

**UNIVERSIDADE REGIONAL DE BLUMENAU**  
**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**  
**CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO**

**CONTROLAR O PONTEIRO DO MOUSE UTILIZANDO O  
MOVIMENTO DE UM OLHO CAPTURADO ATRAVÉS DE  
UMA CÂMERA INFRAVERMELHA FIXA**

**EDUARDO HENRIQUE SASSE**

**BLUMENAU**  
**2010**

**2010/2-12**

**EDUARDO HENRIQUE SASSE**

**CONTROLAR O PONTEIRO DO MOUSE UTILIZANDO O  
MOVIMENTO DE UM OLHO CAPTURADO ATRAVÉS DE  
UMA CÂMERA INFRAVERMELHA FIXA**

Trabalho de Conclusão de Curso submetido à  
Universidade Regional de Blumenau para a  
obtenção dos créditos na disciplina Trabalho  
de Conclusão de Curso II do curso de Ciência  
da Computação — Bacharelado.

Prof. Paulo César Rodacki Gomes, Orientador

**CONTROLAR O PONTEIRO DO MOUSE UTILIZANDO O  
MOVIMENTO DE UM OLHO CAPTURADO ATRAVÉS DE  
UMA CÂMERA INFRAVERMELHA FIXA**

Por

**EDUARDO HENRIQUE SASSE**

Trabalho aprovado para obtenção dos créditos  
na disciplina de Trabalho de Conclusão de  
Curso II, pela banca examinadora formada  
por:

Presidente: \_\_\_\_\_  
Prof. Paulo César Rodacki Gomes, Orientador, FURB

Membro: \_\_\_\_\_  
Prof. Dalton Solano dos Reis, M. Sc – FURB

Membro: \_\_\_\_\_  
Prof. Miguel Alexandre Wisintainer, M. Sc – FURB

Blumenau, 13 de dezembro de 2010

Dedico este trabalho a minha família, pai, mãe, irmão, a minha namorada e claro a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

## AGRADECIMENTOS

A Deus, pela vida.

À minha família e a minha companheira, pelo apoio recebido durante todo o processo de graduação.

À minha empresa o Grupo Zanotti, que me auxiliou e incentivou durante todo o processo da graduação.

Aos meus amigos, pelos empurrões e cobranças.

À Marcelo Gadotti, pelo apoio do desenvolvimento do dispositivo *flash*.

Ao professor Paulo César Rodacki Gomes, pela orientação e apoio no desenvolvimento deste trabalho.

Se quiser derrubar uma árvore na metade do tempo, passe o dobro do tempo amolando o machado.

Provérbio Chinês

## RESUMO

No presente trabalho é apresentado um meio que permite a interação humano-computador através do movimento dos olhos do usuário. Para tal, utiliza-se um dispositivo *flash* infravermelho e uma *webcam* sensível a radiação infravermelha para o monitoramento dos olhos, de modo que em tempo real seja possível a transferência do movimento de um olho para o ponteiro do mouse. O dispositivo *flash* foi construído utilizando o microcontrolador PIC18F4550 e LEDs infravermelhos de 850nm. Para reconhecimento da região dos olhos foi utilizada a SVM como técnica de aprendizagem de máquina. São utilizadas técnicas de processamento de imagens, binarização de imagens digitais, análise de componentes conexos, histograma de imagens entre outras.

Palavras-chave: Tecnologia assistiva. Microcontrolador. SVM. Imagem digital. Processamento de imagens. Monitoramento dos olhos. Webcam.

## **ABSTRACT**

This work presents a way that allows an human-computer interaction through eye movements. Using a infrared flash device and a webcam, sensitive to infrared radiation, to monitor the eyes, turns possible in real time the transfer of the movement of an eye to the mouse pointer. The flash device was build using the microcontroller PIC18F4550 and infrared LEDs of 850nm. To recognize the eyes region was used the learn machine technique named SVM. Techniques of image processing histogram image analyses, digital image binarization, connected components analysis and others.

Key-words: Assistive Technology. Microcontroller. SVM. Digital image. Image processing. Monitoring of the eyes. Webcam.



## LISTA DE ILUSTRAÇÕES

Figura 1 – Bloco diagrama do método de detecção dos olhos .....	19
Figura 2 – Diagrama de seqüência do método de detecção dos olhos combinado com o filtro de Kalmann.....	20
Figura 3 – (a) Convenção dos eixos para representação de imagens digitais e (b) distribuição de valores .....	21
Figura 4 – (a) Vizinhança de 4; (b) vizinhança diagonal e (c) vizinhança de 8 .....	23
Figura 5 – Vetores de suporte e hiperplano ótimo.....	28
Figura 6 – (a) Seqüência de imagens e (b) estimativas de destino .....	30
Figura 7 – Pupila iluminada sobre radiação infravermelha.....	31
Figura 8 – Esquema conceitual de detecção da pupila .....	31
Figura 9 – (a) Tela principal e (b) detecção de fadiga.....	35
Figura 10 – Câmera utilizada no projeto .....	36
Figura 11 – (a) Acionamento eixo interno; (b) acionamento eixo externo e (c) subtração das imagens.....	37
Figura 12 – Olho sobre iluminação infravermelha.....	37
Figura 13 – (a) EyeTech TM3; (b) instalado em notebook e (c) aclopado ao monitor.....	37
Figura 14 – Diagrama de casos de uso .....	39
Quadro 1 – Caso de uso Selecionar dispositivo de captura de imagem...40	
Quadro 2 – Caso de uso Selecciona dispositivo flash.....40	
Quadro 3 – Caso de uso Iniciar o processo de detecção dos olhos.....41	
Quadro 4 – Caso de uso Iniciar modo debug.....41	
Quadro 5 – Caso de uso seleciona opções de visualização .....	42
Figura 15 – Diagrama de classe do pacote gui .....	43
Figura 16 – Diagrama de classe do pacote control .....	44
Figura 17 – Diagrama de classe do pacote eye.....	45
Figura 18 – Diagrama de classe do pacote rxtx.....	45
Figura 19 – Diagrama de classe do pacote smv .....	46
Figura 20 – Diagrama de classe do pacote video .....	47
Figura 21 – Diagrama de seqüência .....	48
Figura 22 – Configuração do sistema <i>flash</i> .....	49

Figura 23 – Simulação do dispositivo <i>flash</i> no Proteus.....	51
Quadro 6 – Código fonte do programa de controle do dispositivo <i>flash</i> .....	54
Quadro 7 – Bits de configuração .....	54
Figura 24 – Diagrama esquemático do dispositivo <i>flash</i> .....	55
Figura 25 – Diagrama circuito elétrico.....	56
Figura 26 – Dispositivo <i>flash</i> .....	57
Figura 27 – Gravadora PIC serial .....	58
Figura 28 – Gravação do microcontrolador através do WinPic800 .....	58
Figura 29 – Lente com o filtro removido .....	59
Figura 30 – (a) Papel fotográfico e (b) lente com filtro.....	59
Quadro 8 – Código fonte do método <code>setSerialPortOpen</code> .....	61
Quadro 9 – Código fonte do método <code>sendData</code> .....	61
Quadro 10 – Código fonte da classe <code>FlashDeviceRXTX</code> .....	62
Quadro 11 – Código fonte do método <code>process</code> .....	63
Quadro 12 – Código fonte do método <code>run</code> .....	64
Figura 31 – (a) Efeito olhos-vermelhos; (b) olhos escuros e (c) resultado da subtração .....	65
Figura 32 – (a) Imagem após nivelamento e (b) histograma.....	66
Figura 33 – (a) Centro dos componentes conexos e (b) função <code>Find Maxima</code> .....	66
Quadro 13 – Subtração de quadros no método <code>detect</code> .....	67
Quadro 14 – Nivelamento do quadro no método <code>detect</code> .....	67
Quadro 15 – Binarização e ROI no método <code>detect</code> .....	67
Quadro 16 – Encaminhamento ao módulo de reconhecimento no método <code>detect</code> .....	68
Figura 34 – (a) Imagens do olho direito e esquerdo e (b) de outras partes do rosto .....	69
Quadro 17 – Código fonte do método <code>crop</code> .....	69
Quadro 18 – Arquivo de entrada da LibSVM .....	71
Figura 35 – Padronização de dados com <i>svm-scale</i> .....	71
Figura 36 – Execução do <code>svm-train</code> .....	72
Quadro 19 – Arquivo gerado pelo <code>svm-train</code> .....	73
Figura 37 – Execução do <code>svm-predict</code> .....	73
Quadro 20 – Código fonte do método <code>classifyImage</code> .....	74
Quadro 21 – Código fonte do método <code>createSVMNodeTemplate</code> .....	74
Quadro 22 – Código fonte do método <code>classify</code> .....	75
Quadro 23 – Código fonte do método <code>setMouseXY</code> .....	75

Quadro 24 – Código fonte do método <code>run</code> .....	76
Quadro 25 – Método responsável pela detecção do <i>click</i> .....	76
Quadro 26 – Execução do <i>click</i> .....	76
Figura 38 – Tela selecionar dispositivo de captura de imagem.....	77
Figura 39 – Tela selecionar dispositivo de captura de imagem.....	78
Figura 40 – Tela principal do sistema .....	78
Figura 41 – Mensagem de <i>webcam</i> não reconhecida .....	79
Figura 42 – Mensagem de Dispositivo <i>Flash</i> inválido.....	79
Figura 43 – Mensagem de porta COM não reconhecida.....	79
Figura 44 – Mensagem de porta COM não reconhecida.....	79
Figura 45 – Início de processos no modo <i>debug</i> . .....	80
Figura 46 – Execução do <i>click</i> .....	81
Figura 47 – Movimentação do ponteiro MODO <code>EYEMOUSE</code> .....	82
Figura 48 – Tela principal do sistema .....	84
Quadro 27 – Resultados dos testes de reconhecimento.....	84
Figura 49 – Funcionamento do sistema com espera de 200ms .....	86
Figura 50 – Funcionamento do sistema com espera de 175ms .....	86
Figura 51 – Funcionamento do sistema com espera de 100ms .....	86
Figura 52 – Destaque do nariz do usuário devido a proximidade da iluminação.....	87
Figura 53 – Reflexo do <i>flash</i> sobre a lente do óculos.....	87
Figura 54 – Sem reflexo do <i>flash</i> sobre a lente do óculos.....	88
Figura 55 – (a) Iluminação superior; (b) lateral; (c) frontal e (b) traseira.....	89
Figura 56 – Fonte de iluminação interferindo no efeito olhos vermelhos.....	90
Figura 57 – Olhos vermelhos em ambos os quadros .....	90

## LISTA DE TABELAS

Tabela 1 – Processo de obtenção do $\gamma$ .....	72
Tabela 2 – Processamento de imagens .....	83

## LISTA DE SIGLAS

ALU - *Arithmetic Logic Unit*

API – *Application Programming Interface*

CCD - Dispositivo de Carga Acoplada, do Inglês *Charge Coupled Device*

CCS - *Custom Computer Service*

CDC - *Communications Device Class*

EEPROM - *Electrically-Erasable Programmable Read-Only Memory*

FRSS - Filtro Retangular de Seis Segmentos

IR - Infravermelha, do Inglês *InfraRed*

LED – *Light Emitting Diode*

RAM – *Random Access Memory*

RBF – *Radial Basis Function*

RF - Requisito Funcional

RNA - Rede Neural Artificial

RNF - Requisito Não Funcional

ROI – Região de Interesse, do inglês *Region of Interest*

SIE - *Serial Interface Engine*

SVM - Máquinas de Vetores de Suporte, do Inglês *Support Vector Machine*

TA – Tecnologia Assistiva

TIC - Tecnologia de Informação e Comunicação

USB - *Universal Serial Bus*

UV – Ultravioleta

# SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>15</b>
1.1 OBJETIVOS DO TRABALHO .....	16
1.2 ESTRUTURA DO TRABALHO .....	16
<b>2 FUNDAMENTAÇÃO TEÓRICA .....</b>	<b>17</b>
2.1 TECNOLOGIA ASSISTIVA .....	17
2.2 TECNOLOGIAS DE INFORMAÇÃO E COMUNICAÇÃO .....	18
2.3 O MÉTODO DE RASTREAMENTO DOS OLHOS .....	18
2.4 PROCESSAMENTO DE IMAGENS .....	20
2.4.1 Representação de imagem digital.....	21
2.4.2 Histograma de imagem .....	22
2.4.3 Limiarização.....	22
2.4.4 Vizinhança de Pixels .....	23
2.5 VISÃO COMPUTACIONAL .....	23
2.5.1 Aquisição de imagens .....	24
2.5.2 Pré-processamento .....	25
2.6 RECONHECIMENTO DE PADRÕES .....	25
2.6.1 Detecção de face .....	26
2.6.2 Máquinas de Vetores de Suporte.....	27
2.6.3 Filtro de Kalman .....	29
2.7 VISÃO DO OLHO SOBRE ILUMINAÇÃO INFRAVERMELHA .....	30
2.7.1 Direção do olhar.....	31
2.8 MICROCONTROLADORES .....	32
2.8.1 Memória .....	32
2.8.2 ALU .....	33
2.8.3 Temporizadores e contadores.....	33
2.8.4 Interfaces de entrada e saída .....	33
2.8.5 Interrupções.....	34
2.9 C PARA MICROCONTROLADORES.....	34
2.10TRABALHOS CORRELATOS.....	34
2.10.1 Ferramenta para detecção de fadiga em motoristas baseada na monitoração dos olhos.....	34

2.10.2	Projeto Visage .....	35
2.10.3	Deteção e rastreamento dos olhos através de suas propriedades fisiológicas, dinâmicas e aparentes.....	36
2.10.4	EyeTech TM3 .....	37
<b>3</b>	<b>DESENVOLVIMENTO DA FERRAMENTA .....</b>	<b>38</b>
3.1	REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	38
3.2	ESPECIFICAÇÃO .....	38
3.2.1	Diagrama de casos de uso .....	39
3.2.2	Seleciona dispositivo de captura de imagem .....	39
3.2.3	Seleciona dispositivo flash.....	40
3.2.4	Iniciar o processo de deteção dos olhos .....	41
3.2.5	Iniciar modo debug .....	41
3.2.6	Iniciar o controle do ponteiro .....	41
3.2.7	Diagrama de classe.....	42
3.2.8	Diagrama de sequência .....	47
3.3	IMPLEMENTAÇÃO .....	48
3.3.1	Técnicas e ferramentas utilizadas.....	49
3.3.1.1	Configuração do sistema <i>flash</i> .....	49
3.3.1.1.1	Microcontrolador PIC 18F4550 .....	50
3.3.1.1.2	Simulação do circuito com Protheus.....	50
3.3.1.1.3	Compilador CCS C .....	51
3.3.1.1.4	Biblioteca de funções .....	52
3.3.1.1.5	Programa para controle do dispositivo flash.....	53
3.3.1.1.6	Projeto do circuito.....	55
3.3.1.1.7	Circuito elétrico.....	56
3.3.1.1.8	LED infravermelho .....	57
3.3.1.1.9	Gravação do microcontrolador PIC18F4550 .....	57
3.3.1.2	Câmara Infravermelha .....	59
3.3.1.3	Comunicação entre computador e dispositivo flash .....	60
3.3.1.3.1	RXTX.....	60
3.3.1.3.2	Comunicação com RXTX.....	60
3.3.1.4	JMF.....	62
3.3.1.4.1	Codec .....	62
3.3.1.4.2	Controle dos quadros .....	63

3.3.1.5 Processamento de Imagens .....	64
3.3.1.5.1 ImageJ .....	64
3.3.1.5.2 Técnicas utilizadas .....	65
3.3.1.5.3 Método de detecção .....	66
3.3.1.6 Reconhecimento de padrões .....	68
3.3.1.6.1 Treinamento da SVM.....	68
3.3.1.6.2 Extração da região dos olhos .....	69
3.3.1.6.3 LibSVM .....	70
3.3.1.6.4 Escala SVM.....	70
3.3.1.6.5 Processo de treinamento da LibSVM.....	71
3.3.1.6.6 Adaptação do algoritmo de escala .....	73
3.3.1.6.7 Classificação da região dos olhos .....	74
3.3.1.7 Movimento do ponteiro do mouse .....	75
3.3.1.7.1 Movimento .....	75
3.3.1.7.2 Click .....	76
3.3.2 Operacionalidade da implementação .....	77
3.3.2.1 Escolhendo um dispositivo de captura de imagem.....	77
3.3.2.2 Escolhendo a porta de comunicação do dispositivo flash.....	77
3.3.2.3 Recursos da ferramenta.....	79
3.3.2.4 Funcionamento da ferramenta .....	80
3.4 RESULTADOS E DISCUSSÃO .....	82
3.4.1 Tempo de processamento .....	83
3.4.2 Reconhecimento .....	83
3.4.3 Movimento do ponteiro.....	85
3.4.4 Velocidade do LED .....	85
3.4.5 Posicionamento do usuário .....	86
3.4.6 Óculos .....	87
3.4.7 Fontes de Iluminação .....	88
<b>4 CONCLUSÕES.....</b>	<b>91</b>
4.1 EXTENSÕES .....	92
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>93</b>

## 1 INTRODUÇÃO

Novas realidades e novos paradigmas emergem na sociedade moderna. Uma sociedade mais permeável à diversidade questiona seus mecanismos de segregação e vislumbra novos caminhos de inclusão social para todos os indivíduos (DAMASCENO; GALVÃO FILHO, 2008, p. 3).

Segundo Henning e Souza (2006, p. 2), indivíduos com deficiência tem se beneficiado com os avanços tecnológicos disponíveis na atualidade. A presença crescente das TICs tem possibilitado diferentes formas de relacionamento entre esses, além de contribuir fortemente na construção do conhecimento.

Em alguns casos, é o piscar e o deslocamento horizontal e vertical dos olhos que são os últimos remanescentes da mobilidade corporal. Este estado é conhecido na literatura médica como Síndrome do Encarceramento<sup>1</sup>, doença que foi retratada no filme - O Escafandro e a Borboleta - (LE ESCAPHANDRE..., 2007). A película conta a história real do editor chefe da revista Elle francesa, Jean Dominique Bauby, que passou a utilizar o piscar do olho esquerdo para interagir com o mundo, chegando escrever um livro.

Existem diversas causas pelas quais os membros superiores podem estar incapacitados de exercer suas funções como, por exemplo, tetraplegia, distrofia muscular, doenças degenerativas, amputação ou até mesmo a ausência dos membros devido a causas naturais (CAETANO; COSTA, 2006, p. 10).

Diante do exposto, o presente trabalho propõe a construção de uma ferramenta, mais especificamente uma interface de comunicação humano-computador, que permita ao usuário movimentar o ponteiro do *mouse* utilizando um de seus olhos. Será utilizada uma câmera de vídeo para obtenção contínua de imagens dos olhos do usuário.

Com o desenvolvimento desta interface, indivíduos que estejam impossibilitados de movimentar os membros superiores serão capazes de interagir com o computador, podendo utilizar-se de recursos que auxiliem em sua comunicação, aprendizagem e interação social, possibilitando a redação desde pequenas mensagens até longos textos, através da seleção de letras na tela do computador, pelo movimentar e piscar dos olhos.

---

<sup>1</sup> Corpo inteiro paralisado com exceção dos olhos e faculdades mentais perfeitas (SINDROME..., 2010).



## 1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver uma ferramenta que possibilite movimentar o ponteiro do *mouse* através dos movimentos do olho, utilizando imagens capturadas em uma *webcam*.

Os objetivos específicos do trabalho são:

- a) captar um vídeo de um usuário utilizando uma *webcam*;
- b) utilizar o *flash* infravermelho para gerar o efeito olhos vermelhos;
- c) detectar e demarcar o olho;
- d) detectar e demarcar a pupila do olho;
- e) calcular a proporção entre o movimento do olho e do ponteiro do *mouse* e mapear o movimento do olho para o ponteiro do *mouse*;
- f) perceber alteração no estado do olho (aberto ou fechado) para caracterizar o *click* simples.

## 1.2 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos. O segundo capítulo contém a fundamentação teórica necessária para o entendimento do trabalho. Nele são discutidos tópicos relacionados a tecnologias assistivas, processamento de imagens, aquisição de imagens, segmentação de imagem, detecção de bordas, reconhecimento de padrões, microcontroladores e uma técnica para obtenção da pupila e da região dos olhos. Por fim, apresentam-se dois trabalhos correlatos.

O terceiro capítulo trata sobre o desenvolvimento da ferramenta, onde são explanados os principais requisitos do problema trabalhado, a especificação contendo diagramas de caso de uso, classe e seqüência. Também são feitos comentários sobre a implementação abrangendo as técnicas e ferramentas utilizadas, operacionalidade e por fim são comentados os resultados e discussão.

O quarto capítulo refere-se às conclusões do presente trabalho e sugestões para trabalhos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os assuntos e técnicas utilizadas para o desenvolvimento da ferramenta. Nas seções iniciais são apresentadas informações sobre TA e TIC. Em seguida são proporcionados conceitos sobre visão computacional, iluminação dos olhos sobre radiação infravermelha, processamentos de imagens digitais e SVM. Por fim, são descritos alguns trabalhos correlatos ao proposto. São abordadas técnicas de computação gráfica que possibilitam o reconhecimento de faces humanas, a detecção dos olhos e da pupila.

### 2.1 TECNOLOGIA ASSISTIVA

TA é o termo utilizado para identificar todo o arsenal de recursos e serviços que contribuem para proporcionar ou ampliar habilidades funcionais de indivíduos com deficiência (BERSCH, 2008).

Em Damasceno e Galvão Filho (2008, p. 5), TA é definida como qualquer produto, instrumento, estratégia, serviço e prática, utilizado por pessoas com deficiência ou pessoas idosas, sendo especialmente produzidas para prevenir, compensar, aliviar ou neutralizar uma deficiência, tendo em vista melhorar a qualidade de vida do indivíduo.

Segundo Galvão Filho (2009, p. 12), as TAs aparecem como um paradigma inclusivo, onde as limitações encontradas pelo indivíduo são analisadas de uma forma mais ampla, em sua funcionalidade e possibilidades de participação, como resultados não só de suas deficiências individuais, mas também de deficiências e barreiras do seu meio, interpostas pelo ambiente e por realidades e condições sócio-econômicas. A pesquisa e o desenvolvimento devem levar estes fatores em consideração, ao estudar soluções, dispositivos, metodologias, entre outras, que compensem ou reduzam as limitações não só do indivíduo, mas também do seu ambiente físico e social.

## 2.2 TECNOLOGIAS DE INFORMAÇÃO E COMUNICAÇÃO

Segundo Damasceno e Galvão Filho (2008, p. 6), as novas TICs vem se tornando, de forma crescente, importantes instrumentos culturais, tendo em sua utilização um meio concreto de inclusão e interação no mundo.

Ainda segundo Damasceno e Galvão Filho (2008, p. 7), esta afirmação torna-se mais evidente tendo em vista indivíduos portadores de deficiência. Nestes casos, as TICs podem ser utilizadas como TA quando o próprio computador é a ajuda técnica para atingir um determinado objetivo. Por exemplo, o computador utilizado como caderno eletrônico, para o indivíduo que não consegue escrever em um caderno de papel comum. Por outro lado, as TICs são utilizadas por meio de TA quando o objetivo final desejado é a utilização do próprio computador, onde são necessárias determinadas ajudas técnicas que permitam ou facilitem esta tarefa. Por exemplo, adaptações de teclado, de *mouse*, softwares especiais entre outros.

A TIC como sistema auxiliar, como uma prótese para a comunicação, é possivelmente a área na qual esta tenha possibilitado os avanços mais significativos até o presente momento. Em muitos casos, o uso destas tecnologias tem se constituído na única maneira pela qual diversas pessoas podem se comunicar com o mundo exterior.

## 2.3 O MÉTODO DE RASTREAMENTO DOS OLHOS

A detecção em tempo real é o primeiro e mais importante passo na construção de uma interface de comunicação através da direção e movimentos dos olhos. Para suportar este primeiro passo e trabalhar em um ambiente de elevado nível de processamento paralelo, como o atual, os algoritmos de detecção devem ser de baixa complexidade computacional.

Para um robusto rastreamento ocular sob condições variáveis de iluminação e diferentes orientações da face aplica-se o reconhecimento de características dos olhos juntamente com a iluminação infravermelha. Combinando os pontos fortes de diferentes técnicas complementares, a fim de superar as suas deficiências, o método utiliza a iluminação infravermelha para produzir o efeito olhos vermelhos.

O efeito olhos vermelhos e as técnicas de reconhecimento de padrões e classificação são utilizados simultaneamente para a detecção e rastreamento do olhar. O método consiste

em duas partes: detecção e rastreamento

A detecção dos olhos é realizada pelo efeito olhos vermelhos. A ideia é capturar imagens sucessivas alternando iluminação comum com iluminação com luz infravermelha da cena. Assim, em um quadro da cena captura-se o rosto do usuário tendo-se a pupila sob iluminação infravermelha, gerando a pupila clara, e no quadro seguinte quando a iluminação infravermelha é trocada pela comum, a pupila aparece escura. As imagens dos dois quadros são então subtraídas a fim de remover o fundo. Após, aplica-se o nivelamento e posteriormente é feita análise dos componentes conexos para demarcação de regiões de interesse na imagem. Após esse processo é então aplicado o reconhecimento de padrões através de SVM. A Figura 1 apresenta uma visão geral do módulo de detecção.

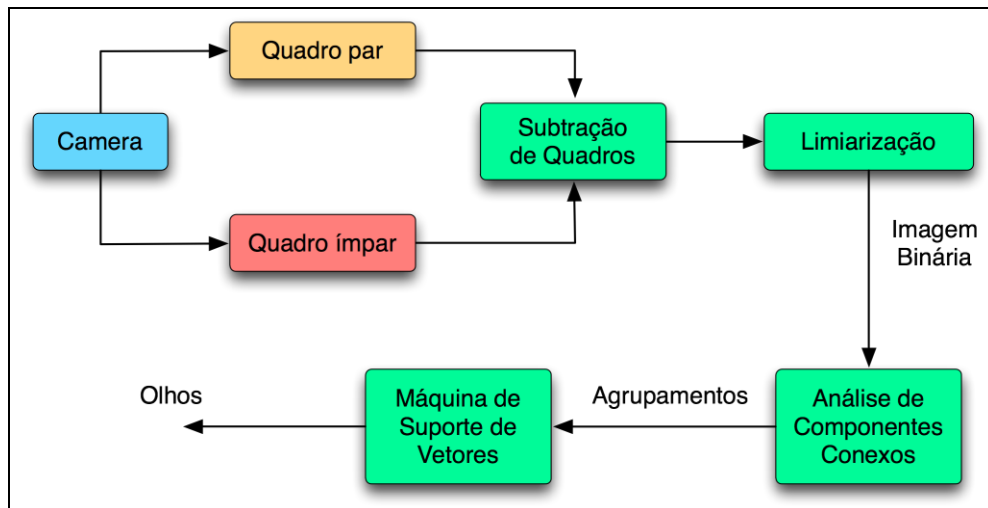


Figura 1 – Bloco diagrama do método de detecção dos olhos

Tendo a detecção dos olhos nos quadros iniciais, os olhos nos quadros subsequentes podem ser monitorados a partir de um quadro para outro. Isto poderia ser feito através da detecção da pupila em cada quadro independentemente, porém este método é do tipo força bruta e tende a desacelerar significativamente a velocidade do rastreamento, dificultando o monitoramento em tempo real, já que é necessário procurar a região de interesse em toda a imagem a cada quadro. Sendo assim, um sistema baseado em predição geralmente é mais eficiente neste caso. A Filtragem de Kalman fornece um mecanismo para fazer esta predição. Apenas em caso de falha, quando ocorre a oclusão das pupilas ou repentinos movimentos de cabeça, todo o algoritmo deve ser novamente executado. A Figura 2 resume o esquema de rastreamento.

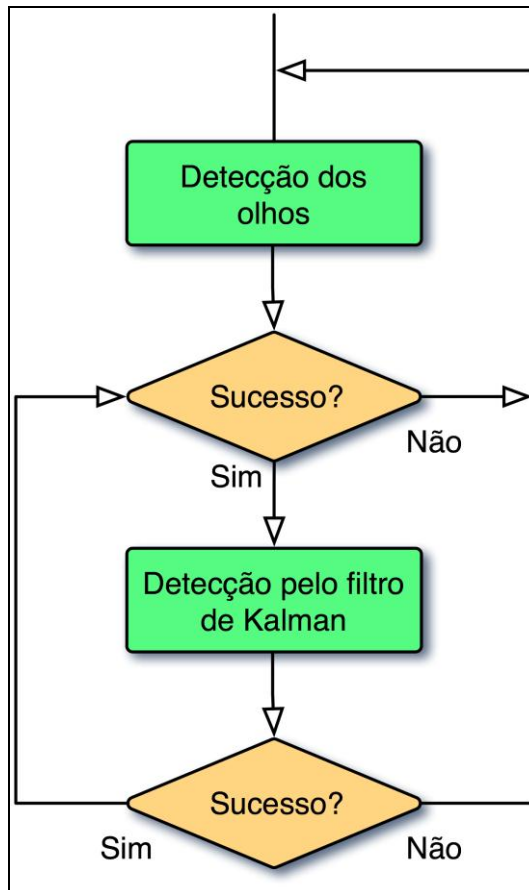


Figura 2 – Diagrama de seqüência do método de detecção dos olhos combinado com o filtro de Kalmann

A iluminação infravermelha utilizando uma fonte próxima a face traz uma série de vantagens ao processo. Dentre eles, pode-se considerar a redução no impacto de diferentes condições de iluminação, incluindo a melhora na qualidade da imagem em condições de pouca iluminação tanto de dia como de noite. Além disso, permite gerar o efeito olhos vermelhos, o que constitui a base do método de detecção.

