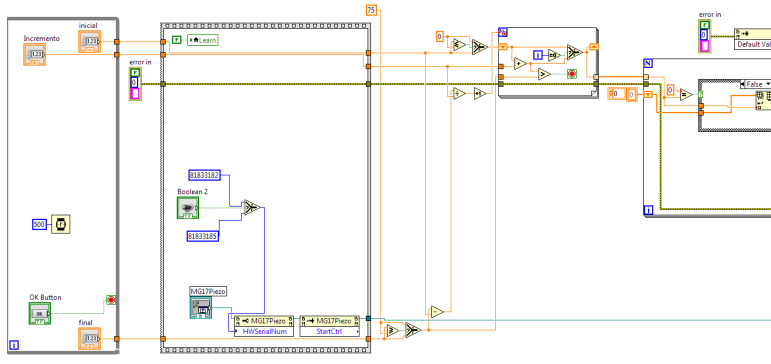


Figure 2.25: Tracking parte 1.

`Loop`.

Dentro do `For Loop` há um `Shift Register` que soma a cada iteração o incremento ao valor inicial, com exceção a primeira iteração que é passada somente o valor inicial, e armazena esses valores em um vetor com a propriedade de `Indexing` do `For Loop` cuja condição de parada é esse novo valor de soma der maior que o valor final. O vetor criado que contém todas as amplitudes em voltagem da onda é indexado em outro `For Loop` para eliminar o valor 0, caso o usuário tenha colocado o 0 como valor inicial. A cada iteração será adicionado um valor no novo vetor com o `Insert into Array`, mas só se o valor do vetor indexado for diferente de 0.

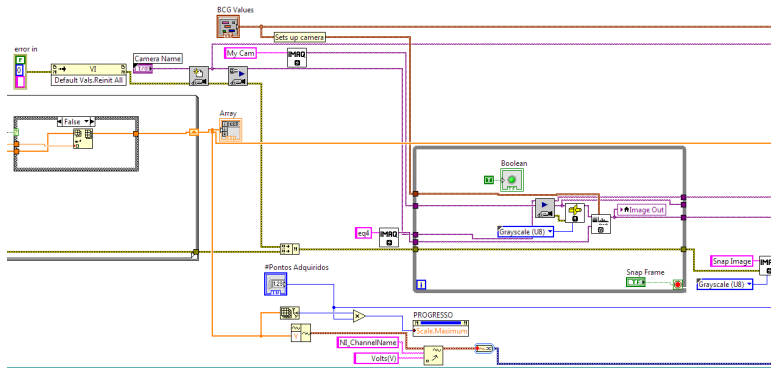


Figure 2.26: Tracking parte 2.

Na parte 2, há a inicialização da comunicação com a câmera, reserva de espaços de memória, aqui também é definido o valor máximo da barra de progresso que é dado pela multiplicação dos pontos a serem adquiridos pelo tamanho do vetor de amplitudes.

Dentro do `While Loop` existe um *led* que é ativado, assim o usuário saberá que pode clicar em `snap frame` para capturar um frame.

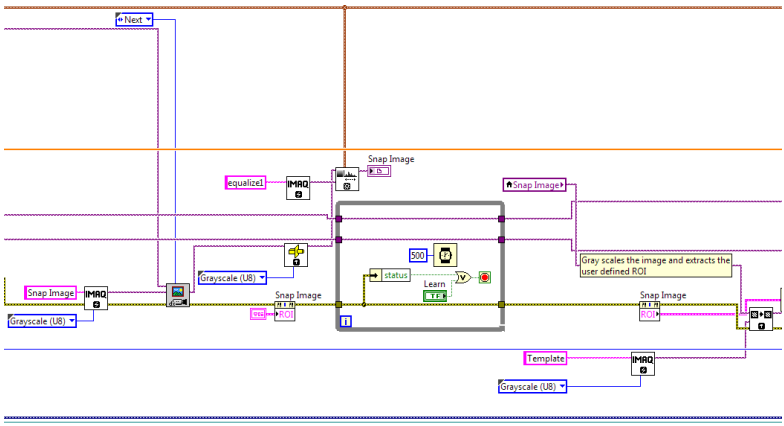


Figure 2.27: Tracking parte 3.

Na parte 3 (figura 2.27), é a função `Get Image` que captura o quadro que é mostrado para o usuário e o mesmo seleciona a região de interesse que é gravada no `Property Node`. O programa fica então aguardando que o usuário aperte `Start (learn)`, depois o quadro capturado tem todas as suas características copiadas para o processamento com `Imaq Copy`.

Nessa parte, do quadro copiado é extraído a região de interesse com o `Imaq extract`, após isso com o `Setup Learn Pattern 2` que define os parâmetros usados durante a fase de aprendizagem da correspondência do padrão, então o `Learn Pattern 2` cria uma descrição da imagem que será procurada durante a fase de verificação de correspondência. Essa informação de descrição é anexada na imagem de entrada. Durante o processo de verificação, a descrição da imagem é extraída dela e usada para achar o padrão na imagem inspecionada.

O valor do período passado é verificado com um `Select` para garantir que o usuário não coloque um demasiadamente pequeno, caso

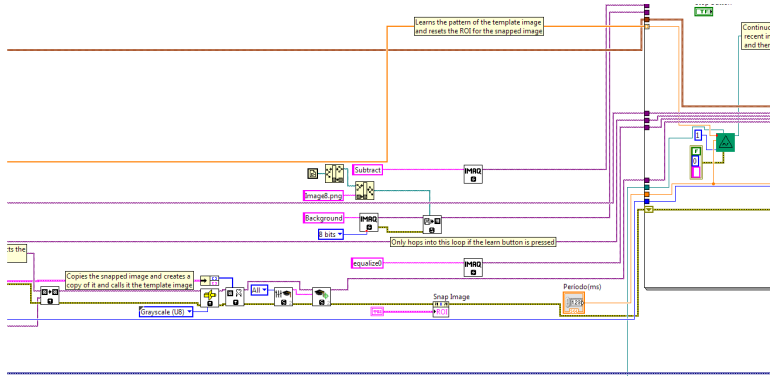


Figure 2.28: Tracking parte 4.

seja, é passado então 600 ms.

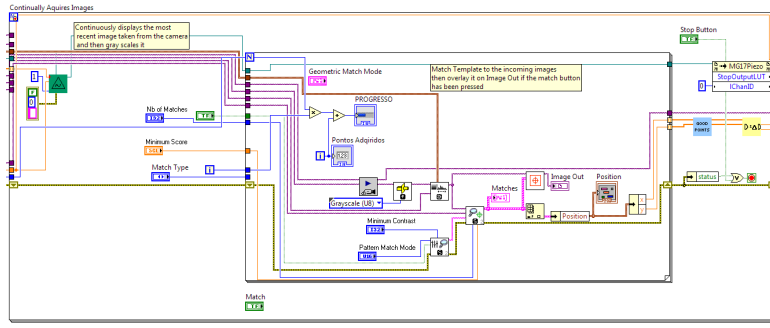


Figure 2.29: Tracking parte 5.

Na parte 5 (figura 2.29), há um for `loop` dentro de outro, no mais externo está a VI `TriangularWave` que gera ondas triangulares (nesse caso no modo contínuo) que é alimentada pelo vetor de amplitudes que está indexado, depois de passar no `For Loop` interno, a onda é interrompida com `StopOutputLUT` para recomeçar na próxima interação com a próxima amplitude. É também no `For Loop` externo que se tem a seleção dos pontos "bons" na VI `GOOD POINTS` e o cálculo da distância e o seu erro na VI `DistanceUncertainty` que são indexados em um `array`.

