GABRIEL PASCHOAL FERRAREZI

SISTEMA DE AQUISIÇÃO DE IMAGEM DO OLHO HUMANO PARA AVALIAÇÃO DA RESPOSTA DA PUPILA SUBMETIDA A ESTÍMULOS LUMINOSOS

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos,
da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

Orientador: Prof. Dr. Adilson Gonzaga

São Carlos, 2010

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

F374s

Ferrarezi, Gabriel Paschoal
Sistema de aquisição de imagem do olho humano para avaliação da resposta da pupila submetida a estímulos luminosos / Gabriel Paschoal Ferrarezi ; orientador Adilson Gonzaga. -- São Carlos, 2010.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2010.

1. Imageamento (Bioengenharia). 2. Olho. 3. Pupila. 4. Visão computacional. 5. OpenCV. I. Título.

Dedico...

Dedico este trabalho a todos os amigos e familiares que me apoiaram e auxiliaram em toda essa caminhada, em especial: minha mãe, Sirlei Aparecida Paschoal, e meu irmão, Danilo Paschoal Ferrarezi.

AGRADECIMENTOS

Agradeço a todos os professores da Escola de Engenharia Elétrica que contribuiram para meu crescimento técnico/cultural, em especial, ao professor Dr. Adilson Gonzaga, pelo suporte e apoio durante a realização deste trabalho e, principalmente, pelo incentivo e confiança. Agradeço também ao professor Dr. José Roberto Monteiro, pelo suporte, orientação e apoio durante a realização de projetos de Iniciação Científica nos anos de 2007 e 2008.

Agradeço imensamente a todos os meus grandes amigos, em especial, Matheus R de Souza, Ilana Fagá, Rodson M Heringer, Norton Revoredo, Rodolfo P Maciel, Amós da Costa, Ulisses Tristão, Alex O Watanabe, Larissa Zeid, Débora F Aumiller e Felipe Gomes.

Meus agradecimentos também a todos os amigos que fiz durante a gestão do Centro Acadêmico Armando de Salles Oliveira entre 2007 e 2008.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Ferrarezi, G. P. **Sistema de aquisição de imagem do olho humano para avaliação da resposta da pupila submetida a estímulos luminosos**. Escola de engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Este trabalho de conclusão de curso refere-se ao desenvolvimento de um sistema de aquisição de imagens do olho humano que possibilita levantar as características da pupila (contração e dilatação) em resposta a diferentes tipos de estímulos luminosos.

Fez-se uso de técnicas de visão computacional, desenvolvendo um software capaz de processar imagens(*frames*) do olho humano para identificar a variação do diâmetro da pupila ao longo do tempo e em resposta à iluminação aplicada. Os algoritmos foram implementados em linguagem de programação C/C++, utilizando-se as bibliotecas OpenCV (*Open Source Computer Vision Library*).

A análise dos dados por processamento das imagens dos olhos permite avaliar a resposta da pupila de acordo com o comprimento de onda da luz utilizada. O presente projeto consistiu em projetar e implementar o *hardware* e *software* do sistema, que possibilite posteriores estudos biométricos a partir da resposta do olho humano para os diferentes estímulos luminosos. Para comprovação da funcionalidade do sistema, levantou-se a resposta do olho humano ao estímulo luminoso para 5 usuários.

Os resultados obtidos permitem concluir que o equipamento desenvolvido poderá ser utilizado para analisar características biométricas da pupila que permitam a identificação de pessoas. É importante ressaltar que nenhum levantamento estatístico foi realizado para tal, mas apenas uma comprovação prática dos diferentes padrões de contração e dilatação da pupila do olho humano quando submetida a diferentes comprimentos de onda de luz.

Palavras-chave: Captação de Imagem, Olho, Pupila, Visão Computacional, OpenCV.

ABSTRACT

Ferrarezi, G. P. **Image acquisition system of the human eye to assess the response of the pupil submitted to light stimulation.** Escola de engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

This work of completion refers to the development of an image acquisition system of the human eye which allows to get the characteristics of the pupil (contraction and dilation) in response to different light stimulation.

It was used techniques of computer vision, developing software capable of processing images (frames) of the human eye to identify the changes in pupil diameter over time and in response of the illumination applied. The algorithms were implemented using programming language C/C++ libraries, using the OpenCV (Open Source Computer Vision Library).

The data analysis processing the images of the eyes permit to assess the response of the pupil according to the wavelength of light used. This project consisted of designing and implementing the hardware and software of the system that enables biometric subsequent studies. To prove the functionality of the system, it was gotten the human eye's response to light stimulation of 5 users.

The results demonstrate that the developed device can be used to analyze biometric characteristics of the pupil to permit identification of persons. It's important to denote that no statistical survey was carried out for this, but just a practical demonstration of different patterns of contraction and dilation of the pupil of the human eye when exposed to different wavelengths of light.

Keywords: Image Capture, Eye,Pupil , Computer Vision, OpenCV.

CAPÍTULO 1

1. INTRODUÇÃO

Biometria é a ciência que cuida das medidas da vida, como o próprio nome diz (bio=vida + metria=medida), ou seja, a ciência que estuda as características dos indivíduos com objetivos estatísticos e/ou de identificação. Para que a leitura de uma característica física ou biológica humana seja caracterizada como biometria, devem ser respeitados, basicamente, os seguintes requerimentos (JANES, 2009):

- Universalidade: cada pessoa obrigatoriamente deve possuir a característica em estudo;

- Distinção: essa característica deve ser suficientemente diferente de uma pessoa para outra, em termos de característica;

- Permanência: as características devem ser suficientemente invariantes durante certo período de tempo;

- Coletabilidade: a característica pode ser medida quantitativamente.

A medida que a biometria foi ganhando força, novas tecnologias foram surgindo e uma das principais foi a identificação por impressões digitais. Apesar da sua enorme difusão e eficácia, a comunidade científica se lançou ao desconhecido e passou a buscar outras características que pudessem ser utilizadas para a identificação, para um melhor

desempenho. Atualmente as técnicas de maior sucesso são: reconhecimento por voz, geometria das mãos, geometria do rosto e reconhecimento pela íris. Uma vantagem das duas últimas técnicas é a identificação sem contato físico.

1.1 Objetivos

O objetivo do presente projeto é propor, projetar e implementar um sistema de *hardware* de aquisição de imagens e um *software* com técnicas de Visão Computacional que possibilite o levantamento de características de dilatação e contração da pupila humana em resposta a estímulos luminosos. Ou seja, desenvolver um sistema que utiliza métodos computacionais para interpretar imagens do olho captadas durante um estímulo luminoso.

Com este sistema, novos estudos poderão ser desenvolvidos para a análise de resposta da íris, de maneira a poder-se utilizá-la como uma característica biométrica em sistemas de reconhecimento.

1.2 Justificativa

Devido à necessidades e oportunidades de evolução e aprimoramento dos sistemas de reconhecimento biométrico pela íris, encontrou-se motivação para desenvolver o presente projeto.

Como a íris humana possui em sua estrutura músculos capazes de se movimentarem quando submetidos a estímulos luminosos, as características visuais geradas por estes músculos (diâmetro da pupila, textura da íris, ...) também se alteram de maneira diferente a cada indivíduo, (DA COSTA, 2009) e (DA COSTA e GONZAGA, 2009). Logo, a contração/dilatação da pupila contribuem para o reconhecimento biométrico.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS

2.1 Visão Computacional

A Visão Computacional é um ramo da Computação que pode ser definida como o conjunto de métodos e técnicas que tornam sistemas computacionais capazes de extrair e interpretar informações de imagens. O desenvolvimento de algoritmos computacionais capazes de extrair informações presentes em uma imagem é fortemente influenciado pela compreensão dos processos de aquisição de imagens e de sua percepção no sistema visual do homem e de outros animais (CARVALHO, et al.). Um dos maiores objetivos da Visão Computacional é buscar um modelo de representação genérico que se aproxime da visão biológica, objetivo esse ainda longe de ser alcançado.

Este ramo da ciência possuiu seus primeiros estudos mais aprofundados e específicos no final da década de setenta, quando os computadores já tinham capacidade de processamento suficiente para processar grandes conjuntos de dados, como são as imagens. Por essa razão, esse ramo é considerado como estando ainda em seus primórdios.

A visão computacional engloba e pode ser aplicada a diversas áreas do conhecimento: inteligência artificial (inteligência computacional), aprendizado de máquina (visão cognitiva), processamento de sinais, física (ótica), matemática, neurobiologia (visão biológica), robótica (visão de robôs).

Aplicações da Visão Computacional podem ser encontradas como parte de sistemas complexos. São exemplos desses sistemas equipamentos para processamento de imagens médicas, aplicações industriais para controle de qualidade ou medição de características de produtos e até mesmo aplicações na área militar (mísseis teleguiados, reconhecimento de oponentes, ...). Além dessas áreas mais proeminentes, a visão computacional pode ser aplicada no reconhecimento de pessoas e padrões, segurança de ambientes, visão de robôs e várias outras atividades. Muitos trabalhos que necessitam da visão humana são repetitivos, alguns perigosos, e até podem prejudicar esse importante sentido. A visão computacional pode auxiliar nestas tarefas com grande eficiência.

2.2 Sistema Ocular

O olho é um maravilhoso e complexo órgão do corpo humano. O olho humano é um órgão capaz de captar a luz refletida por tudo que está em sua direção, transformando esta luz em sinais elétricos e em conjunto com o sistema nervoso interpretá-la como uma imagem.

A entrada da luz no globo ocular se dá pela córnea, um tecido transparente que cobre toda a íris do olho. A luz passa pela pupila e atinge o cristalino, que atua como uma lente de focalização, fazendo todos os raios convergirem em um ponto na retina. A retina é composta por milhões de células fotossensíveis que podem ser de dois tipos: os bastonetes e os cones. Os bastonetes são responsáveis pela percepção de claro e escuro e os cones respondem pela percepção das cores. É na retina que os estímulos de luz são convertidos em impulsos eletroquímicos, que serão transmitidos ao cérebro pelo nervo óptico. No cérebro, o córtex visual recebe os impulsos dos dois olhos, faz o processamento e completa a sensação visual. A informação enviada pelos dois olhos processada simultaneamente dá a percepção de profundidade (KANSKI, 2007).

A íris é um tecido pigmentado (uma rede de trabéculas, rede de filamentos cruzados, formada no 8º mês de gestação) que fica entre a córnea e o cristalino, conforme apresentado na Figura 2.1. Ela tem uma superfície relativamente plana, protegida de agentes externos pela córnea e com um orifício central denominado pupila. A íris possui no seu estroma (trama de tecido localizado em sua circunferência interna) os músculos dilatadores, que servem para contrair e dilatar a pupila. Assim, a função principal da íris é controlar, através da pupila, a quantidade de luz que entra no olho, da mesma forma que o diafragma de uma máquina fotográfica.

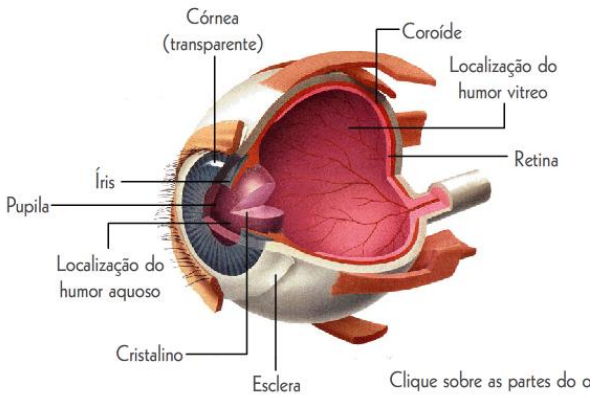

Figura 2.1: Anatomia do olh

Fonte: Disponível em < http://www.gta.ufrj.br/grad/09_1/versao-final/iris/index.html>

A função da íris e da pupila não é apenas controlar a luminosidade que incide na retina, mas também regular a profundidade de foco. Para focalizar objetos distantes o seu diâmetro diminui, e para objetos próximos o diâmetro aumenta, de maneira involuntária, mas altamente precisa. Não coincidentemente o diafragma de uma boa máquina fotográfica manual também pode ser usado para alterar a profundidade focal, algo que todo "bom" fotógrafo está familiarizado.

A rede de filamentos da íris é formada de maneira aleatória, onde suas trabéculas são posicionadas aleatoriamente, não tendo nenhuma ligação com genética ou algo do gênero. A íris é um ótimo identificador biométrico devido a sua unicidade, sua estabilidade, fácil utilização e sua universalidade.

Estudos mostrados em (DAUGMAN, 2006) comprovam que o sistema de reconhecimento biométrico usando características da íris propostos pelo mesmo garantem a unicidade da mesma em uma proporção de 1 erro em 200 bilhões de análises.

Além disso, a íris se forma desde o terceiro mês de gestação até o final do primeiro ano de vida. Após isso, a íris permanece igual até a morte, garantindo sua estabilidade, a não ser que hajam acidentes que a deformem de alguma maneira.

Devido as características mostradas acima a anatomia do olho humano, em especial a formação da íris, conferem a este órgão um alto potencial para aplicações de identificação pessoal.

O reconhecimento pela íris normalmente é baseado em qualidades visíveis como anéis, estrias, manchas, coroas, etc. Basicamente, os algoritmos procuram converter estas características visíveis em um código que será o padrão armazenado para futura verificação, (NEGIN, et al., 2000).

