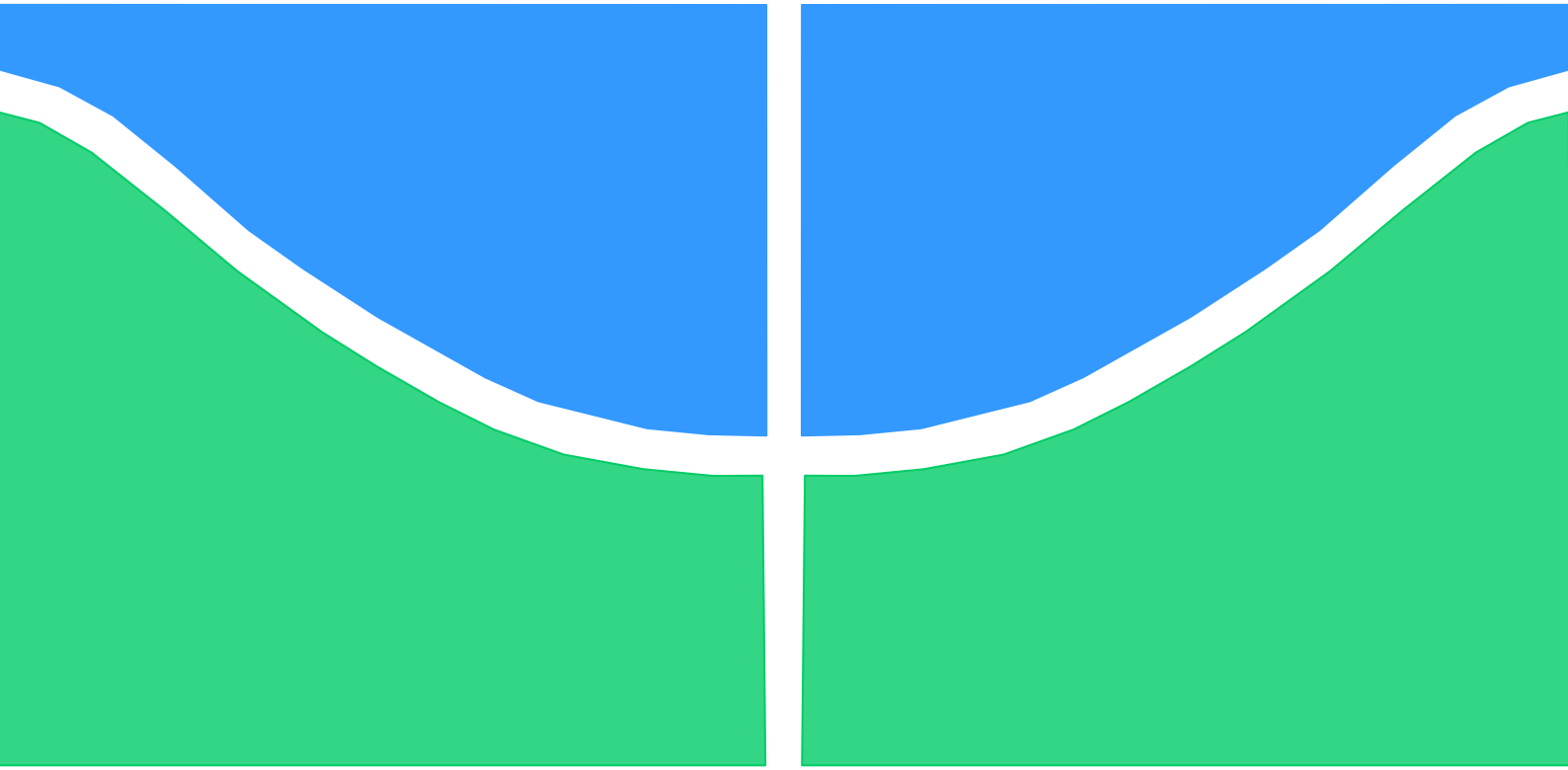

TRABALHO DE GRADUAÇÃO

INTERFACE PARA SÍNTESE MUSICAL POR RECONHECIMENTO DE GESTOS VIA WEBCAM

Daniel Brito Moussallem de Andrade

Brasília, 15 de julho de 2011

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA

Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

INTERFACE PARA SÍNTESE MUSICAL POR RECONHECIMENTO DE GESTOS VIA WEBCAM

Daniel Brito Moussallem de Andrade

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Alexandre R. S. Romariz, UnB/ ENE
(Orientador) _____

Prof. Alexandre Zaghetto, UnB/ CIC
(Co-orientador) _____

Prof. Adson Ferreira da Rocha, UnB/ FGA _____

Dedicatória

Dedico este trabalho às pessoas que veem na música uma fonte de inspiração para a vida e que acreditam que independente do estilo musical, forma de tocar e tecnologia utilizada ela sempre será um objeto de expressão cultural e de agregação social.

Daniel B. M. de Andrade

Agradecimentos

Agradeço aos meus pais e meu irmão pelo grande apoio e paciência, aos meus amigos, pela força e união e aos meus orientadores, que acreditaram neste projeto e deram todo o suporte necessário para que ele se tornasse real.

Daniel B. M. de Andrade

RESUMO

O presente trabalho aborda sobre a execução de música de uma forma diferente, onde não há instrumentos musicais físicos sendo tocados. Utilizam-se apenas as mãos vestidas de luvas, uma webcam, os *softwares* MATLAB e QuteCsound para a criação de música através do posicionamento das mãos.

Uma metodologia baseada no processamento digital de imagens é apresentada para tal.

ABSTRACT

The following work discusses about music execution in a different way, where musical instruments are not being played physically. Just hands wearing gloves, a webcam and the softwares MATLAB and QuteCsound are used to create music through the hands' position.

A methodology based on digital image processing is presented.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

A música é um dos mais importantes e reconhecidos fatores culturais de uma sociedade e através dela, a expressão do artista toma forma trazendo a tona questões de múltiplas áreas do entendimento humano. Essa busca por expressão é um fator decisivo no desenvolvimento dos diversos estilos musicais e na criação dos incontáveis instrumentos com os mais variados timbres.

O uso da tecnologia aliada à música, sempre trouxe experiências interessantes e, não obstante, muitas críticas; no entanto, a vontade de experimentar e criar são o que fazem dessa associação uma área tão inspiradora.

Ao longo da história, passou-se de instrumentos acústicos a instrumentos elétricos, eletrônicos, virtuais, sendo este último o enfoque deste trabalho, que, através de ferramentas computacionais para análise de imagem e *hardware* de vídeo, faz possível a criação de uma interface para a síntese musical através da leitura de movimentos das mãos vistas por uma webcam.