No `For Loop` interno existe a contagem do progresso feito da seguinte forma: soma-se a iteração do `For Loop` interno com a iteração do externo multiplicado pelo número de pontos a serem adquiridos.

É também nessa parte que ocorre a captura normal de vídeo com o matching:

O Setup Match Pattern 2 define os parâmetros que são usados para correspondência do padrão na imagem inspecionada no Match Pattern 2 que possui exatamente essa função de inspeção, então a função Draw Pattern Matches Position desenha para o usuário na imagem de saída um retângulo vermelho no objeto rastreado. A posição do centro do retângulo é indexada em um array no For Loop interno e usado na VI GOOD POINTS.

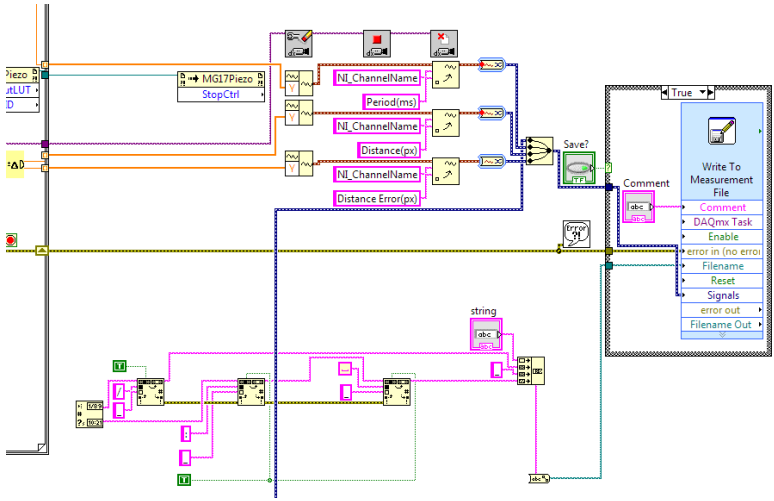


Figure 2.30: Tracking parte 6.

Na parte 6, o motor piezo é encerrado junto com a câmera e os dados (Amplitude(V), Período(ms), Deslocamento(Pixels), Erro do deslocamento(Pixels)) podem ser salvos se usuário quiser com o VI Express Write to Measurement File. Os dados são gravados em Signals cuja entrada é Dynamic Data Type, por isso é preciso transformar os arrays em formas de onda com Build WaveForm e depois Set WaveForm Attribute para se colocar o nome do canal e então converter com Convert to Dynamic Data4. A função Merge Signals une esses sinais para serem gravados em VI Express Write to Measurement File.

O nome do arquivo é conectado em Filename no VI Express Write to Measurement File. A função Get Data/Time String

fornece uma string com as informações de data e hora atuais do computador, em seguida é usado `Search and Replace String` para trocar os caracteres "– ; : ; espaço" por "" e finalmente `Concatenate Strings` para juntar as `strings` do nome que o usuário quiser digitar, da data e da hora e formar um único nome para o arquivo.

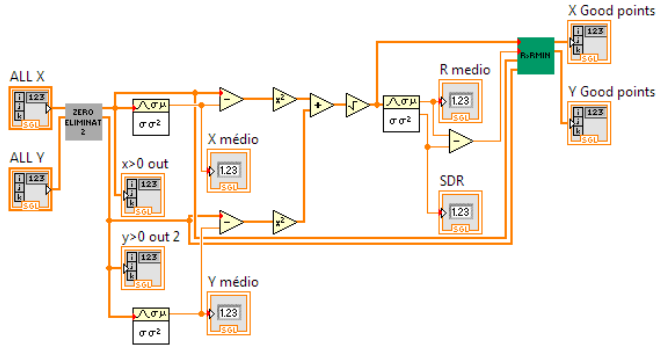


Figure 2.31: Tracking parte 6.

Essa subVI seleciona os pontos bons para o cálculo da distância a partir do critério de que os raios formados por esses pontos, sendo o centro da circunferência dado pelo x médio e y médio das coordenadas, sejam maiores que um raio mínimo dado pelo valor médio dos raios de todos os pontos menos o desvio padrão deles. Depois a subVI `RgreaterThanRMin` fará o trabalho de separar os pontos por meio desse critério.

Pode ser que surjam alguns zeros no `array` de pontos devido ao `matching` não encontrar o padrão da imagem em pequeno instante de tempo, então para que não afete os cálculos são retirados.

Um novo `array` será construído com o `Shift Register` e caso o valor do `array` indexado seja 0 em determinada iteração, o valor dessa iteração não é passado no caso verdadeiro do `Case Structure`, mas para o caso false os valores atuais do `array` indexado são escritos no novo `array` com o `Index into Array`.

É usado a mesma lógica nessa subVI, porém o critério é o `array` dos raios indexados serem menores que o raio minimo. Se um desses raios se mostrar maior que raio minimo, a condição fica verdadeira e esses pontos não são escritos.

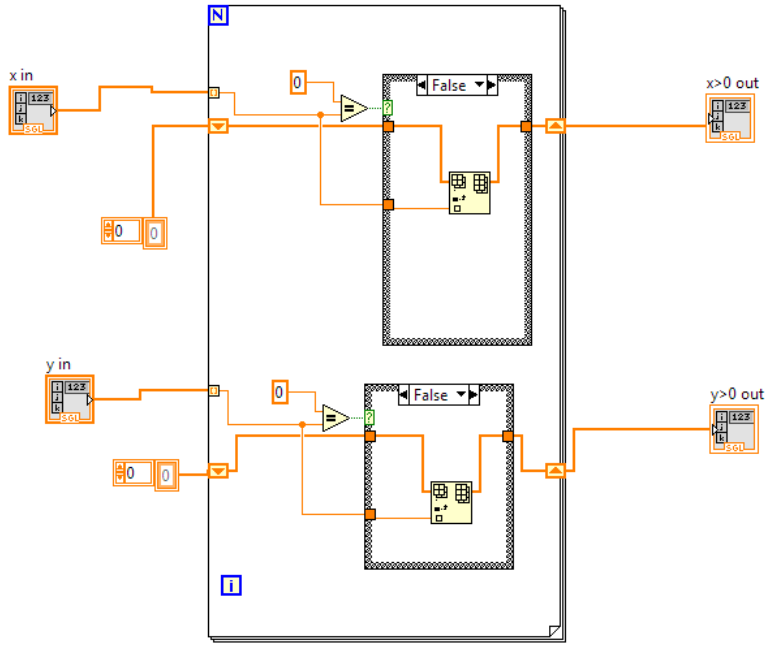


Figure 2.32: SubVI ZeroEliminator.