#### 2.4 PROCESSAMENTO DE IMAGENS

O interesse em métodos de processamento de imagens digitais tem como principais objetivos a melhoria da informação visual para interpretação humana e o processamento de dados de cenas para percepção através dos computadores (GONZALES; WOODS, 2000, p. 1). Um exemplo é a remoção de ruídos ou o melhoramento do contraste da imagem. Segundo Marques Filho e Vieira Neto (1999, p. 23), uma das primeiras aplicações em melhoria da informação visual remonta ao começo do século XIX, onde se buscava formas de aprimorar a

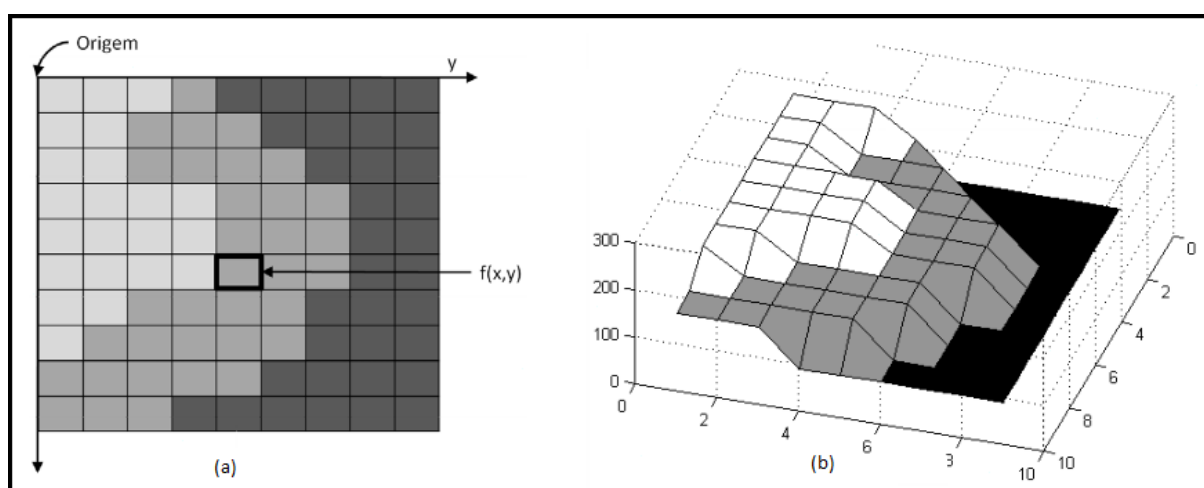
qualidade de impressão de imagens digitalizadas e transmitidas através do sistema Bartlane<sup>2</sup>.

De 1964 aos dias atuais, a área de processamento de imagens vem apresentando crescimento expressivo e suas aplicações permeiam quase todos os ramos da atividade humana, como Medicina, Biologia, Geografia, Sensoriamento Remoto, Geoprocessamento, Meteorologia, dentre outras (MARQUES FILHO; VIEIRA NETO, 1999, p. 24).

Os avanços no poder de processamento e captura da imagem, além da popularização de equipamentos, tem possibilitado o avanço de soluções que utilizam o processamento de imagens, tanto em melhoria da informação como no reconhecimento da informação contida. Por exemplo, sistemas de validação biométrica que utilizam o reconhecimento facial, da impressão digital, retina, entre outros (ROCHA; RAFAEL; SILVA, 2005, p. 6).

#### 2.4.1 Representação de imagem digital

Segundo Gonzales e Woods (2000, p. 4), o termo imagem digital refere-se à “função bidimensional de intensidade de luz  $f(x,y)$ , onde  $x$  e  $y$  denotam as coordenadas espaciais e o valor de  $f$  em qualquer ponto  $(x,y)$  é proporcional ao brilho (ou *níveis de cinza*) da imagem naquele ponto”. Esta representação de suas coordenadas é demonstrada na Figura 3(a). Na Figura 3(b) as regiões planas ou platôs representam as áreas na imagem que possuem diferentes níveis de brilho.



Fonte: adaptado de Paula Junior (2009, p. 3).

Figura 3 – (a) Convenção dos eixos para representação de imagens digitais e (b) distribuição de valores

<sup>2</sup> Sistema Bartlane de transmissão de imagens por cabo submarino entre Londres e Nova Iorque. Os primeiros sistemas datam do início da década de 20, codificavam uma imagem em cinco níveis de intensidade distintos.

### 2.4.2 Histograma de imagem

Segundo Marques Filho e Vieira Neto (1999, p. 75), o histograma de uma imagem é um conjunto de números indicando o percentual de *pixels* de uma imagem que apresentam um determinado nível de cinza. Estes valores são normalmente representados por um gráfico de barras. Através da visualização do histograma de uma imagem obtém-se uma indicação de sua qualidade quanto ao nível de contraste e quanto ao seu brilho médio.

Ainda segundo Marques Filho e Vieira Neto (1999, p. 76), informações quantitativas podem ser extraídas do histograma como o nível de cinza mínimo, médio e máximo, predominância de *pixels* claros ou escuros entre outras. Conclusões de caráter qualitativo necessitam da análise conjunta da imagem que originou o histograma, para que se possa avaliar a qualidade subjetiva global da imagem, presença ou não de ruído entre outras características.

### 2.4.3 Limiarização

Segundo Marques Filho e Vieira Neto (1999, p. 91), o princípio da limiarização consiste em separar as regiões de uma imagem quando esta apresenta duas classes (o fundo e o objeto). Devido ao fato da limiarização produzir uma imagem binária como saída, o processo também é denominado, muitas vezes, binarização. A forma mais simples de limiarização consiste na bipartição do histograma.

Por exemplo, supondo que o histograma corresponda a uma imagem,  $f(x,y)$  composta por objetos iluminados sobre um fundo escuro, de maneira que os *pixels* do objeto e do fundo tenham seus níveis de cinza agrupados em dois grupos dominantes. Uma forma de separar o objeto do fundo é através da seleção de um limiar  $T$  que separe os dois grupos. Então, cada ponto  $(x,y)$  tal que  $f(x,y) > T$  é denominado um ponto do objeto, do contrário é um ponto de fundo (GONZALES; WOODS, 2000, p. 316).

#### 2.4.4 Vizinhaça de Pixels

Segundo Mesquita Filho (2008) a vizinhaça de *pixels* define quais *pixels* podem ser ditos vizinhos de outros *pixels* em uma imagem. Algumas operações são baseadas na suposiçãõ de que determinado objeto é formado por *pixels* vizinhos uns aos outros. Seja  $p$  um *pixel* de uma imagem localizado no ponto, ele pode possuir uma das três vizinhaças apresentadas a seguir:

- vizinhaça de 4: representada por  $N_4(p)$ , é formada por todos os *pixels* que fazem fronteira de borda com  $p$  na horizontal ou na vertical;
- vizinhaça diagonal: representada por  $N_D(p)$ , é formada por todos os *pixels* que fazem fronteira de borda com  $p$  através de suas diagonais;
- vizinhaça de 8: representada por  $N_8(p)$ , é formada pela uniãõ dos *pixels* vizinhos de 4 e vizinhos diagonais a  $p$ .

A representaçãõ visual destas vizinhaças é exibida na Figura 4. Onde na Figura 4(a) sãõ destacados os quatro *pixels* vizinho de  $p$ ; na Figura 4(b) sãõ destacados os *pixels* vizinhos diagonais de  $p$  e na Figura 4(c) sãõ destacados os 8 *pixels* vizinhos de  $p$ .

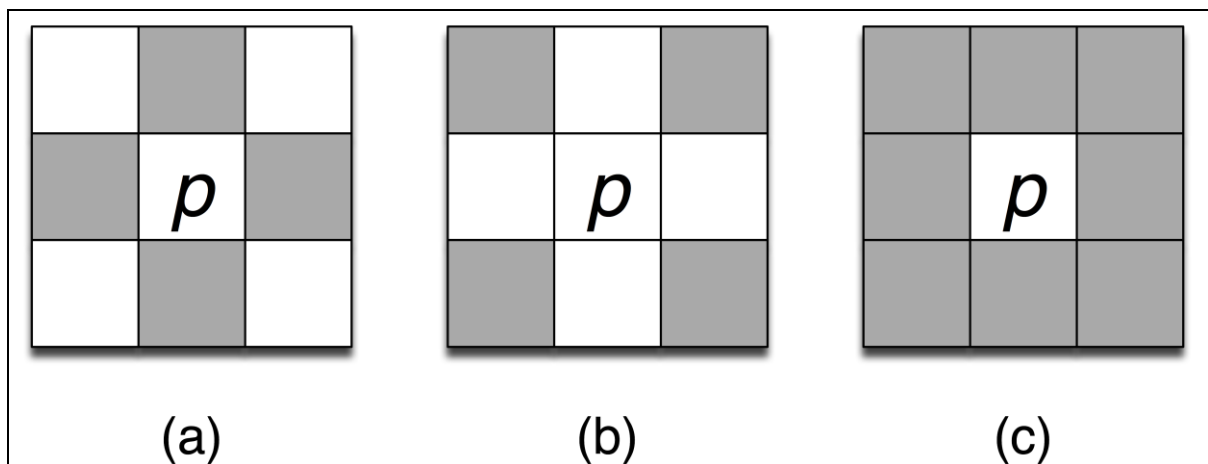


Figura 4 – (a) Vizinhaça de 4; (b) vizinhaça diagonal e (c) vizinhaça de 8

## 2.5 VISÃO COMPUTACIONAL

Segundo Comunello e Wangenheim (2005), define-se visãõ computacional como o “conjunto de métodos e técnicas através dos quais sistemas computacionais podem ser



capazes de interpretar imagens”.

Ainda segundo Comunello e Wangenheim (2005), a interpretação de uma imagem pode ser definida, em termos computacionais, como a transformação de um conjunto de dados digitais representando uma imagem em uma estrutura de dados descrevendo a semântica deste conjunto de dados em um contexto qualquer.

Segundo Tezuka (2009) a visão está relacionada à capacidade que um ser tem de ver e entender o mundo em que habita. A imagem é formada na mente através das organizações físicas, químicas e biológicas dos olhos. O mesmo se sucede na visão computacional, ao invés dos olhos, câmeras são utilizadas para obter a imagem. Sobre estas imagens são aplicadas técnicas computacionais para extrair informações e gerar assim o conhecimento.

Segundo Alves (2005, p. 16), a computação gráfica e o processamento de imagens são duas áreas inerentes à visão computacional. Onde a computação gráfica utiliza informações, como grafos de cena, tipos de materiais, geometria dos objetos, iluminação, entre outras, para gerar imagens a visão computacional faz o caminho contrário, procurando obter informações de cenas a partir de imagens digitais. Para isso quase sempre faz uso de métodos de processamento de imagens. Uma das características mais marcantes do estado da arte em visão computacional é que ainda não existe nenhum modelo genérico de percepção visual passível de ser aplicado na prática.

Atualmente são utilizadas técnicas de aprendizagem de máquina para modelar ou imitar a percepção visual biológica, além da utilização de um conjunto de algoritmos bastante específicos, que são respectivamente responsáveis por realizar tarefas bastante limitadas dentro do processo de interpretação de imagens. Estes algoritmos são divididos em grupos, como filtros de contraste, detectores de bordas de objetos, segmentadores de imagens em regiões, classificadores de texturas e assim por diante (COMUNELLO; WANGENHEIM, 2005).

### 2.5.1 Aquisição de imagens

Segundo Gonzales e Woods (2000, p. 7), dois elementos são necessários para obtenção de uma imagem digital. O primeiro é um dispositivo físico que seja sensível a uma banda do espectro (raios X, ultravioleta, visível, ou banda infravermelha) e que produza uma saída proporcional ao nível de energia recebido. O segundo é o elemento digitalizador, que converte a saída do meio físico para uma forma digital. Pode representar desde um meio sensível à

saída, como o filme de uma câmera fotográfica até um *software* que processe a saída e gerencie as ações a serem realizadas.

Um sistema comum de aquisição de imagem possui uma saída proporcional a intensidade da luz incidente, onde a saída é armazenada em matrizes de estado sólido. A tecnologia usada em sensores de imageamento do estado sólido é baseada em CCD.

Um sensor CCD para captação de imagens é formado por um circuito integrado contendo uma matriz de capacitores acoplados. Os CCDs são utilizados em fotografia digital, imagens de satélites, equipamentos médico-hospitalares (como por exemplo, os endoscópios), e na astronomia (particularmente em fotometria, óptica e espectroscopia UV e técnicas de alta velocidade) (DISPOSITIVO..., 2010).

### 2.5.2 Pré-processamento

A imagem resultante do processo de aquisição pode apresentar diversas imperfeições, tais como presença de pixels ruidosos, contraste e/ou brilho inadequado, caracteres interrompidos ou indevidamente conectados entre outras. A função da etapa de pré-processamento é aprimorar a qualidade da imagem para as etapas subsequentes (GONZALES; WOODS, 2000, p. 9).

As operações efetuadas nesta etapa são ditas de baixo nível porque trabalham diretamente com os valores de intensidade dos pixels, sem nenhum conhecimento sobre quais deles representam a informação desejada. A imagem resultante desta etapa é uma imagem digitalizada de melhor qualidade que a original (GONZALES; WOODS, 2000, p. 9).

## 2.6 RECONHECIMENTO DE PADRÕES

Segundo Baptista, Nogueira e Siqueira (2006) na natureza os padrões se manifestam de diversas maneiras como sons, formas, imagens, cheiros e sabores que são a todo instante percebidos pelos seres humanos, e também outros animais, que interagem com estes padrões de forma extremamente natural. Exemplos disso são as habilidades que o ser humano tem de diferenciar o som do motor de um automóvel do som de uma música, ou ainda, a habilidade

que um animal selvagem tem de distinguir a presa de um predador. A naturalidade inerente a estas habilidades faz com que o ser humano sequer imagine as complexidades cognitivas que estão por trás delas.

Complexidades que se tornam evidentes quando se tenta reproduzi-las artificialmente em um computador, o que há muito desafia a comunidade científica interessada no assunto. O reconhecimento de padrões por computador é uma das mais importantes ferramentas usadas no campo da inteligência de máquina. Atualmente está presente em inúmeras áreas do conhecimento e encontra aplicações diretas em visão computacional.

### 2.6.1 Detecção de face

Alguns métodos de detecção de face surgiram em torno do ano de 1970, onde simples heurísticas eram aplicadas às imagens captadas, porém com algumas restrições. Por exemplo, fundo liso, condição de luz especial e vista frontal. Estes métodos, no entanto, melhoraram ao longo do tempo e tornaram-se mais robustos.

Com grande sucesso como uma aplicação de análise de imagens, a detecção de face passou a ser estudada como uma grande revolução em segurança, tanto ao controle de acesso por sistemas biométricos quanto à segurança contra fraudes. É uma área considerada multidisciplinar, pois acredita-se que o avanço da visão computacional irá fornecer percepções para neurocientistas e psicólogos de como o cérebro humano funciona, e vice versa (RODRIGUES. 2007).

Segundo Gong, McKenna e Psarrou (2005 apud MENEZES, 2009, p. 30) são pelo menos quatro tarefas relacionadas com o reconhecimento de faces:

- a) classificação: consiste na identificação de uma face  $x$  assumindo-se que ela pertence a uma pessoa do conjunto  $F$ , sendo  $F$  o conjunto de todas as classes existentes. Em outras palavras, assumindo-se que  $x$ , um padrão originário de uma face cuja classificação é desconhecida, pode ser classificado como um padrão de alguma classe  $w_j$ , tal que  $w_j \in F$ , a tarefa de classificação consiste em determinar o valor de  $j$ ;
- b) conhecido-desconhecido: objetiva decidir se a face é ou não um membro de  $F$ , ou seja, se  $x$  pode ser classificado como um padrão de alguma classe de  $F$ ;
- c) verificação: dado que a identidade  $w_j$  de uma face  $x$  foi determinada através de outro meio não visual, essa tarefa busca confirmar a identidade dessa pessoa

usando imagens de face, ou seja, confirmar se  $x$  é da classe  $w_j$ . Isso equivale à tarefa conhecido-desconhecido com apenas uma pessoa conhecida dentro da base de dados do conjunto de treinamento;

- d) reconhecimento completo: visa determinar se uma face é de uma classe  $F$ , em caso positivo, determinar sua identidade  $w_j$ .

Os métodos de detecção facial existentes atualmente estão organizados nas seguintes categorias: métodos baseados em características e em imagem.

O primeiro método se destina a encontrar características faciais como, por exemplo, rastros do nariz, olhos, sobrancelhas, lábios, e no fim verificar a sua autenticidade geométrica.

O segundo baseado em imagem procura pela ROI na imagem com uma área específica onde pode se procurar pela face, independentemente da escala.

## 2.6.2 Máquinas de Vetores de Suporte

Segundo Carvalho e Lorena (2005) as SVMs constituem uma técnica de aprendizado que vem recebendo crescente atenção entre as técnicas de aprendizado de máquina. Os resultados da aplicação dessa técnica são comparáveis e muitas vezes superiores aos obtidos por outros algoritmos de aprendizado, como as RNAs. Exemplos de aplicações de sucesso podem ser encontrados em diversos domínios, como na categorização de textos, Bioinformática e na análise de imagens.

Ainda Segundo Carvalho e Lorena (2005) as técnicas empregam um princípio de inferência denominado indução, no qual obtém-se conclusões genéricas a partir de um conjunto particular de exemplos. O aprendizado indutivo pode ser dividido em dois tipos principais: supervisionado e não-supervisionado.

No aprendizado supervisionado tem-se a figura de um supervisor externo, o qual apresenta o conhecimento do ambiente através de um conjunto agrupado de exemplos. O algoritmo extrai a representação do conhecimento a partir desses exemplos. O objetivo é que a representação gerada seja capaz de produzir saídas corretas para novas entradas não apresentadas previamente.

No aprendizado não-supervisionado não há a presença de um supervisor, ou seja, não existem exemplos rotulados. O algoritmo se encarrega de agrupar as entradas segundo uma medida de erro máximo.

A SVM do método de aprendizado supervisionado consiste em definir uma série de pontos em um espaço multidimensional e encontrar um hiperplano ótimo que separe estes pontos em classes diferentes. Ou seja, possua um  $y$  diferente do dado em uma função  $f(x_1, \dots, x_n) = y_e$  onde cada  $x_i$  é chamado vetor de suporte e  $y_e$  é a classe correspondente (RODRIGUES, 2007).

Os valores próximos ao hiperplano são considerados vetores de suporte como pode ser visto na Figura 5. O hiperplano é definido por balancear as distancias entre os vetores positivos e negativos a fim de obter o melhor balanceamento de treino com os dados (RODRIGUES, 2007).

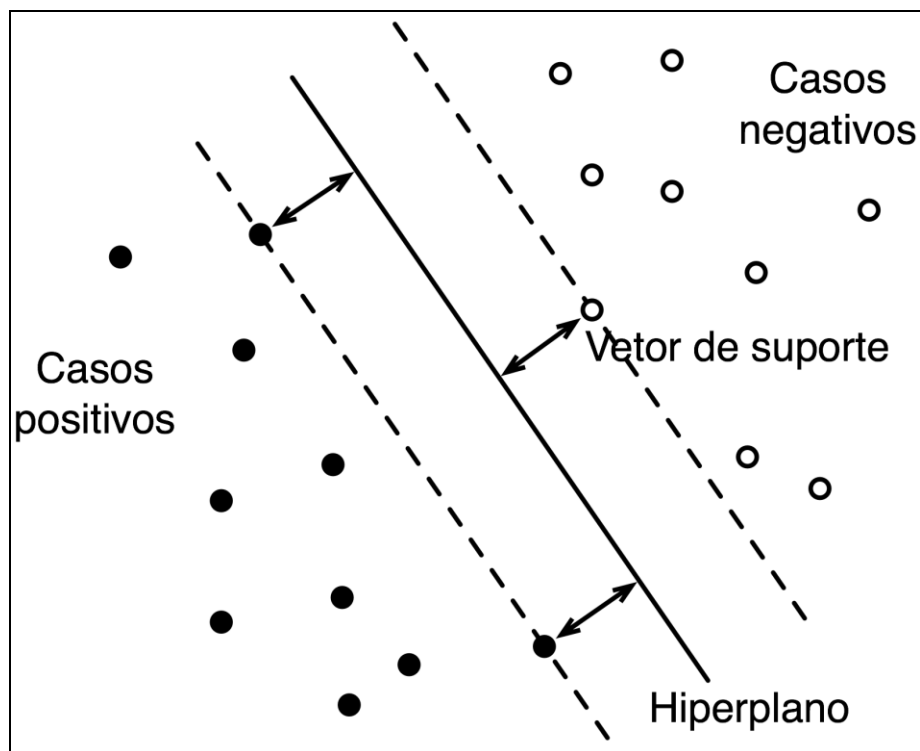


Figura 5 – Vetores de suporte e hiperplano ótimo

Segundo Carvalho e Lorena (2005) em situações reais, é difícil encontrar aplicações cujos dados sejam linearmente separáveis. Isso se deve a diversos fatores, entre eles a presença de ruídos e *outliers* ou à própria natureza do problema, que pode ser não linear. Nestes casos margens rígidas são estendidas para lidar com conjuntos de treinamento mais gerais. Para realizar essa tarefa, permite-se que alguns dados possam violar a restrição rígida com a introdução de variáveis de folga.

Geralmente, o SVM clássico é capaz de separar apenas indivíduos de duas classes distintas. Para a classificação de mais de duas classes, chamado SVM multiclasse, onde são necessárias técnicas de composição de vetores.

Um requisito importante para as SVMs é de lidar com dados imperfeitos, denominados

ruídos. Muitos conjuntos de dados apresentam esse tipo de caso, sendo alguns erros comuns a presença de dados com rótulos e/ou atributos incorretos. Deve-se também minimizar a influência de *outliers* no processo de indução. Os *outliers* são exemplos muito distintos dos demais presentes no conjunto de dados. Esses dados podem ser ruídos ou casos muito particulares, raramente presentes no domínio.

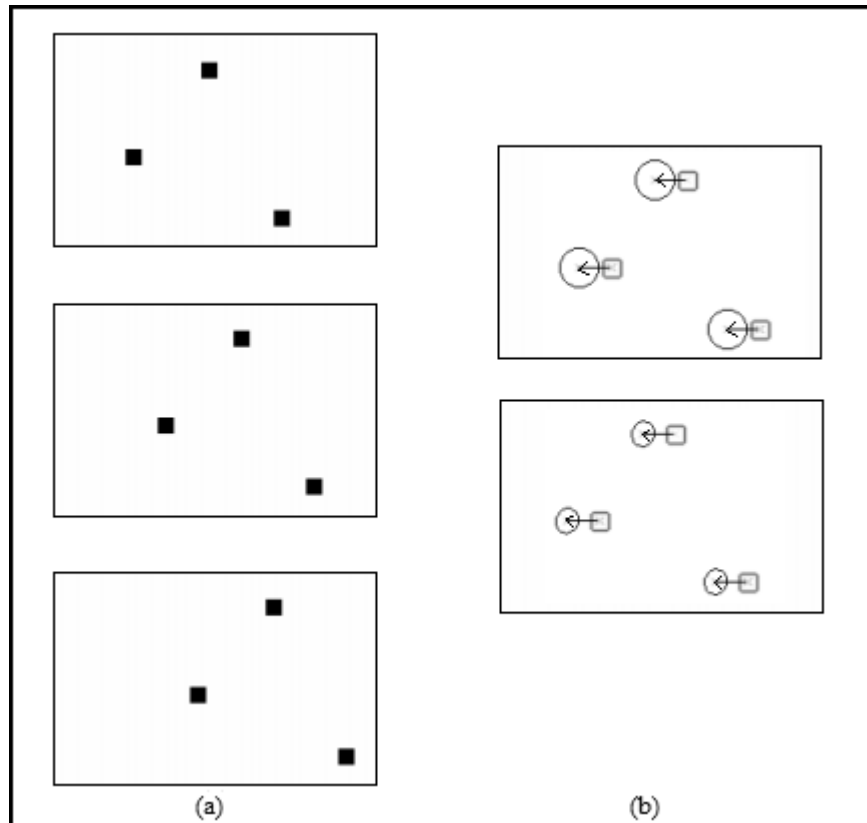
SVM se utiliza de funções denominadas *kernel*. Essas funções são capazes de mapear o conjunto de dados em diferentes espaços, fazendo com que um hiperplano possa ser usado para fazer a separação. Os principais tipos de funções *kernel* são:

- Linear:  $K(x_i, x_j) = x_i^T x_j$ ;
- Polinomial:  $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$ ,  $\gamma > 0$ ;
- Sigmóide:  $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$ ;
- RBF:  $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$ ,  $\gamma > 0$ .

### 2.6.3 Filtro de Kalman

O filtro de Kalman é um conjunto de equações matemáticas que constitui um processo recursivo eficiente de estimação, uma vez que o erro quadrático é minimizado. Através da observação da variável denominada `variável de observação` outra variável, não observável, denominada `variável de estado` pode ser estimada. É um procedimento aplicável quando os modelos estão escritos sob a forma espaço-estado. Além disso, o filtro de Kalman permite a estimação dos parâmetros desconhecidos do modelo através da maximização da verossimilhança via decomposição do erro de previsão (AIUBE, 2005).

A Figura 6 (a) apresenta a seqüência de imagens sintéticas em que se pretende seguir o centro de massa. Resultados da aplicação do filtro de Kalman podem ser vistos respectivamente na Figura 6 (b). As previsões são apresentadas com vetores indicando sua direção da previsão. A incerteza da previsão é definida por um círculo marcando assim seu perímetro (CORREIA; TAVARES, 2005). Os resultados das correções das posições de cada posição de cada ponto estão representados pelo quadrado vazado na Figura 6 (b).



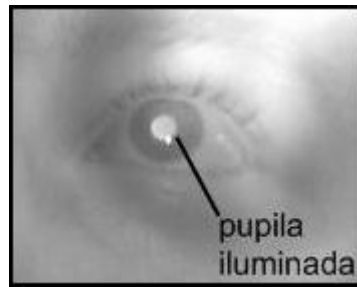
Fonte: adaptado de Tavares e Correia (2005).

Figura 6 – (a) Sequência de imagens e (b) estimativas de destino

## 2.7 VISÃO DO OLHO SOBRE ILUMINAÇÃO INFRAVERMELHA

A radiação infravermelha é uma radiação não ionizante na porção invisível do espectro. Sua região estende-se dos  $3 \times 10^{11}$  Hz até aproximadamente aos  $4 \times 10^{14}$  Hz. O infravermelho está subdividido em três regiões, o próximo (de 780nm até 2500nm), intermédio (de 2500nm até 50000nm) e o longínquo (de 50000nm - 1mm). Por estar fora do espectro detectado pelo olho humano, que se estende dos 400nm até os 700nm, não é visível (RADIÇÃO..., 2010).

Segundo Trindade (2009), ao se iluminar o olho com luz infravermelha a córnea reflete esta luz, produzindo o mesmo efeito conhecido como olhos vermelhos, muito comum em fotos produzidas com o uso de *flash*. Como pode ser visto na Figura 7 este fenômeno produz pupilas com alto brilho em imagens em tons de cinza.



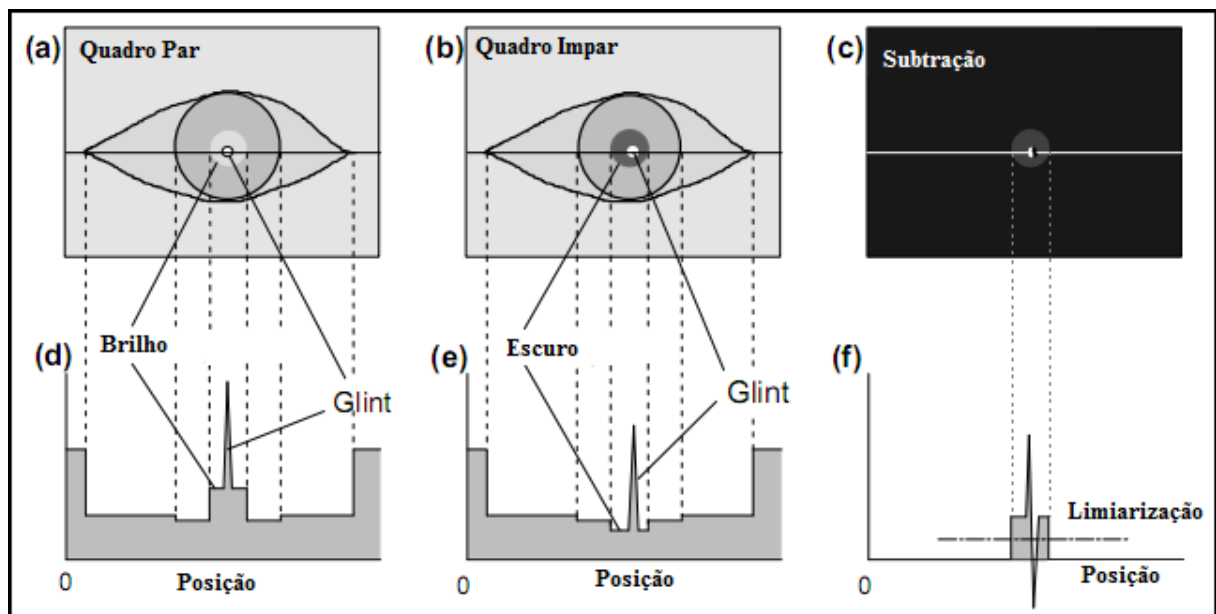
Fonte: Trindade (2009).

Figura 7 – Pupila iluminada sobre radiação infravermelha

### 2.7.1 Direção do olhar

Com o olho sobre uma iluminação infravermelha no mesmo eixo de captura da câmera, a luz que entra na pupila e é refletida na retina e sai através da pupila. A imagem aparece como um disco semi-iluminado contra um fundo escuro, assim chamado de olho claro. Quando a iluminação infravermelha é feita de um eixo externo ao da câmera a pupila aparece escura, temos assim o olho escuro (EBISAWA, 2006).

Em ambas as imagens uma fração da iluminação reflete sobre a superfície da córnea e aparece como um pequeno porém intenso brilho. Este brilho é chamado de *glint* e pode ser visto na Figura 8 (a), (b), (d) e (e) (EBISAWA, 2006).



Fonte: adaptado de Ebsawa (2006).

Figura 8 – Esquema conceitual de detecção da pupila

A posição do *glint* permanece praticamente fixa no campo da imagem, enquanto a



cabeça permanecer estacionária. A posição da pupila muda seguindo a rotação do globo ocular. Assim a direção do olhar pode ser determinada a partir da posição relativa do centro da retina e brilho (EBISAWA, 2006).

Uma vez que as fontes de luz também são pontos fixos, o brilho na córnea do olho pode ser tomado como um ponto de referência, assim, o vetor do brilho para o centro da pupila irá descrever a direção do olhar (MORIMOTO, 1999).

## 2.8 MICROCONTROLADORES

Segundo Santos(2009) o microcontrolador é um circuito integrado composto por um microprocessador e dispositivos periféricos essenciais como memória de programa e de dados; e também periféricos acessórios como interfaces de entrada e saída de dados. Os microcontroladores também são equipados com diversos circuitos eletrônicos tais como conversor analógico digital, temporizadores, comparadores, interfaces de comunicação, geradores de pulsos, entre outros.