Essa nova experiência musical trás a sensação de tocar um theremin virtual.

2 DEFINIÇÕES MUSICAIS

Este capítulo apresenta conceitos e definições que abrangem a organização musical e seus valores.

2.1 O SOM

O som, como matéria prima da música, é o fenômeno acústico que consiste na propagação de ondas mecânicas produzidas por um corpo que vibra em material elástico, especialmente o ar.

Essas ondas são captadas pelo ouvido e interpretadas pelo cérebro, gerando uma sensação auditiva no ouvinte.

Sabendo que ele é periódico no espaço temporal, pode ser representado, em sua forma mais pura, como uma senóide ou como uma soma de ondas.

Suas principais características são:

- Altura: distingue os sons graves e agudos. Sua medida é dada em oscilações por segundo, ou Hertz. Nas escalas musicais mais comuns na música ocidental, a oitava (intervalo correspondente a uma relação 2:1 em frequência) é dividida em 12 semitons, sendo 7 notas principais e 5 acidentes. No entanto, diversos instrumentos musicais como os de sopro, os de corda, conseguem tocar variações contínuas entre uma nota e outra.
- Duração: distingue sons longos e curtos. É medida em tempo (segundos). Na música, a duração é definida pelas figuras de valor.
- Volume / Intensidade: distingue sons fracos e fortes. É medida em decibéis e é definida pelos sinais de dinâmica.
- Timbre: é o conjunto de características que identifica o som produzido por cada fonte sonora. Definido, entre outras coisas, por meio da série harmônica de cada instrumento ou voz e do transiente de ataque.

2.2 A MÚSICA

Música é a arte que usa o som como matéria-prima, de forma a combinar ruído e silêncio, seguindo ou não padrões pré-definidos, com o objetivo de manifestar alguma forma de expressão.

Manifestações sonoras são encontradas em todos os povos, e sendo assim, o ser humano se torna uma sociedade universalmente musical.

Desde a pré-história, o homem busca reproduzir os sons da natureza, em especial dos animais, tanto para comunicação como para rituais, buscando sempre por lógicas e padrões nos ritmos, harmonias e melodias.

O provável primeiro instrumento musical encontrado data de 43000 a.C. e se assemelha a uma flauta feita de osso (fêmur de uma espécie extinta de urso) com quatro orifícios. Já o primeiro instrumento de corda encontrado, é conhecido como Ravanastron, com duas cordas e tocado com arco, e possui 7000 anos de idade.

A criação de novos instrumentos musicais sempre visou à busca por novos timbres e novas formas de reprodução sonora, utilizando-se sempre de novas tecnologias para a criação e reprodução de sons.

A organização musical segue três elementos básicos:

- Harmonia: é um conjunto de sons simultâneos constituídos de acordes e intervalos harmônicos.
- Melodia: é uma sucessão de sons isolados e combinados em diferentes alturas e valores e obedecem um sentido lógico musical ligado à harmonia e ao ritmo. Muitos consideram a melodia como sendo a música por excelência.
- Ritmo: é o andamento da música e seu tempo, constituída através da duração de cada nota da música e define a ordem a qual os sons em diversos valores obedecem dentro do discurso musical.

2.3 O THEREMIN

O instrumento que serviu de base para o desenvolvimento deste projeto, o theremin, foi patenteado em 1921 por Leon Theremin e emite sons sem ser fisicamente tocado, já que seu funcionamento se assenta em duas antenas sensíveis à proximidade que seguem o princípio da heterodinação, o qual consiste em converter altas frequências em baixas frequências para produzir uma nova frequência como conseqüência da mescla de outras duas.

O utilizador, ao aproximar e afastar suas mãos das antenas, age variando os campos eletromagnéticos gerados por elas. Enquanto a proximidade da mão direita junto à antena vertical controla a freqüência de forma contínua, a mão esquerda controla o volume pelo mesmo princípio.

Figura 2.1: Leon Theremin tocando o instrumento criado por ele.

3 PROCESSAMENTO DIGITAL DE IMAGENS

Este capítulo apresenta os principais conceitos e definições do processamento de imagens.

3.1 HISTÓRICO

Uma das primeiras aplicações de imagem digital veio da indústria de jornais, quando algumas figuras foram mandadas via cabo submarino entre Londres e Nova Iorque, reduzindo o tempo de envio de imagens de mais de uma semana para menos de três horas.

Equipamentos de impressão especializados em codificar e decodificar figuras para transmissão em cabo eram utilizados e conseguiam impressões com até 15 níveis de tons de cinza no ano de 1929 (Fig. 3.1).

Embora essas imagens fossem digitais, elas não são consideradas como resultado de processamento digital de imagem devido à falta de um computador digital como intermediário.

Os primeiros computadores poderosos o bastante para esse trabalho surgiram no começo da década de 60, junto com o programa espacial norte-americano e eram utilizados para melhorar imagens da lua enviadas pela sonda espacial Ranger 7 (Fig. 3.2).

Figura 3.1: Imagem sem retoques dos Generais Pershing e Foch, transmitida em 1929 de Londres para Nova Iorque com 15 tons. (McFarlane.)

Figura 3.2: Primeira foto da lua pela sonda espacial Ranger 7 tirada em 31 de julho de 1964, cerca de 17 minutos antes do pouso.

No inicio da década de 70, o processamento digital de imagens começou a ser utilizado na medicina, com a chegada da tomografia computadorizada; na astronomia, com o programa espacial e na geologia, na busca de recursos subterrâneos.

Hoje, essa técnica é fortemente empregada em diversas áreas que dependem da interpretação de imagens.

3.2 FUNDAMENTOS

3.2.1 IMAGEM

O processamento digital de imagens é um processo que utiliza computadores digitais onde ambos, entradas (*inputs*) e saídas (*outputs*), são imagens de onde podem-se extrair atributos (valores).

Cada imagem digital é composta por um número finito de elementos, onde cada um deles tem uma localização e um valor. Esse elemento unitário é mais comumente chamado de pixel.

Uma imagem, bidimensional, é definida por *f(x,y)*. A intensidade da função f, para qualquer par de coordenadas *(x,y)*, é chamada de intensidade ou nível de cinza (*intensity / gray level*). Se todos os valores de *f*, *x*, e *y* forem finitos e com valores discretos, a imagem é dita digital.

No entanto a função *f(x,y)*, Eq.(1), é uma combinação do produto de duas outras funções componentes. A Eq.(2), chamada de Iluminação, *i(x,y)*, denota a quantidade de iluminação incidente de uma fonte de luz própria e a Eq.(3), chamada de Reflectância, *r(x,y)*, representa a quantidade de luz refletida pelo objeto em questão, onde 0 representa absorção total e 1 representa reflexão total.