Os pontos bons são separados em dois grupos que refletem as duas posições que em que a amostra permanece maior tempo durante a movimentação com a onda triangular. É tirado a média de X e Y dos grupos 1 e 2 e esses quatro valores são agrupados em um `cluster` que são jogados da SubVI `Points Distance` para que esse deslocamento seja calculado em *pixels*. E o erro calculado de acordo com a seguinte equação obtida da derivada parcial:

Sendo o deslocamento:

$$d = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} = \sqrt{X^2 + Y^2} \tag{2.3}$$

E a incerteza por definição por definição:

$$\sigma_d = \sqrt{\left(\frac{\mathrm{d}d}{\mathrm{d}X}\right)^2 \sigma_X^2 + \left(\frac{\mathrm{d}d}{\mathrm{d}Y}\right)^2 \sigma_Y^2} \tag{2.4}$$

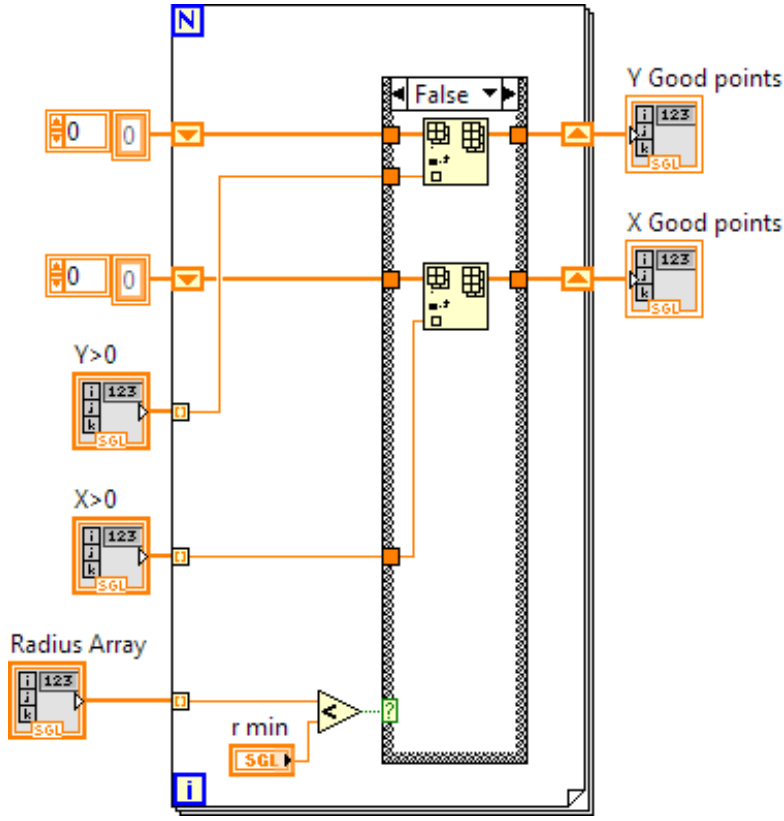


Figure 2.33: SubVI RGreaterhanRMin.

A incerteza será dada por:

$$\sigma_d = \frac{1}{d}\sqrt{(X_1 - X_2)^2\left(\sigma_{X_1}^2 + \sigma_{X_2}^2\right) + (Y_1 - Y_2)^2\left(\sigma_{Y_1}^2 + \sigma_{Y_2}^2\right)} \tag{2.5}$$

As incertezas de X e Y médios dos dois grupos são obtidas estatisticamente a partir da média e variância das medições com a função `Std Deviation and Variance`.

Essa subVI vai separar os pontos bons em dois grupos seguindo a mesma lógica de separação, só que o critério será o resultado do arco tangente dos pontos X e Y, caso seja maior que 0 vai para um grupo no caso verdadeiro do `Case Structure`, caso seja menor que 0 vai para outro grupo no caso falso, isso é explicado devido a origem do centro de coordenadas estar deslocado para o X e Y médio, o que faz com que cada

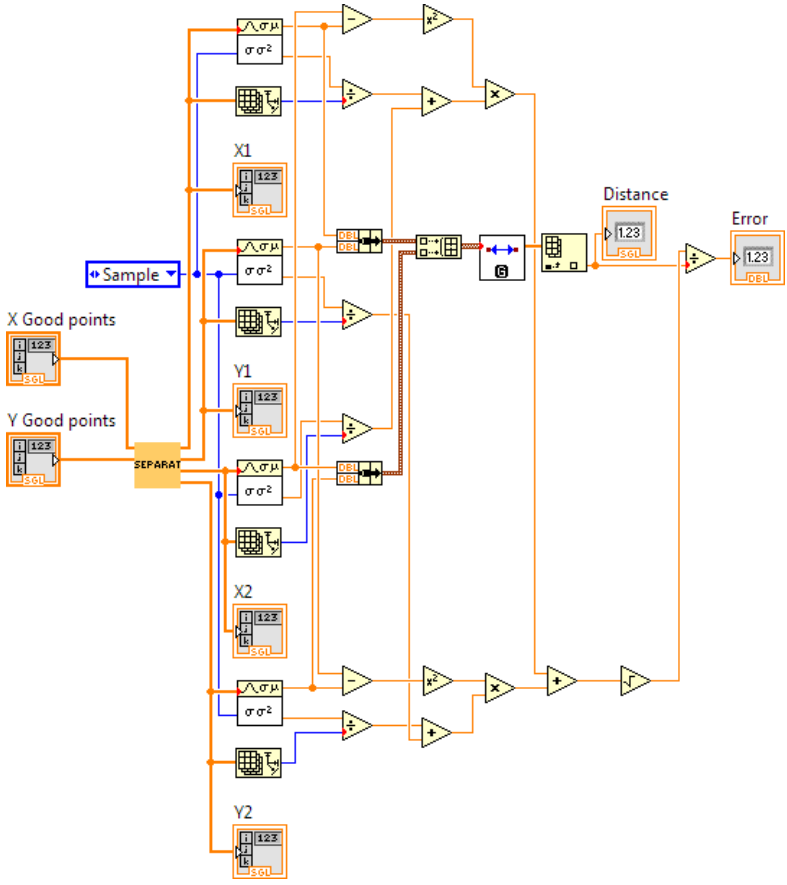


Figure 2.34: SubVI Distance and DistanceUncertainty.

grupo esteja em uma parte oposta do círculo trigonométrico, fazendo com que esse seja um bom critério de separação.

2.3.4. CALIBRAÇÃO PIXEL/VOLT (PIXELVOLT.VI)

O seu objetivo é obter o coeficiente angular e linear e seus respectivos erros para obter a relação de pixel por volts. Este programa é idêntico ao de `Tracking` até a parte 4.

Nessa parte, o `matching` é o igual ao programa de `Tracking`, a diferença está que dessa vez a movimentação do piezo não é dada pela onda triangular mas pela subVI `move piezo`, cuja as posições em

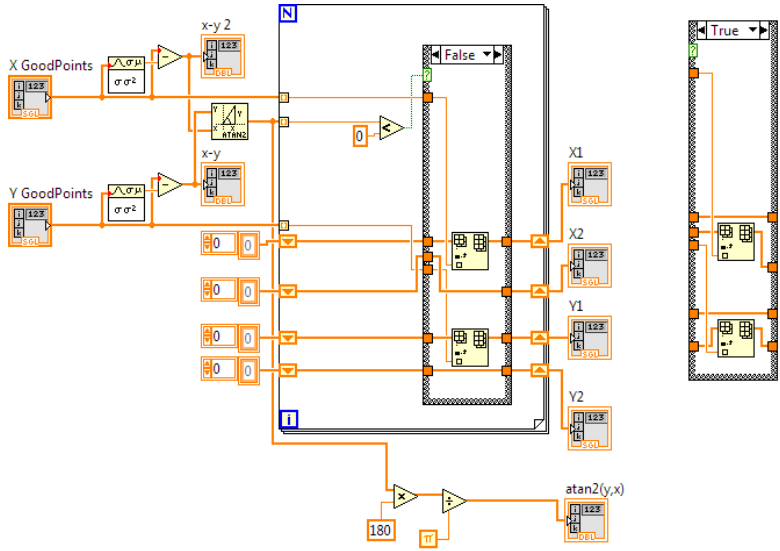


Figure 2.35: SubVI Separator.