São muito populares devido ao seu baixo custo o que os torna soluções viáveis para vários projetos. Por serem programáveis podem ser utilizados nas mais diversas aplicações em sistemas embarcados, como celulares, eletrodomésticos, equipamentos de automação industrial, relógios, alarmes entre outros.

A capacidade de processamento e de armazenamento varia entre os microcontroladores definindo desta forma famílias de processadores com funções semelhantes. Existem famílias que são de linhas compactas, isto é, possuem poucas funções, ocupam menos espaço, consomem menos energia.

Além de terem o baixo custo como vantagem ainda consomem pouca energia, são portáteis, eliminam a necessidade de muitos componentes externos, podem ser reconfigurados com facilidade e necessitam de pouco tempo para o desenvolvimento. Os principais componentes encontrados nos microcontroladores estão citados abaixo.

### 2.8.1 Memória

A memória é um componente essencial em um microcontrolador e é dividida em dois

tipos: memória de programa também conhecida como *flash*, memória de dados também conhecida como RAM e EEPROM.

Na memória de programa são armazenadas as tarefas que o microcontrolador deve executar. Nessa os programas podem modificar configurações, manipular os dispositivos, efetuar comunicação de entrada e saída, executar instruções aritméticas, entre outros. A memória de dados é usada para armazenar resultados e dados que serão usados pelo microcontrolador. Ambas as memórias tem tamanho bem limitado se comparado com outros dispositivos.

### 2.8.2 ALU

A ALU é um módulo do microcontrolador que trabalha com operações lógicas de comparação como maior, menor, igual; operações booleanas como *and*, *or*, *xor*; operações aritméticas como adição, subtração, incrementação, multiplicação e divisão. É considerada a central de processamento.

### 2.8.3 Temporizadores e contadores

São usados para executar rotinas que precisem de noções de tempo ou contadores temporais. Podem gerar pulsos, rotinas em períodos específicos, entre outros. Seus parâmetros são alteráveis, tornando o seu uso programável para uso específico ou geral.

### 2.8.4 Interfaces de entrada e saída

Este tipo de componente é responsável por prover formas de comunicação do microcontrolador com dispositivos externos. É o meio usado para a troca de dados que podem ser transmissão serial e paralela, como o modelo RS232 e USB por exemplo.

### 2.8.5 Interrupções

Este é o componente que controla os pedidos de interrupção. Vários são os dispositivos que estão inclusos dentro de um microcontrolador e a sua maioria dispara pedidos de interrupção o qual pode ser usado para a execução de rotinas específicas.

## 2.9 C PARA MICROCONTROLADORES

Segundo Santos (2009) a linguagem C é uma linguagem de alto nível e estruturada. Sua sintaxe é simples e portátil, isto é, pode ser usado o mesmo programa em várias plataformas. C é muito utilizada para a programação de microcontroladores e também tem o poder de interagir com a plataforma em baixo nível.

As funções desenvolvidas em C são de fácil implementação, pois são utilizados conjuntos de rotinas simples para a execução de rotinas complexas. Os fabricantes de microcontroladores disponibilizam bibliotecas de funções em C para a programação dos microcontroladores.

## 2.10 TRABALHOS CORRELATOS

Dentre os trabalhos pesquisados existem alguns semelhantes ao trabalho proposto. Foram escolhidas as ferramentas descritas por Dattinger (2009), Restom (2006) e Essa, Flickner e Haro (2000). Também se observou a ferramenta comercial EyeTech TM3.

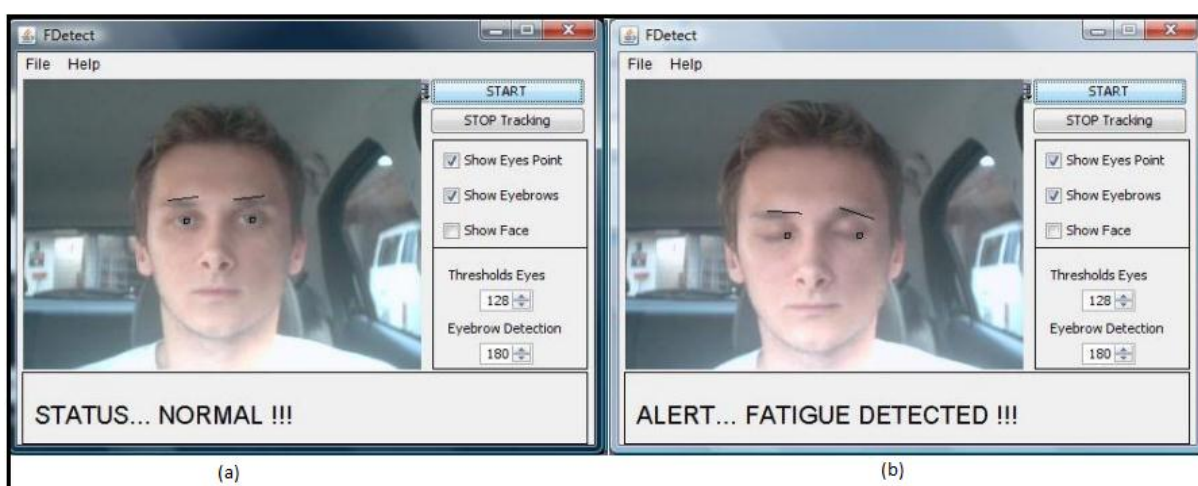
### 2.10.1 Ferramenta para detecção de fadiga em motoristas baseada na monitoração dos olhos

O principal objetivo do trabalho desenvolvido por Dattinger (2009) foi a criação de uma ferramenta para detecção de fadiga em motoristas baseada na monitoração dos olhos, ou

seja, é uma ferramenta em forma de software que realiza a análise dos olhos a partir de uma vídeo capturado do motorista.

O trabalho utilizou-se das classes e da biblioteca do projeto Visage V3.2 (RESTOM, 2006), onde a maioria destas é modificada para se atender o objetivo. Foram utilizados métodos como os de limiarização e transformada de Hough, para a detecção do rosto e da pupila. Além disso, utilizou a técnica FRSS e a heurística presente na SVM, estas se mostraram de grande importância para otimização do processo de detecção da face e da região dos olhos, tornando assim viável o monitoramento da pupila em tempo real.

Como visto na Figura 9(a) a face, os olhos e a pupila são corretamente detectados assim como na Figura 9(b), quando com a detecção do fechamento dos olhos um alerta é apresentado pelo programa.



Fonte: adaptado de Dattinger (2009, p. 58).

Figura 9 – (a) Tela principal e (b) detecção de fadiga

### 2.10.2 Projeto Visage

O projeto de RESTOM (2006) apresenta uma aplicação que é capaz de substituir os tradicionais *mouses* existentes atualmente por uma interface que utiliza o rosto humano como uma nova forma de interação humano-computador.

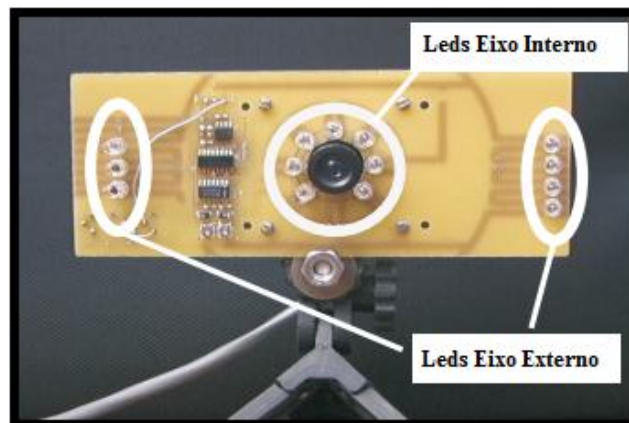
Características faciais como ponta do nariz, olhos e sobrancelhas são detectados e monitorados em tempo real, utilizando-se de uma *webcam* para tal. As coordenadas captadas são utilizadas para fazer o controle do *mouse*. Apertos dos botões do *mouse* são feitos através da detecção da piscada do olho, onde o olho esquerdo equivale ao aperto do botão esquerdo e

o direito equivale ao botão direito.

### 2.10.3 Detecção e rastreamento dos olhos através de suas propriedades fisiológicas, dinâmicas e aparentes

O principal objetivo do trabalho desenvolvido pelos pesquisadores Essa, Flickner e Haro (2000) foi detecção precisa e em tempo real dos olhos e de seus movimentos, com o intuito de precisar a posição da cabeça e o número e intervalo das piscadas, que podem vir a denunciar um estado de sonolência.

Para realizar a captura de forma ágil foi utilizada uma câmera infravermelha em conjunto a um dispositivo que dispara uma *flash* infravermelho com o intuito de gerar o efeito olhos vermelhos como pode ser visto na Figura 10.

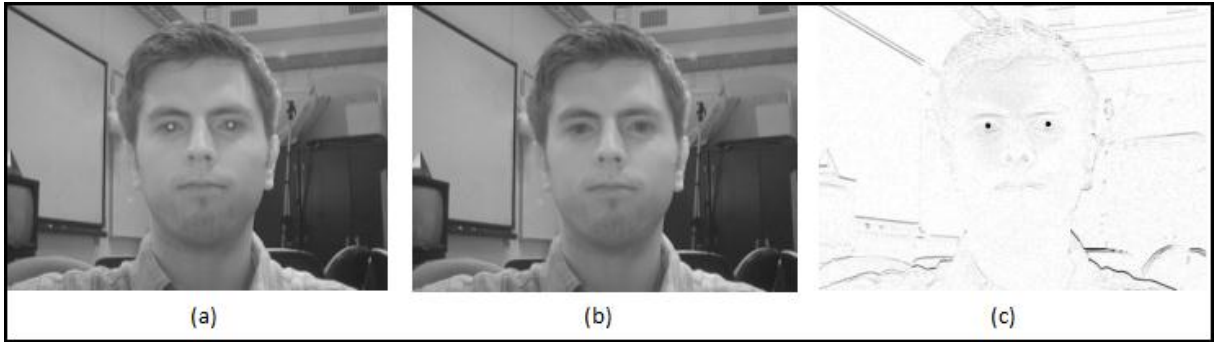


Fonte: adaptado de Essa, Flickner e Haro (2000, p. 3).

Figura 10 – Câmera utilizada no projeto

Dentre as técnicas utilizadas destacam-se a subtração de imagens, como visto na Figura 11(c), onde a Figura 11(a), capturada com o *flash*, produz o efeito olhos vermelhos. Esta imagem é então subtraída da imagem capturada sem este efeito (Figura 11(b)), produzindo uma imagem que destaca a pupila. O circuito responsável pelo chaveamento do *flash* é acionado pelo programa.

O trabalho também utiliza técnicas de processamento de imagens como um algoritmo adaptado de limiarização e a análise do histograma. Entre as técnicas que auxiliam a eliminar as falsas pupilas, encontram-se a APC e o Filtro Kallman.



Fonte: Essa, Flickner e Haro (2000, p. 3).

Figura 11 – (a) Acionamento eixo interno; (b) acionamento eixo externo e (c) subtração das imagens

#### 2.10.4 EyeTech TM3

O EyeTech TM3 é um produto de rastreamento de visão, auxiliando pessoas com deficiência a terem uma maior interatividade com o computador. É um dispositivo de substituição do mouse que permite ao usuário colocar o ponteiro do mouse em qualquer lugar na tela, bastando olhar para o local desejado (EYETECH, 2010).

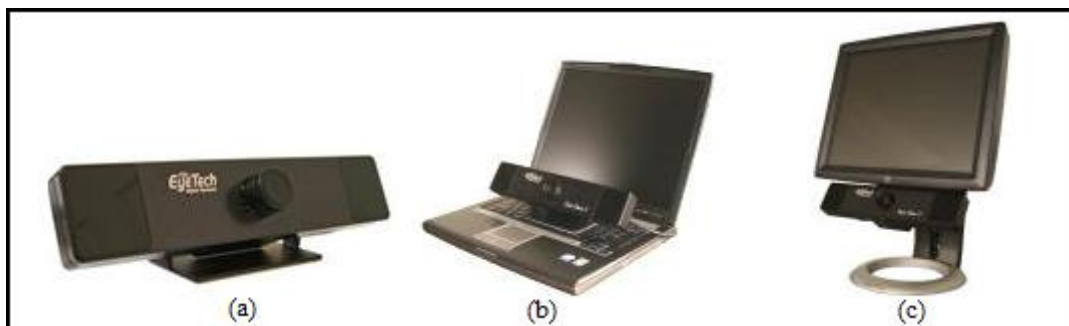
A técnica consiste na utilização de duas fontes de iluminação infravermelha que geram o efeito que pode ser visto na Figura 12.



Fonte: Eyetechn (2010).

Figura 12 – Olho sobre iluminação infravermelha

Pode ser instalado em sistemas operacionais da família Windows e se compromete a realizar todas as tarefas do *mouse*. Custa em torno de U\$7480 e pode ser visto na Figura 13.



Fonte: adaptado de Eyetechn (2010).

Figura 13 – (a) EyeTech TM3; (b) instalado em notebook e (c) acoplado ao monitor

### 3 DESENVOLVIMENTO DA FERRAMENTA

Este capítulo apresentará todo o ciclo de desenvolvimento do aplicativo proposto, contendo os requisitos a serem trabalhados, a especificação, a implementação e por fim os resultados obtidos com o desenvolvimento do aplicativo.

#### 3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O presente trabalho deverá:

- a) captar o vídeo do usuário utilizando uma webcam (RF);
- b) adaptar *webcam* a ser sensível ao espectro infravermelho (RF);
- c) desenvolver o dispositivo *flash* (RF);
- d) identificar a pupila (RF);
- e) verificar se o usuário realizou o *click* simples ao piscar uma vez (RF);
- f) disponibilizar uma interface para permitir o acompanhamento do processo de reconhecimento e movimentação do olho (RF);
- g) implementar a ferramenta utilizando a tecnologia Java (RNF);
- h) utilizar a biblioteca gráfica ImageJ (IMAGEJ, 2010) (RNF);
- i) utilizar a biblioteca de aprendizagem de máquina LibSVM (LIBSVM, 2010) (RNF);
- j) utilizar a porta de comunicação USB para comunicação com o dispositivo *flash* e a biblioteca de comunicação RXTX (RXTX, 2010) (RNF);
- k) utilizar ambiente de programação Eclipse (RNF).

#### 3.2 ESPECIFICAÇÃO

A próxima seção apresenta a especificação do aplicativo e suas funcionalidades, modelada com base em análise orientada a objetos, utilizando-se a UML. Para a análise optou-se pela modelagem dos diagramas de casos de uso, de classe e de sequência através da

ferramenta *Enterprise Architect*. A seguir, são detalhados os diagramas e por consequência as funcionalidades previstas para o aplicativo.

### 3.2.1 Diagrama de casos de uso

A ferramenta apresenta cinco casos de uso conforme a Figura 14, sendo que o primeiro e segundo casos devem ser obrigatoriamente executados pelo usuário. No terceiro caso de uso o usuário pode iniciar o processo de reconhecimento enquanto que no quarto caso de uso é possível acompanhar as etapas deste processo. No quinto caso, o usuário obtém como resultado o movimento do ponteiro a partir do movimento dos olhos.

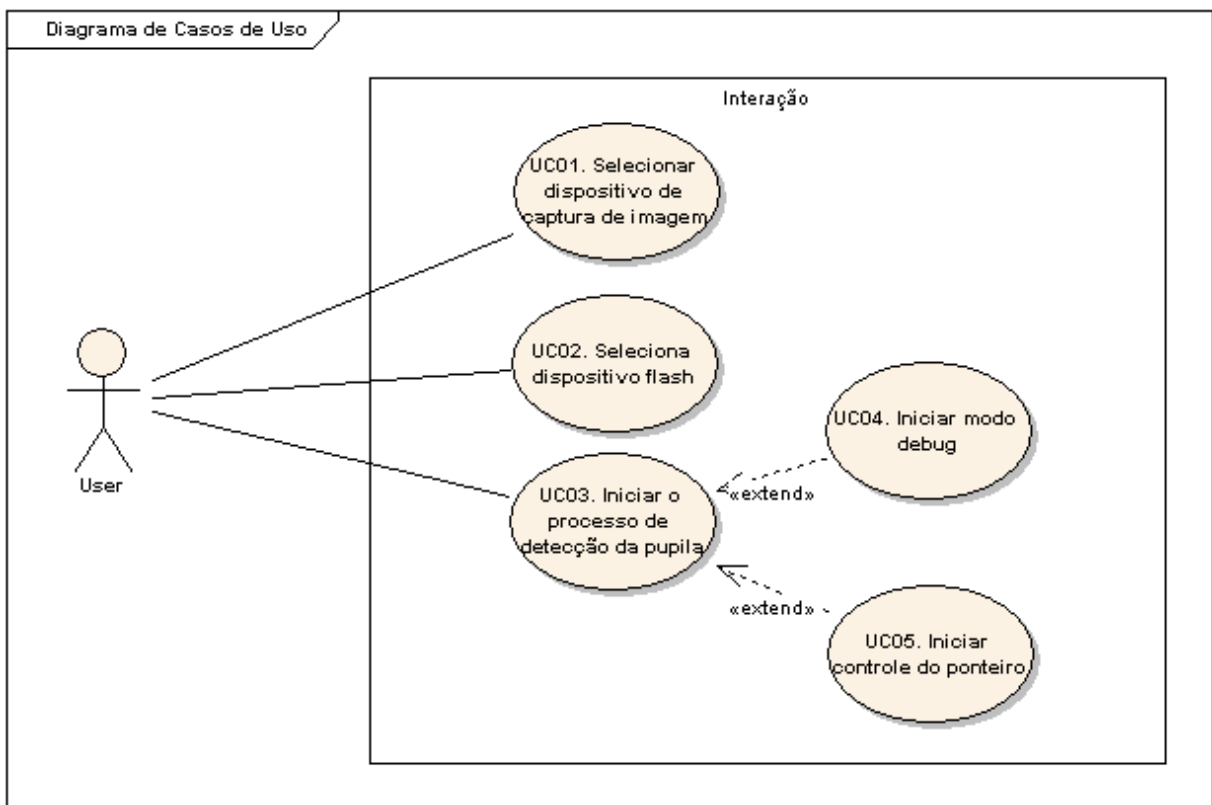


Figura 14 – Diagrama de casos de uso

### 3.2.2 Seleciona dispositivo de captura de imagem

O caso de uso Seleciona dispositivo de captura de imagem (Quadro 1) descreve a forma de seleção da *webcam* a ser utilizada. Este caso de uso possui um cenário



principal, um alternativo e um de exceção.

<b>UC01 - Selecionar dispositivo de captura de imagem:</b> possibilita ao usuário selecionar a webcam a ser utilizada.	
<b>Pré-condição</b>	Um dispositivo de captura de imagem deve estar instalado.
<b>Cenário principal</b>	1) O sistema lista os dispositivos de captura de imagem instalados. 2) O usuário seleciona o dispositivo desejado. 3) O sistema valida o dispositivo selecionado.
<b>Fluxo Alternativo 01</b>	No passo 1, caso exista apenas um dispositivo de captura de imagem instalado, este é selecionado automaticamente pelo sistema.
<b>Exceção 01</b>	No passo 1, caso não exista nenhum dispositivo de captura de imagem instalado, o sistema apresenta uma mensagem informando que não foi encontrado nenhum dispositivo instalado.
<b>Pós-condição</b>	O dispositivo é definido com sucesso.

Quadro 1 – Caso de uso Selecionar dispositivo de captura de imagem

### 3.2.3 Seleciona dispositivo flash

O segundo caso de uso Seleciona dispositivo flash (Quadro 2), descreve a forma de seleção do dispositivo *flash* através de sua porta de comunicação. Além do cenário principal, este caso de uso possui um fluxo alternativo e dois de exceção.

<b>Seleciona dispositivo flash:</b> possibilita ao usuário selecionar o dispositivo flash e a porta a ser utilizada.	
<b>Pré-condição</b>	Um dispositivo flash deve estar instalado.
<b>Cenário principal</b>	1) O sistema lista as portas de comunicação ativas. 2) O usuário seleciona a porta na qual o dispositivo desejado esta conectado. 3) O sistema valida o dispositivo selecionado.
<b>Fluxo Alternativo 01</b>	No passo 1, caso exista apenas uma porta de comunicação ativa esta é selecionada automaticamente pelo sistema e retorna ao passo 3.
<b>Exceção 01</b>	No passo 1, caso não exista nenhuma porta de comunicação ativa, o sistema assume que não há um dispositivo de flash instalado. O sistema apresenta uma mensagem informando que não foi encontrado nenhum dispositivo instalado.
<b>Exceção 02</b>	No passo 3, caso o dispositivo não responda corretamente a requisição de validação, o sistema apresenta uma mensagem informando que não foi encontrado nenhum dispositivo compatível instalado.
<b>Pós-condição</b>	O dispositivo é definido com sucesso.

Quadro 2 – Caso de uso Seleciona dispositivo flash

### 3.2.4 Iniciar o processo de detecção dos olhos

O terceiro caso de uso *Iniciar o processo de detecção dos olhos* (Quadro 3), trata do processo de detecção da pupila, iniciando processo de controle do dispositivo *flash*, processamentos de imagem e reconhecimento e a extração da região do(s) olho(s).

<b>Iniciar o processo de detecção dos olhos:</b> Inicia os processos necessários e realize a detecção dos olhos.	
<b>Pré-condição</b>	Um streaming de vídeo deve ser exibido pelo sistema.
<b>Cenário principal</b>	1) O usuário clica com o mouse no botão Iniciar Detecção. 2) O sistema inicia o monitoramento dos olhos exibindo as coordenadas com sucesso.
<b>Pós-condição</b>	O processo de detecção é iniciado e exibido com sucesso.

Quadro 3 – Caso de uso Iniciar o processo de detecção dos olhos

### 3.2.5 Iniciar modo debug

O quarto caso de uso *Iniciar o modo debug* (Quadro 4), trata de exibir passo a passo o processo de detecção da pupila. Possibilitando acompanhar visualmente todo o processo.

<b>Iniciar modo debug:</b> Inicia o modo debug	
<b>Pré-condição</b>	Um streaming de vídeo deve ser exibido pelo sistema. O processo de detecção dos olhos deve ter sido iniciado.
<b>Cenário principal</b>	1) O usuário clica com o mouse no botão Modo Debug. 2) O sistema inicia a apresentação dos passos realizados para detecção da pupila.
<b>Pós-condição</b>	O processo de detecção é exibido com sucesso.

Quadro 4 – Caso de uso Iniciar modo debug

### 3.2.6 Iniciar o controle do ponteiro

O quarto caso de uso *Iniciar o controle do ponteiro* (Quadro 5), realiza o processo de transferência do movimento do olho para o ponteiro do mouse. Além do cenário principal, este caso de uso possui um fluxo alternativo e nenhum de exceção.

<b>Iniciar o controle do ponteiro:</b> Inicia a transferência do movimento de um olho para o ponteiro do mouse.	
<b>Pré-condição</b>	Um streaming de vídeo deve ser exibido pelo sistema. O processo de detecção dos olhos deve ter sido iniciado.
<b>Cenário principal</b>	1) O usuário clica com o mouse no botão Habilitar Movimento. 2) O sistema inicia a transferência do movimento de um olho para o ponteiro do mouse.
<b>Fluxo Alternativo 01</b>	No passo 1, caso o sistema não detecte nenhum olho durante o processo o ponteiro do mouse deve ficar parado, assim que a detecção retornar volta para o passo 2.
<b>Pós-condição</b>	O processo de transferência do movimento é iniciado com sucesso.

Quadro 5 – Caso de uso seleciona opções de visualização

### 3.2.7 Diagrama de classe

Nesta seção apresenta-se o diagrama de classes, este é responsável por representar a estrutura e relações das classes que servem de modelos para os objetos definidos para a aplicação. Ao todo foram analisadas necessidades de seis pacotes para agrupamento das classes conforme sua natureza e funcionalidade. São eles: `gui`, `control`, `eye`, `rxtx`, `svm` e `video`.

O pacote `gui` (Figura 15) contém as classes responsáveis pela interação com o usuário e captura dos eventos da tela. A classe `JFrameTCC` contém diversos painéis onde são apresentadas as imagens do processamento passo a passo. Cada quadro é exibido em um painel do tipo `JImagePanel`. A classe `EyeSVM` contém a imagem a ser exibida e informações diversas.

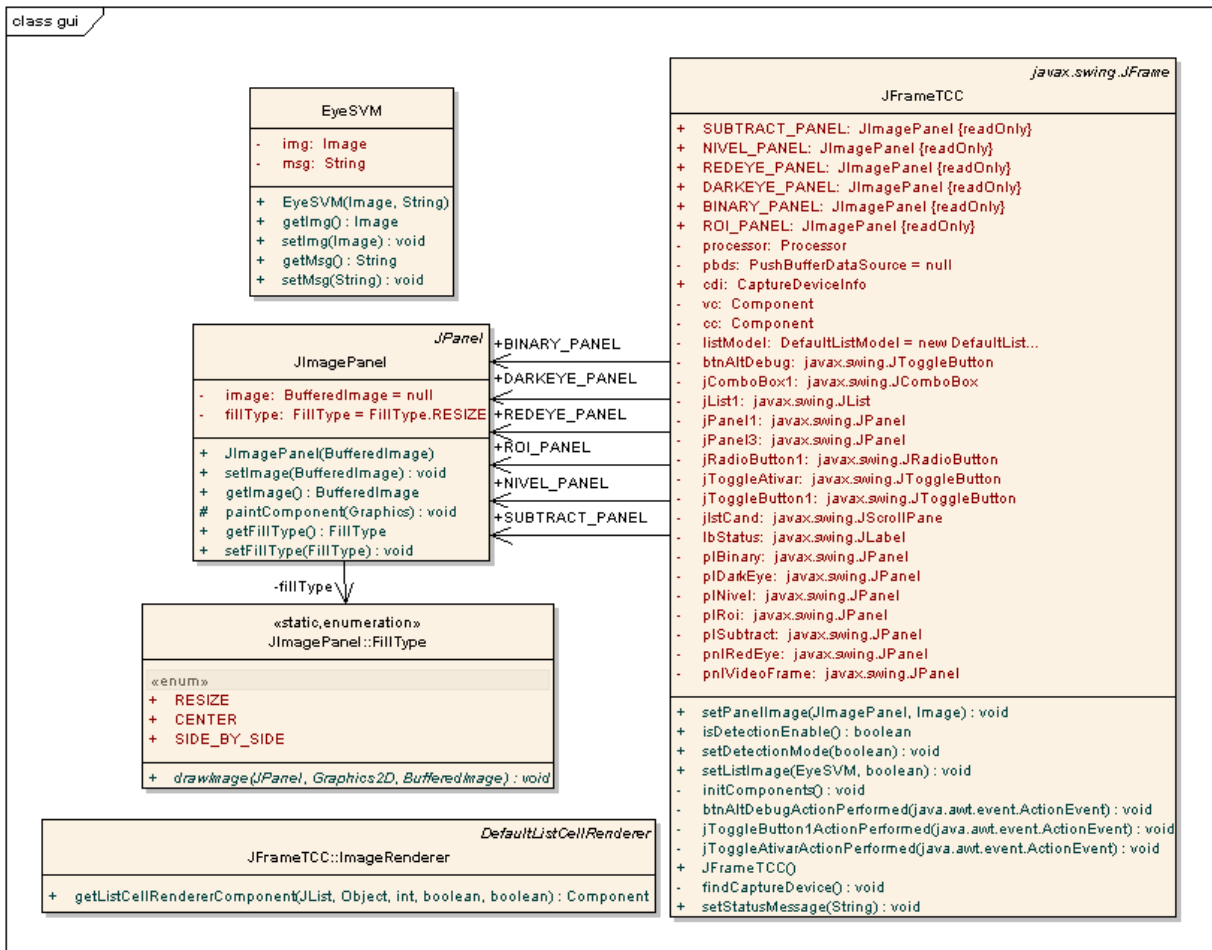


Figura 15 – Diagrama de classe do pacote gui

O pacote `control` (Figura 16) contém as classes utilizadas na centralização de processos comuns ao sistema. A classe `Control` é responsável por iniciar o sistema, a comunicação com a *webcam* e o dispositivo *flash*, além de carregar a base de dados para `LibSVM`, iniciar as variáveis ambientais e apresentar a interface para o usuário. A classe `EyeMouse` se encarrega de controlar os eventos relacionados a movimentação e *click* do *mouse*. A classe `FlashDeviceRXTX` é responsável por abstrair a comunicação com o dispositivo *flash*.

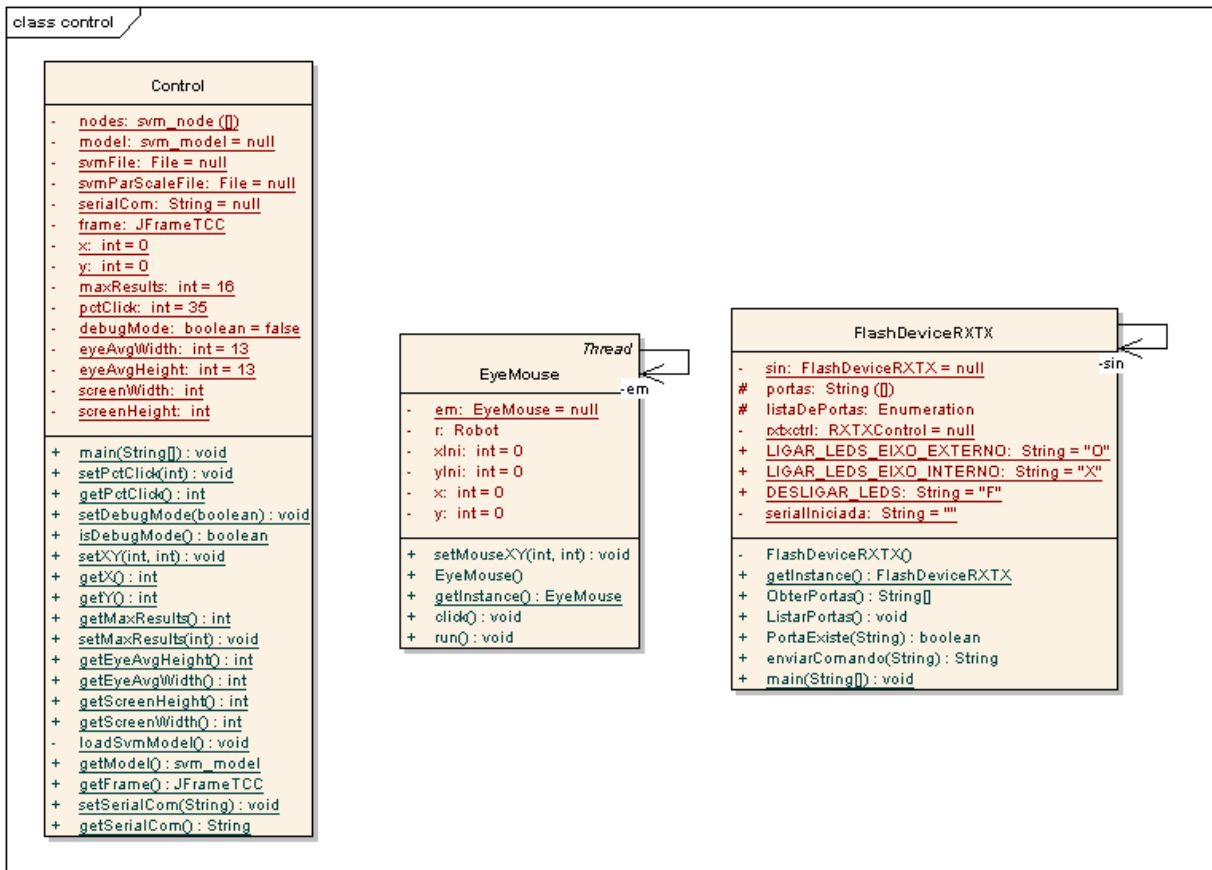


Figura 16 – Diagrama de classe do pacote control

O pacote eye (Figura 17) é responsável pelo processamento das imagens e detecção da região dos olhos. A classe EyeDetectorController recebe as imagens provenientes do dispositivo de captura, onde são aplicados os filtros e é realizado o processo de subtração para que a região dos olhos seja extraída e enviada para a classe EyeDetector. Essa é responsável por classificar a imagem como um olho, ou um falso positivo. A classe ImageProcessing é responsável por realizar os processos úteis no tratamento de imagens não previstos no ImageJ.