2.3 Reconhecimento Biométrico

Os processos mais comuns de reconhecimento pela íris iniciam-se com a aquisição de uma imagem da íris sob iluminação infravermelha, pois esta revela mais detalhes de textura quando sob este tipo de iluminação. A imagem resultante é analisada utilizando algoritmos específicos, que localizam a íris e extraem a informação necessária para criar uma amostra biométrica. (DAUGMAN, 2004)

Alguns métodos de identificação por análise da íris geram aproximadamente 600 pontos (baseados em características de anéis, estrias, manchas e coroas encontradas na íris), enquanto a média de minúcias geradas por uma impressão digital é de 50. O reconhecimento pela íris é mais preciso, no entanto, o custo ainda é elevado para uma utilização em larga escala. O processo de leitura da íris consiste em:

a) Detectar a presença do olho na imagem;
b) Localizar os limites interiores e exteriores da Íris;
c) Detectar e excluir as pálpebras se forem indutoras de erro;
d) Definir um sistema de coordenadas 2D, o qual é mapeado o padrão da íris e gerado o seu código;
e) Armazenar o código da íris em hexadecimal em um banco de dados ou outro tipo de mídia. Uma vez no banco de dados, o código é usado como base para a comparação contra a íris capturada pela câmera no processo de identificação de uma pessoa.

A segmentação da íris é realizada através da localização do raio da íris e da pupila e das coordenadas do centro da imagem. Deve-se levar em consideração que o centro da pupila não é necessariamente o centro da íris, e esta variação pode chegar a até 0,8mm (CARVALHO e JUNIOR, 2008). A Figura 2.2 ilustra esta ocorrência. Para segmentar as fronteiras da íris e separá-la da pupila, Daugman (2004) propôs o uso da Equação 2.1.

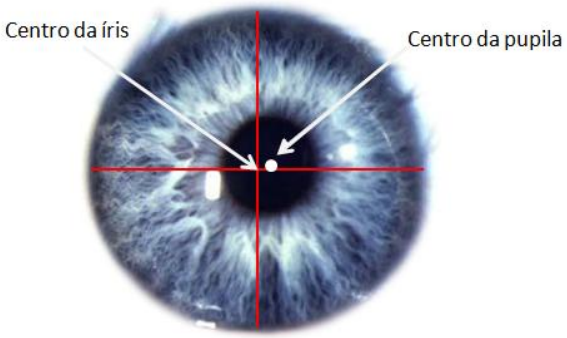

Figura 2.2: Pupila descentralizada com a íris.

Fonte: Disponível em < http://hgculiacan.com/bilblioteca%20medica/retina/anatomia%20retina.htm>

Um dos mais difundidos métodos para codificação da Íris utiliza *Wavelets 2D* de Gabor (Equação 2.1), que reconhece a íris por vetores e adquire as informações relevantes da imagem, como orientação, frequência espacial e posicionamento. Com essas informações é possível mapear o código da Íris. A Equação 2.1 estima o código de uma íris e a imagem da Figura 2.3 é uma representação pictórica de um código de Íris (DAUGMAN, 2004).

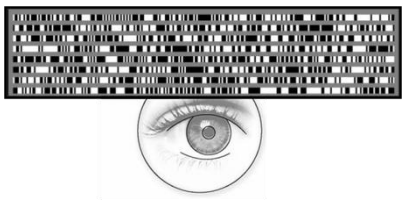

Figura 2.3: Representação Pictórica de um código de Íris.

$$h_{\{\mathrm{Re,Im}\}} = \mathrm{sgn}_{\{\mathrm{Re,Im}\}} \int_{\rho}\int_{\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} \cdot e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho d\rho d\phi \qquad \text{Equação 2.1}$$

Na qual,

- h{Re,Im} é um bit que representa um valor complexo, real ou imaginário pertencente ao conjunto (0, 1, -1), dependendo do sinal da integral.
- I( ) é a imagem no sistema de coordenadas pseudo-polar
- α e β são parâmetros da wavelet 2D multi-escala e
- w é a freqüência da wavelet.

O trabalho de Da Costa (2009) propõe um sistema inovador de Reconhecimento Biométrico, onde é utilizado características dinâmicas da íris humana e não somente dados estáticos como mostrado acima.

O sistema desenvolvido por Da Costa (2009) utiliza o reflexo consensual, analisando a contração ou dilatação da pupila através de estímulos luminosos e extrai as seguintes características:

1)	Circularidade da pupila
2)	Diâmetro da pupila
3)	Tempo para contração/dilatação da pupila
4)	Taxa de contração/dilatação da pupila
5)	Média dos níveis de cinza da íris segmentada
6)	Desvio padrão dos níveis de cinza da íris segmentada
7)	Coeficiente de variação dos níveis de cinza da íris segmentada
8)	Correlação (0º, 45º, 90º e 135º)
9)	Segundo Momento Angular (SMA) (0º, 45º, 90º e 135º)
10)	Entropia (0º, 45º, 90º e 135º)
11)	Contraste (0º, 45º, 90º e 135º)
12)	Momento da Diferença Inverso (MDI) (0º, 45º, 90º e 135º)

Estas 12 (doze) características são analisadas durante os movimentos de contração e dilatação da pupila. A partir destas características gera-se um padrão de resposta para cada pessoa e monta-se um banco de dados para posterior reconhecimento.

2.4 Reflexo Consensual

Na transmissão dos estímulos, o nervo óptico o olho humano direito e esquerdo se "conectam", em uma região denominada quiasma óptico conforme mostrado na Figura 2.4. No quiasma óptico ocorre o cruzamento das fibras médias dos nervos ópticos, de tal forma que fibras do nervo óptico direito passam para o nervo óptico esquerdo e vice-versa. Isso faz com que os olhos estejam "conectados", e os reflexos a estímulos aplicados em um dos olhos sejam apresentados também no outro. Esta função fisiológica é denominada de "Reflexo Consensual" (KANSKI, 2007).

O Reflexo Consensual é responsável por "sincronizar" os movimentos de ambos os olhos, desta forma, se a pupila de um dos olhos contrai ou dilata, a pupila do outro olho realiza o mesmo movimento.

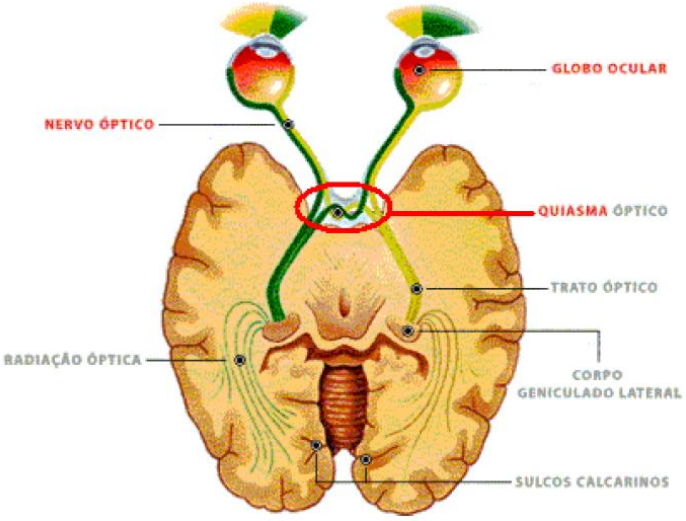

Figura 2.4: Sistema Óptico humano – detalhe da região do quiasma óptico.
Fonte: (Adaptado de STURGES, 2005)

2.5 Visão Computacional (OpenCV)

OpenCV é uma biblioteca *open source* escrita em C e C++ que é utilizada em Plataformas Linux, Windows e Mac OS X. Esta biblioteca foi desenvolvida com foco em aplicação em tempo real e possibilidade de simples uso da infraestrutura de visão computacional, para facilitar a rápida implementação de sofisticadas aplicações de visão computacional, (BRADSKI e KAEHLER, 2008).

A biblioteca OpenCV contém cerca de 500 funções que abrangem várias áreas em visão, incluindo inspeção de produção, imagens médicas, segurança, interface com o

usuário, calibração de câmeras e robótica. Esta biblioteca também possui uma completa sub-biblioteca focada em estatística de reconhecimento de padrões e agrupamento.

Dentre os fundamentos de Visão Computacional e a implementação em OpenCV, alguns tópicos serão apresentados e utilizados neste trabalho.

- ***SMOOTHING***

A operação de *smoothing* quando aplicada a uma imagem tem a intenção de melhorar sua qualidade. Ela busca suavizar a imagem eliminando ruídos de alta frequência. A forma mais comum dessa operação é calcular o valor médio dos pixels que estejam na vizinhança de um pixel que está sendo analisado. Caso o valor do pixel seja muito diferente do valor médio dos pixels vizinhos, ele é interpretado como ruído e seu valor é corrigido. A eliminação de ruídos através da técnica de *smooth* facilita a aplicação de algoritmos de segmentação e reconhecimento posteriores (BRADSKI e KAEHLER, 2008).

No OpenCV temos a seguinte função de *Smooth*:

```
cvSmooth( image, out, CV_GAUSSIAN, 3, 3 );
```

Na qual:

- *image* é a imagem de entrada do sistema;

- *out* é a imagem de saída, após realizada a operação de *smoothing*;

- *CV_GAUSSIAN* é o tipo de operação que será realizada (para o OpenCV existem 5 diferentes metodologias);

- os últimos 2 parâmetros são os valores da altura e largura da janela do filtro.

- ***THRESHOLDING***

Identificar partes de interesse em uma imagem é uma das etapas mais críticas no processamento de imagens. Uma das maneiras mais simples de selecionar um determinado objeto ou parte em uma cena é através da operação de *thresholding*.

Threshold é um valor limite que é usado em um critério de seleção. Todos os pixels de uma imagem são comparados a esse critério de seleção e são alterados conforme a necessidade (BRADSKI e KAEHLER, 2008).

Um exemplo de aplicação de *thresholding* é comparar todos os pixels a um valor limite e caso sejam maiores não alterar seu valor e, caso sejam menores, igualar a zero. Desta forma destaca-se uma região da imagem.

Comumente a operação de *thresholding* gera imagens binárias ou em níveis de cinza. Conforme a aplicação, qualquer outro tipo de destaque da região de interesse pode ser usado para aplicar cores diferentes. Essa operação pertence à etapa de segmentação do processamento de imagens.

No OpenCV tem-se a seguinte função de *Threshold*:

```
double cvThreshold(
                    CvArr* src,
                    CvArr* dst,
                    double threshold,
                    double max_value,
                    int threshold_type);
```

Na qual:

- os dois primeiros índices referem-se a imagem de entrada e saída do sistema, respectivamente;

- o terceiro e quarto índices referem-se aos valores de *threshold*;

- e por último, tem-se o tipo de metodologia de *thresholding*, podendo ser quatro diferentes tipos para o OpenCV - *CV_THRESH_BINARY*, *CV_THRESH_BINARY_INV*, *CV_THRESH_TRUNC*, *CV_THRESH_TOZERO_INV* e *CV_THRESH_TOZERO*.

TRANSFORMADA DE HOUGH (TH)

A transformada de Hough foi desenvolvida por Paul Hough em 1962. Identifica linhas, círculos e elipses em imagens binárias e é uma importante ferramenta no processamento de imagens digitais. A transformada é normalmente aplicada a uma imagem que passou anteriormente pela operação de detecção de bordas (JAMUNDÁ, 2000).

A Transformada de Hough aplica na imagem uma transformação tal que todos os pontos pertencentes a uma mesma curva sejam mapeados em um único ponto de um novo espaço de parametrização da curva procurada. A Figura 2.5 ilustra esta técnica:

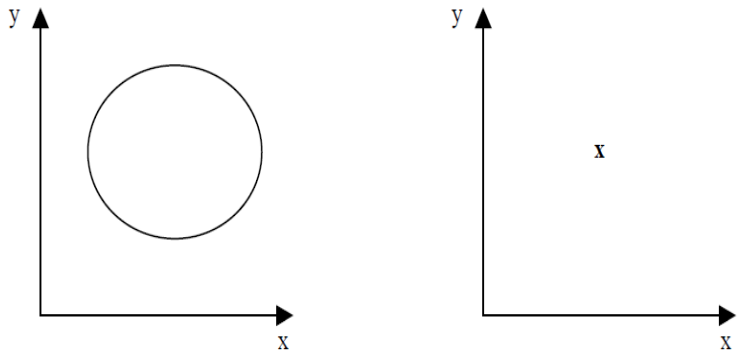

Figura 2.5: Uma curva qualquer parametrizada no espaço x – y , sofre a Transformada de Hough e passa a ser mapeada em um único ponto.

A Transformada de Hough tem como vantagem o fato de que pode ser aplicada ao tratamento de qualquer tipo de curva parametrizável e, além disso, apresenta muita eficiência em imagens ruidosas.

No entanto, tem como desvantagem o fato de que se uma curva é mais complexa, ou seja, com um número maior de parâmetros, o esforço computacional exigido é bem maior.

Neste projeto, devido ao formato da pupila ser quase circular, o interesse é utilizar a Transformada de Hough para detecção de círculos.

A TH pode ser realizada da seguinte maneira*:

- Considere o círculo Δ1 a seguir, de raio R e centro de coordenadas (x_c, y_c), conforme mostra a Figura 2.6.

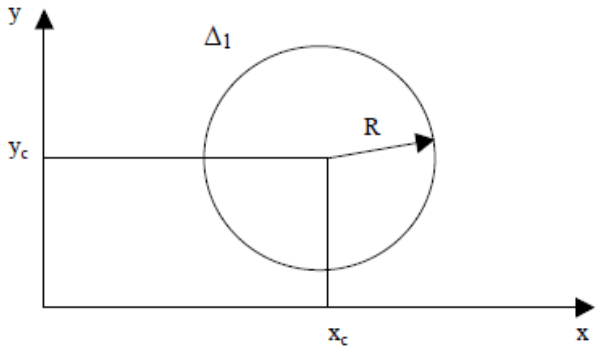

Figura 2.6: Um círculo Δ1 perfeito de raio R e centro de coordenadas (xc, yc).