Têm-se, que:

$$f(x,y) = i(x,y).r(x,y) \quad (1)$$

onde,

$$0 < i(x,y) < \infty \quad (2)$$

e,

$$0 < r(x,y) < 1 \quad (3)$$

3.2.2 VÍDEO

A observação natural de uma cena é um processo contínuo tanto no tempo como no espaço. Já a sua representação digital envolve amostragem no tempo (série de *frames* ou componentes de *frames* amostrados em um intervalo regular de tempo) e no espaço (geralmente uma grade retangular no plano da imagem), como pode ser visto na Fig. (3.3).

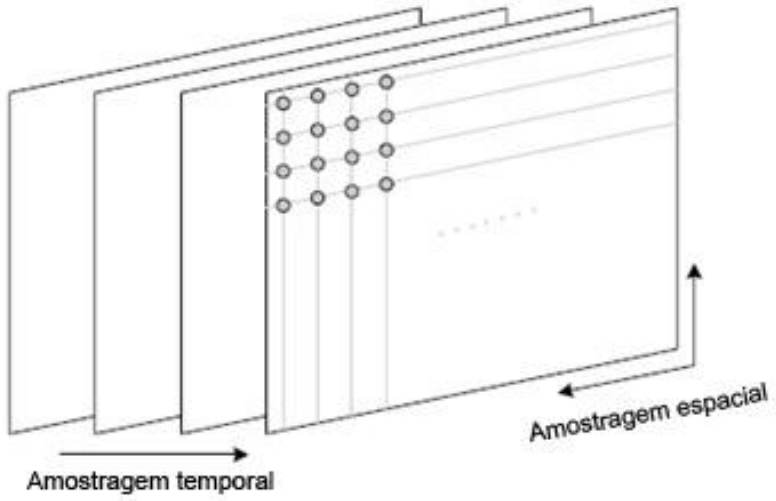

Figura 3.3: Amostragem espacial e temporal de uma sequência de vídeo.

O vídeo digital é a representação da amostragem de uma cena na forma digital, onde cada amostra espaço-temporal é descrita por pixels que carregam informações de brilho (luminância) e cor.

3.3 ESPAÇO DE CORES

Uma imagem monocromática requer somente um número que indique a informação de brilho de cada amostra espacial. No entanto, a maioria das aplicações com vídeos digitais envolvem o uso de vídeo em cores e, sendo assim, é necessário um mecanismo que represente essas informações.

A informação de cor é representada por, pelo menos, três valores por pixel, em diferentes códigos de cores. Tais informações necessitam de, pelo menos, três números por pixel para obter uma representação acurada.

O método escolhido para representar brilho e cor de um pixel é descrito como espaço de cores.

Dois métodos são mais comumente usados:

3.3.1 RGB

Neste espaço de cores, a imagem amostrada é representada por três números que indicam a proporção relativa de Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*), visto que qualquer cor pode ser representada por uma combinação destas.

A componente vermelha consiste de todas as amostras vermelhas, assim como a componente verde consiste de todas as amostras verdes e a componente azul, todas as amostras azuis.

A Figura (3.4) representa uma imagem colorida dividida em suas três componentes RGB.

Figura 3.4: Componentes Vermelho, Verde e Azul, respectivamente, de uma imagem colorida.

A pessoa sentada à direita está usando uma roupa azul, e sendo assim, ela aparece mais brilhosa na imagem da componente azul, enquanto a pessoa à esquerda veste uma roupa vermelha, que aparece mais brilhosa na imagem da componente vermelha.

3.3.2 YCbCr

O sistema visual humano é mais sensível ao brilho e menos sensível a cores.

No espaço de cores RGB, as componentes possuem a mesma importância e por isso são armazenadas geralmente com a mesma resolução. Já no espaço de cores YCbCr, a representação de cores é feita de forma mais eficiente separando o brilho das informações de cores, representando aquele com uma resolução maior que este.

A componente Y representa a luminância, e pode ser calculada como uma média ponderada de R, G e B da seguinte forma:

$$Y = k_r R + k_g G + k_b B \tag{4}$$

onde *k* representa o peso utilizado.

As informações de cores podem ser representadas por componentes de crominância, que são calculadas como a diferença entre R, G ou B e Y:

$$Cb = B - Y \tag{5}$$

$$Cr = R - Y \tag{6}$$

$$Cg = G - Y \tag{7}$$

A descrição completa de uma imagem colorida é dada por Y e três diferenças de cores, Cb, Cr e Cg, que representam a diferença entre a intensidade de cor e a luminância de cada imagem amostrada.

Na Figura (3.5), estão representadas as componentes de crominância Cr, Cg e Cb, correspondentes às componentes R, G e B da Fig. (3.4).

Figura 3.5: Componentes Cr, Cg e Cb, respectivamente, de uma imagem colorida.

O cinza médio (*mid-grey*) representa diferença zero. O cinza claro (*light-grey*) representa diferença positiva e o cinza escuro (*dark-grey*), diferença negativa.

Percebe-se na Fig. (3.5) a grande diferença nas componentes Vermelho e Azul.

Como Cb + Cr + Cg é uma constante, apenas duas das três componentes precisam ser transmitidas, visto que a terceira componente sempre poderá ser calculada a partir das outras duas. No caso do espaço de cores YCbCr, apenas as componentes Y, Cb e Cr são transmitidas, sendo que Cb e Cr podem ser transmitidas com uma menor resolução em relação à Y. Esta é a principal vantagem em relação ao espaço de cores RGB, já que menos dados serão necessários para a transmissão e para o observador casual, não haverá diferença óbvia entre esses dois métodos.

Antes da exibição da imagem, geralmente é necessária a transformação para RGB novamente. Para isso, podem-se usar as Equações (8), (9) e (10).

$$R = Y - \frac{1 - k_r}{0.5}\, Cr \tag{8}$$

$$G = Y - \frac{2k_b\,(1 - k_b)}{1 - k_b - k_r}\, Cb - \frac{2k_r\,(1 - k_r)}{1 - k_b - k_r}\, Cr \tag{9}$$

$$B = Y + \frac{1 - k_b}{0.5}\, Cb \tag{10}$$

A ITU-R (*International Telecommunication Union*) define, através da BT.601-5 (recomendação dos parâmetros de codificação de estúdio para televisão digital nos formatos 4:3 e 16:9), os valores de $k_r = 0.114$ e $k_r = 0.299$

3.4 SEGMENTAÇÃO DE CORES

A segmentação é um processo de particionar uma imagem em regiões. No caso da segmentação de cores, o objetivo é classificar cada pixel de uma dada imagem como tendo uma cor dentro de uma determinada faixa (*range*) de cor ou não, gerando como saída uma imagem binária, preta-e-branca deste resultado.