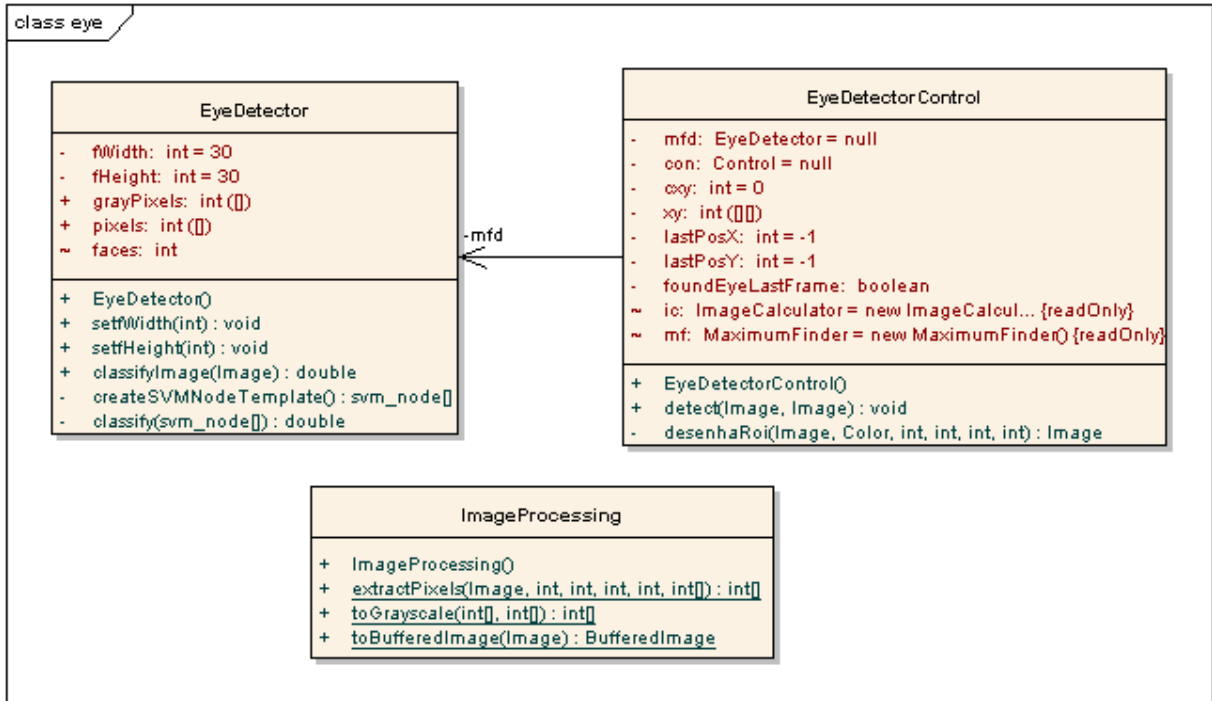


Figura 17 – Diagrama de classe do pacote `eye`

O pacote `rxtx` (Figura 18) contém a classe `RXTXCom` responsável pelo interfaceamento do sistema com a biblioteca RXTX.

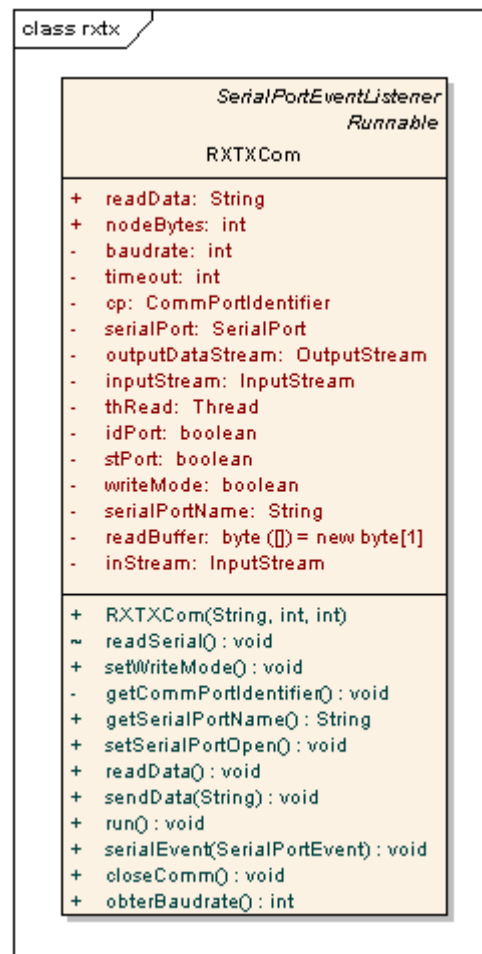


Figura 18 – Diagrama de classe do pacote `rxtx`

O pacote `svm` (Figura 19) contém a classe `MySvmScale` que precisou ser adaptada da LibSVM. Ela tem como função escalonar a imagem de entrada antes do processo de reconhecimento.

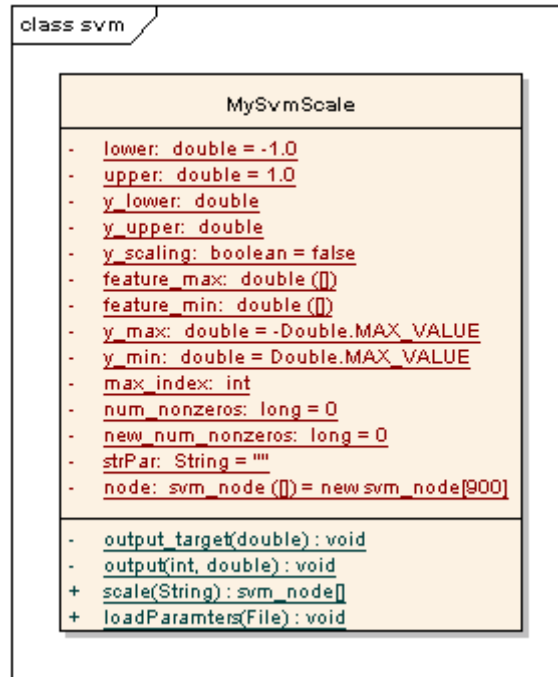


Figura 19 – Diagrama de classe do pacote `svm`

O pacote `video` (Figura 20) contém as classes responsáveis pelo acionamento e controle da *webcam* assim como do *stream* gerado. A classe `ProcessEffectLauncher` recebe o dispositivo reconhecido em `DeviceFinder` e utiliza a classe `ProcessEffect` como um *codec*. Este por sua vez controla o *stream* recebido.

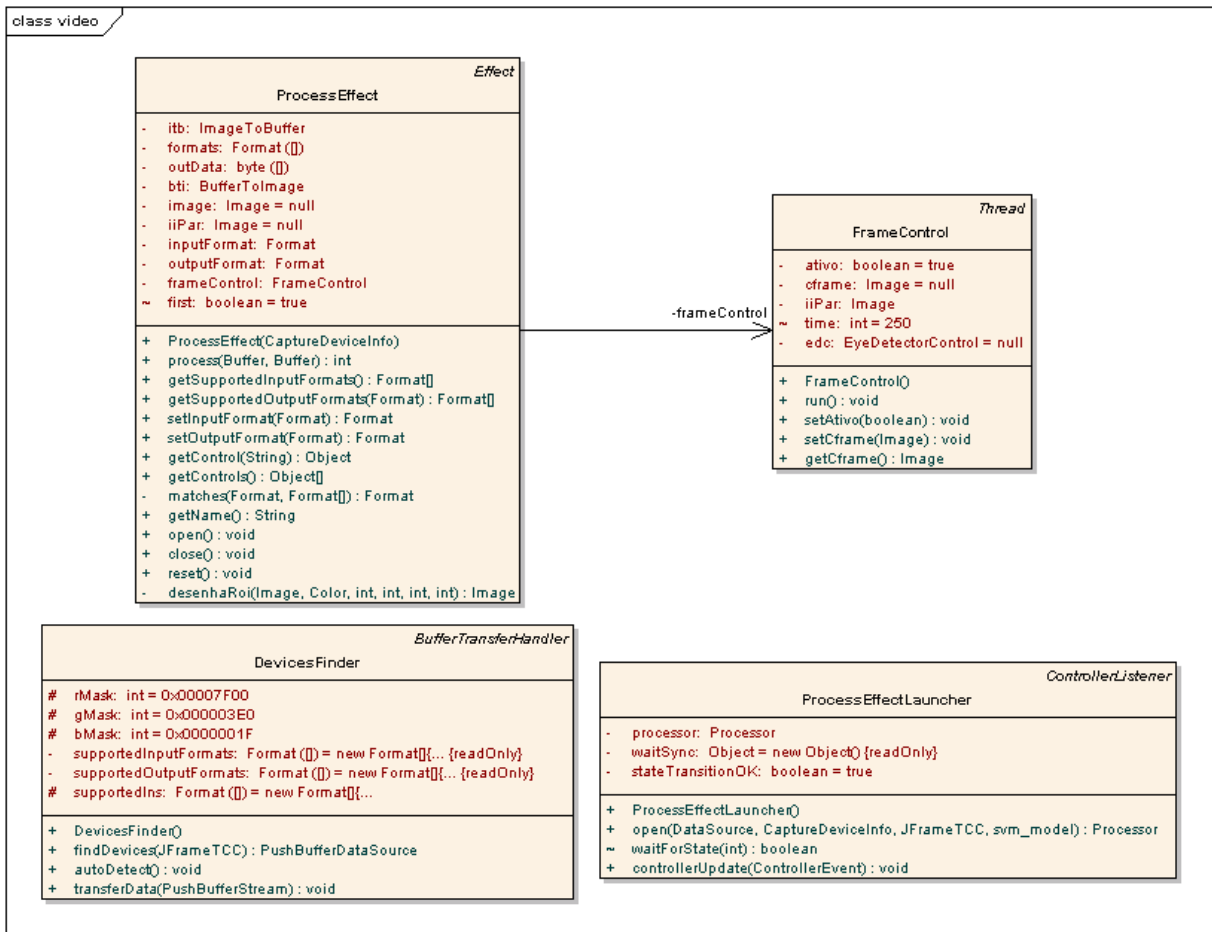


Figura 20 – Diagrama de classe do pacote video

### 3.2.8 Diagrama de seqüência

A Figura 21 mostra o diagrama de seqüência, que permite visualizar o caso de uso Inicia o processo de detecção da pupila de forma resumida para uma melhor visualização.

A partir do momento em que o usuário inicia o sistema e a *webcam* é reconhecida através do método `findCaptureDevice`, o *stream* de vídeo passa pelo método `process` da classe `ProcessEffect` alimentando os parâmetros do tipo *Buffer* utilizados para capturar os quadros. É deste ponto que parte o diagrama de seqüência.



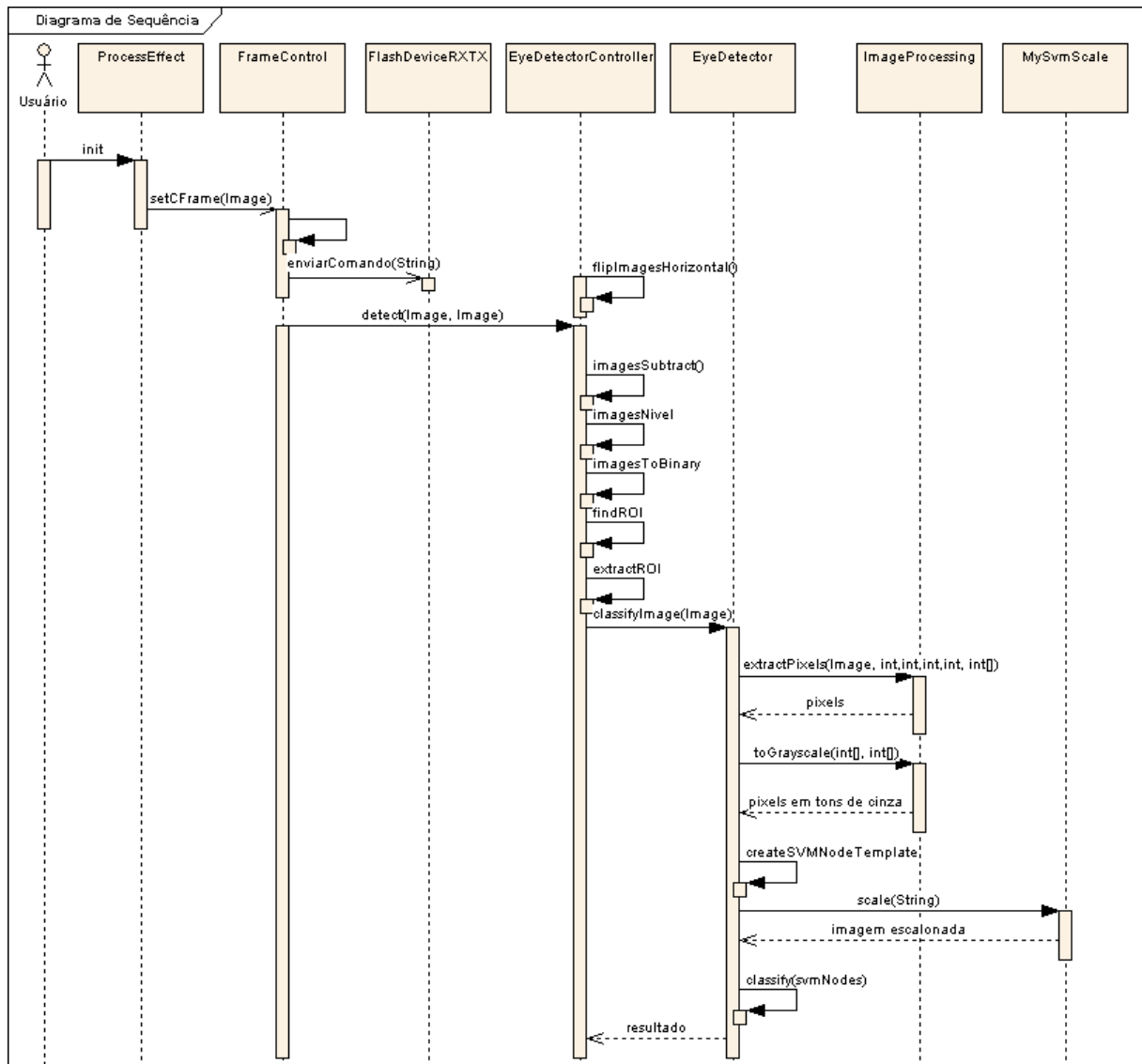


Figura 21 – Diagrama de sequência

### 3.3 IMPLEMENTAÇÃO

Nesta seção são explanadas as técnicas, bibliotecas, ferramentas utilizadas e questões específicas da aplicação. Por fim, faz-se o detalhamento da operacionalidade da implementação.

### 3.3.1 Técnicas e ferramentas utilizadas

Para a implementação do protótipo foi utilizada a linguagem de programação Java juntamente com as bibliotecas JMF, ImageJ, RXTX e LibSVM. O ambiente de desenvolvimento Java escolhido foi o Eclipse SDK Version: 3.4.0, para C para Microcontroladores foi utilizado o CCS PCWHD 4.093. Para desenvolvimento do circuito foi escolhido o Eagle 5.8.0. A gravação do microcontrolador foi realizada através do WinPic800 3.64. A execução de testes e simulação do circuito foram realizadas no Protheus 7.6.

#### 3.3.1.1 Configuração do sistema *flash*

O sistema de detecção da pupila consiste de uma câmera estilo *webcam* e duas fontes de iluminação. Foram utilizados LEDs infravermelhos, invisíveis ao olho humano, e uma *webcam* foi adaptada para ser sensível a iluminação infravermelha.

O sistema de iluminação consiste de 16 LEDs infravermelhos, distribuídos em um anel central e um arranjo lateral, como pode ser visto na Figura 22.

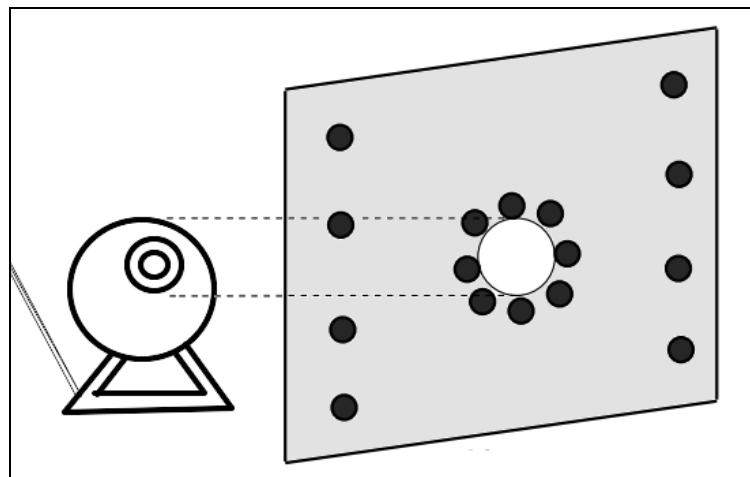


Figura 22 – Configuração do sistema *flash*

O centro do anel coincide com o eixo óptico da câmera, o diâmetro da abertura central é aproximadamente o da *webcam* (15mm). Os quadros da *webcam* são sincronizados com o sistema de iluminação. O centro do anel traz os LEDs suficientemente próximos ao eixo da câmera, o que gera o efeito olhos vermelhos. O arranjo lateral de LEDs está suficientemente afastado, no caso observou-se 4,5 cm, do centro e o efeito não é mais gerado, tendo como única função equilibrar a iluminação na imagem, a fim de facilitar o processamento das

imagens.

No desenvolvimento do circuito utilizou-se um microcontrolador PIC 18F4550. Juntamente foram utilizadas as ferramentas Protheus, Eagle, CSS Compiler e WinPic800 para simulação e desenvolvimento do projeto.

#### 3.3.1.1.1 Microcontrolador PIC 18F4550

O microcontrolador utilizado neste trabalho é o PIC18F4550 MICROCHIP (2004). Suas principais características são:

- a) memória de programação e de dados que permitem apagamento e reescrita de valores milhares de vezes;
- b) auto programável através de um controle interno por software, pelo qual o microcontrolador pode escrever na sua própria memória de programa.
- c) possui um USB SIE compatível com *full-speed* e *low-speed* USB o que possibilita a rápida comunicação entre um host USB e o microcontrolador.
- d) 16 *bits*;
- e) 13 Conversores analógico-digitais de 10 *bits*;
- f) memória de programa *Flash* com 32 *Kbytes*;
- g) memória de dados EEPROM 256 *Bytes*;
- h) memória RAM 2 *Kbytes*;
- i) frequência de operação até 48 *MHz*;
- j) portas bidirecionais de entrada e saída: A, B, C, D e E;
- k) 4 *timers*;
- l) 1 Módulo *Capture/Compare/PWM*;
- m) 2 Comparadores.

#### 3.3.1.1.2 Simulação do circuito com Protheus

O Proteus LabCenter (2010) é uma ferramenta útil para a simulação e a construção de circuitos elétricos. Nele estão disponíveis diversos componentes, incluindo os microcontroladores, com a possibilidade de observar o funcionamento de forma virtual. No Proteus ainda são disponibilizadas ferramentas de laboratório como osciloscópios,

multímetros, geradores de sinais entre outros, permitindo o aprendizado da operação dos principais instrumentos essenciais em uma bancada de eletrônica.

O Proteus é um produto pago mantido pela LabCenter. Esta disponível em uma versão para demonstração onde não é permitido salvar, imprimir ou projetar um novo microcontrolador. Esta versão pode ser obtida no endereço <http://www.labcenter.com>.

No desenvolvimento deste trabalho o Proteus 7.6 juntamente com o módulo virtual USB foi utilizado para simular a comunicação através da porta USB, entre o computador e o dispositivo *flash*. Como pode ser visto na Figura 23, ao iniciar a simulação é gerada uma porta serial COM3.

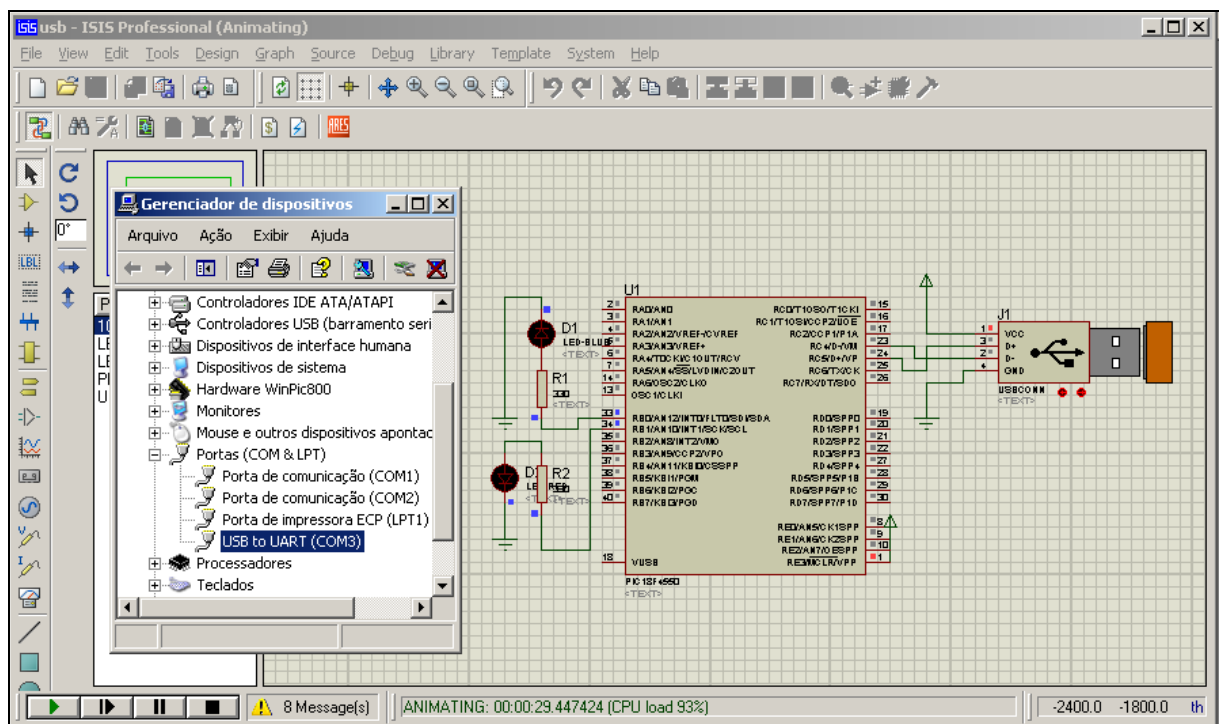


Figura 23 – Simulação do dispositivo *flash* no Proteus

### 3.3.1.1.3 Compilador CCS C

Segundo Santos (2009) o compilador CCS C é um compilador de linguagem C para microcontroladores, especialmente da linha PIC. Desenvolvido pela empresa norte-americana Custom Computer Services, é um software pago e possui uma versão para demonstração limitada a 30 dias de uso e que permite apenas a programação de alguns poucos microcontroladores. Pode ser obtido em <http://www.ccsinfo.com>. Foi utilizado no presente trabalho para o desenvolvimento da rotina de controle do dispositivo *flash*

O CCS C provê completa estrutura lógica para a programação e permite fácil

implementação, sendo composto por operadores que foram empacotados em bibliotecas que são específicas para os microcontroladores PIC, bem como acesso aos recursos de hardware com C.

#### 3.3.1.1.4 Biblioteca de funções

A biblioteca de funções é um conjunto de funções extras que tornam transparente ao programador a manipulação do dispositivo USB. Existem funções de inicialização, transmissão e recepção de dados usando o protocolo de comunicação USB. Esta biblioteca foi desenvolvida, pelo fabricante, para uso em sistemas que usem o microcontrolador de modelo PIC 18F4550 e compilador CCS (SANTOS, 2009, p. 33).

A transmissão de dados neste trabalho é realizada por um microcontrolador que foi programado para se comportar como um dispositivo do tipo CDC através da porta USB. O dispositivo é reconhecido como uma interface de transmissão através de uma porta serial de comunicação.

Existem funções nativas fornecidas com a instalação do compilador. Segundo Santos(2009) algumas das principais funções nativas são:

- a) `usb_enumerated()`: verifica se o dispositivo está pronto para a comunicação;
- b) `usb_detach()`: desconecta o dispositivo. Deve ser usada antes de sua remoção física do computador;
- c) `usb_attach()`: re-conecta o dispositivo, deve ser usada para re-conectá-lo quando o dispositivo foi desconectado, mas ainda não removido literalmente;
- d) `usb_cdc_putc(char c)`: envia um caracter via USB;
- e) `usb_cdc_kbhit()`: verifica se existe algum dado no buffer de recepção;
- f) `usb_cdc_getc()`: recebe um caracter. Deve-se usar o `usb_cdc_kbhit()` descrito anteriormente para verificar se existem dados;
- g) `get_float_usb()`: recebe um numero ponto flutuante;
- h) `get_long_usb()`: recebe um numero inteiro longo;
- i) `get_int_usb()`: recebe um inteiro;
- j) `get_string_usb(char *s, int max)`: recebe uma string;
- k) `gethex_usb()`: recebe um hexadecimal.

### 3.3.1.1.5 Programa para controle do dispositivo flash

O programa de controle responde ligando os LEDs internos e desligando os externos ao receber o caractere “X”, e faz o inverso ao receber o caractere “O”. O caractere “F” desliga todo o conjunto de LEDs. O programa completo pode ser visto no Quadro 6.

```
#include <18F4550.h>
#fuses HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV1, VREGEN
#use delay(clock=48000000)
#include <usb_cdc.h>

void main()
{
    // Caracter recebido
    char rec;
    // Inicializa, configura e enumera o dispositivo USB
    usb_cdc_init();
    usb_init();
    // espera o dispositivo ser reconhecido
    while (!usb_cdc_connected()) {}

    // Controle efetivo do circuito flash
    do {
        // Mantem a conexao ativa
        usb_task();
        // verifica se o dispositivo esta pronto para comunicação
        if(usb_enumerated()){
            // Verifica se existe algum dado no buffer de recepção
            if(usb_cdc_kbhit()){
                // recupera o caractere enviado
                rec = usb_cdc_getc();
                if(rec == 'X'){
                    // Desliga IR circulo eXterno
                    output_low(pin_b2);
                    // Liga IR circulo interno
                    output_high(pin_b1);
                    // Retorno
                    usb_cdc_putc('1');
                } else if (rec == 'O') {
                    // Desliga IR circulo interno
                    output_low(pin_b1);
                    // Liga IR circulo eXterno
                    output_high(pin_b2);
                    // Retorno
                    usb_cdc_putc('1');
                } else if (rec == 'F') {
                    // Desliga IR circulo eXterno
                    output_low(pin_b2);
                    // Desliga IR circulo interno
                    output_low(pin_b1);
                    // Retorno
                    usb_cdc_putc('1');
                } else if (rec == '?') {
                    // Retorno confirmando procedencia
                    usb_cdc_putc('2');
                } else {
                    // Retorno
                }
            }
        }
    } while (1);
}
```

```

        usb_cdc_putc('0');
    }
}

} while(true);
// Nao utiliza nenhuma porta analogica
setup_adc_ports(NO_ANALOGS|VSS_VDD);
// Desliga o conversor Analogico Digital
setup_adc(ADC_OFF);
// Desabilita porta PSP
setup_psp(PSP_DISABLED);
// Desabilita porta SPI
setup_spi(SPI_SS_DISABLED);
// Desabilita o reset em caso de travamento do software
setup_wdt(WDT_OFF);
// Configura o timer 0 para clock interno
setup_timer_0(RTCC_INTERNAL);
// Desabilita TMR1
setup_timer_1(T1_DISABLED);
// Desabilita TMR2
setup_timer_2(T2_DISABLED,0,1);
// Desabilita TMR3
setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
// Desabilita comparadores
setup_comparator(NC_NC_NC_NC);
// Desabilita tensão de referencia
setup_vref(FALSE);}

```

Quadro 6 – Código fonte do programa de controle do dispositivo *flash*

Os bits de configuração ou *fuses* utilizados são detalhados no Quadro 7.