Volts é dada pelo `array` definido inicialmente no programa idêntico ao `Tracking`. Nessa parte as posições de X e Y também são indexados para formar um `array` de posições.

Nessa parte as comunicações da câmera e do piezo são encerradas, os valores de X e Y são elevados ao quadrado, somados e depois é extraído sua raiz quadrada para ter o real deslocamento da amostra, esses valores então são enviados para SubVI `General Linear fit`, sendo que ao mesmo tempo são enviados os valores de uma matriz dos valores de X obtidos pelo `Formula Node` (que funciona como uma fórmula matemática na linguagem C, no caso é atribuído para y0 o valor de uma constante e para y1 o valor de uma variável dependente) depois de terem qualquer valor nulo retirado pela subVI `ZeroEliminator` (caso o matching pare de rastrear por algum motivo o valor nesse instante de tempo é 0). Dessa Subvi é extraído uma matriz de covariância cujo termo (0;0) é o quadrado do erro do coeficiente linear e o termo (1;1) é o termo do coeficiente angular, os valores ajustados e os coeficientes linear e angular. Por meio da função `Index Array` é possível separar cada elemento.

Além dessas funções, é plotado um gráfico com os valores do deslocamento em pixels versus os valores de voltagem em volts já com o

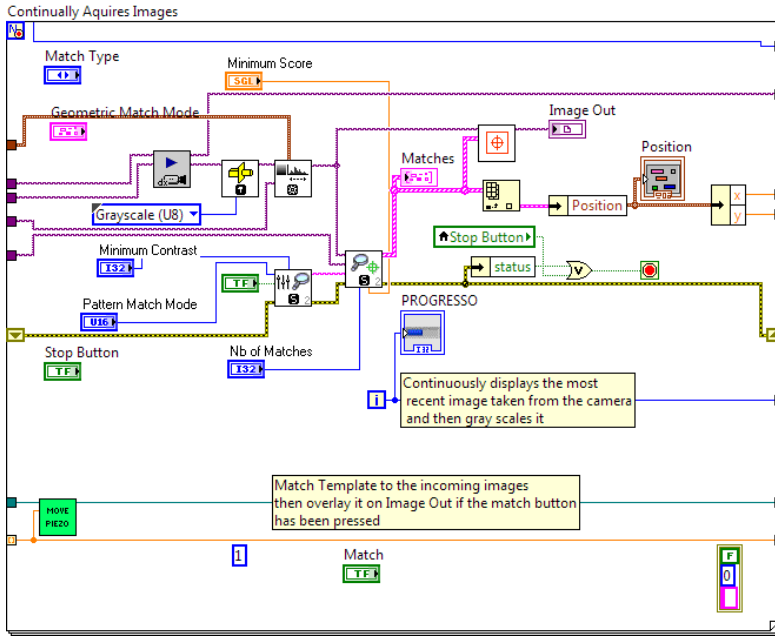


Figure 2.36: Pixel/Volt parte 5.

ajuste linear em `XY Graph`.

Nessa última parte é onde é salvo os dados obtidos (Voltagem(V), Deslocamento(px); Coeficiente linear, Coeficiente angular, Erro dos dois coeficientes). O processo é o mesmo da VI de `Tracking`.

2.4. PROGRAMA FINAL (FINALPROGRAM.VI)

O programa final é a junção de todos os módulos anteriores. Nele o usuário terá todas as funções unificadas e uma interface amigável para o uso.

A figura 2.39 mostra a primeira tela que o usuário verá ao iniciar o programa, nela é mostrada se o motor piezo e a câmera estão prontos para uso, a última atualização das calibrações de pixel/volt, pixel/mícron e da constante elástica, além da última vez em que o plano de fundo foi obtido. Nessa parte também é mostrado o diretório cujo usuário poderá escolher onde gravar seus arquivos, o padrão é a pasta do próprio programa.

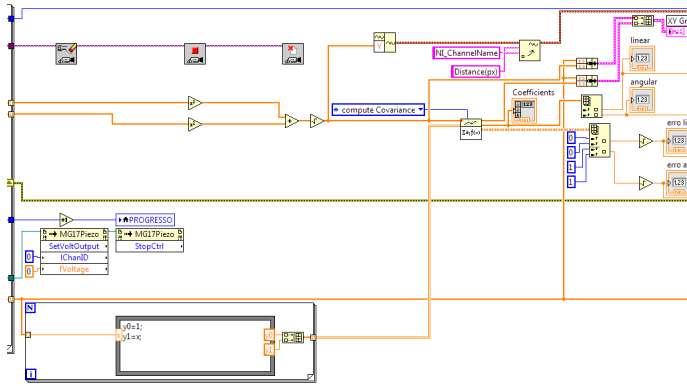


Figure 2.37: Pixel/Volt parte 6.

Em cada módulo existem *leds* que indicam se o usuário pode começar a usar o programa (verde) ou precisa esperar/indicar que o programa encerrou (vermelho), assim como uma barra de progresso em cima do quadro principal que é usada em muitos dos módulos.

Para que cada módulo possa ser usado somente quando necessário, é usado a função `tab` que quando junto de um `case structure` permite que cada módulo esteja associado a um quadro do `case structure`. Conforme figura 2.40 pode-se notar que a cada iteração do `while loop` é aguardado um evento de mudança de aba ou de pressionar o botão parar, isso por meio do `event case` que faz justamente a função de ser acionado quando determinado evento acontece.

Além disso, dentro dos módulos foi adicionado uma subVI que mostra para o usuário uma escala aproximada do tamanho real em mícrons na imagem principal 2.41.

Essa subVI tem como input a altura, a largura e o valor da calibração pixel/mícron. Para a função `Imaq Overlay Line`, que cria uma linha na imagem, são necessárias coordenadas de dois pontos, para o eixo Y de ambos os pontos é o valor de 5% da altura da imagem, já para o primeiro ponto em X é definido como a largura da imagem menos um quarto de seu tamanho e o segundo ponto em X como o primeiro mais um quarto de dele, fazendo com que a linha fique no canto inferior direito da imagem. As mesmas coordenadas são usadas para escrever o valor da escala com o `Imaq Overlay Text`.

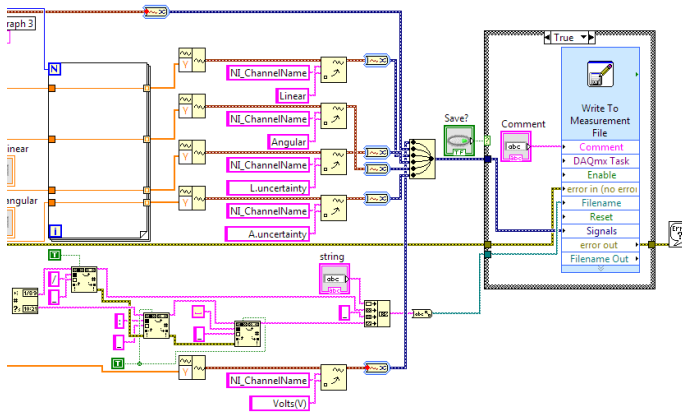


Figure 2.38: Pixel/Volt parte 7.

A escala é sempre múltiplo de 5, ou seja o valor da escala é arredondado quando o tamanho da linha é divido pelo valor de calibração pixel/mícron. Isso é feito verificando o resto da divisão, se for 0 o valor é passado sem alteração, se for menor que 2,5 é arredondado para baixo, mas se for maior que 2,5 é arredondado para cima.

2.4.1. MÓDULOS DE CALIBRAÇÃO

Nessa subseção estão as partes do programa referentes a calibração.