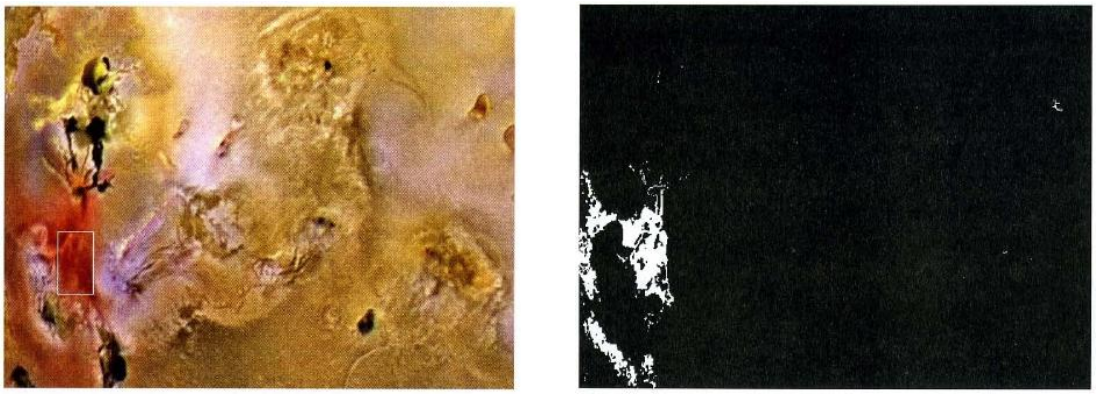

Figura 3.6: Segmentação no espaço de cores RGB: Imagem original à esquerda e imagem segmentada com destaque da cor vermelha.

4 MÉTODO PROPOSTO

Neste capítulo, propõe-se a utilização dos softwares MATLAB e QuteCsound para análise e processamento do vídeo e reprodução do som.

4.1 PROPOSTA

A proposta deste trabalho sugere a criação de música através do movimento das mãos, objetivando alcançar uma experiência semelhante a tocar um theremin. Para isso, utilizaram-se um par de luvas vermelhas, uma *webcam*, o programa matemático MATLAB e o *software* musical QuteCsound e algoritmos que foram criados para esta finalidade.

Iniciando o programa, pede-se que o utilizador vista as luvas e posicione uma das mãos de forma a captar e calibrar as cores que serão utilizadas para a segmentação da imagem, no intuito de identificar somente a cor da luva do utilizador.

Após a calibração, uma nova imagem é mostrada, já segmentada para o tom das luvas e dividida em dois quadrantes, de um para cada mão.

A divisão em quadrantes servirá para calcular a distância entre as duas mãos e esta distância será concatenada a uma frequência, que deverá soar, variando de acordo com a distância entre as mãos. Quanto maior essa distância for, maior a frequência e vice-versa.

4.2 SOLUÇÃO

4.2.1 IDENTIFICAÇÃO DO VÍDEO

Primeiramente, é necessário conhecer o tipo de *hardware* que está sendo utilizado e entender seu funcionamento e suas limitações.

Através do MATLAB, é possível utilizar a função imaqhwinfo para obter as informações dos adaptadores de aquisição de imagem que estão disponíveis no sistema. O adaptador utilizado é a interface entre o MATLAB e os dispositivos de aquisição de imagem. No caso do presente estudo, foi utilizado o adaptador winvideo, disponível como padrão no Windows.

O dispositivo utilizado (Sony Visual Communication Camera) trabalha com o formato YUY2_160x120 como padrão, ou seja, utiliza o espaço de cores YCbCr com uma janela de 160 por 120 pixels, porém, suporta outros 3 formatos: 176x144, 320x240 e 640x480, todos em YCbCr.

Após vários testes de desempenho com o código completo e finalizado, optou-se pelo uso da resolução de 320 por 240 pixels, que apresenta um bom tamanho de imagem e não gera atraso suficiente para prejudicar a execução de música.

4.2.2 DEFINIÇÃO DOS PARÂMETROS DE REPRODUÇÃO DO VÍDEO

Obtidos o dispositivo e o adaptador que serão utilizados, definem-se em seguida os parâmetros de captura do vídeo.

A propriedade FramesPerTrigger indica a quantidade de *frames* que será obtida para formar o *vídeo stream.* Optou-se por utilizá-la definindo seu valor como FramesPerTrigger = inf, fazendo com que o encerramento das atividades não esteja vinculado ao número de *frames*.

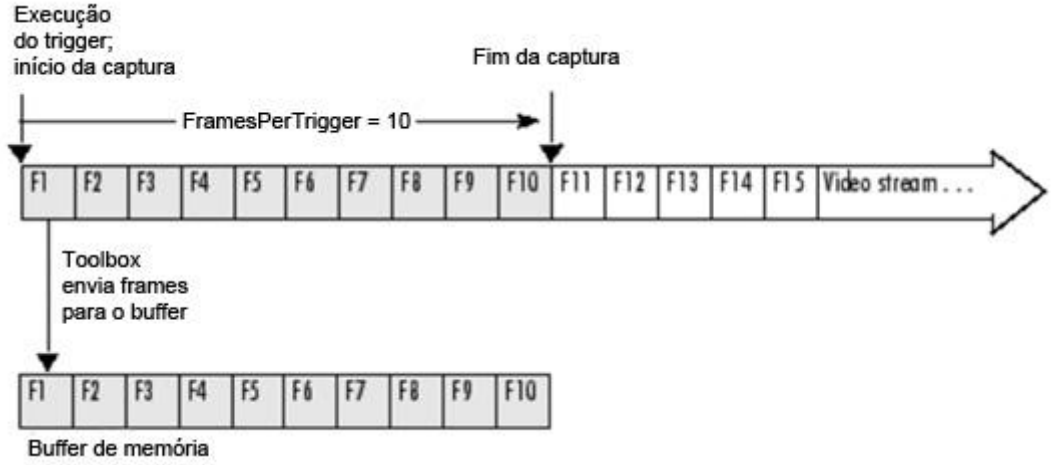

Figura 4.1: Exemplo do funcionamento da propriedade FramesPerTrigger = 10.

A propriedade FrameGrabInterval indica quantos *frames* são adquiridos do *video stream.* Optou-se por adquirir todos os *frames* do *video stream* usando FrameGrabInterval = 1.