Bits de configuração	Descrição
HSPLL	Cristal oscilador de alta velocidade com PLL habilitado
NOWDT	Desativação do cão de guarda
NOPROTECT	Desativação de proteção de código
NOLVP	Desativação da gravação em baixa tensão
NODEBUG	Desativação de depuração em circuito
USBDIV	Ativação do multiplexador USBDIV
PLL5	Uso de PLL pré escalar com valor 5
CPUDIV1	Uso de PLL pós escalar com valor 1
VREGEN	Ativação do uso do Vusb como regulador de tensão interna USB

Quadro 7 – Bits de configuração

Utilizando o modulo PLL pré escalar com a configuração PPL5, o valor de frequência de entrada 20 MHz é dividido por 5 gerando uma frequência de 4 MHz. Este valor é o adequado para o módulo de geração de 96 MHz. O sinal é então encaminhado para dois módulos diferentes. O primeiro módulo é controlado pelo USBDIV, o qual divide o sinal por 2 gerando uma frequência de 48 MHz para o módulo USB. O outro é controlado pela PLL pós escalar. Este valor recebido pelo módulo PLL pós escalar é dividido por 2, conforme CPUDIV1 e é enviado para o núcleo do microcontrolador. Como foi observado o núcleo do microcontrolador e o periférico USB, ambos trabalham com a frequência de 48 MHz (SANTOS, 2009).

### 3.3.1.1.6 Projeto do circuito

Para o desenho, esquemático e layout das trilhas, do circuito foi utilizado o programa EAGLE 5.8.0, desenvolvido pela empresa alemã CADSOFT está disponível para download gratuito no endereço <http://www.cadsoft.de>, onde se tem versões para Windows® e para Linux, nos idiomas inglês e alemão. Apresenta-se como um sistema de projetos eletrônicos disponibilizado gratuitamente em uma versão *light* pelo fabricante, sendo largamente utilizado por estudantes.

O programa permite o desenho do diagrama esquemático do circuito eletrônico e a geração automática da placa de circuito impresso. O desenho esquemático pode ser visto na Figura 24.

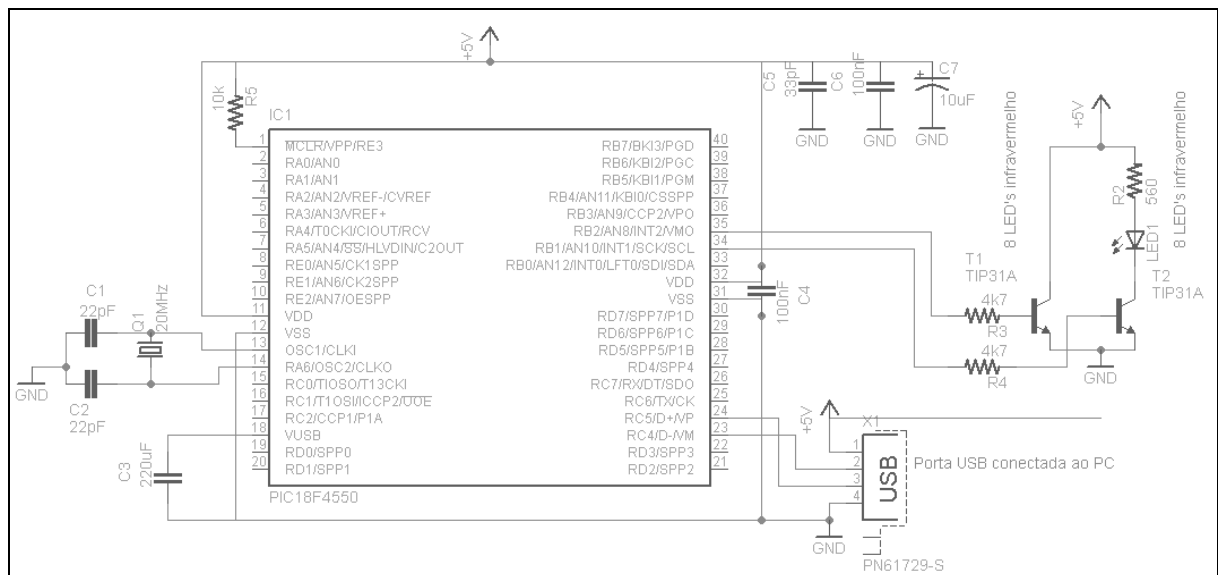


Figura 24 – Diagrama esquemático do dispositivo *flash*

O circuito apresenta 16 LEDs infravermelhos divididos em dois grupos de oito. Para alimentação dos LEDs direcionou-se os +5 Volts, entregues pela porta USB, diretamente ao conjunto. A ação de ligar e desligar os LEDs fica a cargo dos dois transistores TIP31A, que possuem sua base conectada as portas B1 e B2.

O oscilador, responsável por gerar o *clock* de entrada no circuito, utiliza um cristal de 20 MHz e dois capacitores cerâmicos de 22pF. Para eliminação de ruídos e estabilização o circuito, utilizou-se um capacitor eletrolítico de 10uF e dois capacitores cerâmicos, um de 10pF e outro de 100nF, ligados em paralelo.

A entrada Vusb realiza a alimentação dos resistores *pull-up* para a detecção da velocidade da transmissão USB, isto é, detecta se o dispositivo opera em *full speed device* ou em *low speed device*. A não ativação desta função provoca uma necessidade de alimentação



externa deste pino por uma tensão de 3,3V. Ele funciona também como regulador interno de tensão para o dispositivo USB. O valor recomendado é de 220nF para o capacitor conectado a porta Vusb MICROCHIP (2004).

### 3.3.1.1.7 Circuito elétrico

O circuito elétrico envolve todas as conexões, simuladas ou não. A estrutura da construção física do circuito varia conforme disponibilidade de espaço, confecção de trilhas, colocação de componentes e conectores. O desenho da placa de circuito impresso é demonstrado na Figura 25.

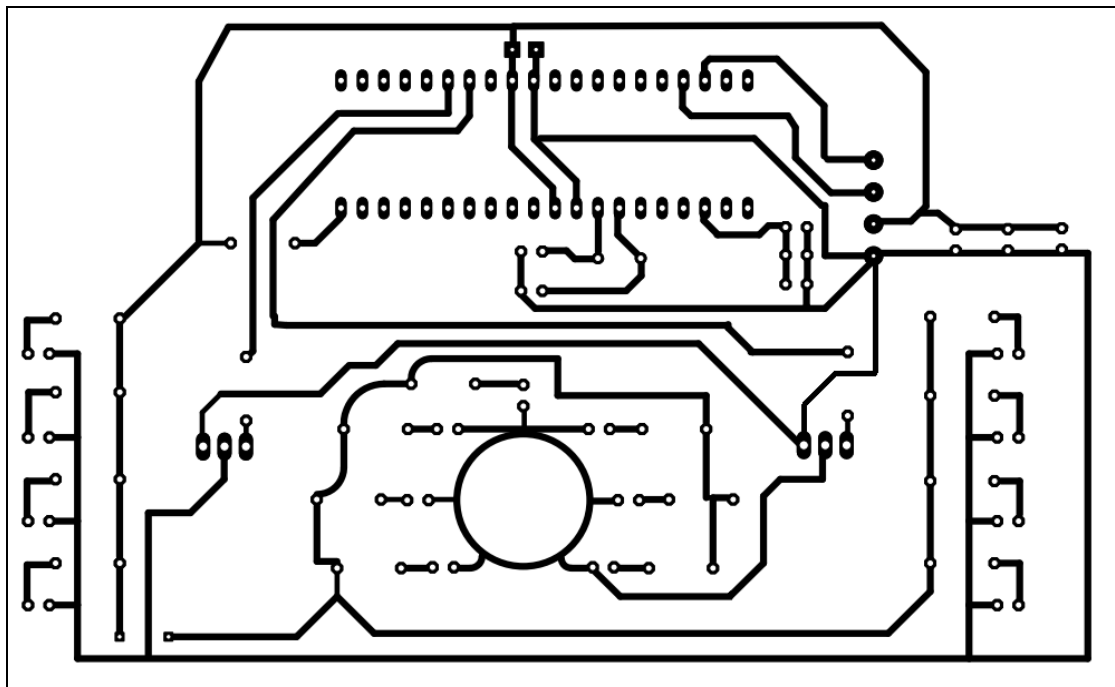


Figura 25 – Diagrama circuito elétrico

O circuito construído pode ser visualizado na Figura 26. Seu funcionamento e instalação são detalhados nas seções posteriores.



Figura 26 – Dispositivo *flash*

#### 3.3.1.1.8 LED infravermelho

Os LEDs utilizados emitem luz infravermelha na banda de 850nm em um ângulo de abertura de 30°. A luz infravermelha emitida por estes LEDs se encontra na faixa de 800nm até 960nm onde a IEC 60825-1 VISHAY(2008) alerta para o perigo de queima da retina e formação de catarata quando a exposição for prolongada.

#### 3.3.1.1.9 Gravação do microcontrolador PIC18F4550

Ao compilar um programa fonte em C é gerado um arquivo em hexadecimal pronto para ser gravado, o sistema de gravação executa o processo da transferência do programa para o microcontrolador. O gravador faz a conversão dos dados gerados pelo compilador em dados a serem armazenados no microcontrolador.

O processo de gravação do microcontrolador PIC18F4550 foi realizado através de uma gravadora serial, a mesma pode ser observada na Figura 27.

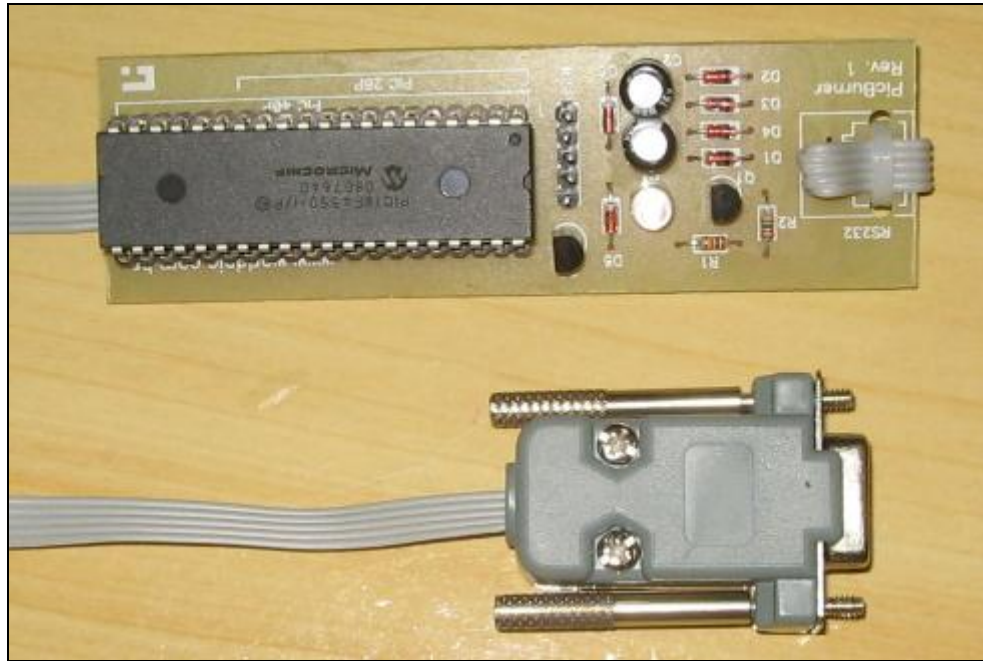


Figura 26 – Gravadora PIC serial

O programa responsável pela gravação foi o WinPic800 3.64 WINPIC800(2010), uma ferramenta desenvolvida pela Microchip está disponível para download gratuito no endereço <http://www.winpic800.com>. Com pode ser visto na Figura 28, com o arquivo hexadecimal aberto e o microcontrolador PIC18F4550 selecionado o mesmo foi gravado com sucesso.

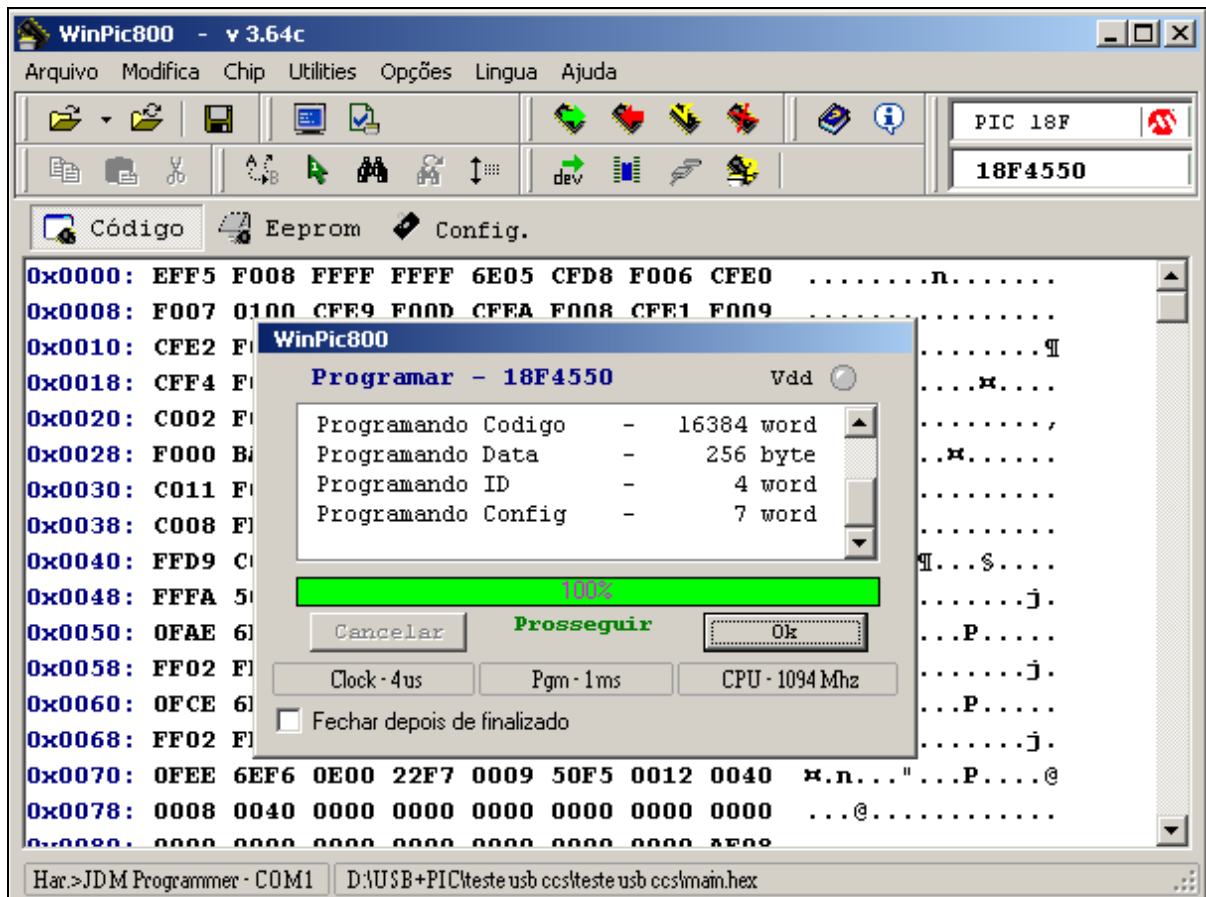


Figura 27 – Gravação do microcontrolador através do WinPic800

### 3.3.1.2 Câmera Infravermelha

Segundo Stern (2006) os sensores CCD são sensíveis a luz visível e infravermelha. Para evitar a saturação dos sensores devido à luz infravermelha, o que produziria um efeito de não naturalidade nas imagens captadas, os dispositivos incluem um filtro infravermelho. Este filtro é conhecido como *hot-mirror*, e está localizado na trajetória óptica para reduzir a quantidade de luz infravermelha recebida pelo CCD. Como o filtro permite que apenas uma pequena quantidade de luz chegue ao CCD, deve-se remover este filtro para utilizar uma *webcam* convencional para capturar o espectro infravermelho

Neste trabalho foi utilizada uma *webcam* convencional modelo JPEG USB Video Camera PK-121 HP. Nesse modelo, o filtro está presente junto à lente. A remoção só foi possível quebrando o filtro. Na Figura 29 é possível ver a lente sem o filtro.



Figura 28 – Lente com o filtro removido

Além de remover o filtro é preciso que se insira um novo filtro, este deve permitir apenas a passagem da luz infravermelha, evitando assim a saturação da imagem. Segundo Johnson(2010) pode ser utilizada a parte escura do filme fotográfico, como o que pode ser visto na Figura 30(a). Dois pedaços do filme foram recortados em forma de círculo e colados junto à lente – Figura 30(b).

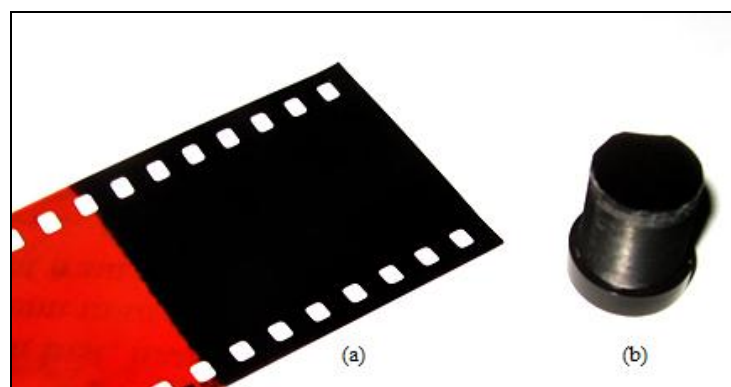


Figura 29 – (a) Papel fotográfico e (b) lente com filtro

### 3.3.1.3 Comunicação entre computador e dispositivo flash

Para realizar a comunicação entre o computador e dispositivo flash foi utilizada a biblioteca RXTX. A comunicação foi estabelecida através da porta COM disponibilizada pelo *driver* do microcontrolador utilizado.

#### 3.3.1.3.1 RXTX

Quando se faz necessário ao Java a tarefa de chamar APIs nativas dos sistemas operacionais, ou ainda, fazer comunicação diretamente com o hardware, começam a aparecer problemas de portabilidade. A *Sun* e demais empresas envolvidas no desenvolvimento Java disponibilizam diversas APIs para facilitar este trabalho, como é o caso da biblioteca RXTX para a comunicação tanto serial quanto paralela (SILVA, 2010).

A biblioteca RXTX é baseada na API Javacomm distribuída pela própria Sun. Porém essa API é destinada apenas para ambientes Linux enquanto a RXTX é portátil para Linux, Windows e Mac (SILVA, 2010).

#### 3.3.1.3.2 Comunicação com RXTX

A biblioteca RXTX abstrai grande parte da comunicação deixando os métodos de interação extremamente simples. Como pode ser visto no Quadro 8 o método de conexão recebe a porta COM, o *baudrate*<sup>3</sup> e o *timeout* informados no construtor.

```

private CommPortIdentifier cp;
private SerialPort serialPort;

public void setSerialPortOpen() {
    try {
        // Obtem identificador da porta serial através do nome
        cp = CommPortIdentifier.getPortIdentifier(serialPortName);
        // Obtem objeto da porta serial
        serialPort = (SerialPort) cp.open(getSerialPortName(), timeout);
        ...
        // configurar parâmetros
        serialPort.setSerialPortParams(baudrate,
                                       serialPort.DATABITS_8,
                                       serialPort.STOPBITS_1,
                                       serialPort.PARITY_NONE);
        serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
    } catch (Exception e) {...}}

```

Quadro 8 – Código fonte do método `setSerialPortOpen`

Da mesma forma o método de envio de dados é bastante simples, como pode ser visto no Quadro 9.

```

public void sendData(String msg) {
    try {
        // Obtem o fluxo para envio de dados
        outputStream = serialPort.getOutputStream();
    } catch (IOException e1) {...}
    try {
        // Escreve no buffer de saida
        outputStream.write(msg.getBytes());
        // Descarrega o buffer no canal de saida
        outputStream.flush();
    } catch (IOException e) {...} }

```

Quadro 9 – Código fonte do método `sendData`

Para o controle do dispositivo *flash* foi criada a classe `FlashDeviceRXTX` responsável por estabelecer a comunicação através da porta informada e manter um meio único de interação com o dispositivo *flash*. As principais funcionalidades dessa classe podem ser vistas no Quadro 10.

---

<sup>3</sup> Termo utilizado como medida de velocidade de transmissão de informação entre computadores. A medida se da em pulsos por segundo BAUD(2010).

```

private static RXTXControl rtxctrl = null;
public static String LIGAR_LEDS_EIXO_EXTERNO = "O";
public static String LIGAR_LEDS_EIXO_INTERNO = "X";
public static String DESLIGAR_LEDS = "F";
...
public static FlashDeviceRXTX getInstance() {
if (sin == null)
    sin = new FlashDeviceRXTX();
// Inicia a porta escolhida
if (Control.getSerialCom() != null) {
    if (!serialIniciada.equals(Control.getSerialCom())) {
        rtxctrl= new RXTXControl(Control.getSerialCom(), 115200, 1000);
        rtxctrl.setWriteMode();
        rtxctrl.setSerialPortOpen();
        serialIniciada = Control.getSerialCom();
    }
}
return sin; }

public void enviarComando(String cmd) {
    leitura.sendData(cmd);
}...

```

Quadro 10 – Código fonte da classe FlashDeviceRXTX

#### 3.3.1.4 JMF

O JMF permite que áudio, vídeo e outras mídias possam ser adicionadas a aplicativos baseados na tecnologia Java. Este pacote de opcionais, que pode captar, reproduzir, transmitir e decodificar vários formatos de mídia, estende a plataforma Java aos desenvolvedores multimídia, fornecendo uma ferramenta poderosa e escalável para desenvolver em tecnologia multiplataforma (JMF,2010).

##### 3.3.1.4.1 Codec

Para obter os quadros capturados pela *webcam* e interferir em seu conteúdo utilizou-se uma classe do tipo *Codec*, que se instala entre o canal de recepção e saída do *buffer* de vídeo, permitindo assim que os quadros sejam capturados, alterados e devolvidos. Uma parte do código fonte do método de inferência pode ser visto no Quadro 11.

```

public int process(Buffer inBuffer, Buffer outBuffer) {
    ...
    this.outData = (byte[]) inBuffer.getData();
    // Cria um Image a partir do buffer
    if (bti == null)
        bti = new BufferToImage((VideoFormat) inBuffer.getFormat());
    // Cria um objeto ImagePlus do ImageJ
    ImagePlus ipq = new ImagePlus("quadro", bti.createImage(inBuffer));
    // Altera o quadro na horizontal
    IJ.run(ipq, "Flip Horizontally", "");
    ...
    final Buffer b =
itb.createBuffer(image, ((VideoFormat).inBuffer.getFormat()).getFrameRate());
    // Copia os atributos de video
    outBuffer.setFormat(b.getFormat());
    outBuffer.setLength(inBuffer.getLength());
    outBuffer.setOffset(inBuffer.getOffset());
    // Copia a imagem para o buffer
    outBuffer.setData(b.getData());
    // Passa o frame para o controlador de quadros
    frameControl.setCframe(bti.createImage(inBuffer));
    return BUFFER_PROCESSED_OK; }

```

Quadro 11 – Código fonte do método process

#### 3.3.1.4.2 Controle dos quadros

A classe `FrameControl` responsável pelo controle dos quadros, no que se refere ao acionamento dos conjuntos de LEDs do dispositivo *flash*, faz ainda o encaminhamento do processo de detecção. Ela foi desenvolvida para se comportar como um processo paralelo. Uma parte do código fonte do método `run()` pode ser visto no Quadro 12.

```

public void run() {
    while (ativo) {
        // Se o modo de deteacao estiver ativo
        if (Control.getFrame().isDetectionEnable()) {
            // Caso a imagem par seja nula
            if (iiPar == null) {
                // Verifica se ha algum problema com a imagem
                if (getCframe() == null) {
                    try {
                        sleep(20);
                    } catch (InterruptedException ex) {...}
                    continue; }
                // Armazena quadro atual
                iiPar = getCframe();
                // Liga LEDs do eixo Externo
                FlashDeviceRXTX.getInstance()
                    .enviarComando(FlashDeviceRXTX.LIGAR_LEDS_EIXO_EXTERNO);
            }
        }
    }
}

```



```

// Limpa frame atual
setCframe(null);
try {
    Thread.sleep(time);
} catch (InterruptedException ex) {...}
} else {
// Verifica se ha algum problema com a imagem
if (getCframe() == null) {
    try {
        sleep(20);
    } catch (InterruptedException ex) {...}
    continue;
}
// Procedimento de detecção
edc.detect(getCframe(), iiPar);
// Liga LEDs do eixo Interno
FlashDeviceRXTX.getInstance()
    .enviarComando(FlashDeviceRXTX.LIGAR_LEDS_EIXO_INTERNO);
try {
    Thread.sleep(time);
} catch (InterruptedException ex) {...}
// Limpa variavel que armazena o quadro par
iiPar = null;
setCframe(null);
...}

```

Quadro 12 – Código fonte do método `run`

O método se encarrega de obter um quadro com os olhos vermelhos e outro com os olhos escuros. É feito o encaminhamento para o método `detect` da classe `EyeDetectorControl`.

### 3.3.1.5 Processamento de Imagens

Para realizar o processamento das imagens obtidas no processo de captura conjunto ao dispositivo *flash*, foram utilizadas as técnicas de processamento de imagens como análise do histograma, binarização de imagens digitais, análise de componentes conexos entre outras. Para realizar o processamento foi utilizada a biblioteca `ImageJ`.

#### 3.3.1.5.1 `ImageJ`

Esta biblioteca consiste em um programa *open-source* desenvolvido em Java para processamento de imagens. Foi desenvolvido e é mantido pela National Institutes of Health, sendo projetado com uma arquitetura aberta que permite extensibilidade através de *plugins* e macros graváveis.

O `ImageJ` pode exibir, editar, analisar, processar, salvar e imprimir imagens. Pode ler e

salvar os mais diversos formatos de imagem, suporta pilhas de imagens (*stacks*), processamento *multithreaded*, pode medir distâncias e ângulos, criar histogramas e gráficos de perfis de linha. Suporta funções de processamento de imagem padrão, tais como operações aritméticas e lógicas entre imagens, manipulação de contraste, convolução, análise de Fourier, nitidez, suavização, detecção de bordas, mediana de filtragem entre outros. Faz transformações geométricas, como escala, rotação e saltos (IMAGEJ, 2010).

### 3.3.1.5.2 Técnicas utilizadas

Para obter as regiões de interesse, a simples subtração dos quadros não gera uma imagem que apresente apenas a pupila. Mesmo utilizando os LEDs do eixo externo para equilibrar a quantidade de luz, um ruído luminoso ainda permanece. Para eliminar este ruído um conjunto de técnicas gráficas foi utilizado.

A subtração do quadro que contém os olhos vermelhos (Figura 31(a)) do quadro olhos escuros (Figura 31(b)) gera uma imagem onde a região da pupila fica fortemente destacada das demais regiões (Figura 31(c)).

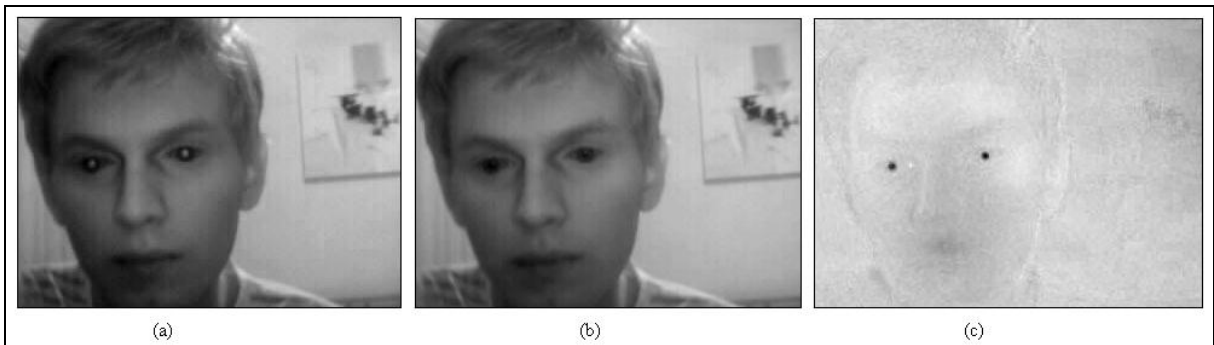


Figura 30 – (a) Efeito olhos-vermelhos; (b) olhos escuros e (c) resultado da subtração

A análise do histograma mostrou o que pode ser observado na Figura 30(c), mesmo com a iluminação não totalmente equilibrada a região mais escura, a pupila, é fortemente destacada das demais. Assim, trazer a janela do histograma para vertical e posicionar o nível exatamente na metade (Figura 32(b)), garante que apenas as partes mais escuras sejam apresentadas (Figura 32(a)).

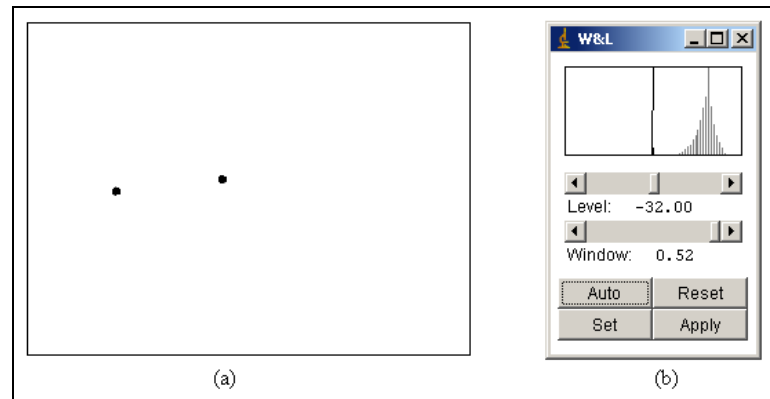


Figura 31 – (a) Imagem após nivelamento e (b) histograma

A imagem gerada no processo anterior é primeiramente convertida para 8 *bits*, para então ser binarizada. Após esse processo a imagem tem suas cores invertidas para ser processada pela função *Find Maxima*. Como pode ser visto na Figura 33(a), esta função indica o centro de componentes conexos espalhados pela imagem.