A imagem 2.42 é referente ao painel frontal do programa de Calibração Espacial, é possível capturar uma imagem para a calibração e salvá-la (pode-se usar a subtração de fundo), ou usar a última imagem gravada na pasta do programa. A calibração é salva num arquivo de extensão `.txt` na pasta do programa e do diretório escolhido, além de aparecer na interface para verificação.

A imagem gerada pela Transformada de Fourier é mostrada ao usuário em um outro quadro.

A figura 2.43, é referente ao painel de controle do programa de Calibração Pixel/Volt. Assim que possuir sinal verde, o usuário tem espaços para inserir a voltagem inicial, o incremento a cada iteração e a voltagem final para o motor piezo, e para decidir se vai deslocar no eixo X ou Y e se vai salvar o valor da calibração.

Na segunda parte, depois de clicar no botão pronto, o usuário captura um frame do vídeo e seleciona uma região de interesse, e ao

Figure 2.39: Apresentação do programa ao usuário

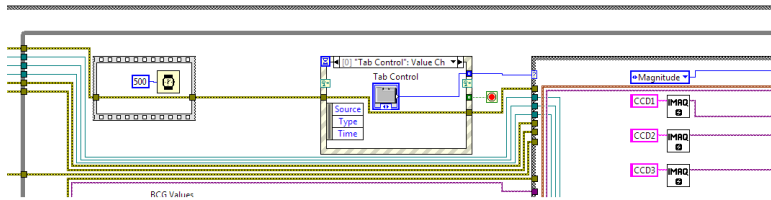


Figure 2.40: Estrutura do programa final

clicar em começar, o motor piezo se move conforme as configurações. Os dados obtidos no posicionamento do objeto rastreado são usados para calcular o ajuste de uma reta no gráfico de deslocamento por volts mostrado na interface, assim é possível obter o coeficiente angular que é justamente o valor desejado. O coeficiente angular é mostrado para o usuário na interface junto com o linear e suas respectivas incertezas.

O painel de controle 2.44 pertence ao programa de `Tracking`. A primeira parte para decidir as voltagens é semelhante ao programa anterior e o usuário poderá decidir o período da onda, o número de pontos que deseja pegar da imagem e também as voltagens para o programa, porém para a onda triangular ao invés do deslocamento.

Na segunda parte, assim que o segundo *led* verde acender depois que o botão pronto for ativado, o usuário deve clicar em diâmetro para que um quadro seja capturado e o usuário possa desenhar uma linha no quadro principal para medir o tamanho aproximado do objeto de

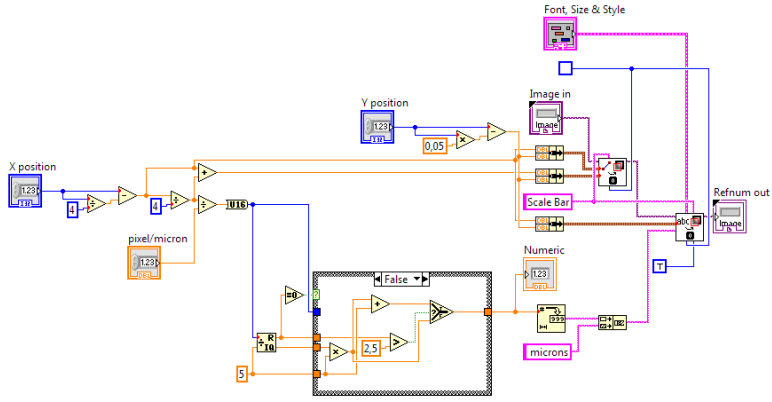


Figure 2.41: Escala em mícrons

interesse, para isso basta clicar diretamente no quadro uma vez que o programa seleciona essa ferramenta automaticamente. Em seguida ao pressionar o botão selecionar região de interesse, a ferramenta no quadro principal mudará de linha para retângulo automaticamente para que o objeto de interesse seja selecionado, por fim basta clicar em começar.

Durante a execução da onda, é mostrado a voltagem atual e os pontos adquiridos durante a iteração

Quando o programa encerra, os resultados são mostrados, a distância (px) e sua incerteza associada (px), e a constante elástica (N/m), assim como um gráfico de distância(px) por velocidade(V/s).

2.4.2. MÓDULOS DE ROTINA

Os módulos dessa subseção são referentes aos programas de rotina.

O painel de controle 2.45 é o de autofocalização. O usuário precisa primeiro selecionar o objeto de interesse e depois clicar em autofocalizar, então o termômetro que indica a pontuação começará a oscilar cabendo ao usuário fazer o ajuste e deixar o termômetro o mais cheio possível. O usuário pode se orientar pelo `Chart`, que é o gráfico da pontuação em função do tempo percorrido.

A imagem 2.46 faz referência ao painel de controle do módulo de vídeo e foto. Ao iniciar esse módulo, o usuário encontra um medidor da taxa de frames por segundo, pode gravar vídeos ao deixar aciona o

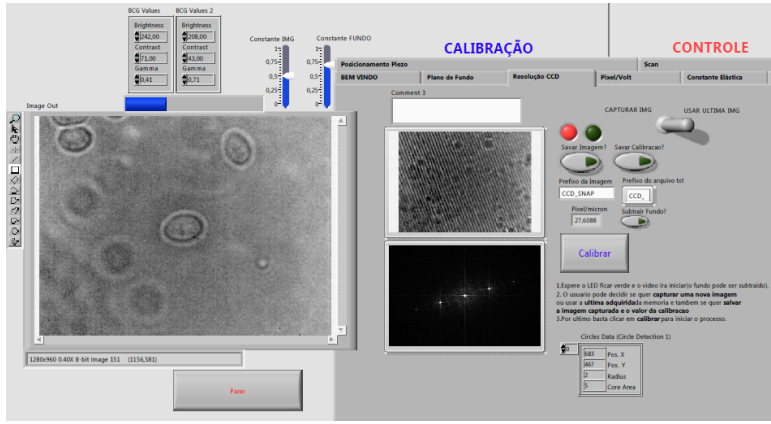


Figure 2.42: Calibração pixel/mícron

botão para tal e quando quiser parar basta desativar o botão. Para tirar uma foto , basta deixar o botão Salvar foto ligado e clicar em capturar e o programa se encerra.

O painel de controle acima 2.47 é o de obtenção de plano de fundo, basta o usuário selecionar o passo desejado para cada motor e clicar em Plano de Fundo que o programa começará a se mover aleatoriamente para capturar as imagens que no final se tornarão o plano de fundo.

Acima 2.48 está o painel de controle da função de posicionamento do piezo, duas barras que se movimentam foram incorporadas para o usuário ter melhor orientação, no entanto ainda pode entrar com valor númerico na caixa se desejar mover o motor desse jeito.

Esse último painel de controle 2.49 é o do módulo de escaneamento. Antes de iniciar o escaneamento, o usuário decide por meio de uma chave seletora se deseja o movimento o aleatório ou o escaneamento do tipo raster, decide o tamanha do passo para os eixos X e Y nas caixas e no caso do raster, a distância a ser percorrida pelos motores piezo. Se o usuário desejar parar o programa, basta clicar em interromper.