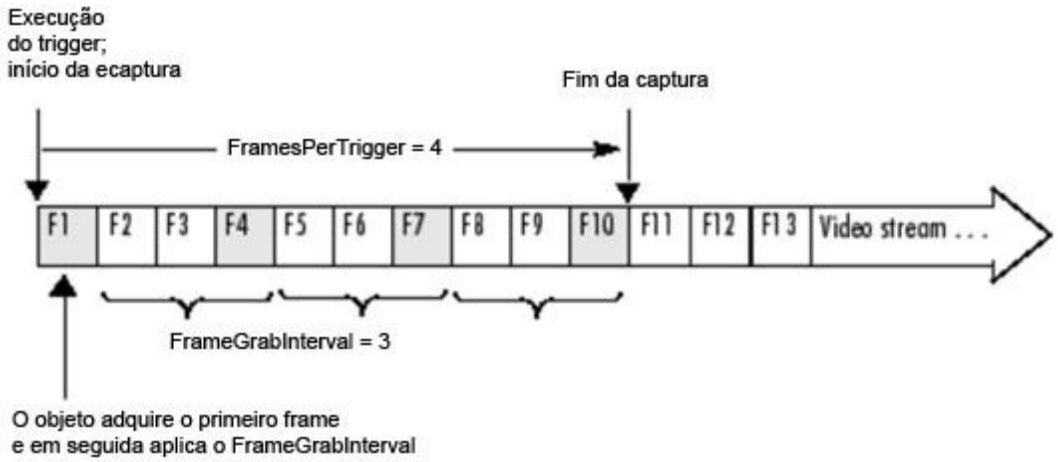

Figura 4.2: Exemplo do funcionamento da propriedade FrameGrabInterval = 3.

4.2.3 CALIBRAÇÃO DE COR

Para a detecção do movimento das mãos, foi utilizado um par de luvas vermelhas para um melhor destaque e diferenciação com os planos de fundo.

A calibração da cor selecionada se dá através da função calibraluva_ycbcr, vide código em anexo, onde um espaço de 60 *frames* será utilizado. Os 40 primeiros *frames* se destinam a buscar a correta posição da calibração e os próximos 20 *frames* são utilizados para o ato da calibração.

Nesta fase, já com a janela de vídeo aberta e funcionando, um pequeno quadrado negro com 10 pixels de lado aparecerá no centro da imagem e uma contagem regressiva se inicia para que a cor utilizada seja posicionada de forma a cobrir por completo o quadrado. Acabando a contagem regressiva de posicionamento, uma contagem progressiva se inicia, captando somente a imagem posicionada dentro do quadrado (Fig. 4.3).

Uma média das componentes Y, Cb e Cr dos 20 últimos *frames* é feita e armazenada nos coeficientes coefY, coefCB e coefCR.

Com o intuito de realizar um ajuste na calibração caso o uso dos valores médios fique insatisfatório para a detecção e apresentação de todos os pontos relativos à luva, foi criada uma linha de código que permite expandir os valores da média armazenada em coefY, coefCB e coefCR em X por cento, onde X é um valor arbitrário definido pelo usuário.

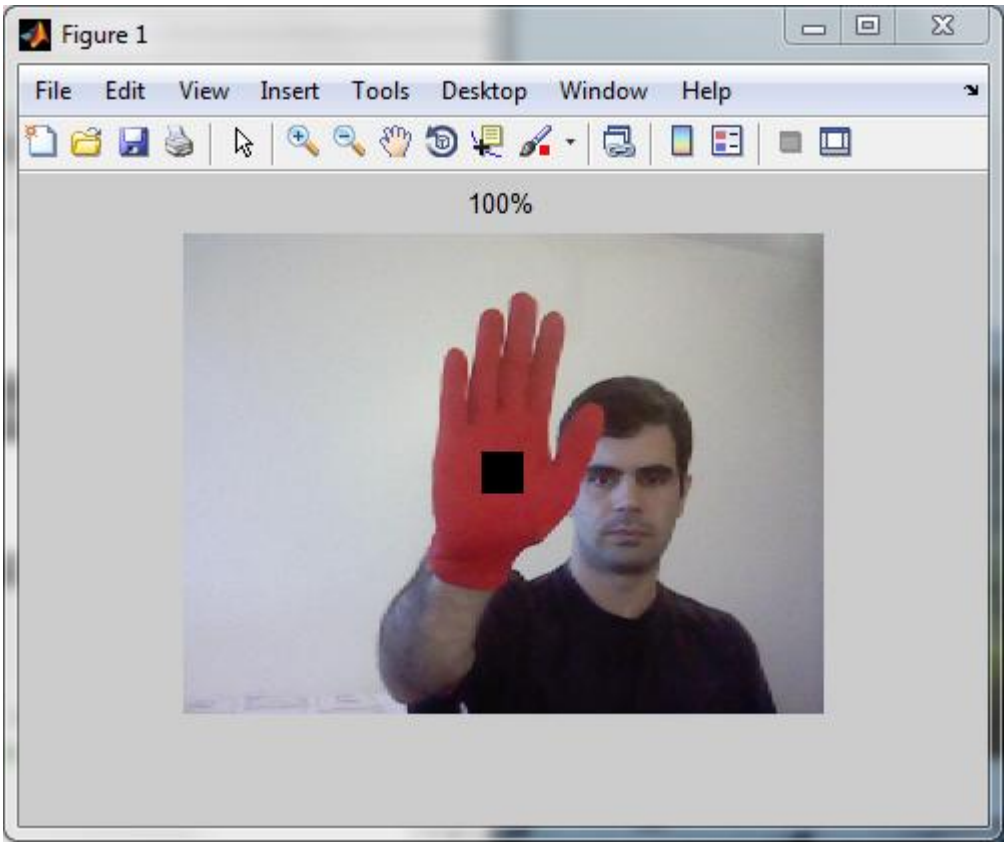

Figura 4.3: Tela de calibração.

4.2.4 SEGMENTAÇÃO DA IMAGEM

A segmentação se dá após o intervalo das médias de Cb e Cr serem escolhidas e com posse desses valores, igualam-se todos os pixels encontrados nesse intervalo ao valor 1 (branco). Os pixels remanescentes são igualados ao valor 0 (preto).

Essa técnica irá separar a luva do restante do plano, oferecendo uma imagem em preto-e-branco que destacará somente a luva (Fig. 4.4).