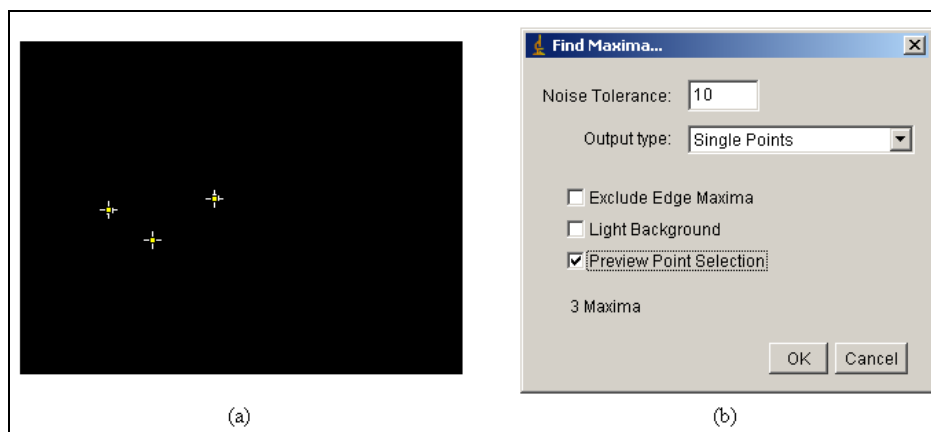


Figura 32 – (a) Centro dos componentes conexos e (b) função Find Maxima

As regiões destacadas na Figura 33(a), são denominadas como ROI, e são enviadas ao módulo de reconhecimento de padrões descrito nas seções subseqüentes.

### 3.3.1.5.3 Método de detecção

O método responsável pelo tratamento das imagens é o `detect` da classe `EyeDetectorControl`. O método recebe os quadros par e ímpar, respectivamente com olhos vermelhos e olhos escuros. O primeiro processo é o de subtração, que pode ser visto no Quadro 13.

```

public void detect(Image iiPar, Image iiImpar) {
    final ImagePlus ipPar, ipImpar, ipSubtractedImg, ipNivel;
    int[][] eyecand;
    // IMAGEM PAR - Efeito olhos vermelhos
    ipPar = new ImagePlus("par", iiPar);
    ...
    // IMAGEM IMPAR - Olho escuro
    ipImpar = new ImagePlus("impar", iiImpar);
    ...
    // PROCESSO DE SUBTRACÇÃO
    // Subtrai as imagens
    ipSubtractedImg = ic.run("Subtract create 32-bit", ipImpar, ipPar);
    ...
}

```

Quadro 13 – Subtração de quadros no método detect

O nivelamento do histograma é realizado no quadro gerado pelo processo de subtração. Obtém-se os máximos e mínimos da imagem, estes são divididos por dois, e então redefinidos como o novo nível através do método `setMinAndMax`, como pode ser visto no Quadro 14.

```

...
// Divide histograma de forma a resaltar as areas mais claras
final int h = (int) Math.round((ipSubtractedImg.getProcessor().getMax() +
ipSubtractedImg.getProcessor().getMin()) / 2);
ipSubtractedImg.getProcessor().setMinAndMax(h - 1, h);
ipNivel = new ImagePlus("nivel", ipSubtractedImg.getProcessor());
...

```

Quadro 14 – Nivelamento do quadro no método detect

O terceiro passo consiste na binarização e busca pelas ROIs. Os resultados obtidos pelo processo *Find Maxima* são armazenados como candidatos a olho. O trecho de código referente a esse processo pode ser visto no Quadro 15.

```

// BINARIZACAO
// Torna binario
IJ.run(ipNivel, "8-bit", "");
IJ.runPlugIn(ipNivel, "ij.plugin.Thresholder", "Make Binary");
// ROI
// Inverte as cores para procurar os maximos
IJ.run(ipNivel, "Invert", "");
// Procura os maximos
IJ.run(ipNivel, "Find Maxima...", "noise=10 output=List exclude");
// Obtem resultados
ImagePlus ipROI = new ImagePlus("ROI", ipNivel.getImage());
final ResultsTable rt = ResultsTable.getResultsTable();
rt.show("Results");
eyecand = new int[rt.getCounter()][2];

```

Quadro 15 – Binarização e ROI no método detect

Com as ROI definidas, é preciso extrair os quadros a serem utilizados para reconhecimento. Empiricamente se obteve que uma redução de seis vezes na escala é a que melhor se adequa a uma distância média de 40 cm, na qual o usuário se posiciona da *webcam*. A redução se deu seguindo consecutivamente a proporção de 1/1.2. As imagens candidatas a olho são então encaminhadas ao módulo de reconhecimento através do método `classifyImage`. O trecho de código referente a esse processo pode ser visto no Quadro 16.

```

// LOCALIZACAO DO OLHO
for (int i = 0; i < eyecand.length; i++) {
// IMAGEM IMPAR
ImagePlus ipId = new ImagePlus("Impar", iiImpar);
...
// Reduz a escala da imagem
float ind = (float) Math.pow(0.833333f, 6);
IJ.run(ipId, "Size...", "width=" + (ipId.getWidth() * ind) + " height=" +
(ipId.getHeight() * ind) + " constrain interpolation=Bilinear");
// Escala a posição
int x = Math.round(eyecand[i][0] * ind);
int y = Math.round(eyecand[i][1] * ind);
// Centraliza no caso da imagem passar da borda
if (x < 15)
    x = x + (x - 15);
if (y < 15)
    y = y + (y - 15);
...
// Define a região de interesse
ipId.setRoi(x - 15, y - 15, 30, 30);
// Retira a area definida com o comando anterior
IJ.run(ipId, "Crop", "");
// Padrão utilizado na base de treinamento
IJ.run(ipId, "32-bit", "");
IJ.run(ipId, "8-bit", "");
// repassa a imagem para a SVM
final double res = mfd.classifyImage(ipId.getImage());
...

```

Quadro 16 – Encaminhamento ao módulo de reconhecimento no método detect

### 3.3.1.6 Reconhecimento de padrões

Para reconhecimento do olho e exclusão de falsas regiões, foi utilizada a técnica de aprendizagem de máquina conhecida como SVM. Foi utilizada a biblioteca LibSVM e realizado o treinamento através de um banco de imagens da região dos olhos. O processo de obtenção das imagens e treinamento é descrito nas seções subseqüentes.

#### 3.3.1.6.1 Treinamento da SVM

Para formar a base de treinamento para a SVM foi utilizado um banco de imagens da face humana disponibilizado no projeto *The ORL Database of Faces* (CAMBRIDGE, 1994).

Para delimitação da região dos olhos foi utilizado o projeto Visage (RESTOM, 2006), onde as classes responsáveis pelo reconhecimento da face e extração da região dos olhos foram modificadas. Juntamente utilizou-se o ImageJ para tratamento das imagens.

### 3.3.1.6.2 Extração da região dos olhos

Para formar a base de treinamento para SVM foram extraídas imagens de 30x30 *pixels* que possuem, como centro aproximado, a pupila do olho direito e esquerdo. Das 401 faces do banco de imagens, foram obtidas 515 imagens positivas e 505 imagens negativas. Na Figura 34(a) as imagens positivas com o olho direito e esquerdo, e na Figura 34(b) imagens de outras partes do rosto obtidas empiricamente.

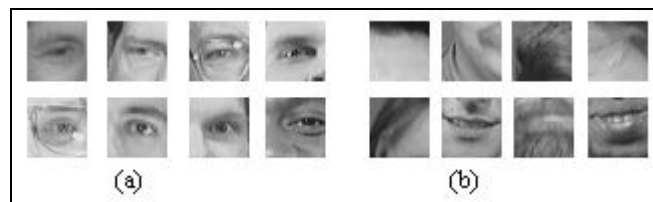


Figura 33 – (a) Imagens do olho direito e esquerdo e (b) de outras partes do rosto

Para obter a região dos olhos, a classe `FaceDetector` presente no trabalho de Restom (2006) teve os padrões do filtro FRSS alterados para obter um melhor resultado com as imagens do banco ORL.

O método responsável pela extração das imagens de treinamento positivas faz o recorte baseado no retorno do método de detecção de faces. Posteriormente as imagens recortadas foram avaliadas visualmente para garantir o sucesso na extração. O ImageJ é utilizado para tratamento com as imagens. Uma parte do código fonte do método `crop` pode ser visto no Quadro 17.

```
public static boolean crop(ImagePlus face, String pathDest, String name){
    // Metodo responsável pela detecção da face e pupila
    mfd.detectAllFaces(face.getImage());
    // Contem a posição obtida da pupila
    final int[] facexy = mfd.getFace();
    if (facexy == null)
        return false; // não detectou os olhos
    // Imagens para recorte do olho
    final ImagePlus ipREye = new ImagePlus("Right Eye",face.getImage());
    final ImagePlus ipLEye = new ImagePlus("Left Eye",face.getImage());
    // Marca a ROI na imagem do olho direito.
    // Tendo como centro a pupila do olho
    ipREye.setRoi(facexy[0]-15, facexy[1]-15, 30,30);
    //Marca a ROI na imagem do olho esquerdo.
    // Tendo como centro a pupila do olho
    ipLEye.setRoi(facexy[2]-15, facexy[3]-15, 30,30);
    // Função Crop do ImageJ recorta a ROI
    IJ.run(ipREye, "Crop", "");
    IJ.run(ipLEye, "Crop", "");
    ...
    return true;}

```

Quadro 17 – Código fonte do método `crop`

Algumas imagens de olhos fechados foram inclusas para forçar o padrão a reconhecer

estes casos como uma região dos olhos.

#### 3.3.1.6.3 LibSVM

LibSVM é uma biblioteca de funções para classificação, regressão e estimação de distribuição de SVM. Possui uma interface com o usuário de simples manipulação. Contem softwares auxiliares como o *svm-scale* que pode ser utilizado para normalizar a base de dados.

O procedimento de utilização envolve a transformação dos dados para o formato de entrada da LibSVM, a escolha aleatória de uma função de *kernel* e de seus parâmetros.

Em *A Practical Guide to Support Vector Classification*, Chang, Hsu e Lin (2007) propõe o seguinte procedimento:

- a) transformar dados para o formato de um software SVM;
- b) utilizar o *svm-scale* para normalizar os dados;
- c) considerar a função *kernel* RBF;
- d) utilizar a validação cruzada para testar os parâmetros;
- e) utilizar o melhor parâmetro  $C$  e  $\gamma$  (*gamma*) para treinar a base de dados;
- f) realizar testes.

#### 3.3.1.6.4 Escala SVM

Segundo Sarle (1997) a padronização de um vetor, na maioria das vezes, significa subtrair uma medida de localização e de divisão por uma medida de escala. Por exemplo, se o vetor contém valores aleatórios com uma distribuição de gaussiana, é possível subtrair a média e dividir pelo desvio padrão, obtendo assim um "padrão normal" em uma variável aleatória com média 0 e desvio padrão 1.

A contribuição de uma entrada depende fortemente da sua variabilidade em relação a outros insumos. Se uma entrada tem um intervalo  $[0,1]$ , enquanto a outra entrada tem uma escala de  $[0,100]$ , a contribuição da primeira entrada à distância é neutralizada pela segunda entrada.

O dimensionamento das entradas para  $[-1,1]$  irá funcionar melhor do que  $[0,1]$ , embora qualquer escala que define a zero a média ou a mediana, seja melhor e mais robusta na

estimação de localização e de escala para as variáveis de entrada com *outliers* extremos.

Chang, Hsu e Lin (2007) comentam sobre as vantagens do escalonamento ao evitar que atributos em maiores intervalos numéricos dominem aqueles em intervalos menores. Falam sobre a vantagem de evitar dificuldades durante o cálculo numérico, pois os valores *kernel* geralmente dependem do produto interno de vetores de características. Também existe a recomendação de utilizar a escala no intervalo de  $[-1, +1]$  ou  $[0, 1]$ .

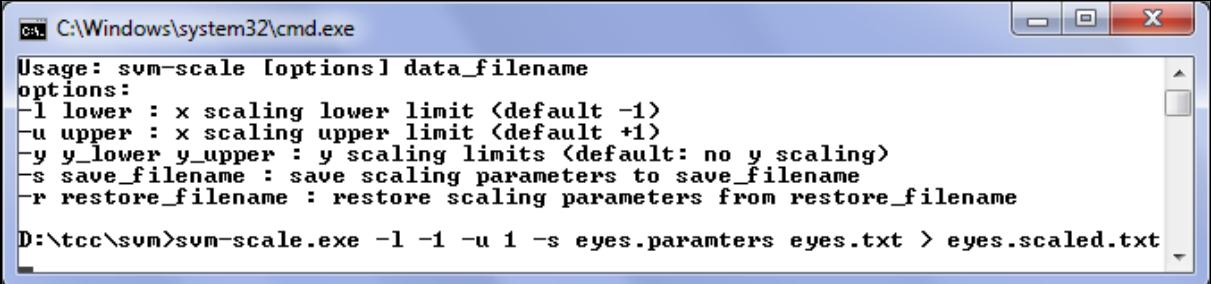
### 3.3.1.6.5 Processo de treinamento da LibSVM

As imagens obtidas para o treinamento foram convertidas para o padrão de entrada da LibSVM. Cada imagem foi convertida em uma matriz unidimensional disposta linearmente. No início de cada linha é informado o grupo ao qual a imagem pertence. Foi utilizado “-1” para identificar as imagens da região dos olhos e “1” para as demais. Como pode ser visto no Quadro 18.

```
-1 1:0.4196078431372549 2:0.4352941176470588... 900:0.5176470588235295
-1 1:0.5607843137254902 2:0.5529411764705883... 900:0.6705882352941176
...
1 1:0.4549019607843137 2:0.4196078431372549... 900:0.7764705882352941
1 1:0.1529411764705882 2:0.1764705882352941... 900:0.7568627450980392
...
```

Quadro 18 – Arquivo de entrada da LibSVM

Após transformar dados para o formato de entrada, é preciso padronizar os mesmos. A LibSVM disponibiliza a ferramenta `svm-scale`. Como pode ser visto na Figura 35 a ferramenta permite que seja definido o intervalo de escala. Os parâmetros obtidos durante o processo podem ser armazenados em um arquivo e são utilizados posteriormente para padronizar as entradas de teste. A execução gera sua saída, que é a base de entrada após o escalonamento, em tela e deve ser direcionada para um arquivo.



```
C:\Windows\system32\cmd.exe
Usage: svm-scale [options] data_filename
options:
-l lower : x scaling lower limit (default -1)
-u upper : x scaling upper limit (default +1)
-y y_lower y_upper : y scaling limits (default: no y scaling)
-s save_filename : save scaling parameters to save_filename
-r restore_filename : restore scaling parameters from restore_filename
D:\tcc\svm>svm-scale.exe -l -1 -u 1 -s eyes.paramters eyes.txt > eyes.scaled.txt
```

Figura 34 – Padronização de dados com `svm-scale`

A função *kernel* escolhida para o treinamento da base é a RBF. Esta função necessita de dois parâmetros  $C$  e  $\gamma$ . Segundo recomendam Chang, Hsu e Lin (2007) deve ser formada



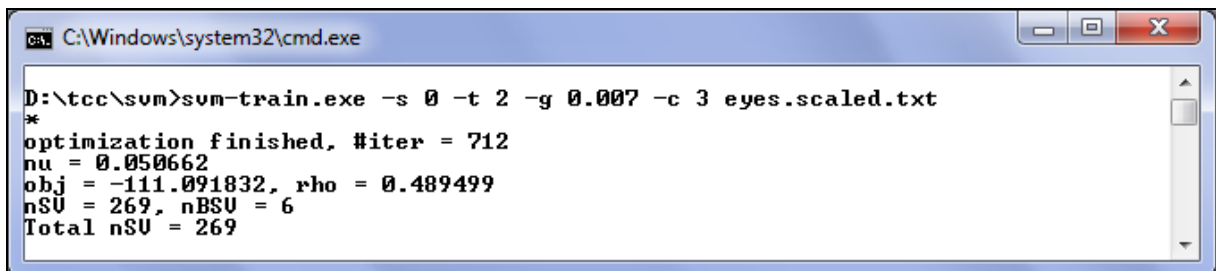
uma seqüência de testes utilizando a validação cruzada<sup>4</sup>. Basicamente pares de  $C$  e  $\gamma$  são julgados e aquele com a melhor precisão de validação cruzada é escolhido. Nos testes realizados pelos pesquisadores revelou-se que tentar crescer exponencialmente os valores de  $C$  e  $\gamma$  é um método prático para identificar bons parâmetros. O método também pode ser facilmente paralelizado, pois  $C$  e  $\gamma$  são independentes.

As interações realizadas até a obtenção do melhor valor para  $\gamma$  são apresentadas na Tabela 1. Iniciou-se pelo valor recomendado de  $2^{-15}$  e atingiu-se o equilíbrio na iteração 7 com o  $\gamma$  em  $2^{-3}$ . Buscou-se então por valores próximos até encontrar um novo equilíbrio na iteração 14 com  $7^{-3}$  como o valor final do  $\gamma$ . A variável  $C$  foi previamente obtida.

Tabela 1 – Processo de obtenção do  $\gamma$

PROCESSO DE OBTENÇÃO DO $\gamma$			
Iteração	$\gamma$	C	Acurácia
1	$2^{-15}$	3	74,5726%
2	$2^{-13}$	3	74,5726%
3	$2^{-11}$	3	72,2080%
4	$2^{-9}$	3	78,2764%
5	$2^{-7}$	3	78,2764%
6	$2^{-5}$	3	91,0969%
7	$2^{-3}$	3	98,2906%
8	$2^{-1}$	3	66,6667%
9	$1^{-3}$	3	97,2222%
10	$3^{-3}$	3	98,6467%
11	$4^{-3}$	3	98,7892%
12	$5^{-3}$	3	99,2877%
13	$6^{-3}$	3	99,1453%
14	$7^{-3}$	3	99,2877%
15	$8^{-3}$	3	99,2165%

O treinamento da LibSVM foi efetuado utilizando a função `svm-train`. Como pode ser visto na Figura 36 os parâmetros informados são respectivamente o tipo de SVM (C-SVC), a função *kernel*(RBF), o  $\gamma$  e o  $C$  obtidos e a base de dados.



```

C:\Windows\system32\cmd.exe
D:\tcc\svm>svm-train.exe -s 0 -t 2 -g 0.007 -c 3 eyes.scaled.txt
*
optimization finished, #iter = 712
nu = 0.050662
obj = -111.091832, rho = 0.489499
nSV = 269, nBSV = 6
Total nSV = 269
  
```

Figura 35 – Execução do `svm-train`

<sup>4</sup> Validação cruzada é uma técnica que propicia estimar a capacidade de generalização de um classificador. Essa técnica consiste em dividir o conjunto de treinamento em partes aproximadamente iguais. Uma dessas partes será o subconjunto a ser utilizado para teste. Cada execução do experimento altera esse conjunto (ROCHA, 2005).

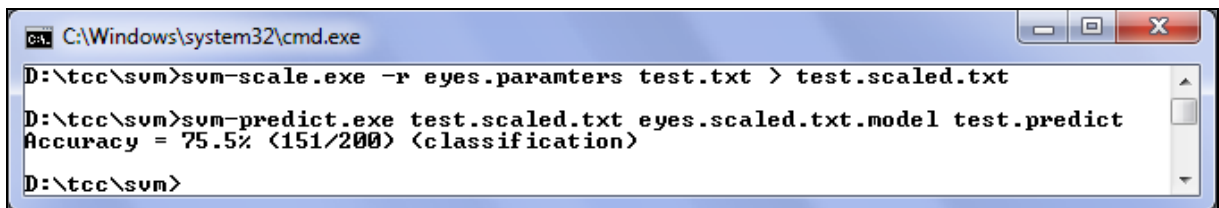
Uma parte do arquivo gerado pelo treinamento pode ser visto no Quadro 19. Os campos descrevem os parâmetros que configuração da LibSVM.

```
svm_type c_svc
kernel_type rbf
gamma 0.007
nr_class 2
total_sv 269
rho 0.489499
label -1 1
nr_sv 139 130
SV
2.560612889194792 1:-0.0536585...
```

Quadro 19 – Arquivo gerado pelo `svm-train`

Após a execução do processo de treinamento é possível medir a acurácia da SVM ao aplicar entradas que não participaram do processo de treinamento. Para formar a base de teste foi utilizado um banco de imagens da face humana disponibilizado no projeto *FEI Face Database* (FEI, 2006). A região dos olhos foi extraída com o método descrito na sessão 3.3.1.6.2.

Como pode ser visto na Figura 37, utilizou-se os parâmetros gerados no treinamento da base para normalizar a base de teste. A acurácia é obtida através do método `svm-predict` ao passar a base de teste normalizada seguida do modelo gerado pelo processo de treinamento.



```
C:\Windows\system32\cmd.exe
D:\tcc\svm>svm-scale.exe -r eyes.paramters test.txt > test.scaled.txt
D:\tcc\svm>svm-predict.exe test.scaled.txt eyes.scaled.txt.model test.predict
Accuracy = 75.5% (151/200) (classification)
D:\tcc\svm>
```

Figura 36 – Execução do `svm-predict`

### 3.3.1.6.6 Adaptação do algoritmo de escala

O algoritmo da ferramenta `svm-scale` esta contido na LibSVM apenas para processamento em modo de arquivo de entrada. Foi preciso adaptar o algoritmo para atender a recomendação sobre o escalonamento dos dados.

A classe `MySvmScale` contem o método `scale` que recebe os pixels da imagem em um `String` no mesmo formato do arquivo de entrada da LibSVM.

### 3.3.1.6.7 Classificação da região dos olhos

O método `classifyImage` se encarrega de classificar a imagem definida no processamento gráfico como uma ROI. A imagem é convertida em tons de cinza, transformada para o formato de entrada e escalada pelo `MySvmScale` e então enviada ao método de classificação. O método descrito pode ser visto no Quadro 20.

```
public double classifyImage(Image eyeImg) {
    pixels = new int[fWidth * fHeight];
    grayPixels = new int[fWidth * fHeight];
    // transfere os pixels para uma array unidimensional
    pixels = ImageProcessing.extractPixels(eyeImg, 0, 0, fWidth, fHeight,
    pixels);
    // faz a conversão para grayscale e transfere os pixels para uma array
    unidimensional
    grayPixels = ImageProcessing.toGrayscale(pixels, grayPixels);
    // Carrega o array template que sera utilizado no processo de
    classificação
    svm_node[] template = createSVMNodeTemplate();
    // Prepara entrada para o algoritmo que escala os dados
    String toScale = "0 ";
    for (int i = 0; i < 900; i++)
        toScale += ((i + 1) + ":" + template[i].value + " ");

    svm_node[] templateScaled = null;
    try {
        templateScaled = MySvmScale.scale(toScale);
    } catch (IOException e) {...}
    // Realiza o processo de classificação
    double result = classify(templateScaled);
    return result;}

```

Quadro 20 – Código fonte do método `classifyImage`

O padrão de entrada da LibSVM consiste em uma matriz unidimensional do tipo `svm_node`. O método responsável por sua criação pode ser visto no Quadro 21.

```
private svm_node[] createSVMNodeTemplate() {
    svm_node node;
    svm_node template[] = new svm_node[900];
    for (int y = 0; y < 30; y++) {
        for (int x = 0; x < 30; x++) {
            node = new svm_node();
            node.index = y * 30 + x + 1;
            node.value = grayPixels[y * fWidth + x] / 255d;
            template[y * 30 + x] = node;
        }
    }
    return template;}

```

Quadro 21 – Código fonte do método `createSVMNodeTemplate`

O método de classificação (Quadro 22) recebe a matriz do tipo `svm_node` e encaminha para o método `svm.svm_predict_values` da LibSVM. O resultado é obtido multiplicando o retorno na matriz `decValues`, passada como parâmetro, com o *label* definido para o valor positivo no arquivo de configuração.

```
private double classify(svm_node[] template) {
    double cResults;
    double[] decValues = new double[1];
    svm.svm_predict_values(Control.getModel(), template, decValues);
    cResults = decValues[0] * Control.getModel().label[0];
    return cResults;}

```

Quadro 22 – Código fonte do método `classify`

### 3.3.1.7 Movimento do ponteiro do mouse

O controle do movimento do mouse foi realizado por meio de uma classe nativa do Java, a `java.awt.Robot`. Para abstrair a interação foi criada classe `EyeMouse`, que realiza o controle do movimento e do *click*. Esta classe se comporta como um processo paralelo.

#### 3.3.1.7.1 Movimento

O movimento é realizado conforme a direção e a distância da mudança no movimento do olho em relação ao quadro anterior. Para transferir o movimento do olho para o ponteiro do mouse é preciso acertar as proporções. No Quadro 23 é apresentado o método que recebe o valor da diferença deste movimento, e realiza o cálculo de proporção.

```
public void setMouseXY(int x, int y) {
    this.x = (x * (Control.getScreenWidth()/Control.getEyeAvgWidth())) +
    xIni;
    this.y = (y * (Control.getScreenHeight()/Control.getEyeAvgHeight())) +
    yIni;
    ...
}

```

Quadro 23 – Código fonte do método `setMouseXY`

O tamanho da tela é obtido ao iniciar o sistema. O tamanho do olho foi obtido empiricamente ao analisar as imagens da base de treinamento.

A transferência do movimento não é efetuada diretamente para a nova posição. É utilizada a equação paramétrica da reta<sup>5</sup> para criar a impressão de um movimento suave. Um trecho do código responsável pelo movimento pode ser visto no Quadro 24.

---

<sup>5</sup> As equações paramétricas são formas de representar as retas através de um parâmetro, ou seja, uma variável irá fazer a ligação de duas equações que pertencem a uma mesma reta (MUNDO..., 2006).

```

public void run() {
    while (true) {
        int xN = 0, yN = 0;
        for (float t = 0.1f; t < 1; t += 0.1f) {
            // Equação paramétrica da reta
            xN = Math.round(xIni + (x - xIni) * t);
            yN = Math.round(yIni + (y - yIni) * t);
            // Se a posição for diferente da anterior
            if (xN != xIni || yN != yIni)
                r.mouseMove(xN, yN);
            ...
        }
        // Armazena a nova posição
        xIni = xN;
        yIni = yN;
    }
}

```

Quadro 24 – Código fonte do método `run`

### 3.3.1.7.2 Click

O processamento do *click* ocorre quando uma região previamente detectada como um olho, não está incluída como uma região candidata no quadro posterior. Se esta região estiver definida como o olho, e ainda estiver 35% distante da região anterior, valor obtido empiricamente ao observar o sistema operando nas distâncias recomendadas, é caracterizado o *click*.

Um trecho do método responsável pelo *click* pode ser visto no Quadro 25.

```

if (foundEyeLastFrame) {
    // Verifica o quão distante a nova região esta da anterior
    float pctx = 100f * ((float) (lastPosX - fx) / (float) fx);
    if (pctx < 0)
        pctx *= -1;
    if (pctx > Control.getPctClick()) {
        Control.getFrame().setStatusMessage("Click simples!");
        EyeMouse.getInstance().click();
        // Emite beep
        Toolkit.getDefaultToolkit().beep();
        // Zera contador
        cxy = 0;
    }
    ...
}

```

Quadro 25 – Método responsável pela detecção do *click*

O método responsável pelo *click* em `EyeMouse` pode ser visto no Quadro 26.

```

public void click(){
    r.mousePress(InputEvent.BUTTON1_MASK);
    r.mouseRelease(InputEvent.BUTTON1_MASK);
}

```

Quadro 26 – Execução do *click*

### 3.3.2 Operacionalidade da implementação

Esta seção tem por objetivo mostrar a operacionalidade da implementação em nível de usuário. Nas próximas seções serão abordadas as principais funcionalidades da ferramenta.

#### 3.3.2.1 Escolhendo um dispositivo de captura de imagem

Ao se iniciar o projeto, caso o usuário possuir mais de um dispositivo de captura de imagem instalado no seu computador, a primeira tela apresentada para o usuário é a tela de seleção do dispositivo de captura de imagem como pode ser visto na Figura 38.

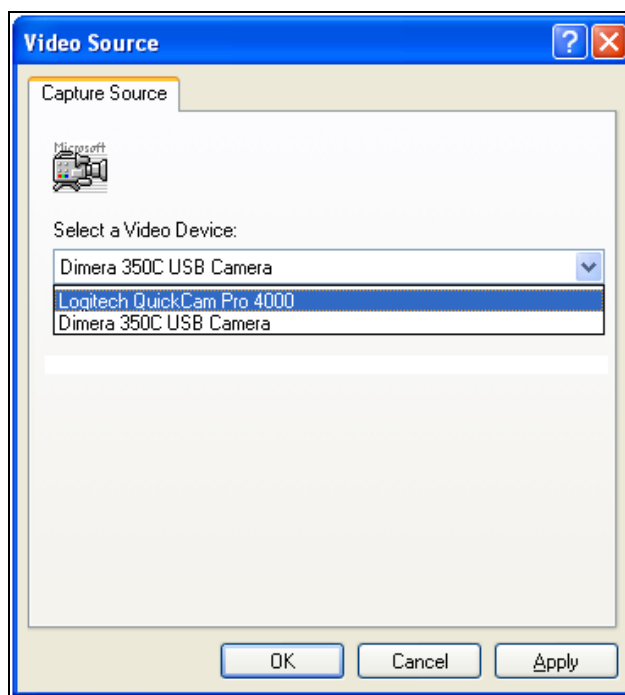


Figura 37 – Tela selecionar dispositivo de captura de imagem

Ao se selecionar o botão OK na interface, ou se o usuário possuir somente um dispositivo de captura de imagem instalado no seu computador, o sistema seleciona este dispositivo e prossegue.

#### 3.3.2.2 Escolhendo a porta de comunicação do dispositivo flash

Ao se iniciar o projeto, caso exista mais de uma porta de comunicação do tipo COM

disponível no computador, a segunda tela apresentada para o usuário é a tela de seleção da porta COM como pode ser visto na Figura 39.

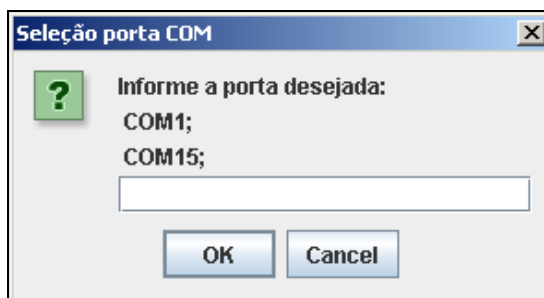


Figura 38 – Tela selecionar dispositivo de captura de imagem

Ao informar a porta desejada e selecionar o botão OK na interface, ou se o usuário possuir somente uma porta COM instalada, o sistema seleciona esta porta e se tudo ocorrer normalmente deverá ser apresentado à tela principal do sistema, conforme mostra a Figura 40.