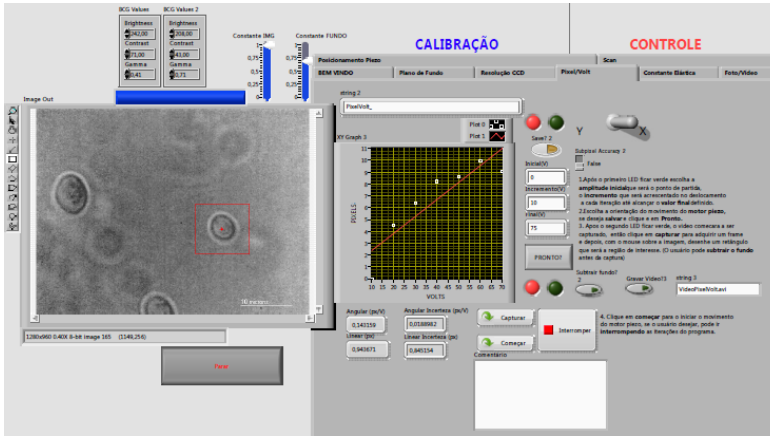


Figure 2.43: Calibração pixel/volt

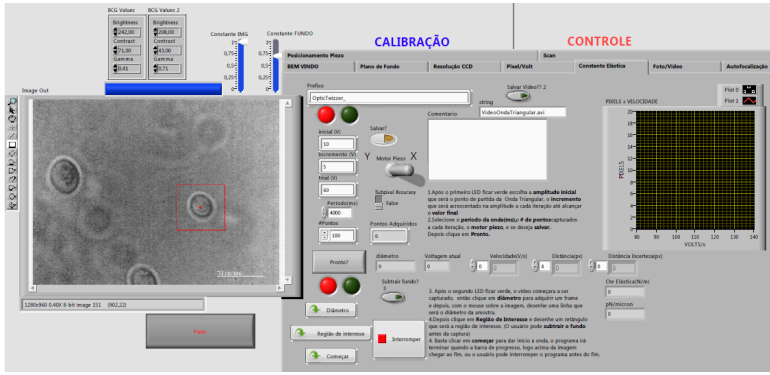


Figure 2.44: Calibração da constante elástica

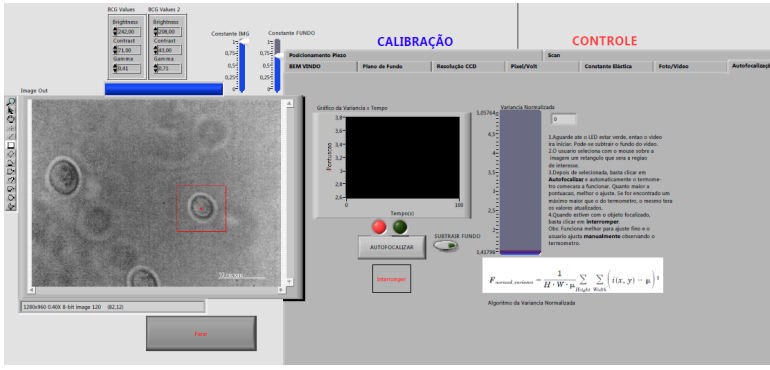


Figure 2.45: Módulo de Autofocalização

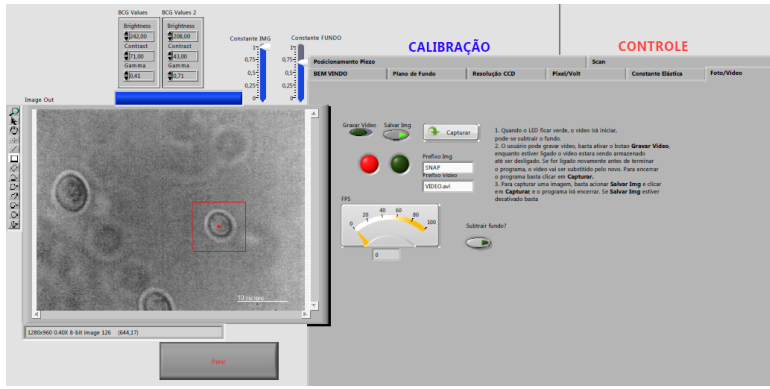


Figure 2.46: Módulo de foto e vídeo

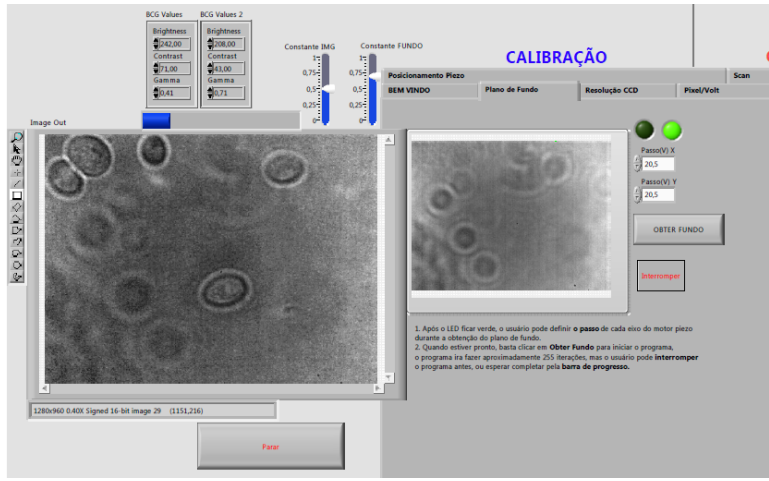


Figure 2.47: Módulo de obtenção de plano de fundo

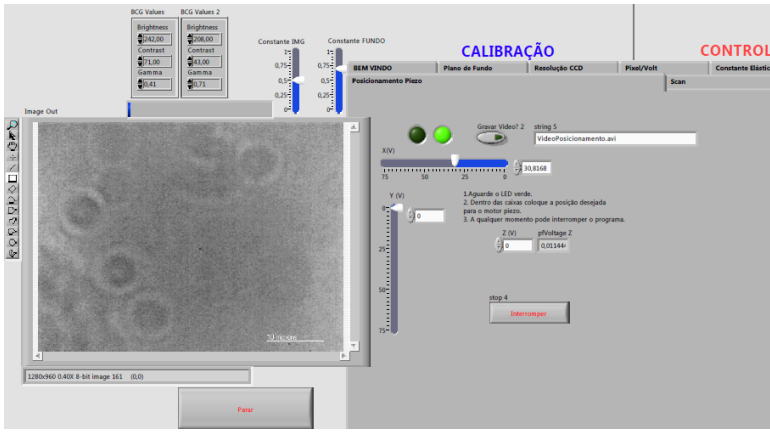


Figure 2.48: Módulo de Posicionamento do Piezo

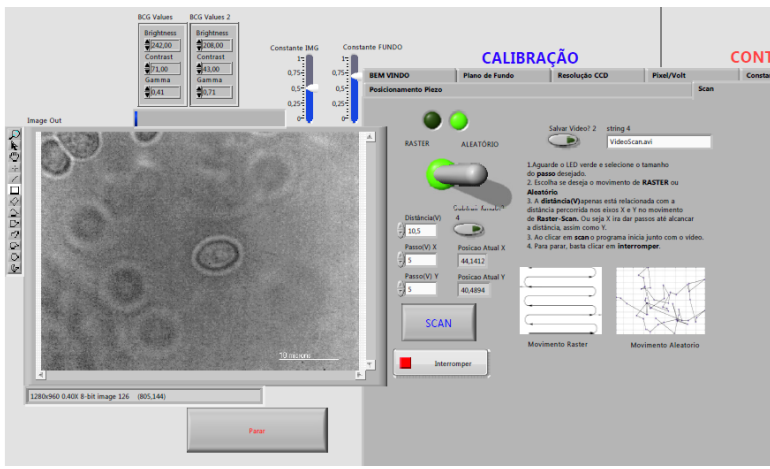


Figure 2.49: Módulo de Escaneamento

3

RESULTADOS

Neste capítulo estão os resultados obtidos até o momento do término desta Iniciação Científica, uma breve descrição do que cada módulo executa corretamente, o que eles podem oferecer ao usuário, o que não saiu como o esperado e também possíveis sugestões para melhoramentos.