- Traçando-se uma tangente t ao círculo Δ1 e um vetor gradiente G perpendicular à t, tem-se a geometria apresentada na Figura 2.7:

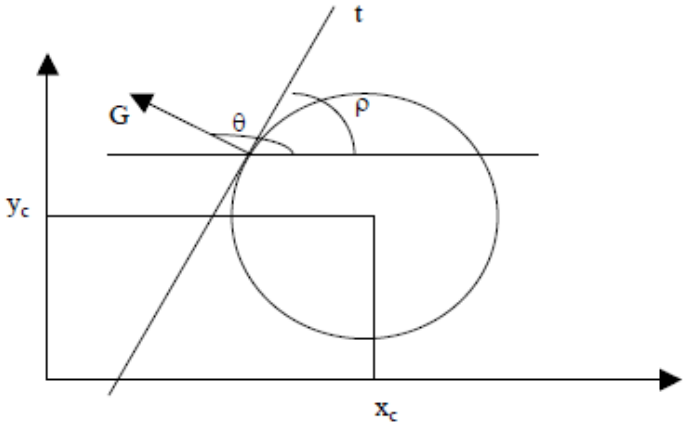

Figura 2.7: São traçados ao círculo Δ1 uma tangente t e um vetor gradiente G, perpendicular à t.

θ é o ângulo formado entre o vetor gradiente G e o eixo horizontal x e ρ é o ângulo formado entre a tangente t e o eixo horizontal x.

- Da geometria elementar tem-se que a equação paramétrica do círculo é:

$$(x - x_c)^2 + (y - y_c)^2 = R^2 \qquad (1)$$

- Aplicando-se a derivada de primeira ordem, relativa ao gradiente (em função de x), em (1), chega-se a:

$$2(x - x_c) + 2(y - y_c)\frac{dy}{dx} = 0 \qquad (2)$$

*Disponível em <http://www.ppgia.pucpr.br/~facon/Hough/Hough-Circulo.pdf>

onde temos que:

$$\frac{dy}{dx} = tg\ \rho \tag{3}$$

e

$$\theta = \frac{\pi}{2} + \rho, ou\ seja, \rho = \theta - \frac{\pi}{2} \tag{4}$$

- Substituindo-se (4) em (3), tem-se que:

$$\frac{dy}{dx} = tg\left(\theta - \frac{\pi}{2}\right), \text{ logo}$$

$$\frac{dy}{dx} = -\frac{1}{tg\theta} \tag{5}$$

- Substituindo-se (5) em (2), tem-se:

$$(y - y_c) = tg\theta(x - x_c) \tag{6}$$

- Substituindo-se (6) em (1), chega-se a:

$$(x - x_c)^2 + (x - x_c)^2 tg^2\theta = R^2 \tag{7}$$

- Isolando-se $(x - x_c)^2$ em (7), chega-se a:

$$(x - x_c)^2[1 + tg^2\theta] = R^2 \tag{8}$$

- Como $tg^2\theta = \frac{sen^2\theta}{cos^2\theta}$:

$$(x - x_c)^2\left[1 + \frac{sen^2\theta}{cos^2\theta}\right] = R^2$$

$$(x - x_c)^2\left[\frac{sen^2\theta + cos^2\theta}{cos^2\theta}\right] = R^2$$

$$(x - x_c)^2\left[\frac{1}{cos^2\theta}\right] = R^2$$

$$(x - x_c)^2 = R^2.cos^2\theta$$

Simplificando-se:

$$(x - x_c) = R.cos\theta$$

ou

$$(y - y_c) = R.sen\theta \tag{9}$$

x e y definem as coordenadas do ponto no novo espaço de parametrização do círculo, x_c e y_c definem as coordenadas do centro e R é o raio do mesmo.

Em OpenCV tem-se a seguinte função para a Transformada Circular de Hough:

```
CvSeq* cvHoughCircles(
    CvArr* image,
    void* circle_storage,
    int method,
    double dp,
    double min_dist,
    double param1,
    double param2,
    int min_radius,
    int max_radius);
```

Na qual:

- *image* é a imagem na qual será realizado a TH;

- *circle_storage* é uma variável para armazenamento dos valores após a detecção;

- *method* trata-se do parâmetro de escolha do método do cálculo para detecção de circunferências;

- *dp* é a resolução do acumulador da imagem;

- *min_dist* é a distância mínima que deve haver entre 2 círculos;

- *param1* e *param2* são os valores de *threshold* do detector de bordas (Canny) e do acumulador, respectivamente;

- *min_radius* e *max_radius* são os valores mínimos e máximo de raio aceitos pela TH.

CONVERSÃO RGB PARA A ESCALA DE CINZA

Imagens coloridas podem ser representadas no espaço de cores RGB (*Red, Green,* Blue) onde cada *pixel* é representado por um vetor de 3 dimensões. No entanto, algumas operações de processamento de imagens são facilitadas quando os pixels tem apenas uma dimensão, ou seja, a imagem é em escala de cinza ou binária. A aplicação da Transformada de Hough é um exemplo: processar três canais de cor para identificar retas e círculos seria complexo e dispendioso, além de envolver processamentos desnecessários.

Existem diversas maneiras para converter imagens RGB em escala de cinza. A idéia é somar valores proporcionais de cada uma das componentes de cor para obter o valor da luminância de um pixel. No caso dessa aplicação, a seguinte proporção é utilizada na soma das componentes RGB (LYU e FARID, 2003):

$$Y = 0{,}299R + 0{,}587G + 0.114B$$

Dessa forma, as três informações referentes a um pixel colorido são transformadas em uma informação referente a um pixel em escala de cinza. A informação de luminância pode ser representada por valores inteiros ou reais.

Uma função do OpenCV para conversão de escala de cor é a:

```
void cvCvtColor(
        const CvArr* src,
        CvArr* dst,
        int code);
                        [Conversion Code - CV_RGB2GRAY]
```

Na qual:

- *src* é a imagem na qual será realizado a conversão;

- *dst* é a imagem de saída*;*

- e *code* é o tipo de conversão que sera realizado – de RGB para escala de cinza, utiliza-se CV_RGB2GRAY .

CAPÍTULO 3

3. MATERIAL E MÉTODOS

O objetivo deste trabalho é propor, construir e avaliar um sistema de aquisição de imagens da íris durante estímulos luminosos, para posterior interpretação através de técnicas de Visão Computacional.

O diagrama em blocos da Figura 3.1 representa a proposta geral deste trabalho

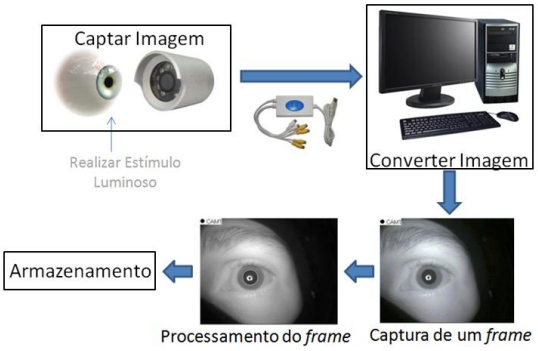

Figura 3.1: Diagrama de blocos da aplicação.

Os três blocos finais (Captura de um *frame*, processamento do *frame* e Armazenamento) do diagrama da Figura 3.1 correspondem às implementações de *software*. Toda a parte de *software* foi desenvolvida utilizando-se bibliotecas, compiladores e ambiente de desenvolvimento livre, no caso a OpenCV (*Open Source Computer Vision Library*) e o Microsoft Visual C++ 2008. Já os três blocos iniciais do diagrama correspondem às implementações de *hardware.*

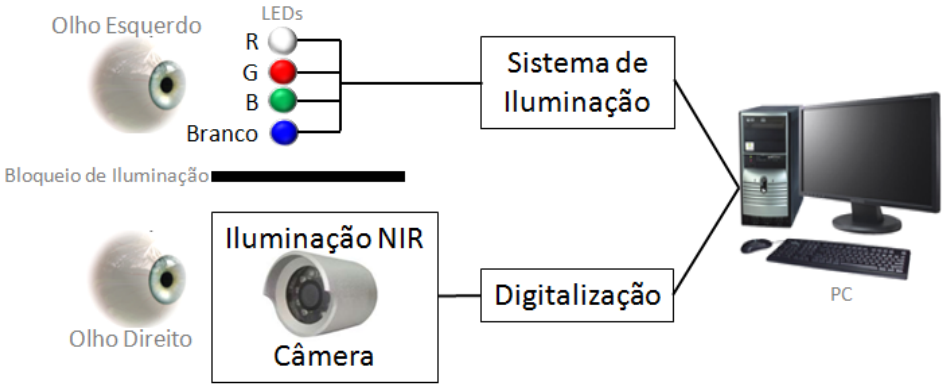

Figura 3.2: Sistema proposto neste trabalho.

O sistema de aquisição de imagem do olho humano proposto neste trabalho que permitirá realizar a avaliação da resposta da pupila (contração/dilatação) submetida a estímulos luminosos é apresentada na Figura 3.2. No olho esquerdo é realizado o estímulo luminoso com leds Vermelho(R), Verde(G), Azul(B) e Branco, independentemente e controlados por um programa instalado no microcomputador. Devido ao reflexo consensual, a cada estímulo luminoso independente aplicado ao olho esquerdo, os dois olhos vão se comportar de maneira semelhante, contraindo-se ou dilatando-se a pupila conforme a aplicação ou retirada do estímulo (KANSKI, 2007). Do olho direito obtém-se, sincronamente ao impulso luminoso aplicado ao olho esquerdo, a imagem com iluminação NIR (*Near Infra Red),* que permite obter-se imagens do olho com a ausência de luz visível, de maneira a evitar reflexos excessivos na córnea. O *software* de processamento de imagem deverá então analisar o comportamento da pupila a cada diferente estímulo.

3.1 Desenvolvimento do *Hardware*

Para se atingir os objetivos propostos neste trabalho fez-se necessário projetar e implementar um *hardware* dedicado e específico para a captura da imagem e controle da iluminação que estimula o olho.

3.1.1. Captura de Imagem

A parte mais importante do sistema é a aquisição da imagem. Se houver uma deficiência no mesmo, torna-se inviável obter respostas precisas. Ko, Gil e Yoo (2006) com principios bastante simples, em seu sistema de reconhecimento pela íris, utilizou uma câmera com circuito CCD (*charge-coupled device* - circuitos integrados de estado sólido que contém vários diodos que atuam armazenando a luz a que são expostos, fotodetectores) e iluminação infravermelho para capturar imagens com resolução de 320 x 240 pixels onde o diâmetro da íris apresenta aproximadamente um raio de 170 pixels. Com

um equipamento de baixo custo, atingiram o objetivo de obter imagens adequadas para a identificação.

Baseando-se no trabalho de Da Costa (2009) e no trabalho de Ko, Gil e Yoo(2006), decidiu-se qual seria a especificação do sistema de Aquisição de Imagem do presente projeto que contemplaria as necessidades:

- Câmera infravermelho NIR, para possibilitar aquisição da imagem com níveis de luminosidade próximas a 0 LUX, reduzindo reflexos na córnea;

- Câmera com leds, para gerar a iluminação NIR, permitindo a aquisição de imagens nesta faixa de frequência;

- Câmera para uso em pequenas distâncias, com poucos Leds IR, evitando reflexos excessivos e mais segura ao usuário;

- Câmera com perfomance e qualidade adequada, permitindo uma maior definição de imagens;

Após a análise do custo benefício de diferentes marcas e modelos de câmeras, decidiu-se por desenvolver o sistema de Aquisição de Imagem com uma Câmera CCD Color Sony modelo KIR-1004N da marca KODO, para uso em sistemas de vigilância - com alcance de até 5 metros. A Tabela 3.1 apresenta os dados técnicos da câmera e a Figura 3.3 a vista geral da mesma.

Tabela 3.1: Dados Técnicos da Câmera KODO.

MODELO	**KIR - 1004N**
SENSOR	1/3" SONY SUPER HAD CCD
TOTAL PIXEL	510 (H) X 492 (V) 500 (H) X 582 (V)
SISTEMA DE ESCANEAMENTO	NTSC 525 linhas PAL 625 linhas
SINCRONISMO	Interno
RESOLUÇÃO	330 linhas
SAÍDA DE VÍDEO	VBS 1.0 Vp-p Composto (75 ohms)
RUÍDO	Maior que 48dB (AGC Off)
ILUMINAÇÃO MÍNIMA	1.0 Lux (F=2.0), menor que 0.000 Lux (modo IR)
VISIBILIDADE	10 LED´s (850nm): 5m
GAMA	r = 0.45
WHITE BALANCE	White Balance Automático
LENTE	f: 4mm F=2.0
SHUTTER SPEED	1/60 ~ 1/100,000 seg
TEMPERATURA DE OPERAÇÃO	-10°C ~ 50°C
UMIDADE	90%
CONSUMO	DC 12V +/- 10% (Max 1.9Watts, 2.1Watts com IR LED)

Figura 3.3: Câmera KODO - KIR 1004N.

Para digitalizar a imagem capturada pela câmera, utilizou-se uma placa de Captura RCA/USB com 4 canais a 30fps (*frames per second*), conforme mostra a Figura 3.4.

Figura 3.4: Placa de Captura USB 4x30fps (Marca: Globo, Linha Silver).

3.1.2. Circuito de Iluminação

Após ser definido o sistema de aquisição de imagens, foi projetado o circuito de iluminação, para aplicação do estímulo luminoso sobre o olho.

Utilizando-se a porta paralela implementou-se um circuito de iluminação com 4 LEDS – branco, vermelho, azul e verde e um cabo paralelo, como mostrado na Figura 3.5.