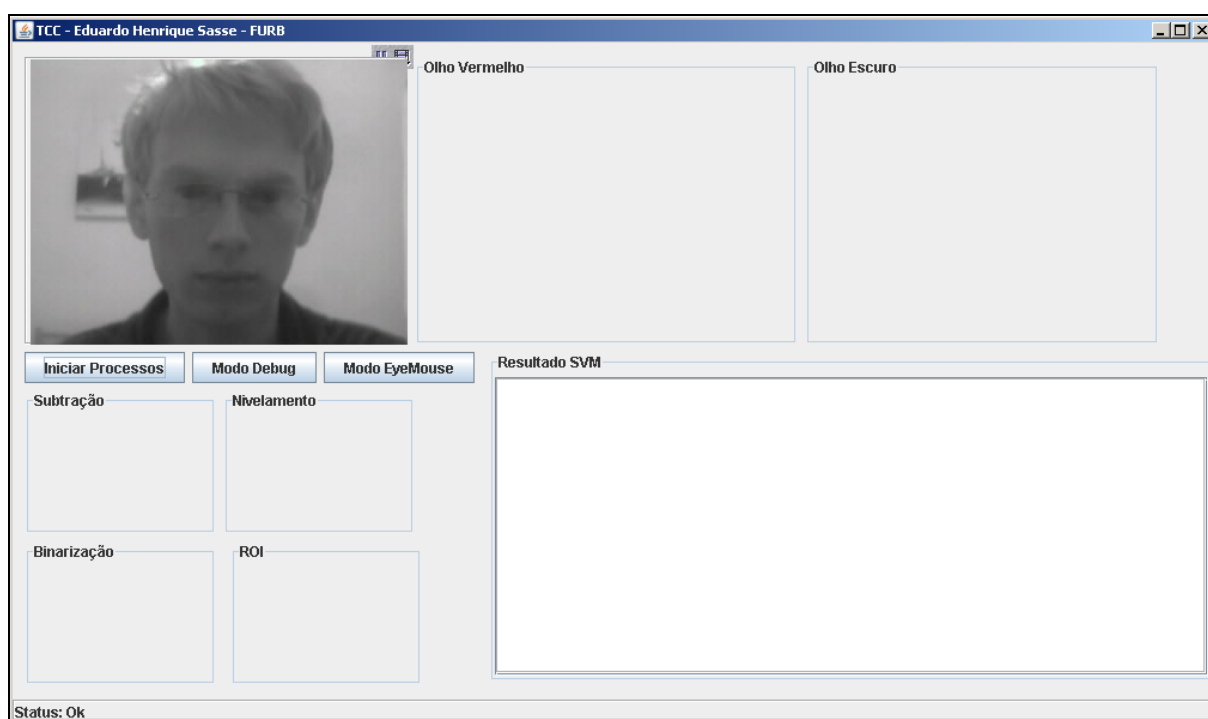


Figura 39 – Tela principal do sistema

Caso algum dos dispositivos não for encontrado ou o dispositivo não for adequado para o funcionamento correto do sistema, são apresentadas as seguintes mensagens ao usuário. Em caso de problemas com a *webcam* será apresentada a mensagem vista na Figura 41.



Figura 40 – Mensagem de *webcam* não reconhecida

Para o caso da porta serial informada não ser a correta, é realizada a validação do dispositivo *flash*. Caso esse não responda corretamente é considerado um dispositivo *flash* inválido e será apresentada a mensagem vista na Figura 42.

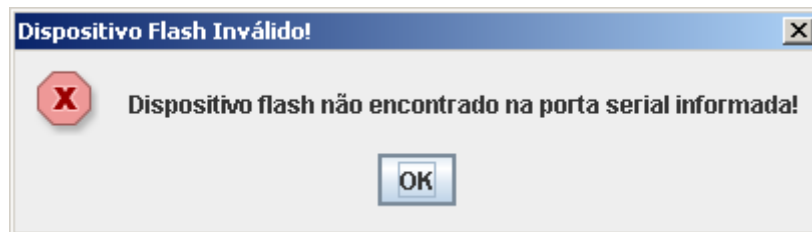


Figura 41 – Mensagem de Dispositivo *Flash* inválido

Caso nenhuma porta COM estiver disponível será apresentada a mensagem vista na Figura 43.

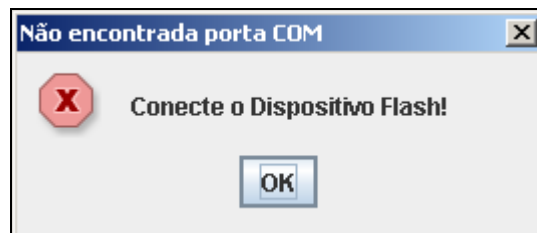


Figura 42 – Mensagem de porta COM não reconhecida

### 3.3.2.3 Recursos da ferramenta

A ferramenta disponibiliza na sua interface três botões do tipo `JToggleButton` que podem ser vistos na Figura 44. O primeiro `INICIAR PROCESSOS` é responsável pelo início de todos os processos referentes ao reconhecimento da pupila. O botão `MODO DEBUG` demonstra a execução passo a passo de cada etapa dos processos de reconhecimento. O botão `MODO EYEMOUSE` habilita a transferência do movimento gerado pela pupila para o ponteiro do mouse.



Figura 43 – Mensagem de porta COM não reconhecida



### 3.3.2.4 Funcionamento da ferramenta

Após a tela principal (Figura 40) ser apresentada para o usuário, este pode iniciar o processo de detecção da pupila ao pressionar o botão `INICIAR PROCESSOS` na interface do sistema. De forma aparente nada vai acontecer porem todos os processos de reconhecimento estarão iniciados e executando. Ativando o botão `MODO DEBUG` será possível acompanhar estes processos, conforme pode ser visto na Figura 45.

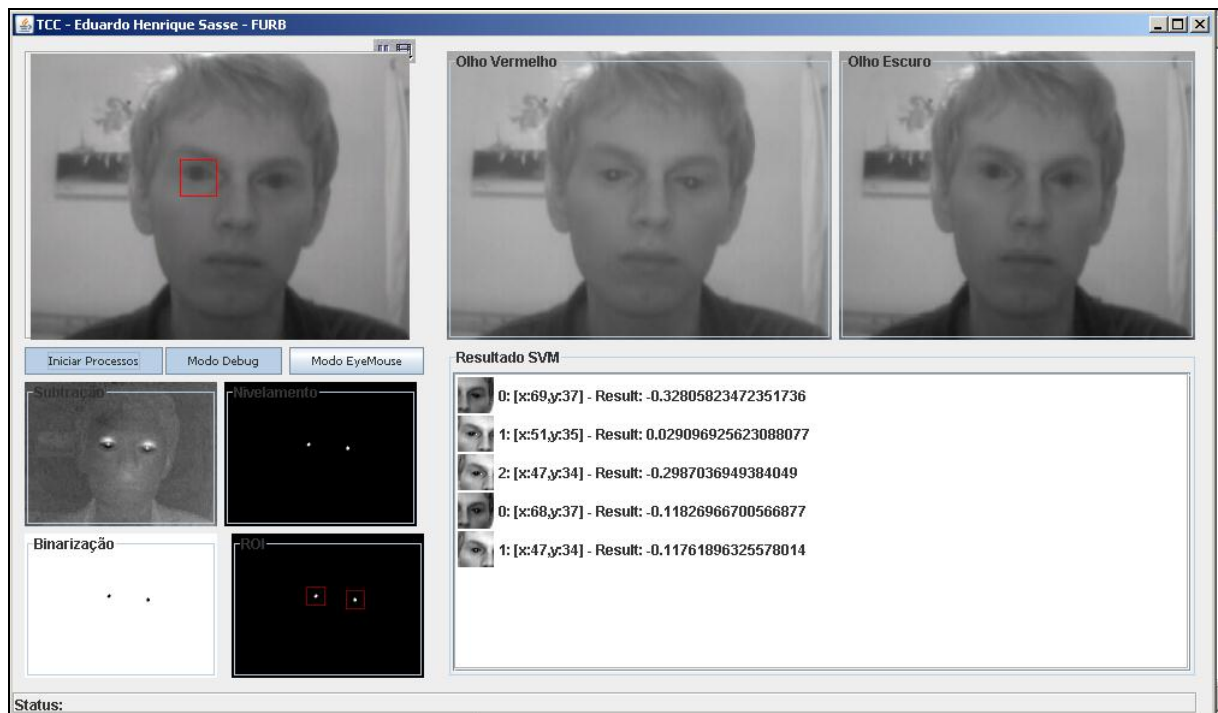


Figura 44 – Início de processos no modo *debug*.

O primeiro painel traz o vídeo proveniente da *webcam*, o quadrado em seu interior demarca o olho escolhido para o movimento do mouse. Como padrão no desenvolvimento utilizou-se o olho detectado na menor posição do eixo *x*.

Nos painéis `OLHOS VERMELHOS` e `OLHOS ESCUROS` é possível acompanhar os quadro capturados de forma alternada através dos acionamentos do dispositivo *flash*.

Os painéis `SUBTRAÇÃO`, `NIVELAMENTO`, `BINARIZAÇÃO` e `ROI` permitem o acompanhamento no processamento das imagens, para obtenção dos pontos candidatos a pupila.

No painel `RESULTADO SVM` esta inclusa a lista com as imagens provenientes do ROI, já classificadas pela LibSVM, contendo inclusive o resultado da mesma.

A ação do *click* é apresentado através de um sinal sonoro, e também uma mensagem na barra de *status*, conforme demonstra a Figura 46.

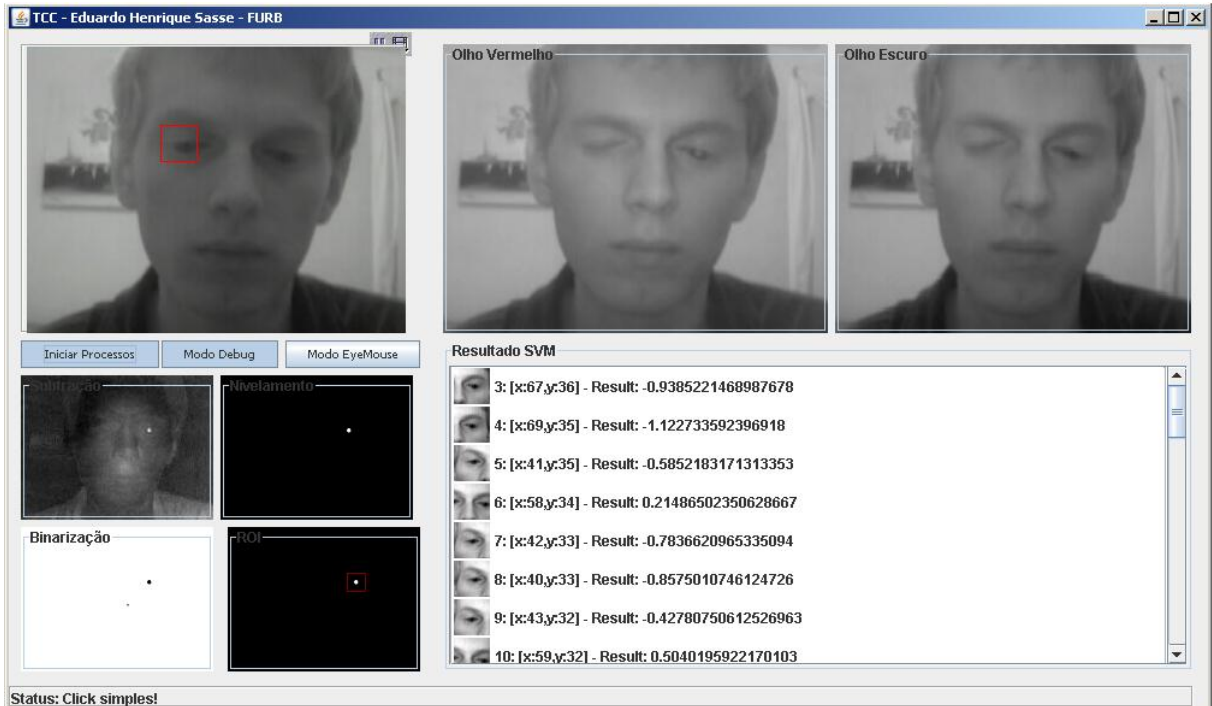


Figura 45 – Execução do *click*

Ao habilitar a opção `MODO EYEMOUSE` o sistema inicia a transferência do movimento para o ponteiro do mouse. Como pode ser visto na Figura 47 o ponteiro do mouse é movimentado do Botão 01 para o Botão 02.

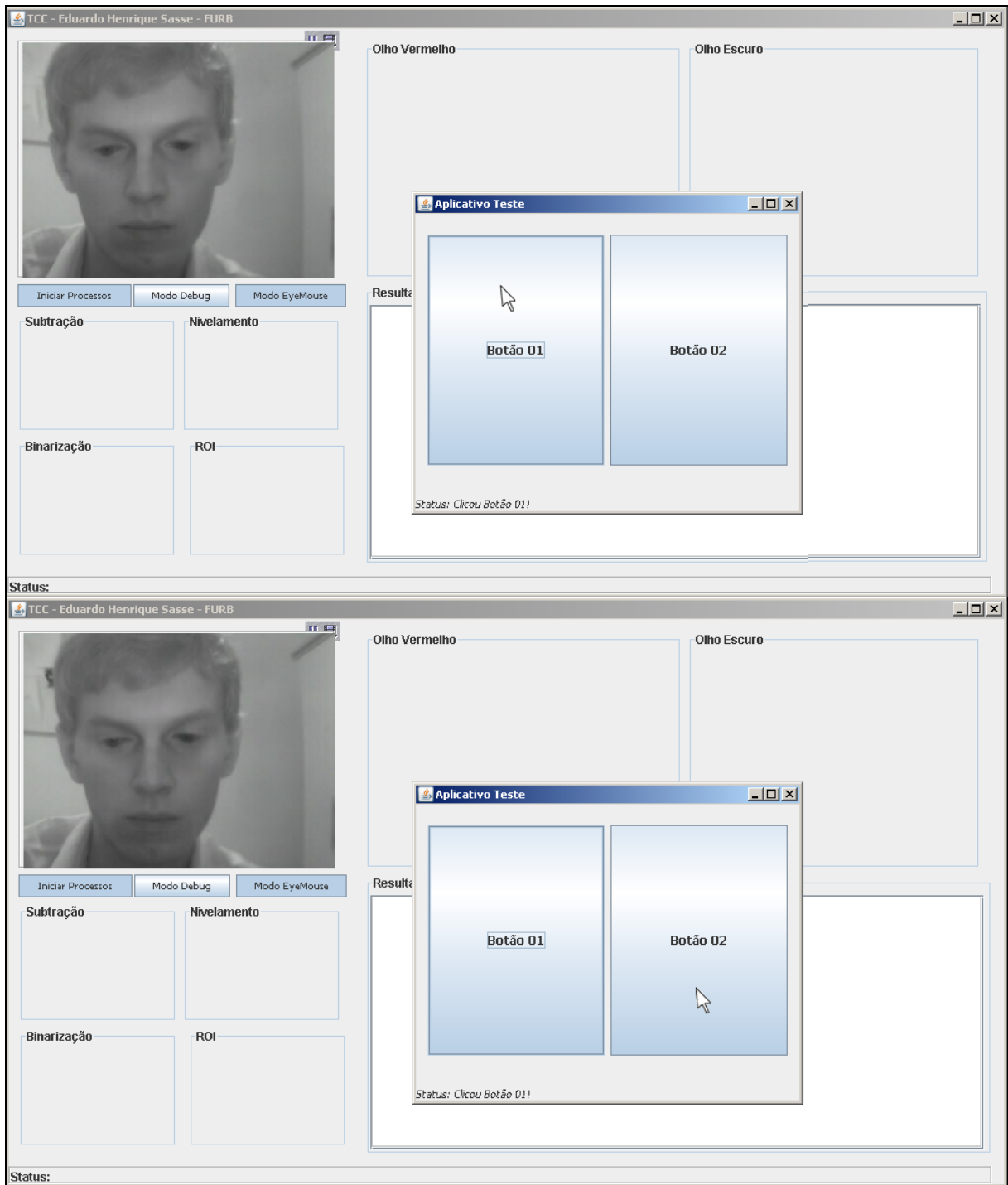


Figura 46 – Movimentação do ponteiro MODO EYEMOUSE

### 3.4 RESULTADOS E DISCUSSÃO

Esta seção tem por objetivo apresentar os resultados e discussões obtidos através de testes realizados na ferramenta. Nas próximas seções serão abordadas questões de

desempenho e precisão no reconhecimento.

### 3.4.1 Tempo de processamento

Ao se iniciar o projeto notou-se que a velocidade de processamento era um dos fatores críticos a ser tratado. No teste foi utilizado um notebook com processador AMD TurionX2 de 2.0Ghz com 2 Gb de memória RAM.

Para verificar o tempo de processamento de cada etapa do processo de reconhecimento anteriormente descrito, foram realizadas tomadas de tempo no início e no final de cada processamento, a fim de verificar qual é a etapa em que está sendo consumido o maior e o menor tempo de processamento. Utilizou-se a função `System.currentTimeMillis()` nativa do Java. Os resultados dos testes estão apresentados no Tabela 2.

Tabela 2 – Processamento de imagens  
PROCESSAMENTO DE IMAGENS

Roi	Video(ms)	RXTX(ms)	Subtração(ms)	Nivelamento(ms)	ROI(ms)	Escala(ms)	Classf.(ms)	Total(ms)
2	15	0	15	0	4	14	500	31
4	16	0	16	0	6	16	767	61
8	16	0	16	0	5	32	1280	94
16	15	0	16	0	6	38	2885	257

Fica evidente que o algoritmo adaptado de `svm-scale` é o gargalo de todo processamento. Ao desconsiderar este método diminuimos a acurácia do processo de classificação da LibSVM, porém obtém-se um algoritmo que realiza a detecção em menos de 100ms. Em comparação com o projeto Visage de RESTOM (2006), que tem um tempo de reconhecimento médio de 175ms, e que não utiliza o escalonamento da imagem antes do reconhecimento, e também considerando a passagem de apenas uma imagem através do processo de classificação, o método presente neste trabalho se mostrou 221% mais rápido.

### 3.4.2 Reconhecimento

Com o objetivo de testar o comportamento da ferramenta na tarefa de reconhecimento da posição das pupilas com diversas pessoas, foi organizado um grupo de 8 pessoas com diferentes características, tais como: cabelo comprido e curto, com e sem a presença de barba,

indivíduos com pele clara e morena, do sexo masculino e feminino, entre outras características únicas. A Figura 48 demonstra a tela principal da ferramenta durante a realização de um destes testes.

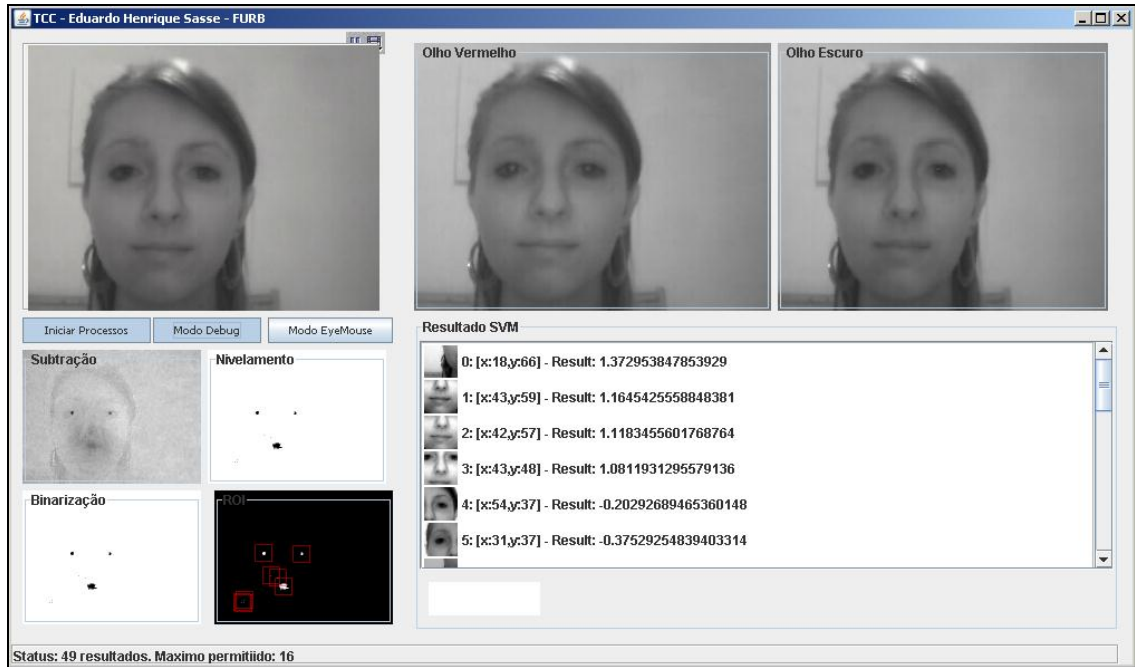


Figura 47 – Tela principal do sistema

Os testes foram realizados em ambiente interno com iluminação superior e durante o dia e consistem de uma verificação dos resultados de classificação após uma interação de 10 segundos com a ferramenta. As porcentagens de acerto nestes testes podem ser vistos no Quadro 28.

Usuário	Características	Sexo	Percentual de acerto		Média
			Olho	Outro	
1	Cabelo comprido escuro liso amarrado, pele clara, 22 anos	Feminino	100%	99%	<b>99,5%</b>
2	Cabelo curto claro, pele clara, 22 anos	Masculino	100%	100%	<b>100%</b>
3	Utilizando boné, pele clara, 28 anos	Masculino	100%	96%	<b>98%</b>
4	Cabelo comprido escuro liso solto, pele escura, 29 anos	Feminino	73%	98%	<b>85,5%</b>
5	Cabelo comprido escuro encaracolado solto, pele clara, olho pequeno, 21 anos	Feminino	84%	97%	<b>90,5%</b>
6	Cabelo comprido claro liso solto, pele clara, 21 anos	Feminino	100%	98%	<b>99%</b>
7	Cabelo curto grisalho, pele clara, 62 anos	Masculino	94%	98%	<b>96%</b>
8	Cabelo curto escuro, pele clara, 44 anos	Feminino	98%	92%	<b>95%</b>

Quadro 27 – Resultados dos testes de reconhecimento

Durante os testes observou-se que o reconhecimento da região dos olhos e outras regiões se mostrou extremamente satisfatória. Em alguns momentos a região da boca e partes da testa apresentam falsos positivos, porem muito próximos a mediana.

### 3.4.3 Movimento do ponteiro

Tomando as proporções existentes entre a área de movimento do globo ocular e área da tela, mesmo trabalhando em uma resolução de 800x600 pixels, por menor que seja o movimento no globo ocular, inclusive muitas vezes movimentos não intencionais, refletem em significativas alterações na trajetória do ponteiro.

Manter o cursor parado, mesmo com o usuário estático, é outra dificuldade, devido principalmente a imprecisão dada pela distancia e baixa resolução da *webcam* utilizada. Acarretando em oscilações no centro detectado quadro a quadro.

Contudo a ferramenta se mostrou eficaz em realizar o movimento do ponteiro. A utilização da equação paramétrica auxiliou impedindo que um efeito de salto do ponteiro pela tela viesse a ocorrer.

### 3.4.4 Velocidade do LED

O tempo de acionamento e desligamento dos LEDs utilizados no projeto se mostrou alto demais. No trabalho desenvolvido pelos pesquisadores Essa, Flickner e Haro (2000) foi utilizado um LED IR modelo HSDL-4200 Series (HSDL, 2000). O tempo de acionamento informado no *datasheet* para este LED é de 40ns. O LED HSDL-4200 não foi encontrado para utilização neste projeto.

Utilizando uma câmera configurada para obter 40 quadros por segundo, ou seja um quadro a cada 25ms. Realizou-se uma análise quadro a quadro e obtiveram-se os tempos de acionamento e desligamento para o modelo de LED utilizado.

Para que o LED desligue completamente são necessários em torno de 50ms. O acionamento, para que o brilho seja total, precisa de cerca de 100ms. Totalizando assim um tempo de 150ms para que se possa obter um novo quadro que contenha o efeito esperado.

Este fato conjunto a uma limitação dada pela quantidade real de quadros capturados pela *webcam*, que se mostrou em média de 20 FPS, aumenta o tempo de espera em mais 50ms (1/20). Portanto a soma destes tempos limitou o sistema a operar uma espera de 200ms após o acionamento de cada um dos conjuntos de LEDs. Como pode ser visto na Figura 49, o sistema opera relativamente bem com este tempo de espera.



Figura 48 – Funcionamento do sistema com espera de 200ms

Reduzindo o tempo de espera para 175ms o sistema já apresenta um problema, como pode ser visto no painel OLHO ESCURO na Figura 50.



Figura 49 – Funcionamento do sistema com espera de 175ms

Reduzindo novamente o tempo de espera, agora para 100ms, o sistema já apresenta os dois painéis OLHO VERMELHO e OLHO ESCURO com ambos os conjuntos de LEDs ainda acesos, como pode ser visto na Figura 51.



Figura 50 – Funcionamento do sistema com espera de 100ms

### 3.4.5 Posicionamento do usuário

Observou-se que para o melhor funcionamento da ferramenta o usuário deve posicionar-se a uma distância superior a 40cm e inferior a 90cm. Caso a distância seja inferior a 40cm ocorre o efeito que pode ser visto na Figura 52. Onde o nariz concentra o brilho do eixo interior, e não recebe a iluminação necessária dos LEDs do eixo exterior. Acaba assim tendo maior destaque na subtração das imagens.

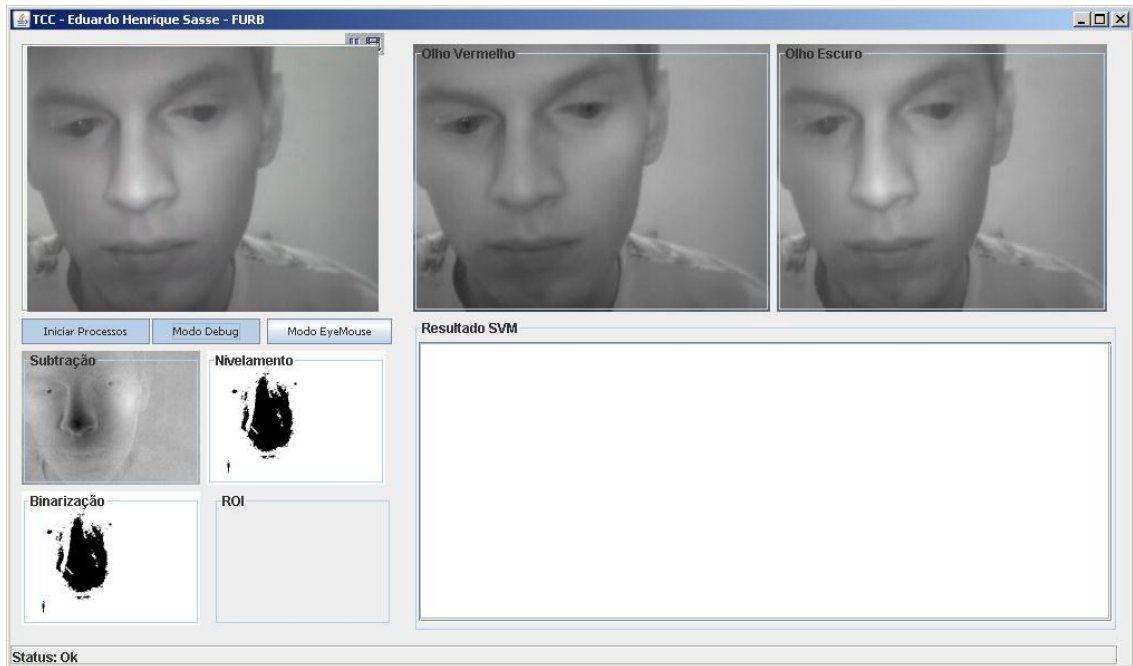


Figura 51 – Destaque do nariz do usuário devido a proximidade da iluminação

### 3.4.6 Óculos

Quando o usuário faz a utilização de óculos verifica-se o reflexo do dispositivo *flash* sobre a lente dos mesmos. Dessa forma o ponto refletido fica mais brilhante que a pupila, como pode ser visto na Figura 53.

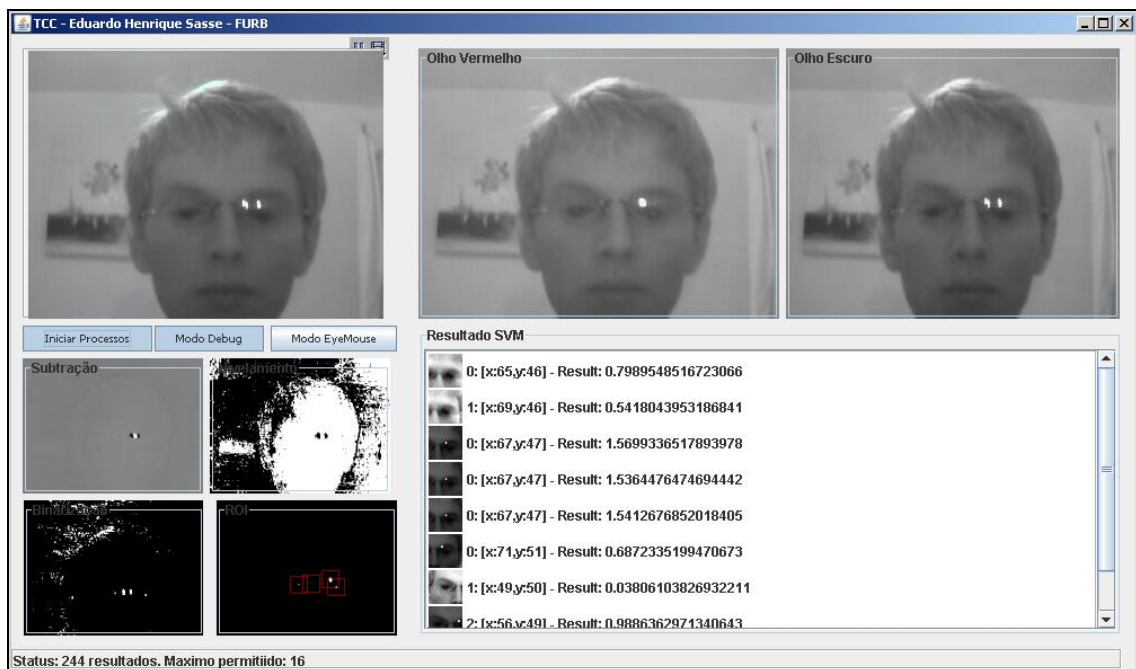


Figura 52 – Reflexo do *flash* sobre a lente do óculos

Para evitar o efeito basta posicionar-se um pouco acima ou abaixo do eixo reto dos



conjuntos de LEDs, assim o processo ocorre normalmente como pode ser visto na Figura 54, onde o usuário está utilizando óculos.