3.1. MÓDULOS

Os módulos estão executando corretamente a sua função no programa final:

Posicionamento: varredura máxima por eixo de 20 micrômetros, corresponde a uma variação de tensão entre 0 e 75 volts do estágio motor piezelétrico (NanoMax-TS TPZ001, THORLABS), onde por meio deste é possível inserir valores de entrada para os eixos x, y e z, em Volts, ou mover um *slider*, onde será a posição do estágio, com resolução de aproximadamente 3 nanômetros.

Eliminação e subtração de imagem de fundo: Digitalização de aproximadamente 100 fotos (Webcam HD-C270, Logitech), após mover o estágio aleatoriamente nos eixos x e y (ou em z) enquanto obtém *frames* e os soma com o anterior. Com as imagens em tons de cinza, (16bits, 1280 x 720 *pixels*) a média é utilizada para subtrair dos vídeos subsequentes eliminando assim grande parte do ruído [7]. Depois de obtida a imagem é possível usá-la para subtrair o fundo nos outros módulos do programa.

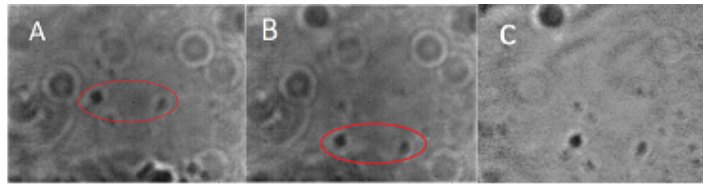


Figure 3.1: Eliminação do plano de fundo.

Em A pode-se ver dois mitoplasto indicadas, já em B ocorre um deslocamento arbitrário do estágio, e em C está a figura B sem a imagem de fundo. O método elimina boa parte dos "anéis" que poluem a imagem, esses "anéis" nada mais são do que possíveis partículas de poeira ou sujeira na objetiva ou na lâmina refratando a luz.

Varredura da pinça: Para capturar com o laser, aleatoriamente micro objetos, através de uma varredura tipo *raster-scan*, em uma determinada região de interesse através do controlador piezelétrico. O objetivo é capturar vários mitoplastos e agrupá-las para estudo posterior.

Auto-focalização: Definida uma região de interesse, o programa inicia o cálculo do algoritmo de variância normalizada [11] indicando para o usuário a posição do foco ideal. No entanto funciona melhor para ajustes finos.

Calibração espacial: Ajustando a função seno pelo método dos mínimos quadrados, de uma imagem de uma grade de difração de período conhecido podemos obter a relação pixel/mícron, por meio do software *ImageJ* foi possível extrair linha da matriz de *pixels* da imagem após deixar as grades perpendiculares com o sentido horizontal [1]. No atual projeto de IC foi possível estender esta calibração utilizando todos os pixels da imagem, ou seja, usando todas as linhas dessa matriz por meio do LabVIEW. O método usado foi o de Transformada de Fourier.

Calibração Pixel/Volt: Com uma imagem fixa ao estágio é possível movimentá-la em um sentido na direção X ou Y, enquanto a mesma é rastreada por um algoritmo, obtendo um gráfico linear do seu deslocamento pela voltagem utilizada sendo que a coeficiente angular do gráfico é a calibração em pixel/mícron.

Calibração da constante elástica: O usuário pode, tendo um objeto de interesse imerso em um fluído e em uma armadilha óptica, rastreá-lo enquanto é produzido uma onda triangular pelo motor piezo, sendo que após a obtenção dos dados é possível calcular a constante elástica pelo

igualando a força elástica com o arraste de Stokes.

Isso é possível porque a onda triangular implica um deslocamento linearmente proporcional ao tempo, portanto velocidade constante, ao contrário de outros tipos de onda, como as senoidais, permitindo a calibração da pinça óptica através do deslocamento de Stokes [1].

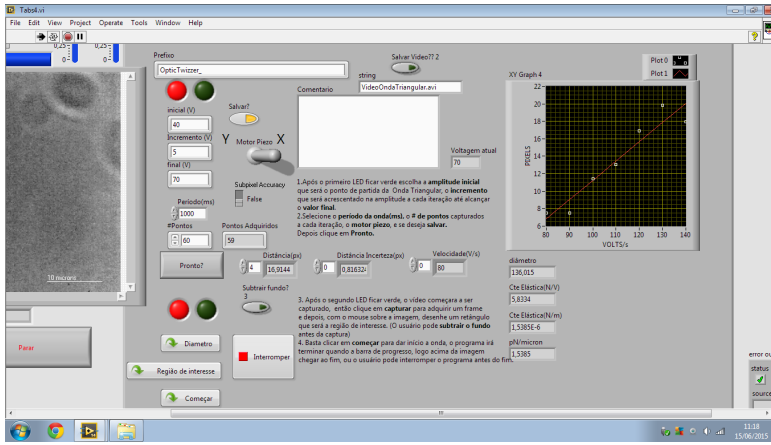
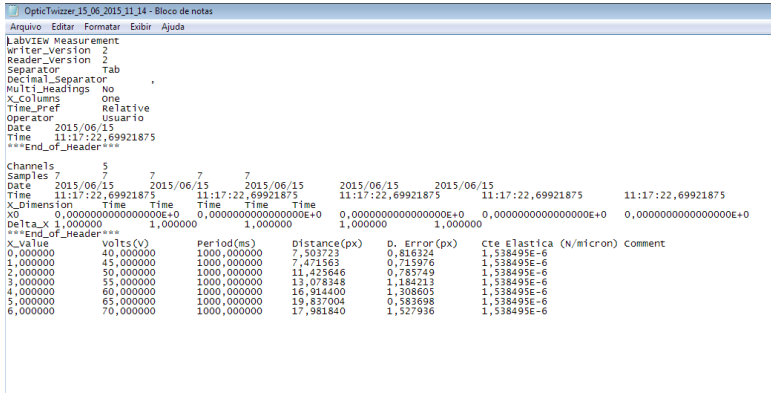


Figure 3.2: Obtenção da constante elástica e do Gráfico linear


OpticTwizzer_15_06_2015_11_14 - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

```
LabVIEW Measurement
Writer_Version  2
Reader_Version  2
Separator       Tab
Decimal_Separator       ,
Multi_Headings  No
X_Columns       One
Time_Pref       Relative
Operator        Usuario
Date    2015/06/15
Time    11:17:22,69921875
***End_of_Header***

Channels        5
Samples 7       7       7       7       7
Date    2015/06/15      2015/06/15      2015/06/15      2015/06/15      2015/06/15
Time    11:17:22,69921875       11:17:22,69921875       11:17:22,69921875       11:17:22,69921875       11:17:22,69921875
X_Dimension     Time    Time    Time    Time    Time
X0      0,00000000000000000E+0  0,00000000000000000E+0  0,00000000000000000E+0  0,00000000000000000E+0  0,00000000000000000E+0
Delta_X 1,000000        1,000000        1,000000        1,000000        1,000000
***End_of_Header***
X_Value         Volts(V)        Period(ms)      Distance(px)    D. Error(px)    Cte Elastica (N/micron) Comment
0,000000        40,000000       1000,000000     7,503723        0,816324        1,538495E-6
1,000000        45,000000       1000,000000     7,471563        0,715976        1,538495E-6
2,000000        50,000000       1000,000000     11,425646       0,785749        1,538495E-6
3,000000        55,000000       1000,000000     13,078348       1,184213        1,538495E-6
4,000000        60,000000       1000,000000     16,914400       1,308605        1,538495E-6
5,000000        65,000000       1000,000000     19,837004       0,583698        1,538495E-6
6,000000        70,000000       1000,000000     17,981840       1,527936        1,538495E-6
```

Figure 3.3: Calibração da constante elástica salva em arquivo .txt

3.2. CONCLUSÃO

Analisando o progresso até o fim do projeto, os módulos cumprem a função para o qual foram programados, no entanto ainda há melhoras a

serem feitas neles como a otimização dos *frames* por segundo nos vídeos e ajustar ou até mudar algumas características funcionais no programa final. Também foi percebido que quanto maior a complexidade de um programa, mais processamento é exigido do computador.