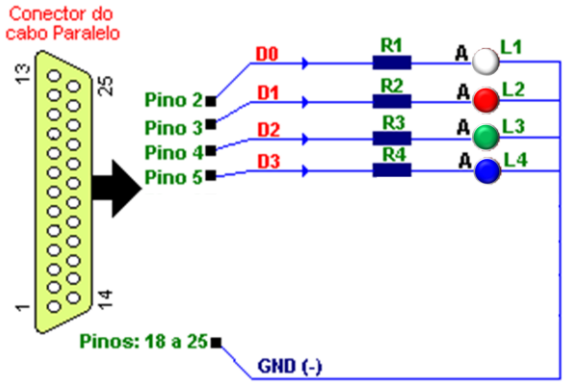

Figura 3.5: Circuito de Iluminação.

Porta Paralela

As portas paralelas foram originalmente desenvolvidas pela IBM como uma maneira de conectar a impressora ao computador. Quando a IBM estava no processo de projeto do computador, a empresa queria que ele funcionasse com impressoras produzidas pela Centronics, uma fabricante de primeira linha de impressoras na época. A IBM decidiu não usar no computador a mesma interface de portas que a Centronics usava na impressora (TYSON, 2001).

Figura 3.6: Conector fêmea D-Sub de 25 pinos.

Em vez disso, os engenheiros da IBM uniram um conector de 25 pinos, DB-25, a um conector Centronics de 36 pinos para criar um cabo especial para conectar a impressora ao computador. Outros fabricantes de impressoras acabaram adotando a interface da Centronics, tornando esse cabo híbrido improvável de se firmar como um padrão.

Quando um computador envia dados para a impressora ou outro dispositivo usando a porta paralela, ele envia 8 bits de dados (1 byte) de uma vez. Esses 8 bits são transmitidos em paralelo. A porta paralela padrão é capaz de enviar de 50 a 100 kilobytes de dados por segundo.

A função de cada pino da porta, mostrado na Figura 3.6: Conector fêmea D-Sub de 25 pinos.Figura 3.6, é:

- O pino 1 carrega o sinal de estrobo (*strobe*). Ele mantém um nível entre 2,8 a 5 Volts, mas cai abaixo de 0,5 volts toda vez que o computador envia um byte de dados. Essa queda de tensão diz à impressora que os dados estão sendo enviados;

- **Os pinos 2 ao 9 são usados para carregar dados**. Para indicar que um bit tem valor 1, uma tensão de 5 volts é enviada através do pino. A ausência de tensão no pino indica valor 0;

- O pino 10 envia o sinal de reconhecimento (*acknowledge*) da impressora para o computador. Como o pino 1, ele mantém uma tensão entre 2,8 e 5 Volts e leva a tensão abaixo de 0,5 volts para permitir que o computador saiba que a impressora está pronta para receber mais dados;

- Se a impressora estiver ocupada (*busy*), haverá tensão no pino 11. Depois, a impressora vai levar a tensão para um valor abaixo de 0,5 volts para permitir que o computador saiba que ela está pronta para receber mais dados;

- Se o computador recebe uma tensão no pino 13, ele sabe que o dispositivo está online;

- Os pinos 18 a 25 são terra e são usados como um sinal de referência para a tensão baixa (abaixo de 0,5 volts).

Para o cálculo dos valores das resistências, utilizou-se corrente de polarização de 5 mA, os respectivos valores de Tensão de condução dos LEDs(U), conforme mostra a Figura 3.7, a Lei de Ohm ($U = R.I$), e o valor de tensão dos pinos 2 a 9 da porta paralela (5V), chegando-se aos valores apresentados na Tabela 3.2:

Tabela 3.2: Calculo dos Resistores - Circuito de Iluminação.

LED BRANCO – L1	U = 2,7V $R = \left(\frac{5-2,7}{0,005}\right) = 460ohms$
LED VERMELHO – L2	U = 1.8V $R = \left(\frac{5-1,8}{0,005}\right) = 640ohms$
LED VERDE – L3	U = 2.7 $R = \left(\frac{5-2.7}{0,005}\right) = 460ohms$
LED AZUL – L4	U = 2.84V $R = \left(\frac{5-2.84}{0,005}\right) = 432ohms$

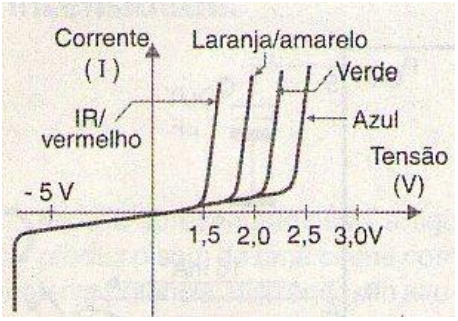

Figura 3.7: Curva da Tensão de Condução dos LEDs.

Portanto, os resistores utilizados foram de 470 e 620 ohms, isto porque foi necessário diminuir o incômodo ao usuário ao passar pelos procedimentos de aquisição de dados, problema que foi reduzido utilizando-se uma corrente de polarizaçao de apenas 5mA.

As cores utilizadas no circuito de iluminação, foram assim definidas (R, G, B e Branco), baseado na resposta fisiológica do olho humano à luz, (MILLER, 1985) e (ALVES, 2002). Fundamentalmente, a luz é um espectro contínuo de comprimento de onda, o que significa que na realidade existe um número quase infinito de cores. Entretanto, o olho humano normal tem apenas três tipos de receptores sensíveis a cor, chamados cones. Tais cones respondem a larguras de ondas específicas das cores vermelha, verde e azul – por isto, o uso dessas cores como estímulo luminoso. Embora a sensibilidade máxima dos cones não se produz exatamente nessas três freqüências de cores, essas cores são consideradas as cores primárias em fontes de luz (sistema RGB), porque cada uma delas pode estimular os cones de forma praticamente independente, proporcionando uma ampla gama de cores.

3.1.3. Desenvolvimento do Protótipo do Sistema

Atualmente existem diversos modelos de produtos comerciais para o reconhecimento de pessoas através da biometria da Íris. A Figura 3.8 mostra a aparência de alguns deles.

Figura 3.8: Equipamentos Comerciais - Reconhecimento Biométrico da Íris.
Fonte: Disponível em <http://www.irisid.com/ps/products/index.htm> e
<http://www.panasonic.com/business/security/bm-et200_iris_reader_animation/index.html>

Os produtos da linha *IrisAccess* da marca LG e o modelo *BM-ET200* da PANASONIC possuem como características:

- uso de imagens estáticas da íris;
- reconhecimento sem contato entre o usuário e o produto;
- alta velocidade de processamento, menos de 5 segundos para processamento do algorítimo;
- banco de dados de 50 pessoas por produto;
- usa imagem do olho esquerdo e direito;
- possibilidade de geração de sistema em rede;
- pequenos, leves e portáteis.

Baseando-se nos produtos existentes no mercado, mostrados acima, e pesquisas no ramo de Identificação biométrica tomou-se como pontos chave para o desenvolvimento do Protótipo do Projeto:

- ser transportável;
- ser ergonômico, para a maior quantidade possível de tipos de rostos;
- causar o mínimo desconforto possível ao usuário;
- possibilitar avanços e/ou customizações em projetos posteriores;

Como mostrado anteriormente, o protótipo utiliza os seguintes componentes eletrônicos:

- Uma câmera CCD Color Sony modelo KIR-1004N da marca KODO, com leds IR de alcance de até 5 metros;

- Fonte de alimentação da Câmera;
- Sistema de iluminação, com leds branco, vermelho, verde e azul.
- Placa de captura RCA – USB, 4 canais;

E para a montagem de todo o sistema, construiu-se uma base de madeira com as seguintes características:

- Madeira de 3 mm;
- Dimensões – A=18cm, B=12cm, C=30cm e D=5cm, Figura 3.9;
- Tecido preto para isolação da iluminação ambiente;
- Base de óculos de proteção – tamanho único para adultos.

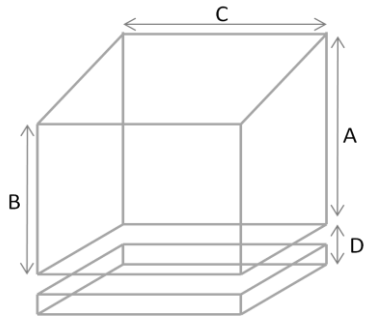

Figura 3.9: Dimensional do Protótipo.

Após a montagem da base do protótipo, instalou-se a câmera de maneira a capturar imagens do olho direito e os leds, de maneira a estimular o olho esquerdo, como verifica-se na Figura 3.10 e Figura 3.11.

Estímular um olho e capturar a imagem em outro evita que o reflexo de luz visível na córnea prejudique a qualidade das imagens e consequentemente dos dados obtidos na mesma. Daí a exploração do Reflexo Consensual (KANSKI, 2007), explicitado anteriormente. Portanto, aplicando-se o estímulo luminoso em um dos olhos, pode-se levantar a resposta ao estímulo em ambos os olhos – no caso, o estímulo luminoso no lado esquerdo auxiliará o levantamento da característica de resposta dinâmica do olho direito.

Figura 3.10: Protótipo final do Projeto.

Para o funcionamento do sistema, é necessário um computador com plataforma Windows XP e os drivers da Câmera e Porta LPT1 instalados. Para possibilitar a comunicação entre o Software desenvolvido em Linguagem C/C++ e o mesmo.

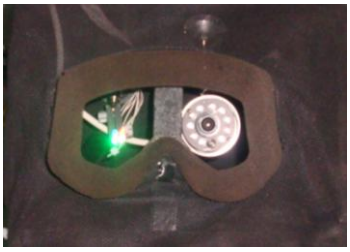

Figura 3.11: Prótipo - Acionamento dos LEDs.

A Figura 3.12 apresenta um exemplo da imagem de um *frame* gerado pelo dispositivo. Os pontos brancos e cinza na pupila são os reflexos da iluminação *NIR* utilizada pela câmera (*LED's* infra-vermelhos), utilizados para iluminar o olho e permitir a captura do vídeo. Estes reflexos não comprometem a avaliação, pois o dispositivo foi projetado para que eles estejam sempre no centro da pupila, mesmo com sua máxima contração. A informação importante sobre a pupila é seu diâmetro e circularidade, não importando os reflexos em seu interior. É possível, também, observar que a íris não apresenta reflexos que possam prejudicar sua segmentação ou extração de características.

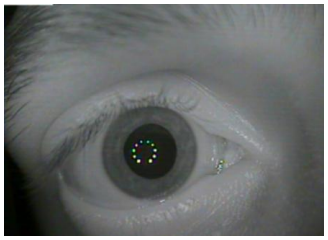

Figura 3.12: Exemplo de imagem de um *frame* gerado pelo dispositivo de Captura.

3.2 Desenvolvimento do *Software*

Wildes et al. (1994) realizaram a localização da íris usando a transformada de Hough circular e o uso da transformada parabólica de Hough para detectar pálpebras, aproximando os limites superiores e inferiores das pálpebras com arcos parabólicos através da polarização das derivadas na direção horizontal para detecção das pálpebras e na direção vertical para a detecção dos limites circulares da íris. A motivação para tal vem do fato de as pálpebras serem normalmente alinhadas horizontalmente, e também o mapa de bordas das pálpebras irão corromper o mapa de bordas dos limites circulares da íris se usados todos os dados do gradiente. Somente com os gradientes verticais para localização dos limites da íris, reduz-se a influência das pálpebras quando realizada a transformada circular de Hough.

Após basear-se nos trabalhos mostrados acima, decidiu-se utilizar a Transformada Circular de Hough para obter os dados do diâmetro da Íris, de maneira a construir um *software* veloz e consistente com o objetivo do projeto (obter o valor do diâmetro da pupila).

3.2.1. Microsoft Visual C++

Para o desenvolvimento de técnicas de Visão Computacional, existem diversos softwares no mercado, com diferentes características e compiladores. Devido a sua grande abrangência e por possuir versão aberta para desenvolvimento estudantil, utilizou-se no presente projeto o *Microsoft Visual C++ 2008 Express Edition*.

O *Visual C++* é um produto da IDE da Microsoft usado para as linguagens de programação C, C++ e C++/CLI. O *Visual C++ 2008* fornece um ambiente de desenvolvimento eficaz e flexível para a criação de aplicativos com base no Microsoft Windows e no Microsoft .NET.

O *Visual C++* faz parte de um pacote de Softwares da Microsoft constituindo o Visual Studio. Este pacote se baseia em três pilares para proporcionar melhor experiência para os programadores:

1º. Melhorias na produtividade do desenvolvedor.

2º. Gerenciamento do ciclo de vida do aplicativo.

3º. Utilização das mais recentes tecnologias

3.2.2. Software de Aquisição da Imagem

O software desenvolvido gera estimulos que são aplicados ao olho esquerdo com luz visível (branca, verde, vermelha ou azul) em períodos de tempo pré-determinados de 300 frames (10 segundos), como mostrado na Figura 3.13, enquanto a imagem do olho direito é digitalizada em uma sequência de vídeo, sob iluminação NIR, como é possível verificar na Figura 3.11.

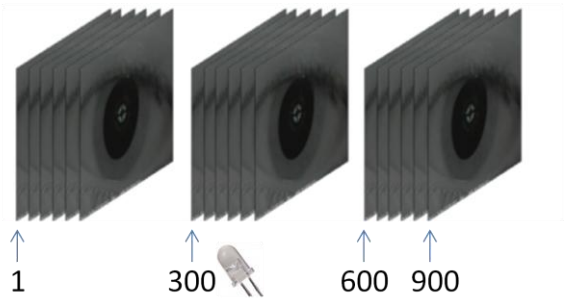

Figura 3.13: Sequência de Frames.

O LED, branco, vermelho, azul ou verde é acionado apenas entre os *frame* 300 e 599, sendo os frames 600 a 900 apenas para checar a acomodação da íris, estes valores foram definidos empiricamente. Para os estudos desejados, realizou-se 4 vídeos, para cada voluntário, com 900 frames cada, a uma taxa de aquisição de imagem de 30

frames/segundo, onde cada um dos vídeos foi acionado uma cor de LED – vermelho, azul, verde e branco.