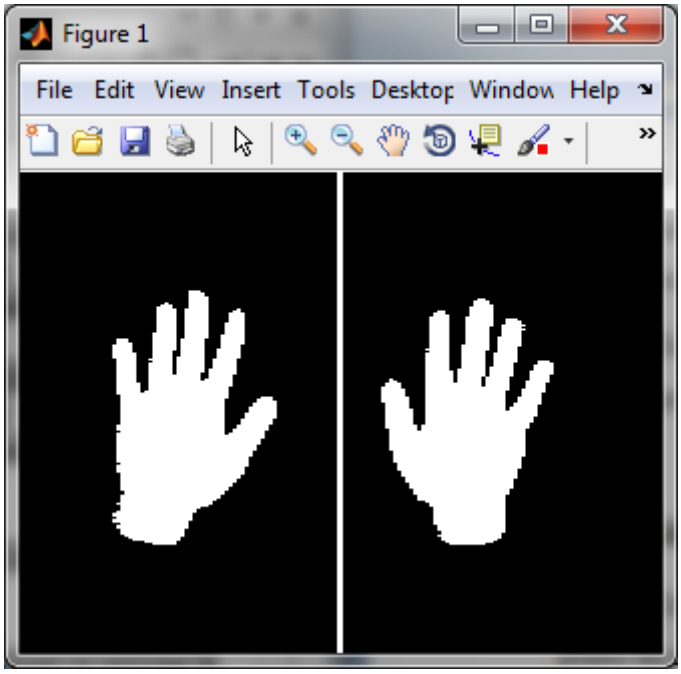

Figura 4.4: Tela segmentada destacando as luvas.

4.2.5 CÁLCULO DA DISTÂNCIA ENTRE AS MÃOS

Uma linha branca foi criada para dividir a imagem entre o semi-plano da direita e o semi-plano da esquerda. Essa divisão visa o cálculo da distância entre as mãos, que deverão estar cada uma em um dos quadrantes.

Com as mãos devidamente posicionadas e com a cor utilizada segmentada em branco, faz-se uma média da posição dos pixels segmentados em cada quadrante e é calculada a distância somente em relação ao eixo horizontal, utilizando esses valores, através da Eq. (4.1).

$$dist = \sqrt{(med2(2) - med1(2))^2} \tag{4.1}$$

Onde med1(2) é a média calculada no semi-plano esquerdo e med2(2) é a média calculada no semi-plano direito (Fig. 4.5).

Figura 4.5: Cálculo da distância entre as luvas através de seus pontos médios.

4.2.6 REPRODUÇÃO DO SOM

Inicialmente, utilizou-se a função *sound*, que é própria do programa MATLAB, no entanto, os resultados não foram satisfatórios pois esta função consumia muito tempo de processamento além de ter que ser reproduzida para cada *frame* analisado. Esta condição atrapalhou bastante a experiência sonora por fazer com que o som não fosse reproduzido de forma contínua e gerando um *lag* muito grande entre o movimento das mãos e a resposta sonora, trazendo grande dificuldade para o controle das notas tocadas.

A solução encontrada se deu com a utilização do programa QuteCsound, que permite a emissão sonora criada por meio de códigos programáveis. Através dele, criou-se uma barra de rolagem (*slide*) com tamanho e frequência programável para o controle do som gerado por uma senóide pura.

As frequências escolhidas para a barra de rolagem variam de 110 Hz a 880 Hz, que são frequências que não desagradam o ouvido humano, oferecem uma variação de três oitavas e facilitam o controle das notas, já que o tamanho do slide foi definido como o dobro da distância máxima entre as luvas (Fig. 4.6).

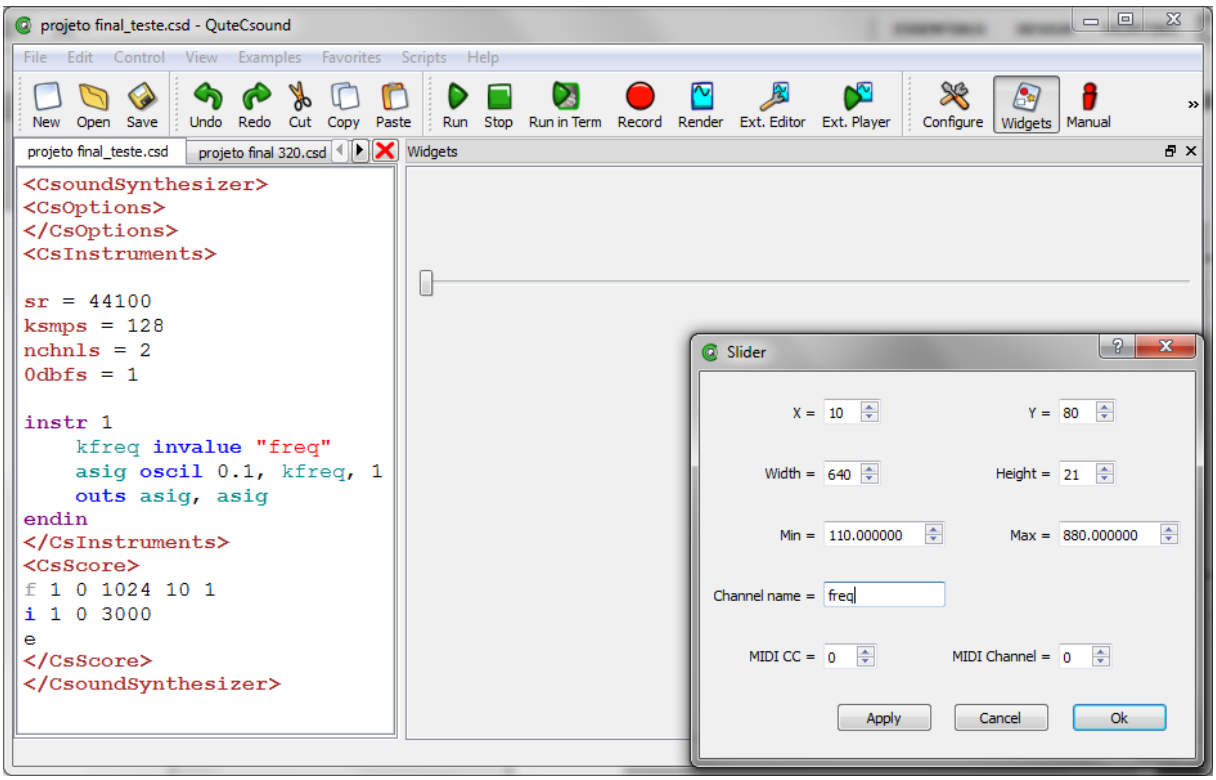

Figura 4.6: Interface do programa QuteCsound, utilizado para o controle e reprodução do som.

4.2.7 CONTROLE DO SOM

O controle da posição e dos botões do *mouse* feitos pelo MATLAB se dá através das classes java.awt.Robot e java.awt.event.*, que devem ser importadas para o MATLAB, visto que ele não possui essas habilidades disponíveis.

De posse desse controle, o programa é capaz de dar início e fim ao reprodutor sonoro do programa QuteCsound através do *click* do *mouse* nos ícones *Run* e *Stop* respectivamente e variar a posição da barra de rolamento, que é responsável pela variação da frequência, mantendo o *click* pressionado nela durante todo o processo de execução musical.