Com base nessas observações, foi perfeitamente possível realizar uma interface intuitiva para o usuário utilizando esses módulos apesar das dificuldades encontradas, por questão de hardware e também programáticas, mas por se tratar de uma estrutura mais complexa ainda é preciso que seja elaborado da maneira mais eficaz e o intuito é que o programa desenvolvido seja aprimorado para novas versões.

3.3. DIFICULDADES

Durante o projeto uma das dificuldades encontradas foi o defeito apresentado pela câmera da THORLABS, que estava originalmente no projeto no lugar da *webcam* que está sendo utilizada, o que nos forçou a adaptar o programa para as restrições de uma *webcam* comum. Atrelada a *webcam* e ao processamento do computador vimos que os *frames* por segundo chegam ao máximo em 30, ainda menos se o programa ficar mais complexo, caindo a 15 *fps* ou menos caso se adicione algum filtro de tratamento, isso acarreta travamento no vídeo e faz com que haja salto nos *buffers*. Além disso, a baixa resolução da *webcam* comum dificultou alguns algoritmos testados para identificar a borda de objetos circulares.

Sobre o programa da calibração da constante elástica e dificuldade de identificar objetos circulares, faltou uma função melhor elaborada para obter o raio do objeto de interesse, para substituir isso temporariamente o próprio usuário desenha a linha no tamanho do diâmetro aproximado. Ainda sobre o programa da constante elástica, existe um pequeno atraso de vídeo a cada iteração durante o movimento da onda triangular no programa de calibração da constante elástica.

A autofocalização, algumas vezes, também se mostra um pouco problemática na hora de obter a posição melhor focalizada para ajuste grosso. Alguns algoritmos de pontuação também não foram capazes de realizar a autofocalização apesar de estarem bem avaliados, como o algoritmo de Vollath e o de Brenner [11], mas o que funcionou melhor foi o de Variância Normalizada.

No programa de calibração da CCD, primeiramente foi feito sem sucesso o ajuste da periodicidade das bandas claras e escuras por meio

de uma função seno que não retornou o ajuste adequado.

Na parte de rastreamento não foi possível verificar se a função de precisão em *subpixel* da vi `Imaq Setup Match Pattern 2` realmente teve um resultado efetivo. Ainda sobre o rastreamento, é dado para os valores de posição X e Y zero, momentos antes do início da rastreamento e quando *matching* não é correspondente, para evitar que isso afete no cálculo é usado um algoritmo que elimina esses zeros, mas talvez o ideal fosse corrigir isso antes que acontecesse.

Sobre o tratamento da imagem foi experimentado usar alguns filtros de convolução de imagem para limpar a imagem de ruídos de baixa e alta frequência, porém esses filtros tem um alto custo operacional deixando inviável para o hardware utilizado.

Quando se usa a subtração do plano de fundo é feito também um fator de correção da imagem para melhorar os ruídos somando uma porcentagem da imagem do *frame* atual com o restante dessa portagem do *frame* anterior de modo que resulte 100% [7], porém isso atrapalha o *matching* da região de interesse no programa de calibração pixel/volt e da constante elástica porque como há menos *frames* de modo contínuo, o movimento do estágio faz com que a região de interesse apareça dois lugares diferentes já que se soma a imagem do *frame* anterior com a do atual. Uma solução provisória foi truncar esse fator em 1, de modo que só se utilize da imagem do *frame* atual.

O programa final possui uma falha de travamento se o usuário iniciar um módulo e não aguardar o seu término e for ativando outro módulo. Ao mesmo tempo não é possível executar o mesmo módulo duas vezes exceto se o usuário clicar em outro módulo, encerrá-lo e depois clicar no desejado. Além dessas dificuldades mais específicas, também é preciso se preocupar em conseguir fazer o programa com o melhor processamento.

3.4. PROJEÇÕES FUTURAS

Para as versões posteriores deste programa pode ser interessante visto as dificuldades melhorar alguns módulos e adicionar outros.

Um módulo para ser implementado seria o controle dos espelhos galvanométricos (GVS002, THORLABS) via USB que interligaria o gerador de funções (Tektronix, AFG 3021B) tornando possível posicionar a armadilha óptica em qualquer lugar dentro do alcance do laser.

O desenvolvimento de uma biblioteca própria para o tratamento da

imagem em *subpixel* melhorando a informação visual e o refinamento de pesquisa (O LabVIEW dispõe dessas bibliotecas) sobre o *matching* na hora de encontrar o padrão desejado na imagem.

O desenvolvimento de um programa que possa obter o raio médio de uma determinada amostra automaticamente selecionando a região de interesse ao invés do próprio usuário estimar o raio a partir de uma linha desenhada com mouse.

Um melhoramento no módulo de autofocalização para ajuste grosso.

Melhor ajuste entre a captura de vídeo e o motor piezo sobre o atraso a cada iteração da calibração da constante elástica .

Estudar a possibilidade de adicionar filtros para melhorar o tratamento das imagens.

Consertar o problema de travamento no programa final melhorando a dinâmica entre os módulos e a possibilidade de executar o mesmo módulo duas vezes seguida.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. M. C. de Arruda Farias Bittar, *Montagem e calibração de uma pinça óptica*, Iniciação Científica (Santo André, 2014).

[2] Thorlabs.com, *Controlador piezo do t-cube,* (2015).

[3] N. C. Braga, *Como funcionam os materiais piroelétricos e piezoelétricos,* (2015).

[4] K. Nice, T. V. Wilson, and G. Gurevich, *How digital cameras work,* (2014).

[5] MSPC, *Câmeras digitais,* (2014).

[6] THORLABS.COM, *User guide to labview & apt,* (2015).

[7] C. N. H. Candia, S. Tafoya Martínez, and B. Gutiérrez-Medina, *A minimal optical trapping and imaging microscopy system,* PLoS ONE **8**, e57383 (2013).

[8] Y. Sun, S. Duthaler, and B. J. Nelson, *Autofocusing in computer microscopy: Selecting the optimal focus algorithm,* Microscopy Research and Technique **65**, 139 (2004).

[9] J. S. Figueira, *Movimento browniano: uma proposta do uso das novas tecnologias no ensino de física,* Revista Brasileira de Ensino de Física **33**, 4314 (2011).

[10] J. G. Mangum, D. T. Emerson, and E. W. Greisen, *The on the fly imaging technique,* Astronomy and Astrophysics **474**, 679 (2007).

[11] Y. Sun, S. Duthaler, and B. J. Nelson, *Autofocusing in computer microscopy: Selecting the optimal focus algorithm,* Microscopy Research and Technique **65**, 139 (2004).