O *software* de controle do sistema pode ser representado pelo diagrama de blocos da Figura 3.14.

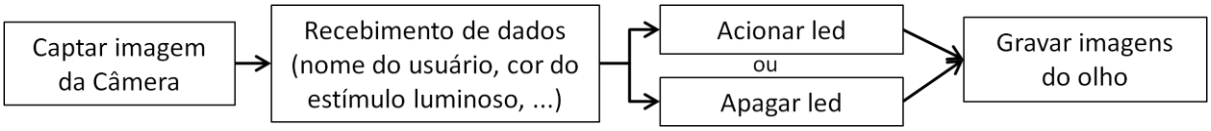

Figura 3.14: Diagrama de blocos do *software* de controle do Sistema.

O *software* desenvolvido faz o controle da câmera, da gravação do vídeo e também do acionamento e desacionamento do led que realiza o estímulo luminoso.

Para realizar o controle da câmera e a gravação do vídeo, utilizou-se funções da biblioteca OpenCV, cvCaptureFromCAM(), cvVideoWriter e cvCreateVideowriter. Já para o controle dos leds que realizam o estímulo luminoso, usou-se linhas de programação em C que controlam a porta LPT1 através do uso da *inpout32.dll* (implementação específica para uso em Windows XP ou Vista).

Ou seja:

- a função *cvCaptureFromCAM()* capta as imagens da câmera *frame* a *frame*;

- a função *cvCreateVideowriter* e *cvVideoWriter* gera o arquivo para gravação do vídeo;

- durante a gravação do vídeo, realiza-se o controle dos leds, acendendo o led no *frame* 300 e o apagando no *frame* 600 – enviando um sinal (baseado no posicionamento dos leds no conector da porta paralela) através da variável

```
oup32 = (oupfuncPtr) GetProcAddress(hLib, "Out32").
```

- e a função *cvVideoWriter* grava os *frames* captados;

As linhas de código em C/C++ são apresentadas no Apêndice B, com seus devidos comentários.

3.2.3. Software de Tratamento de Imagem

Diferente de bases de imagens estáticas, que são capturadas em ambientes controlados, a captura em tempo real, ou tempo de vídeo, oferece obstáculos mais complexos. Os movimentos involuntários, ou não, do olho são constantes, exigindo que a tarefa de pré-processamento de imagem seja bastante específica para descartar *frames* que possam comprometer o método e selecionar apenas os *frames* que sejam adequados.

Basicamente, os problemas que podem levar um *frame* a ser desconsiderado podem ser:

- olhos “serrados” (Figura 3.15 a);

- olhos fechados (Figura 3.15 b);

- olhos “desviados”, angulação inadequada do olhar (Figura 3.15 c);

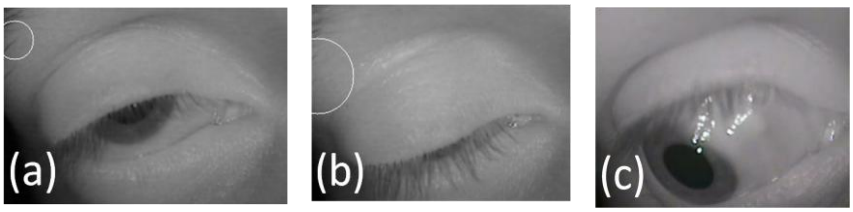

Figura 3.15: Exemplo de imagens que geram dados a serem desconsiderados.

Devido a qualidade das imagens obtidas e principalmente a necessidade de rapidez de processamento (análise em tempo de vídeo), decidiu-se utilizar algorítimos matemáticos para eliminação dos dados que estiverem fora do esperado – ou seja, utilizar um algoritmo que elimina valores que possuem uma variância muito brusca em relação aos seus antecessores (15% maior ou menor que o último valor).

Ou seja:

- implementar um algoritmo que compare o valor de raio calculado com o valor de raio do *frame* anterior, evitando-se variâncias bruscas.

```
SE (ATUAL>=0.85*ANTERIOR)e(ATUAL <=1.15*ANTERIOR){
            ANTERIOR = ATUAL;
            Grava valor ATUAL no arquivo *.txt;
      SE NÃO
            Grava valor ANTERIOR no arquivo *.txt;
```

Após realizar-se a etapa de pré-processamento dos vídeos, eles podem ser processados de maneira a interpretar os seus *frames* separadamente e então levantar a resposta da íris em reação ao estímulo luminoso.

A resposta do olho humano ao estímulo luminoso é verificado na variação do diâmetro da pupila, portanto, esta é a característica que o *software* necessita extrair nas imagens adquiridas.

As medidas dos diâmetros da pupila são feitas em todos os quadros que compõem o vídeo. Cada quadro é tratado como uma imagem estática e para tal, o quadro precisa ser capturado do vídeo original.

Foram implementadas duas funções para essa finalidade. A biblioteca OpenCV fornece uma estrutura chamada *capture* que no caso de captura a partir de vídeos é iniciada pela função *cvCreateFileCapture*. Essa estrutura endereça o fluxo de vídeo (*stream*) que será utilizado para capturar as imagens que compõem cada quadro do vídeo. Com o auxílio da função da Biblioteca OpenCV, a *cvQueryFrame* o quadro é capturado do fluxo de vídeo e alocado em uma estrutura de imagem.

Ou seja:

- *cvCreateFileCapture* inicia o arquivo *.avi de vídeo gravado no *software* de Aquisição da Imagem;

- *cvQueryFrame* interpreta o arquivo *.avi como um ponteiro, o separando *frame* a *frame.*

A estrutura de dados básica que é utilizada para as imagens é chamada de *IplImage*. Nesta estrutura, os dados sobre cada pixel estão armazenados na forma de um vetor e todas as informações necessárias sobre a imagem como tamanho, profundidade de bits, tipo de dado utilizado estão presentes na mesma.

Uma vez capturado um quadro pode-se passar para o processamento e cálcular o valor do diâmetro da pupila.

Cálculo do diâmetro da pupila

O principal movimento que ocorre no globo ocular com a alteração de iluminação é o movimento de contração e dilatação da pupila. A função da pupila é de trabalhar de maneira análoga ao diafragma de uma máquina fotográfica controlando a quantidade de luz que adentra o olho. Além disso, a pupila também tem a função de regular a profundidade de foco (FÖRSTER, GROSS e HÖRING, 2005).

Para focalizar objetos distantes o seu diâmetro diminui, e para objetos próximos o diâmetro aumenta, de maneira involuntária. No entanto, neste trabalho é considerado apenas o movimento que ocorre com alterações na iluminação. Assim, o método proposto apenas avalia a variação da pupila em função da iluminação.

Para o cálculo do diâmetro da pupila, utilizou-se a Transformada Circular de Hough. Anteriormente converte-se as imagens para a escala de cinza através de uma conversão RGB – níveis de cinza, e então melhora-se sua qualidade com a função *Smooth* (descritos no Capítulo 2).

A Transformada Circular de Hough utiliza diversos parâmetros para a detecção de círculos, sendo os valores mínimos e máximos de *Threshold* cruciais para a realização da transformada. Empiricamente, chegou-se aos valores de *Threshold* para a detecção de círculos das imagens aquisitadas.

A TH possui como resposta um vetor com 3 informações, coordenadas cartesianas x e y do centro da circunferência e o raio do círculo (p[0], p[1] e p[2] respectivamente). Para o presente projeto, a saída utilizada para o obter o diâmetro da pupila foi p[2] (valor do raio do círculo detectado). Para converter p[2] e obter o valor correto em milímetros, torna-se necessário realizar uma conversão dos valores obtidos. A conversão foi realizada utilizando uma imagem base que possui um círculo de 10mm de diâmetro, como mostrado na Figura 3.16 e a partir dela calculado a conversão de todos os outros valores, com a Equação 3.1.

$$Diâmetro = \frac{\text{Raio\_Hough} \times 10}{19{,}21 \times 3}$$

Equação 3.1

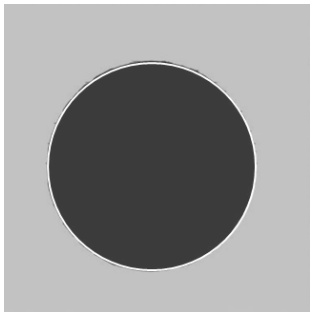

Figura 3.16: Detecção do círculo de 10mm.

Ou seja:

- capturou-se a imagem da figura de um círculo de 10mm na mesma posição na qual o olho do usuário é posicionado;

- realizou-se a TH para detecção do círculo e conseqüente cálculo do raio do mesmo;

- calculou-se a Equação 3.1 para conversão dos valores – conversão para a escala em milímetros;

Todos os dados obtidos são gravados em um arquivo de texto (*.txt) para posterior análise gráfica acerca da variação do diâmetro pupilar no tempo, em resposta aos estímulos luminosos. Com isso pode-se avaliar se existe um padrão de resposta para cada usuário. Este instrumento também pode ser usado para melhor previsão da área cirúrgica ideal dos olhos em casos de cirurgia refrativa, como os pupilômetros fazem.

As linhas de código em C/C++ do software que realiza a interpretação das imagens do olho humano e o conseqüente cálculo do diâmetro da pupila, estão apresentadas no Apêndice A, com seus devidos comentários. A Figura 3.17 mostra a detecção da pupila, com o desenho de um círculo calculado pela TH.

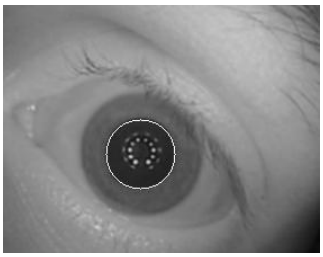

Figura 3.17: Exemplo de cálculo do diâmetro da pupila pela Transformada de Hough.

CAPÍTULO 4

4. RESULTADOS E CONCLUSÕES

Para testar e mostrar a eficiência do sistema desenvolvido foram utilizados vinte diferentes filmes dos olhos de 5 voluntários - a Tabela 4.1 mostra as características dos voluntários. A cada quadro do filme o diâmetro da pupila foi calculado e armazenado em arquivos textos. Como foram utilizados filmes com taxa de quadros de 30 frames por segundo, com 900 frames cada, cada filme possui 30 segundos e cada arquivo texto 900 valores de diâmetro da pupila. Com os dados do arquivo texto, foram realizadas análises gráficas utilizando o *software* Microsoft Excel 2007.

Tabela 4.1: Características dos voluntários analisados.

	SEXO	IDADE	COR DO OLHO	DISTURBIO VISUAL
Voluntário 1	Masculino	22	verde	não possui
Voluntário 2	Masculino	23	marrom escuro	0,75' astigmatismo
Voluntário 3	Masculino	23	marrom	0,5' miopia
Voluntário 4	Feminino	22	marrom claro	1,5' miopia e 1,25' astigmatismo
Voluntário 5	Feminino	22	marrom	0,75' astigmatismo

Baseando-se nas características utilizadas por Da Costa (2009), utilizou-se como pontos de diferenciação três características da pupila, durante sua contração e dilatação através de estímulos luminosos:

1) Diâmetro da pupila
2) Tempo para contração/dilatação da pupila
3) Taxa de contração/dilatação da pupila

O diâmetro foi levantado diretamente do software desenvolvido, como mostrado no Capítulo 3. E o tempo para contração/dilatação da pupila e a taxa de contração/dilatação da pupila foram calculados da seguinte maneira:

- Calculou-se a média do diâmetro da pupila a cada 20 frames dos vídeos;

- A partir da análise dos valores do diâmetro obteve-se os valores de diâmetro máximo e mínimo, e o valor do *frame* na qual estes valores são encontrados, como mostra a Figura 4.1;

- calculou-se a variação temporal (subtraindo-se o valor do *frame* na qual tem-se o diâmetro mínimo pelo valor do *frame* na qual tem-se o diâmetro máximo) na escala de *frames* (lembrando-se que cada 30 frames representam 1 segundo);

- calculou-se a taxa de contração/dilatação da pupila, através da tangente dos gráficos, ou seja, variação do diâmetro dividido pela variação temporal, como mostra a Equação 4.1.

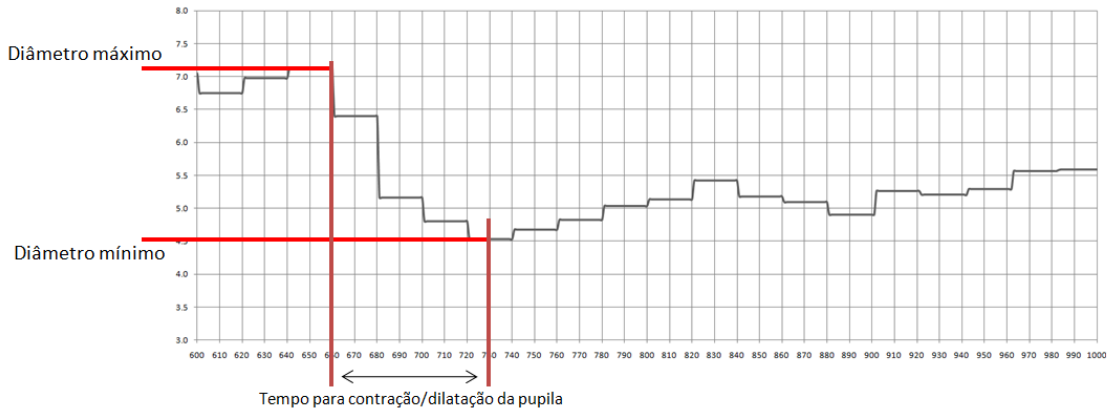

Figura 4.1: Exemplo de análise gráfica realizada.

$$Taxa = \frac{Diâmetro\ máx - Diâmetro\ mín}{\Delta tempo}$$

Equação 4.1

A valor do diâmetro da pupila de cada usuário foi obtido e estes passaram a compor tabelas e a partir dessas tabelas criados gráficos, como mostrados da Figura 4.2 à Figura 4.26.