4.2.8 FINALIZAÇÃO DO PROCESSO

Com o intuito de finalizar o programa no momento escolhido pelo utilizador, criou-se um identificador de imagem que verifica se as luvas ainda se encontram na tela.

Como os pixels pretos possuem valor zero e os pixels brancos da luva possuem valor um, faz-se uma soma dos pixels brancos na tela. Caso o valor seja maior do que zero, o programa continua sendo executado. Caso contrário, o programa verifica por 20 *frames* seguidos se o valor permanece com soma zero. Tendo a confirmação da tela escura por esse período, o programa é encerrado.

5 RESULTADOS

Neste capítulo, discute-se a realização do projeto e as diversas modificações que foram introduzidas a fim de deixá-lo coerente com a proposta introduzida.

No início do desenvolvimento do trabalho, percebeu-se um gargalo na velocidade de apresentação do vídeo, que era mostrado sempre com vários segundos de atraso visto que, entre outros, a transformação do espaço de cores YCbCr, utilizado pela câmera de vídeo, para RGB consumia muito tempo de processamento. Ainda no espaço RGB, testes foram feitos variando o parâmetro FrameGrabInterval para valores entre 1 e 4. O valor 1 apresentava a resposta mais lenta, com muito atraso e inviabilizava a execução musical. À medida que esse valor ia aumentando, ganhava-se velocidade na apresentação do vídeo, porém perdia-se fluidez, visto que alguns *frames* eram cortados da análise. Optou-se neste caso por desenvolver todos os cálculos baseados no espaço YCbCr. Esta mudança trouxe tanto velocidade quanto fluidez e permitiu ao utilizador um controle mais fiel das notas tocadas.

Na fase de calibração, pensou-se primeiramente na utilização de histogramas, em RGB, para analisar a melhor faixa de cores a ser considerada para o processo de segmentação. Mesmo apresentando bons resultados, essa opção foi descartada, pois a identificação dessas faixas era feita de modo manual (observando os histogramas e definindo os valores dentro do código) e variavam para cada condição de luz e ambiente. O uso da média de valores em YCbCr trouxe resultados satisfatórios, rápidos e automáticos e o uso da linha de porcentagem cria a possibilidade de melhorar o ajuste feito, caso seja necessária a expansão dos valores calculados para a inclusão de mais pixels na luva.

A reprodução de som foi a fase mais crítica do projeto. A função sound do próprio MATLAB não mostrou ser uma boa opção para este caso. Além de ser o processo que mais exigia memória e processamento, não ofereceu a possibilidade de reprodução contínua de sons, resultando que a cada *frame* analisado, o processo iniciava, tocava o som por um tempo previamente especificado no código e encerrava, gerando um resultado final que mesclava sons e pausas, com a reprodução de vários *beeps*. A utilização do QuteCsound resolveu os problemas de reprodução, liberando o MATLAB desta função e contribuiu muito positivamente para melhorar o controle do som, visto que as barras de rolagem são programáveis tanto em seu tamanho como na definição das frequências.

Para fazer a integração entre os dois programas, o controle do *mouse* foi implementado no código do MATLAB de forma a controlar o início e o final da reprodução sonora e a barra de rolagem do QuteCsound durante toda a execução do programa. Esta técnica se mostrou a mais eficiente para o caso em questão, tendo em consideração que não é possível conectar os programas através de códigos e funções. Em contrapartida, o uso do *mouse* limita o controle a apenas um parâmetro, que neste caso foi a frequência.

A fim de testar o potencial do instrumento virtual criado neste trabalho, ligou-se a saída de áudio do computador a uma pedaleira de efeitos para guitarra e esta, a um amplificador. Foi possível a agregação de vários efeitos sobre o som, como o uso de *overdrive* (distorção do som através do aumento de ganho), *reverb* (reverberação) e *delay* (eco, ou repetição do som tocado com um atraso controlado), resultando em uma gama de novos timbres interessantes, aproximando o resultado final deste trabalho ao uso real do theremin que pode ser visto em diversas músicas que o utilizam com esses efeitos.

Após diversos testes e modificações nos códigos do programa e com a adição do QuteCsound, conseguiu-se, por fim, um excelente resultado, com uma execução limpa, fluida e de rápida resposta.

6 CONCLUSÕES

Através de técnicas de processamento de imagens, conseguiu-se criar uma interface capaz de produzir música através do reconhecimento espacial das mãos vistas por uma webcam.

O MATLAB se mostrou uma boa interface para criar e testar o programa, no entanto suas limitações para processar os sons fizeram com que fosse necessário o uso do QuteCsound especificamente para esta função. Esses dois programas trabalhando em conjunto foram capazes de suprir todas as necessidades para a implementação da interface de síntese musical, de forma que fosse possível controlar cada uma das etapas especificadas no trabalho da forma mais conveniente possível para se alcançar o resultado final com êxito.

Observa-se, mais uma vez, a grande integração entre música e tecnologia e abre-se espaço para novas pesquisas que possam complementar este trabalho ou surgir a partir dele.

6.1 PROJETOS FUTUROS

Para uma melhor aproximação deste projeto com o instrumento theremin, pode-se pensar em usar cada semi-plano da janela de segmentação para exercer uma função diferente, onde um deles seria para o cálculo da frequência e o outro poderia controlar o volume.

Pode-se pensar também na utilização do acessório Kinect, do vídeo-game Xbox, que utiliza uma câmera RGB e outra infravermelho, como dispositivo de aquisição de imagem. A Microsoft já disponibiliza *drivers* oficiais e pacotes para desenvolvedores visando a ligação deste acessório ao computador e já se fala em lançar uma versão própria para ele, enquanto a Asus já anunciou o Wavi Xtion para 2011, que utiliza a mesma tecnologia e é próprio para computadores.

A reprogramação em outras linguagens deve melhorar consideravelmente a velocidade de processamento das imagens e assim permitir que janelas maiores, com mais resolução, possam ser usadas para a visualização do vídeo.

O QuteCsound abre a possibilidade para a criação, através de sua programação, de diversos timbres que podem atribuir novas qualidades sonoras ao projeto, enriquecendo a experiência de se tocar um instrumento virtual.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, Maria B.M.; ARRUDA, Susana M. **Como fazer referências:** bibliográficas, eletrônicas e demais formas de documentos. Universidade Federal de Santa Catarina. Disponível em: < http://bu.ufsc.br/framerefer.html >. Acesso em: 04 julho 2011.

CAESAR, Wesley Lely. **Guitarra**: Noções elementares. [S. I.: Irmãos Vitale], 2003.