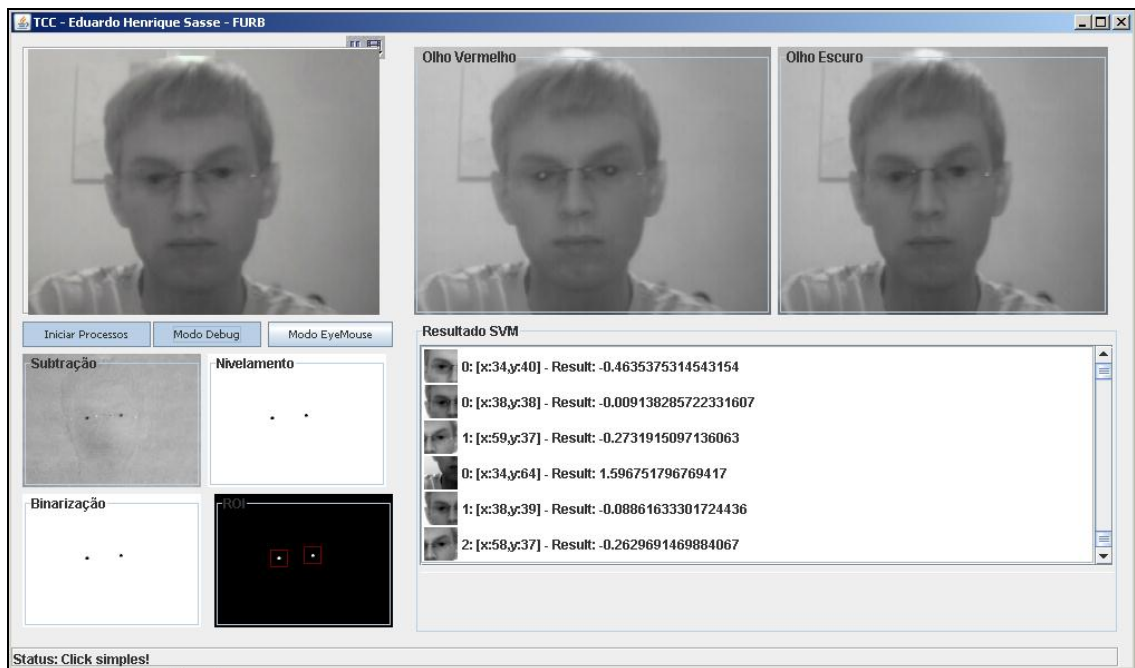


Figura 53 – Sem reflexo do *flash* sobre a lente do óculos

### 3.4.7 Fontes de Iluminação

Nos testes foram exploradas as condições de luminosidade e o posicionamento da luz no ambiente. Os testes realizaram-se em ambiente fechado, pois a exposição a iluminação solar cega completamente a *webcam*, impedindo assim a utilização em ambiente externo. Para os testes foram utilizadas uma lâmpada incandescente de 40W e uma luminária fluorescente de 14W.

A iluminação utilizando a lâmpada fluorescente não alterou significativamente o efeito, independentemente de seu posicionamento. A Figura 55 apresenta uma visão dos posicionamentos da lâmpada incandescente em relação a *webcam*.

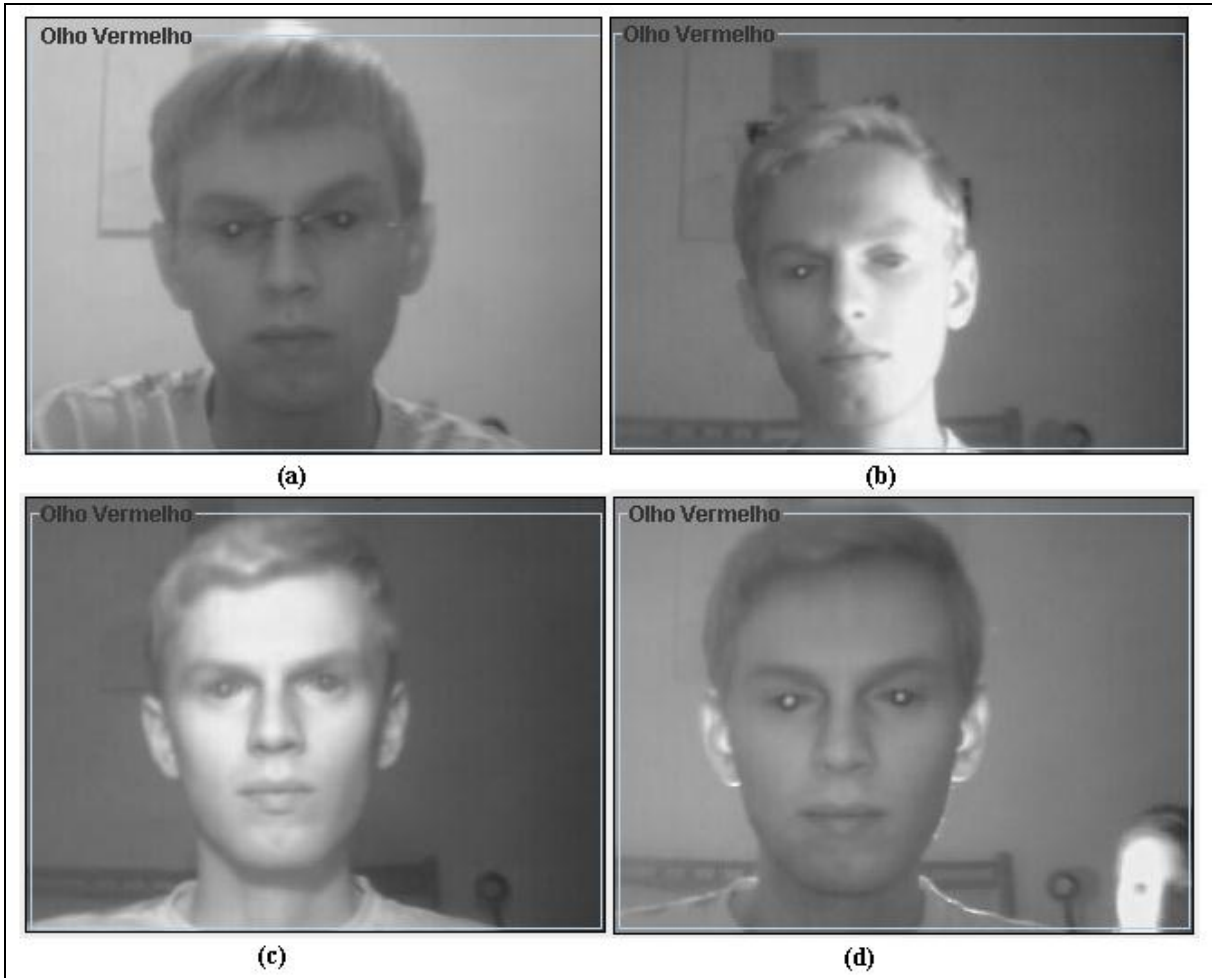


Figura 54 – (a) Iluminação superior; (b) lateral; (c) frontal e (b) traseira

Com a iluminação superior – Figura 55(a) – obtém-se os resultados mais favoráveis tanto na geração do efeito olhos vermelhos quanto para reconhecimento através da LibSVM. A iluminação lateral – Figura 55(b) – impede a geração do efeito olhos vermelhos na parte iluminada da face, a outra parte não foi afetada pois o próprio dispositivo *flash* fez iluminação necessária. A iluminação frontal – Figura 55(c) – traz dois principais impactos na detecção, o tamanho da pupila é reduzido devido a iluminação direta ao olhos e clareamento do rosto faz com que a LibSVM não tenha sucesso no reconhecimento, principalmente quando o escalonamento esta desabilitado. A iluminação traseira – Figura 55(d) – apresenta resultados semelhantes ao da iluminação superior pelo fato de equalizar a iluminação no ambiente, deixando para o próprio dispositivo *flash* a iluminação da face.

Outro teste foi realizado colocando uma fonte luminosa que emita radiação infravermelha em foco, como visto na Figura 56, a iluminação sobrepões o efeito olhos vermelhos impedindo o reconhecimento.

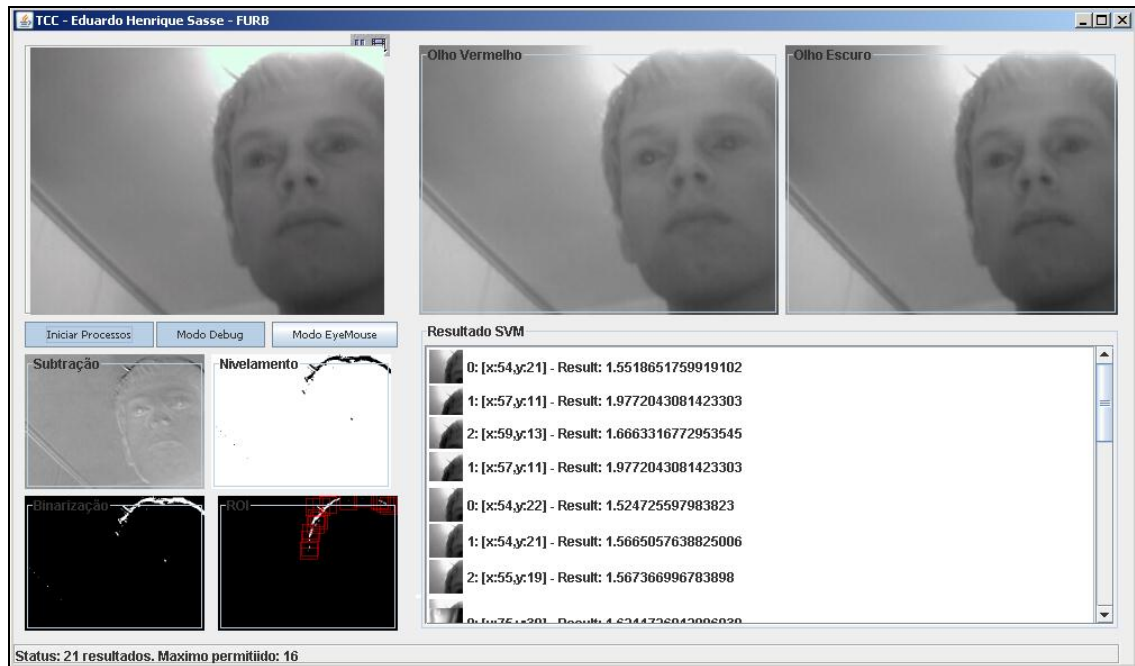


Figura 55 – Fonte de iluminação interferindo no efeito olhos vermelhos

Em um quarto escuro, tendo como única fonte de iluminação a emitida pelo dispositivo *flash*, o efeito olhos vermelhos é gerado corretamente, porém ocorre uma alteração no tempo de abertura do obturador da *webcam*, acarretando uma redução no numero de quadros para 6 FPS em média. Com isto o efeito acaba não ocorrendo de forma sincronizada, impedindo que a subtração dos quadros indique o ponto referente a pupila (Figura 57).

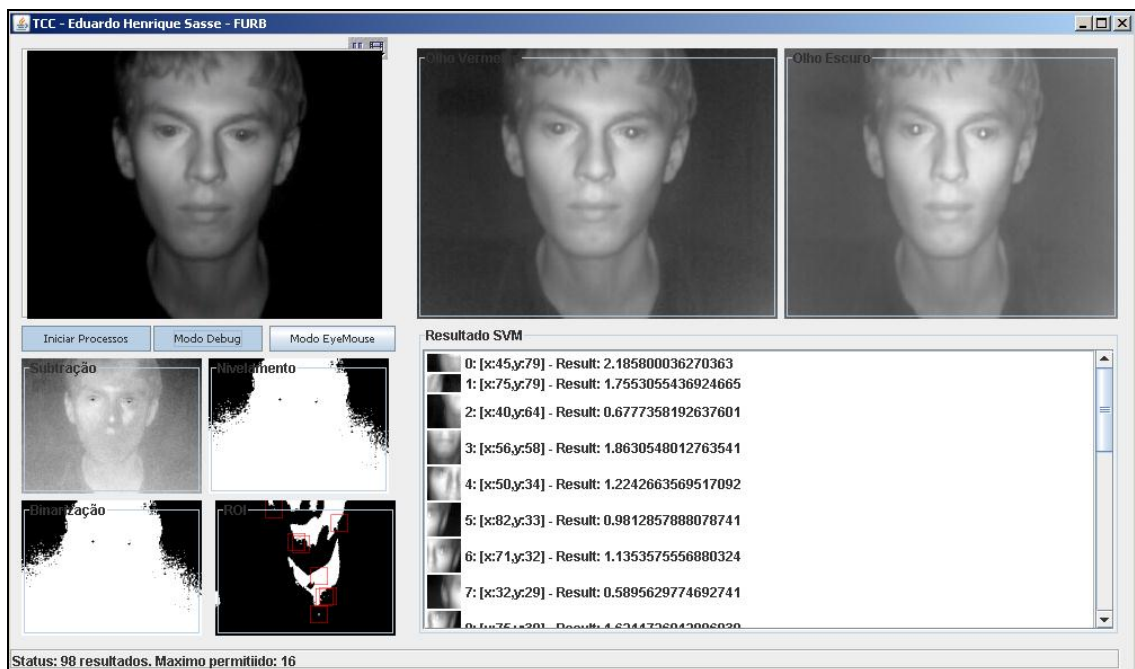


Figura 56 – Olhos vermelhos em ambos os quadros

## 4 CONCLUSÕES

O presente trabalho possibilitou desenvolver uma ferramenta capaz de monitorar a região dos olhos, e transferir o movimento de um olho para o ponteiro do mouse. Para o funcionamento utilizou-se um dispositivo de captura de imagem do tipo *webcam*, que foi adaptada para ser sensível a radiação infravermelha. Também se desenvolveu um dispositivo *flash* com o intuito de gerar o efeito olhos vermelho, a fim de destacar a pupila.

A implementação da ferramenta teve como base teórica o trabalho dos pesquisadores Essa, Flickner e Haro (2000). Métodos como os de subtração de imagens e análise do histograma foram fundamentais para detecção da pupila. A utilização da heurística presente na LibSVM mostrou grande desempenho no processo de reconhecimento das ROI, tornando viável a detecção e monitoramento da pupila em tempo real. Outros métodos como, por exemplo, a diminuição da escala nas imagens para o reconhecimento, também contribuíram para um melhor desempenho da ferramenta.

A utilização das bibliotecas JMF, ImageJ, RXTX e LibSVM juntamente com a linguagem de programação Java, onde todas possuem vários métodos e rotinas de exemplo bem documentadas foi de grande ajuda para a elaboração da ferramenta. A transferência do movimento para o mouse foi nativamente tratado pelo Java.

O desenvolvimento do dispositivo *flash* foi facilitado pelas muitas ferramentas para simulação e desenvolvimento presentes no mercado, muitas inclusive gratuitas para estudantes. A simulação no Proteus permitiu que a comunicação através da biblioteca RXTX e a integração com o sistema, fosse testada antes mesmo do circuito ser construído. A construção física do circuito se mostrou mais complicada que o esperado devido a muitas condições simuladas não refletirem claramente a realidade.

O uso das ferramentas presentes no *Enterprise Architect* para a elaboração das especificações do trabalho se mostraram ágeis e fáceis de usar, não apresentando problemas durante sua execução, atendendo desta forma a todas as necessidades do projeto.

A ferramenta se mostrou eficaz quando as condições mínimas de operação são atendidas, entre elas o ângulo do rosto, a região dos olhos deve estar no mesmo eixo óptico da câmera, e não ultrapassar um grau de rotação maior que trinta graus, referente ao grau de abertura dos LEDs infravermelhos, o fato do usuário que estiver utilizando óculos, ou outro acessório que cause o reflexo do dispositivo *flash*, posicione o mesmo fora de eixo de reflexo. A distância entre o dispositivo de captura e o usuário deve estar entre 40cm e 90cm. Deve

existir uma fonte luminosa posicionada fora da área de captura da câmera, e de preferência superior ao usuário.

#### 4.1 EXTENSÕES

A ferramenta presente no trabalho possibilita a continuação do projeto de forma a melhorar sua precisão de movimento e acurácia na detecção do *click*, além da inclusão do *click* duplo. Os LED utilizados podem ser substituídos por outros de maior velocidade, como descrito na seção 3.4.2.

Como visto na seção 3.4.1 o tempo de execução do algoritmo *svm-scale* é o principal gargalo de desempenho. Como tema para trabalho futuros, sugere-se que sua adaptação seja revista e otimizada. É também uma possibilidade a não utilização da *svm-scale*, por consequência se reduz a acurácia da identificação, gerando em alguns casos a detecção errônea. Com a diminuição dos custos e melhoria dos dispositivos de captura de imagem, acredita-se que o uso de uma câmera especialmente desenvolvida para visão noturna se tornaria viável.

O projeto pode ser melhorado caso seja utilizado o filtro de Kalman, a fim de reduzir as ROI otimizando assim o processo de reconhecimento da região do olho. Outra possibilidade é a implementação do software utilizando C/C++ ao invés de Java, afim de obter um melhor desempenho.

## REFERÊNCIAS BIBLIOGRÁFICAS

AIUBE, Fernando A. L. **Modelagem dos preços futuros de commodities**: abordagem pelo filtro de partículas. Rio de Janeiro, 2005. Disponível em: <[http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0124838\\_05\\_Indice.html](http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0124838_05_Indice.html)>. Acesso em: 21 ago. 2010.

ALVES, Gabriel T. M. **Um estudo das técnicas de obtenção de forma a partir de estéreo e luz estruturada para engenharia**. Rio de Janeiro, 2005. Disponível em: <[http://www.maxwell.lambda.ele.puc-rio.br/Busca\\_etds.php?strSecao=resultado&nrSeq=6951@1](http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=6951@1)>. Acesso em: 14 ago. 2010.

BAPTISTA, Vanessa. NOGUEIRA, Aislan. SIQUEIRA, Simara. **Um overview sobre reconhecimento de padrões**. Rio de Janeiro, 2006. Disponível em: <[http://www.aedb.br/seget/artigos06/688\\_SEGET\\_Um\\_Overview\\_Sobre\\_Reconhecimento\\_de\\_Padr\\_es.pdf](http://www.aedb.br/seget/artigos06/688_SEGET_Um_Overview_Sobre_Reconhecimento_de_Padr_es.pdf)>. Acesso em: 14 set. 2010.

BAUD. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2010. Disponível em: <[en.wikipedia.org/wiki/Baud](http://en.wikipedia.org/wiki/Baud)>. Acesso em: 21 set. 2010.

BERSCH, Rita. **Tecnologia assistiva**. Northridge, 2008. Disponível em: <[www.assistiva.com.br/](http://www.assistiva.com.br/)>. Acesso em: 16 mar. 2010.

CAETANO, Rogério; COSTA, Linara S. da. Mouse ocular: vida e tecnologia em um piscar de olhos. In: FÓRUM DE TECNOLOGIA ASSISTIVA E INCLUSÃO SOCIAL DA PESSOA DEFICIENTE, 1., 2006, Belém, UFPA. **Anais...** Belém, UFPA, 2006. p. 10-21. Disponível em: <[www2.uepa.br/nedeta/ANAIS.pdf](http://www2.uepa.br/nedeta/ANAIS.pdf)>. Acesso em: 16 mar. 2010.

CAMBRIDGE. **The database of faces**. Estados Unidos, 1994. Disponível em: <<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>>. Acesso em: 30 ago. 2010.

CARVALHO, André C. P. L. F., LORENA, Ana C. **Uma introdução às support vector machines**. Portugal, 2005. Disponível em: <[http://seer.ufrgs.br/index.php/rita/article/viewFile/rita\\_v14\\_n2\\_p43-67/3543](http://seer.ufrgs.br/index.php/rita/article/viewFile/rita_v14_n2_p43-67/3543)>. Acesso em: 30 ago. 2010.

CHANG, Chung; LIN, Jen; HSU, Wei. **A practical guide to support vector classification**. Taiwan, 2007. Disponível em: <<http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>>. Acesso em: 19 ago. 2010.

COMUNELLO, Eros; WANGENHEIM, Aldo V. **Introdução à visão computacional**. Florianópolis, 2005. Disponível em: <[www.inf.ufsc.br/~visao/](http://www.inf.ufsc.br/~visao/)>. Acesso em: 20 mar. 2010.

CORREIA, Miguel F. P. V., TAVARES, João M. R. S. **Filtro de kalman no seguimento de movimento em visão computacional**. Portugal, 2005. Disponível em: <<http://repositorio-aberto.up.pt/handle/10216/241>>. Acesso em: 21 ago. 2010.

DAMASCENO, Luciana L.; GALVÃO FILHO, Teófilo A. **Tecnologia assistiva em ambiente computacional**: recursos para a autonomia e inclusão socio-digital da pessoa com deficiência. [Bauru], 2008. Disponível em: <<http://bauru.apaebrasil.org.br/arquivo.phtml?a=9457>>. Acesso em: 16 mar. 2010.

DATTINGER, Rafael. **Ferramenta para detecção de fadiga em motoristas baseada na nonitoração dos olhos**. 2009. 62 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DISPOSITIVO de carga acoplada. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2010. Disponível em: <[pt.wikipedia.org/wiki/Dispositivo\\_de\\_carga\\_acoplada](http://pt.wikipedia.org/wiki/Dispositivo_de_carga_acoplada)>. Acesso em: 14 ago. 2010.

EBISAWA, Yoshinobu. **Detecting and tracking eyes by using their physiological properties, dynamics, and appearance**. [S.l.], 2006? Disponível em: <[www.cc.gatech.edu/cpl/projects/pupil/pupil.pdf](http://www.cc.gatech.edu/cpl/projects/pupil/pupil.pdf)>. Acesso em: 18 ago. 2010.

ESSA, Irfan; FLICKNER, Myron; HARO, Antonio. **Unconstrained pupil detection technique using two light sources and the image difference method**. Atlanta, 2000. Disponível em: <[www.sys.eng.shizuoka.ac.jp/~ebisawa/Seika/Videa95/PROC.PDF](http://www.sys.eng.shizuoka.ac.jp/~ebisawa/Seika/Videa95/PROC.PDF)>. Acesso em: 19 mar. 2010.

EYETECH. **EyeTech**. [S.l.], 2010. Disponível em: <<http://www.eyetechds.com/assistivetech/products/qg3.htm>>. Acesso em: 14 set. 2010.

FEI. **FEI face**. São Paulo, 2006. Disponível em: <<http://www.fei.edu.br/~cet/facedatabase.html>>. Acesso em: 19 mar. 2010.

GALVÃO FILHO, Teófilo A. **A tecnologia assistiva: de que se trata?** Porto Alegre, 2009. Disponível em: <[www.galvaofilho.net/assistiva.pdf](http://www.galvaofilho.net/assistiva.pdf)>. Acesso em: 29 mar. 2010.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais**. Tradução Roberto Marcondes Cesar Junior, Luciano da Fontoura Costa. São Paulo: Edgard Blücher, 2000.

HENNING, Jamine E.; SOUZA, Rodrigo R. R. de. **Tecnologias assistivas**: desenvolvendo as potencialidades das pessoas com necessidades educativas especiais. Curitiba, 2006. Disponível em: <[www.boaaula.com.br/iolanda/fotumta/apresenta/jamine.doc](http://www.boaaula.com.br/iolanda/fotumta/apresenta/jamine.doc)>. Acesso em: 19 mar. 2010.

HSDL. **High-performance t-13/4 (5 mm) ts algas infrared (875 nm) lamp.** [S.l.], 2000? Disponível em: <<http://www.digchip.com/datasheets/parts/datasheet/720/HSDL-4200.php>>. Acesso em: 19 set. 2010.

IMAGEJ. **Imagej: image processing and analysis in java.** [S.l.], 2010. Disponível em: <<http://rsbweb.nih.gov/ij/>>. Acesso em: 25 maio 2010.

JMF. **Java se desktop technologies.** [S.l.], 2010. Disponível em: <<http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html>>. Acesso em: 15 out. 2010.

JOHNSON, Geoff. **How to make a webcam work in infra red wikimedia foundation.** [S.l.], 2010. Disponível em: <<http://www.hoagieshouse.com/IR/>>. Acesso em: 1 out. 2010.

LABCENTER. **Labcenter eletronic.** [S.l.], 2010. Disponível em: <[www.labcenter.com](http://www.labcenter.com)>. Acesso em: 30 out. 2010.

LE ESCAPHANDRE et le Papillon. Direção: Julian Schnabel. França: Pathé Renn Productions: Alternative Films, 2007. VHS NTSC.

LIBSVM. **LIBSVM: a library for support vector machines.** Taiwan, 2010. Disponível em: <<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>>. Acesso em: 23 ago. 2010.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento digital de imagens.** Rio de Janeiro: Brasport, 1999. Disponível em: <[pessoal.utfpr.edu.br/hvieir/download/pdi99.pdf](http://pessoal.utfpr.edu.br/hvieir/download/pdi99.pdf)>. Acesso em: 19 mar. 2010.

MENEZES, Gilliano G. S. **Reconhecimento de faces usando análise de componentes principais e morfologia matemática.** Pernambuco, 2009. Disponível em: <[http://dsc.upe.br/~tcc/20092/TCC\\_final\\_Gilliano.pdf](http://dsc.upe.br/~tcc/20092/TCC_final_Gilliano.pdf)>. Acesso em: 14 set. 2010.

MESQUITA FILHO, Julio. **Um método para melhoria de qualidade de imagens médicas utilizando a transformada wavelet.** São Paulo, 2008. Disponível em: <<http://www.ime.usp.br/~hitoshi/framerate/>>. Acesso em: 18 ago. 2010.

MICROCHIP. **PIC18F2455/2550/4455/4550.** [S.l.], 2004. Disponível em: <<http://ww1.microchip.com/downloads/en/DeviceDoc/39632b.pdf>>. Acesso em: 19 set. 2010.

MORIMOTO, C. H. et al. **Frame-rate pupil detector and gaze tracker.** São Paulo, 1999. Disponível em: <<http://www.ime.usp.br/~hitoshi/framerate/>>. Acesso em: 18 ago. 2010.

MUNDO EDUCAÇÃO. **Equações paramétricas.** São Paulo, 2006. Disponível em: <<http://www.mundoeducacao.com.br/matematica/equacoes-parametricas.htm>>. Acesso em: 21 out. 2010.



RADIAÇÃO infravermelha. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2010. Disponível em: <[pt.wikipedia.org/wiki/Radiação\\_infravermelha](http://pt.wikipedia.org/wiki/Radiação_infravermelha)>. Acesso em: 20 mar. 2010.

RESTOM, A. **Visage, using the face as a mouse**. [S.l.], 2006. Disponível em: <<http://www.mirror-service.org/sites/download.sourceforge.net/pub/sourceforge/v/vi/visage-hci/>>. Acesso em: 25 maio 2010.

ROCHA, Ana; RAFAEL, João; SILVA, Vanessa. **Biometria e segurança**. Caldas Rainha, Portugal, 2005. Disponível em: <[www.scribd.com/doc/331610/BIOMETRIA](http://www.scribd.com/doc/331610/BIOMETRIA)>. Acesso em: 20 mar. 2010.

RODRIGUES, Vinícius L. **Reconhecimento facial usando SVM**. Rio de Janeiro, 2007. Disponível em: <<http://pandeiro.learn.fplf.org.br/AM/images/4/44/FaceRecognition.pdf>>. Acesso em: 14 set. 2010.

RXTX. [S.l.], 2010. Disponível em: <<http://www.rxtx.org/>>. Acesso em: 25 ago. 2010.

SANTOS, Leonadro S. L. **Sistema de comunicação USB com microcontrolador**. Pernambuco, 2009. Disponível em: <<http://dsc.upe.br/~tcc/20091/TCC%20-%20Leonardo%20Santos.pdf>>. Acesso em: 01 out. 2010.

SARLE, Warren S. **Part 2 of Sarle's neural networks faq**. Estados Unidos, 1997. Disponível em: <<http://www.creative.net.au/mirrors/neural/FAQ2.html>>. Acesso em: 20 ago. 2010.

SILVA, Petter V. R. **Utilizando a api rxtx para manipulação da serial**. [S.l.], 2010. Disponível em: <[www.devmedia.com.br/post-6722-Utilizando-a-API-RXTX-para-manipulacao-da-serial-Parte-I.html](http://www.devmedia.com.br/post-6722-Utilizando-a-API-RXTX-para-manipulacao-da-serial-Parte-I.html)>. Acesso em: 19 set. 2010.

SINDROME do encarceramento. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2010. Disponível em: <[pt.wikipedia.org/wiki/Síndrome\\_do\\_encarceramento](http://pt.wikipedia.org/wiki/Síndrome_do_encarceramento)>. Acesso em: 22 mar. 2010.

STERN, Zach. **How to make a digital toy infrared camera**. [S.l.], 2006. Disponível em: <<http://home.comcast.net/~zachstern/toyir/toyir.html>>. Acesso em: 19 set. 2010.

TEZUKA, Érika S. **Um modelo de visão computacional para identificação do estágio de maturação e injúria no pós colheita de bananas**. São Carlos, 2009. Disponível em: <[www.btdt.ufscar.br/htdocs/tedeSimplificado//tde\\_arquivos/3/TDE-2009-12-14T085125Z-2760/Publico/2695.pdf](http://www.btdt.ufscar.br/htdocs/tedeSimplificado//tde_arquivos/3/TDE-2009-12-14T085125Z-2760/Publico/2695.pdf)>. Acesso em: 14 ago. 2010.

TRINDADE, Fernando. **Técnicas de visão computacional para rastreamento de olhar em vídeos**. [S.l.], 2009. Disponível em: <[almerindo.devin.com.br/index.php?option=com\\_content&view=article&id=78:tecnicas-de-computacao-visual-para-rastreamento-de-olhar-em-videos&catid=43:trabalhos-de-alunos&Itemid=18&limitstart=2](http://almerindo.devin.com.br/index.php?option=com_content&view=article&id=78:tecnicas-de-computacao-visual-para-rastreamento-de-olhar-em-videos&catid=43:trabalhos-de-alunos&Itemid=18&limitstart=2)>. Acesso em: 20 mar. 2010.

VISHAY. **Eye safety risk assessment of light or infrared emitting diodes**. [S.l.], 2008. Disponível em: <[www.vishay.com/docs/81935/eyesafe.pdf](http://www.vishay.com/docs/81935/eyesafe.pdf)>. Acesso em: 20 set. 2010.

WINPIC800. [S.l.], 2010. Disponível em: <<http://www.winpic800.com/>>. Acesso em: 15 out. 2010.