VOLUNTÁRIO 1

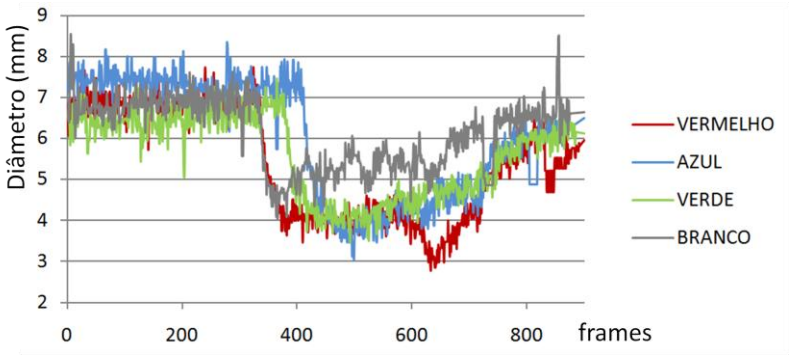

Figura 4.2: Diâmetro da pupila em resposta aos 4 estímulos - Voluntário 1.

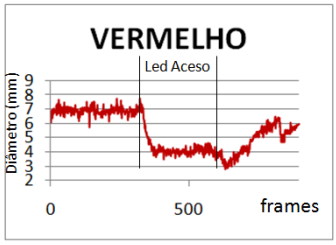

Figura 4.3: Diâmetro da pupila em resposta ao Led Vermelho - Voluntário 1.

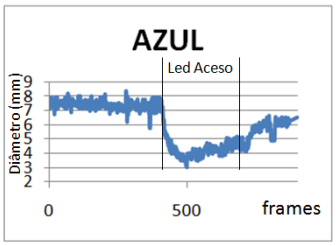

Figura 4.4: Diâmetro da pupila em resposta ao Led Azul - Voluntário 1.

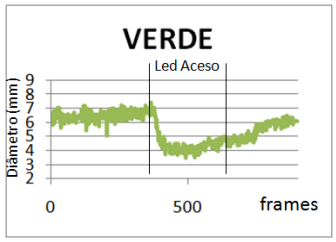

Figura 4.5: Diâmetro da pupila em resposta ao Led Verde - Voluntário 1.

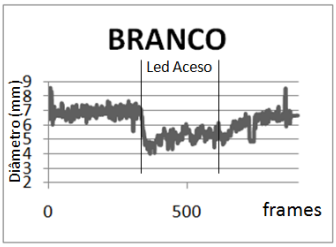

Figura 4.6: Diâmetro da pupila em resposta ao Led Branco - Voluntário 1.

VOLUNTÁRIO 2

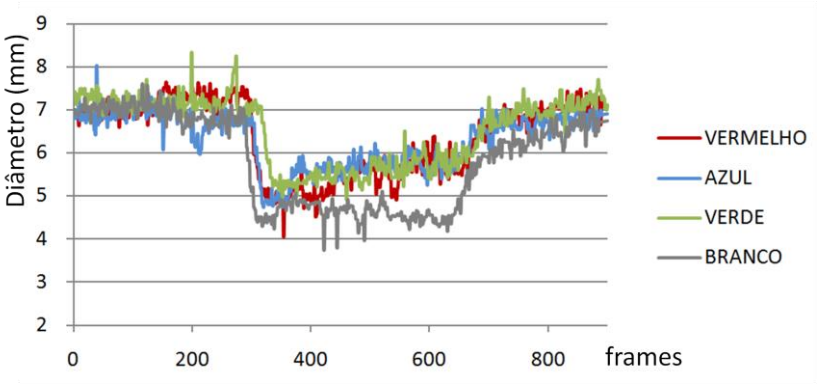

Figura 4.7: Diâmetro da pupila em resposta aos 4 estímulos - Voluntário 2.

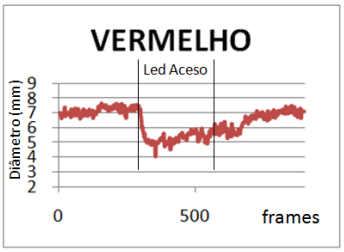

Figura 4.8: Diâmetro da pupila em resposta ao Led Vermelho - Voluntário 2.

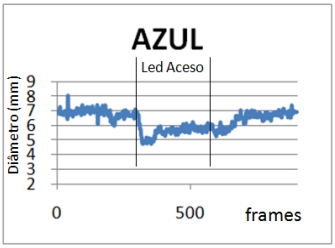

Figura 4.9: Diâmetro da pupila em resposta ao Led Azul - Voluntário 2.

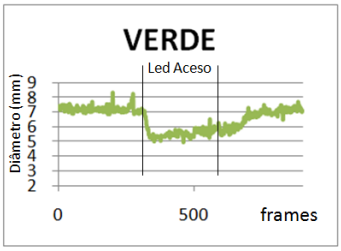

Figura 4.10: Diâmetro da pupila em resposta ao Led Verde - Voluntário 2.

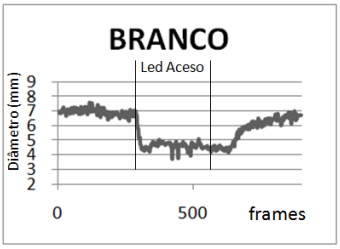

Figura 4.11: Diâmetro da pupila em resposta ao Led Branco - Voluntário 2.

VOLUNTÁRIO 3

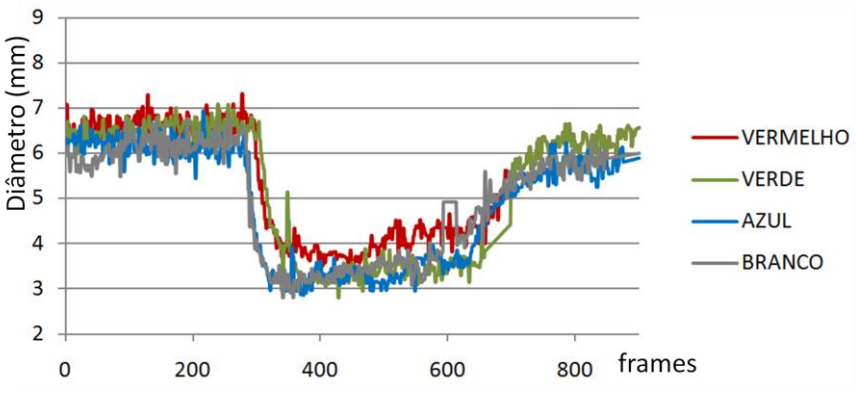

Figura 4.12: Diâmetro da pupila em resposta aos 4 estímulos - Voluntário 3.

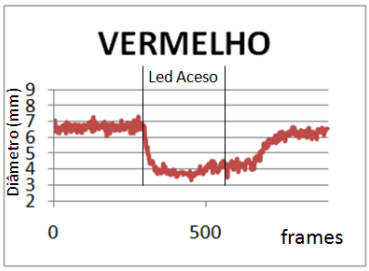

Figura 4.13: Diâmetro da pupila em resposta ao Led Vermelho - Voluntário 3.

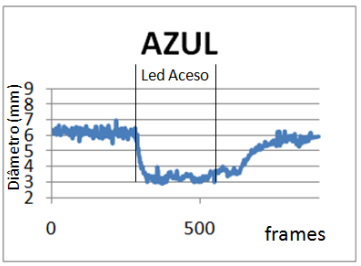

Figura 4.14: Diâmetro da pupila em resposta ao Led Azul - Voluntário 3.

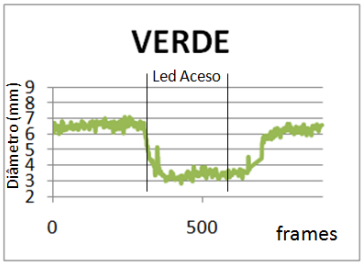

Figura 4.15: Diâmetro da pupila em resposta ao Led Verde - Voluntário 3.

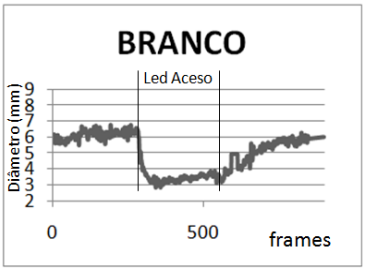

Figura 4.16: Diâmetro da pupila em resposta ao Led Branco - Voluntário 3.

VOLUNTÁRIO 4

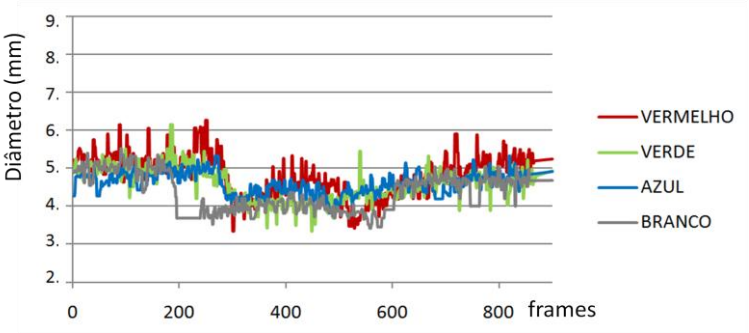

Figura 4.17: Diâmetro da pupila em resposta aos 4 estímulos - Voluntário 4.

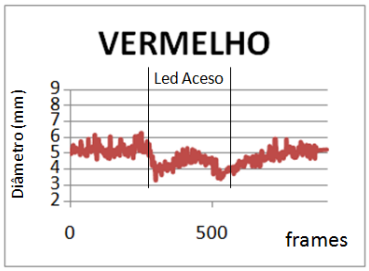

Figura 4.18: Diâmetro da pupila em resposta ao Led Vermelho - Voluntário 4.

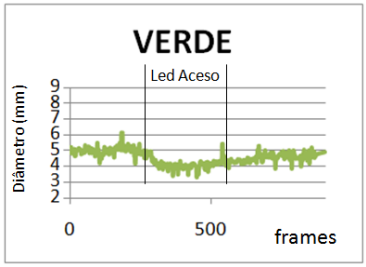

Figura 4.19: Diâmetro da pupila em resposta ao Led Verde - Voluntário 4.

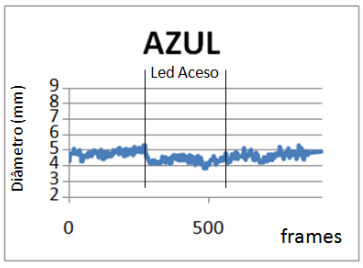

Figura 4.20: Diâmetro da pupila em resposta ao Led Azul - Voluntário 4.

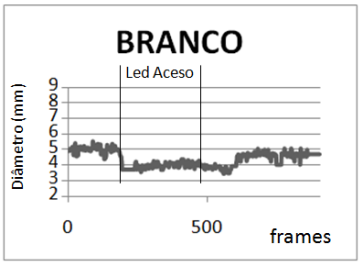

Figura 4.21: Diâmetro da pupila em resposta ao Led Branco - Voluntário 4.

VOLUNTÁRIO 5

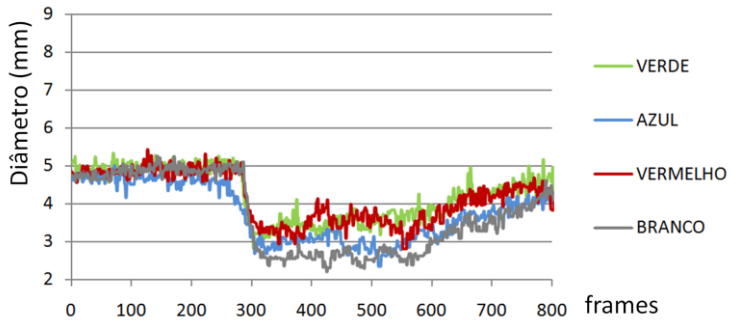

Figura 4.22: Diâmetro da pupila em resposta aos 4 estímulos - Voluntário 5.

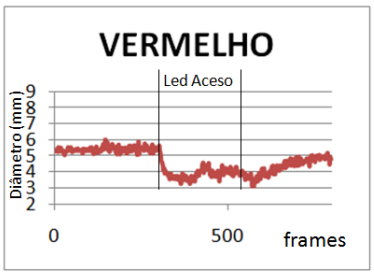

Figura 4.23: Diâmetro da pupila em resposta ao Led Vermelho - Voluntário 5.

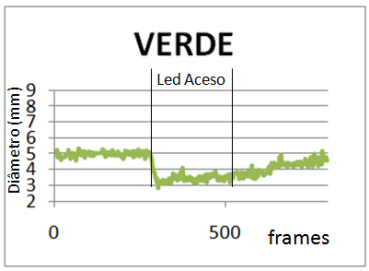

Figura 4.24: Diâmetro da pupila em resposta ao Led Verde - Voluntário 5.

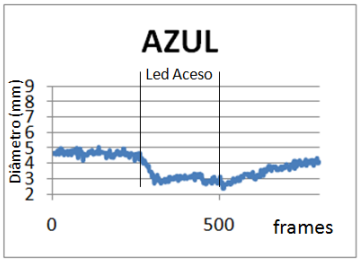

Figura 4.25: Diâmetro da pupila em resposta ao Led Azul - Voluntário 5.