DUFFIN, Ross W. **How equal temperament ruined harmony**: (And why you should care). United States of America: W. W. Norton & Company, 2007

ESTRADA, Víctor. **Guía de introducción al theremin**. Disponível em: <http://web.mac.com/estudioserin/theremines/Presentación.html>. Acesso em: 04 julho 2011.

FREDERICO, Edson. **Música**: Breve história. [S. I.: Irmãos Vitale], 1999.

GONZALEZ Rafael C.; WOODS, Richard E. **Digital Image Processing**. New Jersey: Prentice Hall, 2002.

MATHWORKS. **Controlling logging parameters**. Disponível em: <http://www.mathworks.com/help/toolbox/imaq/f15-69957.html>. Acesso em: 04 junho 2011.

MATHWORKS. **FrameGrabInterval**. Disponível em: <http://www.mathworks.com/help/toolbox/imaq/framegrabinterval.html>. Acesso em: 04 junho 2011.

MATHWORKS. **How can I programmatically control mouse motion and clicks with MATLAB?** Disponível em: <http://www.mathworks.com/support/solutions/en/data/1-2X10AT/index.html?solution=1-2X10AT>. Acesso em: 04 junho 2011.

QUTECSOUND. Disponível em: < http://qutecsound.sourceforge.net/>. Acesso em: 04 julho 2011.

RICHARDSON, Iain E. G. **H264 and MPEG-4 video compression**: Video coding for next-generation multimedia. England: Wiley, 2003.

THE MUSICAL MUSEUM. Disponível em: <http://www.musicalmuseum.co.uk/theremin.html>. Acesso em: 04 julho 2011.

WISNIK, José Miguel. **O som e o sentido**: Uma outra história das músicas. São Paulo: Companhia das Letras, 2007.

ANEXO

CÓDIGO UTILIZADO NO MATLAB

```
clear all;
close all;
clc;

w = 320;                                    %Tamanho da janela:
h = 240;                                    %Pixels horizontais e verticais

tamanho = ['YUY2_' num2str(w) 'x' num2str(h)];

[coefY, coefCB, coefCR] = calibraluva_ycbcr(w, h);  % Função de calibração


video = videoinput('winvideo',1,tamanho);           %Entrada de vídeo
set(video,'FrameGrabInterval',1);  %Adquire todos os frames do video stream
set(video,'FramesPerTrigger',inf); %Video stream composto por infinitos
frames

start(video);

close all

import java.awt.Robot;                          %Controles do mouse
import java.awt.event.*;
mouse = Robot;
%screenSize = get(0, 'screensize');

mouse.mouseMove (1310, 70);                     %Posição do botão Run
mouse.mousePress(InputEvent.BUTTON1_MASK);      %Botão esquerdo do mouse
mouse.mouseRelease(InputEvent.BUTTON1_MASK);    %Botão esquerdo do mouse

mouse.mouseMove(1503, 295);                     %Posição inicial do slide
mouse.mousePress(InputEvent.BUTTON1_MASK);      %Botão esquerdo do mouse


cont = 1;
while(cont<20)

    data = getdata(video,1);

    percent = 5;                                %Porcentagem de ajuste
    luva = ( data(:,:,2)<(1+percent/100)*coefCB & data(:,:,2)>
(1-percent/100)*coefCB & data(:,:,3)< (1+percent/100)*coefCR & data(:,:,3)>
(1-percent/100)*coefCR); %Segmentação da imagem

    linha = luva;
    linha(:, (w/2 - 1):(w/2 + 1), :)=1;         %Linha de separação

    figure(1)
    imshow(linha)                               %Visualização do vídeo
```

```
    plano1 = luva(:, 1:(w/2 - 2));              %Semi-plano esquerdo
    [a1 b1] = find(plano1==1);                   %Segmentação
    med1 = mean([a1 b1]);                        %Média do semi-plano

    plano2 = luva(:, ((w/2)+2):w);              %Semi-plano direito
    [a2 b2] = find(plano2==1);                   %Segmentação
    med2 = mean([a2 b2]);                        %Média do semi-plano
    med2(2) = med2(2) + w/2;

    dist = sqrt(((med2(2)-med1(2))^2));          %Cálculo da distância

    y = 2*dist + 1323;                           %Relação entre o slide e
                                                 %a distância das mãos
    mouse.mouseMove(y, 295);                     %Nova posição do slide


    final = sum(luva);                  %Soma dos valores binários de luva
    if final == 0                       %Se a soma for zero,
       cont = cont + 1;                 %Aumenta a contagem
    end

end


mouse.mouseMove (1503, 295);                     %Posição inicial do slide
mouse.mouseRelease(InputEvent.BUTTON1_MASK);     %Botão esquerdo do mouse

mouse.mouseMove (1345, 70);                      %Posição do botão Stop
mouse.mousePress(InputEvent.BUTTON1_MASK);       %Botão do mouse
mouse.mouseRelease(InputEvent.BUTTON1_MASK);     %Botão do mouse

close

stop(video);
delete(video);
```

FUNÇÃO calibraluva_ycbcr:

```
function [coefY, coefCB, coefCR] = calibraluva_ycbcr(w, h)

tamanho = ['YUY2_' num2str(w) 'x' num2str(h)];

video = videoinput('winvideo',1,tamanho);

set(video,'FrameGrabInterval',2);
set(video,'FramesPerTrigger',inf);

start(video);

cont = 1;

while (video.FramesAcquired<60)

    data = getdata(video,1);
    dataRGB = ycbcr2rgb(data);

    dataaux = dataRGB;
    dataaux((h/2 - 10):(h/2 + 10), (w/2 - 10):(w/2 + 10), 1)=0;
    dataaux((h/2 - 10):(h/2 + 10), (w/2 - 10):(w/2 + 10), 2)=0;
    dataaux((h/2 - 10):(h/2 + 10), (w/2 - 10):(w/2 + 10), 3)=0;

    imshow(dataaux, 'Border', 'loose')

    if (video.FramesAcquired>40)
        title([num2str(100*(cont/20)) '%'])
    else
        title([num2str((100*(40-video.FramesAcquired)/40)) '%'])
    end

    if (video.FramesAcquired>40)
        calib = data((h/2 - 16):(h/2 + 16), (w/2 - 16):(w/2 + 16), :);

        Y = calib(:,:,1);
        CB = calib(:,:,2);
        CR = calib(:,:,3);

        mY(cont) = mean(double(Y(:)));
        mCB(cont) = mean(double(CB(:)));
        mCR(cont) = mean(double(CR(:)));

        cont = cont+1;
    end
end

coefY = mean(mY);
coefCB = mean(mCB);
coefCR = mean(mCR);

stop(video);
delete(video);
```