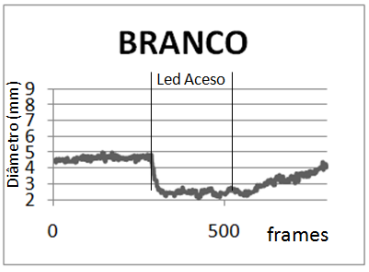

Figura 4.26: Diâmetro da pupila em resposta ao Led Branco - Voluntário 5.

Nesses gráficos é possível observar a resposta da pupila ao estímulo de luz. Os momentos em que o diâmetro da pupila diminuem correspondem aos momentos em que a luz do equipamento de captura de imagens do olho é acionada. Mesmo após desligar a luz que estimula a reação dos olhos, o diâmetro da pupila necessita de um tempo para voltar ao mesmo valor em que estava anteriormente.

Observou-se que cada usuário possui respostas específicas para cada cor, como por exemplo:

- o Voluntário 1 responde com maior diferença entre o estímulo branco e o vermelho (verde e azul similares ao vermelho), como mostra a Figura 4.27;

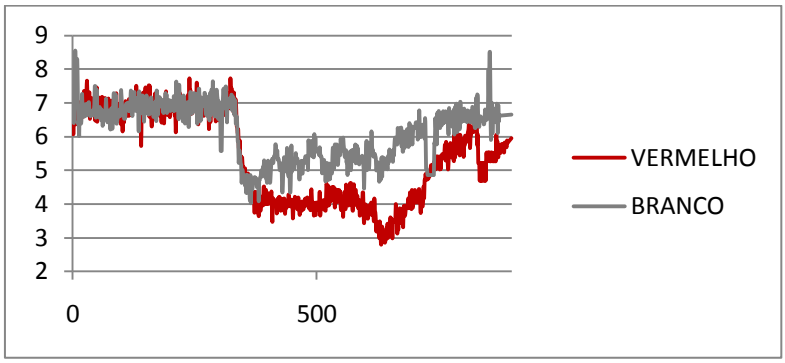

Figura 4.27: Diferença entre o estímulo vermelho e branco para o Voluntário 1.

- o Voluntário 2 responde com maior diferença entre o estímulo branco e verde (azul e vermelho similares ao verde), inversamente ao que ocorre com o Voluntário 1, como mostra a Figura 4.28;

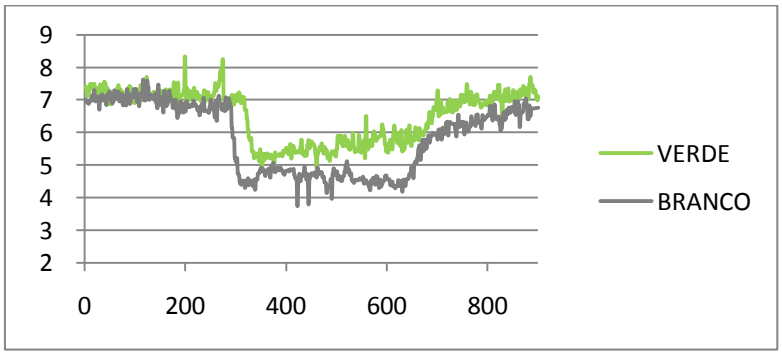

Figura 4.28: Diferença entre o estímulo verde e branco para o Voluntário 2.

- o Voluntário 3 responde com maior diferença entre o estímulo vermelho e verde (azul e branco similares ao verde), como mostra a Figura 4.29;

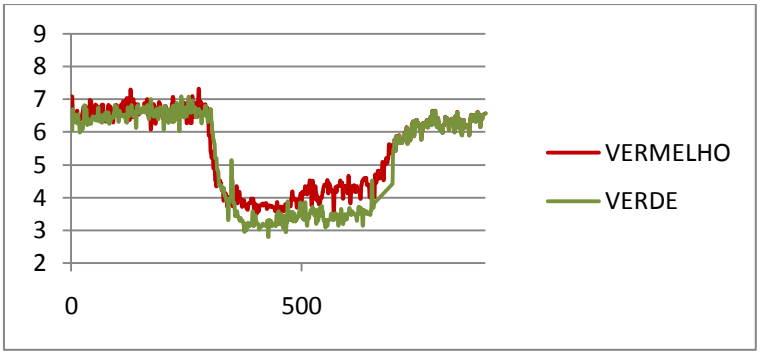

Figura 4.29: Diferença entre o estímulo vermelho e verde para o Voluntário 3.

- o Voluntário 4 não responde com grandes variações entre os diferentes estímulos;

- e o Voluntário 5 responde com maior diferença entre o estímulo branco e vermelho, como mostra a Figura 4.30 - similar ao que ocorre com o Voluntário 2, contudo não há grande similaridade entre as respostas para o estímulo vermelho, verde e azul e os valores do diâmetro são muito diferentes para os dois voluntários, cerca de 2mm de diferença.

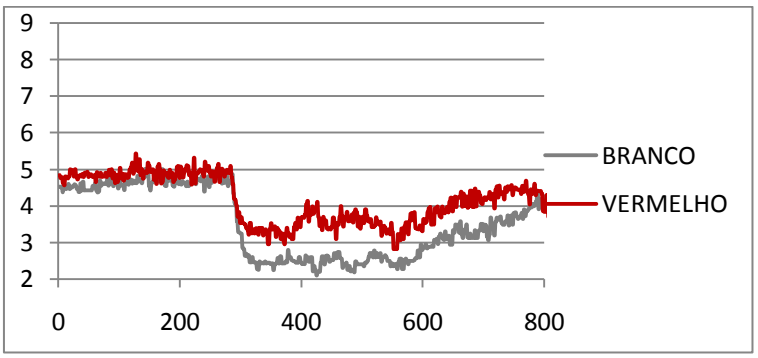

Figura 4.30: Diferença entre o estímulo vermelho e branco para o Voluntário 5.

Os gráficos mostram diferentes padrões de contração da pupila para diferentes estímulos luminosos e/ou de pessoa para pessoa. Isto é um indicativo da possibilidade de se utilizar estes padrões em sistemas de reconhecimento biométrico.

Além do padrão verificado pela análise gráfica do diâmetro da pupila de cada usuário, calculou-se também o tempo para contração/dilatação da pupila(2) e a taxa de contração/dilatação da pupila(3), de maneira a quantificar os padrões analisados graficamente, como mostrado da Tabela 4.2 à Tabela 4.6. O que mostra em números os diferentes padrões de contração da pupila para os diferentes estímulos e pessoas.

Tabela 4.2: Características (2) e (3) para o Voluntário 1.

	VERMELHO	VERDE	AZUL	BRANCO
Diâmetro máx	6.9	6.9	7.4	7.1
Diâmetro min	4.1	4.0	3.5	4.5
Variação do diâmetro	2.8	2.9	3.9	2.6
Tempo para contração/dilatação da pupila(n. frames)	100	190	160	70
Taxa de contração/dilatação da pupila	0.0285	0.0153	0.0241	0.0371

Tabela 4.3: Características (2) e (3) para o Voluntário 2.

	VERMELHO	VERDE	AZUL	BRANCO
Diâmetro máx	7.4	6.8	7.5	6.9
Diâmetro min	4.9	4.9	5.2	4.4
Variação do diâmetro	2.5	2.0	2.2	2.4
Tempo para contração/dilatação da pupila(n. frames)	80	40	90	60
Taxa de contração/dilatação da pupila	0.0310	0.0494	0.0245	0.0403

Tabela 4.4: Características (2) e (3) para o Voluntário 3.

	VERMELHO	VERDE	AZUL	BRANCO
Diâmetro máx	6.7	6.7	6.2	6.4
Diâmetro min	3.8	3.2	3.0	3.1
Variação do diâmetro	2.9	3.5	3.2	3.3
Tempo para contração/dilatação da pupila(n. frames)	150	110	90	70
Taxa de contração/dilatação da pupila	0.0195	0.0319	0.0361	0.0473

Tabela 4.5: Características (2) e (3) para o Voluntário 4.

	VERMELHO	VERDE	AZUL	BRANCO
Diâmetro máx	5.9	5.1	5.0	5.0
Diâmetro min	4.1	4.0	4.2	3.7
Variação do diâmetro	1.8	1.2	0.8	1.4
Tempo para contração/dilatação da pupila(n. frames)	80	240	80	70
Taxa de contração/dilatação da pupila	0.0231	0.0049	0.0099	0.0194

Tabela 4.6: Características (2) e (3) para o Voluntário 5.

	VERMELHO	VERDE	AZUL	BRANCO
Diâmetro máx	5.0	5.1	4.6	5.0
Diâmetro min	3.2	3.2	2.8	2.6
Variação do diâmetro	1.7	1.8	1.8	2.4
Tempo para contração/dilatação da pupila(n. frames)	100	60	100	80
Taxa de contração/dilatação da pupila	0.0173	0.0308	0.0180	0.0305

4.1 Discussão

Os dados obtidos nos mostrou inicialmente que o olho reage diferentemente para cada faixa de comprimento de onda. Esta variação depende também da intensidade luminosa a qual estamos expondo o olho, portanto para análise total do resultado, torna-se necessário quantificar quantos LUX cada LED emite.

O Protótipo mostrou ser eficiente para a presente proposta de trabalho, com grande potencial de melhorias em software (principalmente para aplicações em olhos que possuem pupila com circularidade diferente de 1, não perfeitamente redonda – na qual a TH mostra não ser eficiente) e também em hardware.

Isto porque para usuários com pupilas não simétricas (alteração de forma), pupilas patológicas, pós-operados com corectopia ou discoria, o *software* deste instrumento

reconheceria círculos de maneira indevida. Como o princípio utilizado calcula o melhor raio de curvatura como uma média de todos os raios para 360 graus, o sistema sempre vai partir da hipótese de que a pupila do paciente é aproximadamente circular. Para casos em que isso não é verdade a aproximação pela melhor circunferência pode induzir em aproximações não otimizadas opticamente para o centro de visão e para o diâmetro total da região.

O *software* realiza estímulos luminosos durante 10 segundos e faz a análise para um tempo total de 30 segundos, isto porque o presente projeto deseja-se analisar a resposta da pupila/íris em todos os momentos do estímulo e também conseguir analisar a dilatação da pupila e não apenas a contração, e se o estímulo fosse inferior a 10 segundos, tería-se dados insuficientes. Contudo, para uso em sistemas de reconhecimento biométrico, torna-se necessário estudar estes valores e os dados necessários para caracterizar o usuário e então reduzí-los ao menor tempo possível, de maneira a tornar-se viável o uso do mesmo.

Apesar da eficiência do sistema desenvolvido, não há vídeos e nem dados suficientes para que seja feita alguma conclusão estatística sobre o comportamento da pupila em resposta aos estímulos. Os testes realizados permitem apenas sugerir possíveis padrões que, por ventura, possam existir no comportamento pupilar. Para chegar a conclusões, diversos testes devem ser feitos com dados e vídeos preparados de forma correta.

4.2 Conclusão

Após análise dos resultados, verificamos a eficiência do sistema desenvolvido, assim como sua importância e potencialidade para uso em estudos que comprovem padrões de resposta do olho para diferentes estímulos luminosos e/ou de pessoa para pessoa. Podendo posteriormente utilizar estes padrões para uso em sistemas de reconhecimento biométrico.

Os objetivos deste trabalho foram atingidos. Uma aplicação capaz de identificar e medir o diâmetro da pupila foi desenvolvida. A aplicação pode ser utilizada para processar vídeos com imagens do olho humano e gerar informações para posterior análise. Porém, diversas melhorias ainda são bem vindas.

Os dados que são obtidos pela execução dessa aplicação podem indicar padrões e comportamentos anômalos de uma pupila. Porém, conclusões sobre o significado desses dados não podem ser feitas com base apenas na observação estatística e carecem de estudos de diversas áreas além da computação.

4.3 Sugestão de Trabalhos Futuros

Propostas para melhoria/uso do sistema:

- Adicionar sensores de intensidade e comprimento de onda, de maneira a nos responder exatamente qual a relação entre o espectro de intensidade do estímulo luminoso e a resposta do sistema ocular;

- E/ou utilizar uma segunda Câmera para captar a intensidade e comprimento de onda da iluminação através de seu sensor CCD – calibrando-o com espectrômetos;

- Utilizar câmeras com foco auto ajustável, de maneira a fornecer imagens de melhor qualidade para todos os tipos de rosto e olho;

- Existem estudos que levantam outras características como influências para a resposta da íris, como por exemplo a cor da mesma. Alguns autores relacionam a cor da íris como fator importante para resposta em relação ao estímulo luminoso. Portanto, um ponto importantíssimo para os trabalhos que usarão o sistema desenvolvido para a análise da resposta da íris, seria aumentar a quantidade de pessoas analisadas e com isto, responder com maior clareza qual a real diferença de resposta da íris e qual a relação com as cores de olhos. Além disso, seria necessário verificar a diferença devido a disturbios visuais (miopia, astigmatismo e hipermetropia);

- Utilizar o sistema para a área médica (como por exemplo um pupilômetro). Possam servir ao médico como mais uma técnica no diagnóstico de patologias do olho e também no planejamento personalizado de cirurgias refrativas, ou em outros casos onde seja necessário o conhecimento preciso do diâmetro da pupila para condições adversas de luminosidade;

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, C. A. R. Seminário Oftalmológico J. Britto. **Neuroftalmologia**, 2002. Disponivel em: <http://www.fm.usp.br/oftalmo/neurof_2.php>. Acesso em: Junho 2010.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV - Computer Vision with the OpenCV Library**. First Edition. ed. United States: O'Reilly Media, Inc, 2008.

CARVALHO, L. A. V. D.; JUNIOR, A. P. Resultados preliminares de um sistema computadorizado e estereoscópico para pupilometria in vivo. **Arquivos Brasileiros de Oftalmologia**, v. 71, p. no.6, nov. 2008.

CARVALHO, P. C. et al. Instituto de Matemática Pura e Aplicada. **Visão Computacional, Imageamento, Computação Gráfica**. Disponivel em: <http://milenio.impa.br/novo/portugues/areas_visao.htm>. Acesso em: abr. 2010.

DA COSTA, R. M. **Nova abordagem para reconhecimento biométrico baseado em características dinâmicas da íris humana.** Escola de Engenharia de São Carlos da Universidade de São Paulo. São Carlos. 2009.

DA COSTA, R.; GONZAGA, A. Extraction and selection of dynamic features of the human iris. **Sibgrapi 2009 (XXII Brazilian Symposium on Computer Graphics and Image Processing)**, Rio de Janeiro, p. 202-208, Outubro 2009.

DAUGMAN, J. How iris recognition works. **International Conference on Image Processing**, v. 1, p. 33 - 36, 2004.

DAUGMAN, J. Probing the Uniqueness and Randomness of IrisCodes: Results From 200 Billion Iris Pair Comparisons. **Proceedings of the IEEE**, v. 94, n. 11, November 2006.

FÖRSTER, S.; GROSS, H. H.; HÖRING, L. Extended depth of focus as a process of pupil manipulation. **Proceedings of SPIE**, 2005.

JAMUNDÁ, T. **Reconhecimento de Formas:** A Transformada de Hough. Seminário Visão Computacional - CPGCC/UFSC. [S.l.]: [s.n.]. 2000.

JANES, R. **Estudo sobre sistemas de segurança em instalações elétricas automatizadas**. Escola Politécnica da Universidade de São Paulo. São Paulo, p. 35. 2009.

KANSKI, J. J. **Clinical Ophthalmology:** A Systematic Approach. 6th. ed. [S.l.]: [s.n.], 2007.

KO, J.; Y., G.; J, Y. Iris Recognition Using Cumulative Sum Based Change Analysis. In: _____ **International Signal Processing and Communications**. [S.l.]: [s.n.], 2006. p. 275 - 278.

LYU, S.; FARID, H. **Detecting Hidden Messages Using Higher-Order**. Dartmouth College, USA. Hanover. 2003.

MILLER, N. R. The autonomic nervous system: pupillary function, accomodation and lacrimation. **Clinical Neuro-Ophthalmology**, Baltimore, 1985. 385 - 556.

NEGIN, M. et al. An Iris Biometric System for Public and Personal Use. **Computer IEEE Press**, v. 33, p. 70-75, Feb 2000.

REÍLLO, S. R. Identificación biométrica y su union con las tarjetas inteligentes. Disponivel em: <http://www.revistasic.com/revista39/pdf_39/SIC_39_agora.PDF>. Acesso em: March 2010.

STURGES, B. K. **Neuro-ophthalmology:** The Visible Nervous System. 2nd Annual Veterinary Neurology Symposium. University of California, Davis, USA: [s.n.]. 2005.

TYSON, J. **How Parallel Ports Work**, 2001. Disponivel em: <http://www.vasa.abo.fi/itintro/How%20Parallel%20Ports%20Work.pdf>. Acesso em: Fevereiro 2010.

WILDES, R. P. Automated iris recognition: An emerging biometric technology. **Proceedings of the IEEE**, v. 85, n. 9, p. 1348–1363, September 1997.

APÊNDICE A - FUNÇÃO PARA DETECÇÃO DE CIRCULO E GRAVAR DADOS

```
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>
#include <math.h>
#include <stdio.h>

void main (void)
{       //Gera as variáveis
        CvCapture* capture;
        IplImage* frame;
        CvSize imgSize;
        imgSize.width = 320;
        imgSize.height = 240;
        IplImage* gray_frame = cvCreateImage( imgSize, IPL_DEPTH_8U, 1);
        double reg = 0;
        int j = 0;

        // Carrega o arquivo .avi e .txt(os arquivos devem estar na mesma pasta que o Projeto)
        capture = cvCreateFileCapture("VIDEO.avi" );
        FILE *f = fopen ("PLANILHA.txt", "wt");

        cvNamedWindow( "Example", CV_WINDOW_AUTOSIZE );
        printf ("DIÂMETRO DA PUPILA\n");

        while(1) {
                // pega o .avi como um ponteiro - frames
                frame = cvQueryFrame( capture );

                if( !frame ) break; // Verifica erro ao carregar o vídeo

                // Tratamento da imagem
                cvSmooth(frame,frame);
                cvCvtColor(frame, gray_frame, CV_RGB2GRAY);

        //_____________________________________DET CIRCULO_____________________________________//
                CvMemStorage* storage = cvCreateMemStorage(0);
                if(gray_frame->origin != IPL_ORIGIN_TL){cvFlip(gray_frame, gray_frame, 0);}

                CvSeq* results = cvHoughCircles( gray_frame,
                                storage,CV_HOUGH_GRADIENT,1,gray_frame->width,100,1,10,50);
                //(image,void* circle_storage,int method,double dp,double min_dist,
                double param1 = 100,double param2 = 300,int min_radius = 0,int max_radius = 0

                printf ("\n N.%i\n\n", j);

                // Calculo dos valores do vetor calculado na Transf de Hough e
                // Impressão do circulo na imagem
                for( int i = 0; i < results->total; i++ ) {
                        float* p = (float*) cvGetSeqElem( results, i );
                        CvPoint pt = cvPoint( cvRound( p[0] ), cvRound( p[1] ) );
                        cvCircle(gray_frame,pt,cvRound( p[2] ),CV_RGB(0xff,0xff,0xff));

                        //Calculo para conversão para o valor real do diâmetro da pupila
                        p[2] = (p[2]*10)/(19.21*3);

                //_____ROTINA PARA GRAVAR DADOS E EVITAR USO DE VALORES NÃO REAIS_______//
                                if (j==0){
                                        reg = p[2];
                                        printf ("\n%f\n\n", p[2]);
                                        fprintf (f, "%.4f\n", p[2]);}
                                else{
                                        if ((p[2]>=0.8*reg)&(p[2]<=1.2*reg)){
                                                reg = p[2];
                                                printf ("\n%f\n\n", p[2]);
                                                fprintf (f, "%.4f\n", p[2]);}
                                        else
                                                printf ("\n%f\n\n", reg);
                                                fprintf (f,"%.4f\n", reg);}
                //_____________________________________________________________________//
                }
        //_____________________________________________________________________________________//
```

```
                cvShowImage( "Example", gray_frame);

                cvWaitKey(0);

                j++;
                char c = cvWaitKey(33);
                //ESC key ASCII=27 - para gerar o break da execução do loop
                if( c == 27 ) break;
        }
        //Livra a memória q tinha sido alocada e fecha o arquivo .txt
        fclose (f);
        cvReleaseImage( &gray_frame );
        cvReleaseCapture( &capture );
        cvDestroyWindow( "Example" );
}
```

APÊNDICE B - FUNÇÃO PARA CAPTAR IMAGEM DA CAM E GRAVAR UM .AVI

```
#include <cvcam.h>
#include <cv.h>
#include <cvaux.h>
#include <cxcore.h>
#include <highgui.h>
#include <conio.h>
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <time.h>

// Para enviar dados para a Paralela.
typedef short _stdcall (*inpfuncPtr)(short portaddr);
typedef void _stdcall (*oupfuncPtr)(short portaddr, short datum);

int main(){
//_____________________________CAPA_____________________________//
        IplImage* capa = NULL;
        capa = cvLoadImage("capa_prog.jpg",-1);
        cvNamedWindow( "TCC Software", CV_WINDOW_AUTOSIZE );
        cvShowImage( "TCC Software", capa);
        cvWaitKey(0);
        cvReleaseImage( &capa );
        cvDestroyWindow("TCC Software");
        printf("              TRABALHO DE CONCLUSAO DE CURSO - 2010 \n\n");
//_____________________________________________________________//
    // Dados para a PARALELA.
    HINSTANCE hLib;
    inpfuncPtr inp32;
    oupfuncPtr oup32;
    // Para carregar a dll. Esta dll precisa estar na mesma pasta do programa
    // ou você deve especificar o caminho onde ele se encontra. Dados para a PARALELA.
    hLib = LoadLibrary("inpout32.dll");
    // Para criar um 'ponteiro' para a porta paralela.
    oup32 = (oupfuncPtr) GetProcAddress(hLib, "Out32");
    (oup32)(0x378,0x00);
    IplImage* tmp_frame = NULL;
    CvCapture* cap = NULL;
    IplImage *gravaFrame;

    cvcamSelectCamera(0);

    cap = cvCaptureFromCAM(0);

    if (!cap)
       {
       printf("Erro ao conectar com o dispositivo.");
       getch();
       exit(0);
       }
    tmp_frame = cvQueryFrame(cap);

    if(!tmp_frame){
        printf("Erro ao Ler o dispositivo. \n");
       exit(0);
        }

    char usuario[100];
    char led[1];
    int cor;
    //Capta o nome do usuário para gravar o arquivo
    printf(" Informe o nome do usuario: ");
    scanf("%s", usuario);
    //Gravar o arquivo com o nome digitado e na pasta correta
    strcat(usuario, ".avi");
    //Capta o tipo de LED desejado
    printf("\n 1- WHITE / 2- RED / 3- GREEN / 4- BLUE ? ");
    scanf("%s", led);
    cor = led[1];
    if (cor==1){cor=0x01;}
    if (cor==2){cor=0x02;}
    if (cor==3){cor=0x03;}
```

```
    if (cor==4){cor=0x04;}

CvVideoWriter*video1=cvCreateVideoWriter(usuario,CV_FOURCC('D','I','V','X'),30,cvSize(320,240));

    cvNamedWindow("Video", 1);
    time_t t, t1, t2, t3, t4;
    int fr1, fr2, fr3, fr4;
    char texto[100];
    int status=0;
    CvFont font, font0;
    cvInitFont(&font, CV_FONT_HERSHEY_SIMPLEX, 0.35, 0.35, 0, 1);
    cvInitFont(&font0, CV_FONT_HERSHEY_SIMPLEX, 0.3, 0.3, 0, 1);

    printf("Tecle <Esc> para comecar a gravar...");
    for( int fr = 1;tmp_frame; tmp_frame = cvQueryFrame(cap), fr++ )
    {
        cvShowImage("Video", tmp_frame);
        int k = cvWaitKey(1);
        if( k == 27 ) break;
        };

    time(&t);
    time(&t1);
    fr1 = 1;
    printf("\n Comecou: %s\n", ctime(&t));
    printf("\n (Pressione ESC para finalizar a qualquer momento) %s\n", ctime(&t));
    (oup32)(0x378,0x00);

    for( int fr = 1;tmp_frame; tmp_frame = cvQueryFrame(cap), fr++ )
    {
         time(&t);
         if (fr == 300){
               time(&t2);
               fr2 = fr;
               printf("\n   Ligou: %s   Frame: %d\n", ctime(&t), fr);
               status = 1;
               (oup32)(0x378,cor); //Liga LED desejado
                     }
         if (fr == 600){
               time(&t3);
               fr3 = fr;
               printf("\nDesligou: %s   Frame: %d\n", ctime(&t), fr);
               status = 0;
               (oup32)(0x378,0x00);
           }
        if( fr >= 900 ) {
               time(&t4);
               fr4 = fr;

        sprintf(texto,"Usuario: %s", usuario);
        cvPutText(tmp_frame, texto, cvPoint(5, 450), &font, cvScalar(255, 255, 255));
        sprintf(texto,"Hora Inicial: %s", ctime(&t1));
        cvPutText(tmp_frame, texto, cvPoint(5, 410), &font, cvScalar(255, 255, 255));
        sprintf(texto,"Frame Inicial: %d - Luz Desligada", fr1);
        cvPutText(tmp_frame, texto, cvPoint(5, 390), &font, cvScalar(255, 255, 255));

        sprintf(texto,"Hora Liga: %s", ctime(&t2));
        cvPutText(tmp_frame, texto, cvPoint(5, 350), &font, cvScalar(255, 255, 255));
        sprintf(texto,"Frame Liga: %d - Luz Ligada", fr2);
        cvPutText(tmp_frame, texto, cvPoint(5, 330), &font, cvScalar(255, 255, 255));

        sprintf(texto,"Hora Desliga: %s", ctime(&t3));
        cvPutText(tmp_frame, texto, cvPoint(5, 290), &font, cvScalar(255, 255, 255));
        sprintf(texto,"Frame Desliga: %d - Luz Desligada", fr3);
        cvPutText(tmp_frame, texto, cvPoint(5, 270), &font, cvScalar(255, 255, 255));

        sprintf(texto,"Hora Final: %s", ctime(&t4));
        cvPutText(tmp_frame, texto, cvPoint(5, 230), &font, cvScalar(255, 255, 255));
        sprintf(texto,"Frame Final: %d", fr4);
        cvPutText(tmp_frame, texto, cvPoint(5, 210), &font, cvScalar(255, 255, 255));

        printf("\n  Acabou: %s\n", ctime(&t));
        }
        if (status == 0){
        sprintf(texto,"Sem Luz - %s - Frame: %d - Usuario: %s", ctime(&t), fr, usuario);
        }
        else{
```

```
          sprintf(texto,"Com Luz - %s - Frame: %d - Usuario: %s", ctime(&t), fr, usuario);
            }
         cvPutText(tmp_frame, texto, cvPoint(5, 5), &font0, cvScalar(255, 255, 255));

        gravaFrame = cvCloneImage(tmp_frame);
        cvWriteFrame(video1, gravaFrame);

         cvShowImage("Video", tmp_frame);
         int k = cvWaitKey(1);
        if( k == 27 ) break;
        if( fr == 900 ) break;
         }

         cvWaitKey(0);
         cvReleaseVideoWriter(&video1);
         cvDestroyWindow("Video" );
         cvReleaseCapture(&cap);
         (oup32)(0x378,0x00);
}
```