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

Eduardo Henrique Sasse– Acadêmico

Paulo César Rodacki Gomes - Orientador



Roteiro

- Introdução
- Objetivos;
- Fundamentação teórica;
- Desenvolvimento da ferramenta;
- Conclusão;
- Extensões.



Introdução

- Inclusão social;
- Benefícios tecnologia;
- Formas de relacionamento;
- Construção de conhecimento;
- Movimento dos olhos;
- Membros superiores debilitados;
 - tetraplegia;
 - distrofia muscular;
 - amputação;
- Interface de comunicação humano-computador (ferramenta)
 - Monitoramento dos olhos;
 - Detecção do movimento;
 - Transferência do movimento;



Objetivos do trabalho

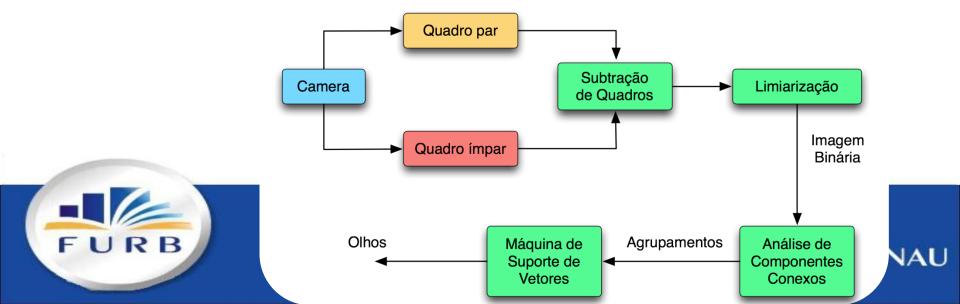
- Captar um vídeo de um usuário utilizando uma webcam;
- Utilizar o flash infravermelho para gerar o efeito olhos vermelhos;
- Detectar e demarcar o olho;
- Detectar e demarcar a pupila do olho;
- Calcular a proporção entre o movimento do olho e do ponteiro do mouse e mapear o movimento do olho para o ponteiro do mouse;
- Perceber alteração no estado do olho (aberto ou fechado) para caracterizar o click simples.

Fundamentação teórica



Método de Detecção

- Premissas
 - Detecção em tempo real;
 - Baixa complexidade computacional;
- Técnicas
 - Detecção e rastreamento;
 - Efeito Olhos-Vermelhos;
 - Iluminação infravermelha;
 - Subtração de quadros;
 - Reconhecimento de características;



Visão computacional

- Conceito
 - Conjunto de técnicas;
 - Interpretar imagens;
 - Estrutura de dados semântica.
- Aquisição de imagens
 - Dois elementos;
 - CCD (Charge Coupled Device);
 - · Elemento digitalizador.
- Pré-processamento
 - Imagem com imperfeições;
 - Operações de baixo nivel;
 - Aprimorar a qualidade.



Reconhecimento de Padrões

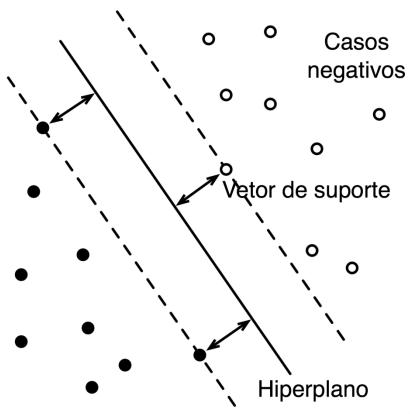
- Reconhecimento de padrões
 - Conjunto de técnicas;
 - Identificação semelhanças.
- Problemas típicos
 - Objeto contido;
 - Apresenta característica.
- Máquinas de Vetores de Suporte SVM
 - Técnica de aprendizado de maquina;
 - Comparável/Superior RNA;
 - Conclusões genéricas de conjunto de exemplos;
 - Aprendizado supervisionado;
 - Aprendizado não-supervisionado.



Reconhecimento de Padrões

- Máquinas de Vetores de Suporte SVM
 - Pontos em um espaço multidimensional;
 - Hiperplano separa classes;
 - Vetores de suporte;
 - Tipos de margem;
 - Rigida;
 - Suaves.
 - Função kernel;
 - Outliers.

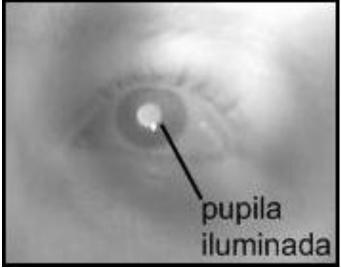
Casos positivos





Visão do Olho sobre Iluminação Infravermelha

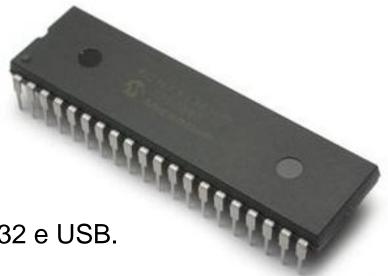
- Radiação Infravermelha
 - Não ionizante;
 - Região do espectro entre 3x10¹¹ Hz e 4x10¹⁴ Hz;
 - Próximo de 780nm até 2500nm;
 - Não é visível pelo olho humano (400nm até os 700nm);
 - Córnea reflete a luz;
 - Efeito olhos-vermelhos.





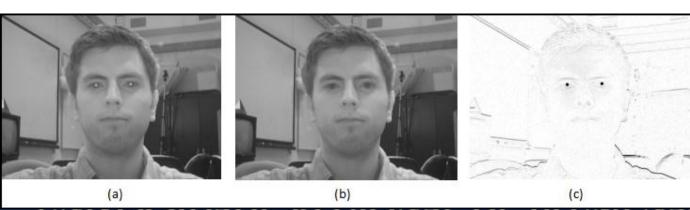
Microcontroladores

- Microcontrolador
 - Circuito integrado;
 - Microprocessador;
 - Periféricos essenciais;
 - Baixo custo;
 - Programáveis;
 - Baixo consumo de energia;
 - Interface entrada/saida RS232 e USB.
- C para Microcontroladores
 - Linguagem de alto nível e estruturada;
 - Bibliotecas de funções.



Trabalhos correlatos

- Detecção e rastreamento dos olhos através de suas propriedades fisiológicas, dinâmicas e aparentes(ESSA, FLICKNER E HARO, 2000)
 - Detecção de fadiga;
 - Tempo real;
 - Numero de piscadas;
 - Câmera infravermelha;
 - Dispositivo Flash Infravermelho;
 - Subtração de quadros;
 - Técnicas
 - Limiarização;
 - Analise do Histograma;
 - PCA Principal Component Analysis;
 - Filtro de Kallman.







Trabalhos correlatos

- Projeto Visage (RESTOM, 2006)
 - Rosto como interface Homem-Computador;
 - Ponta do nariz;
 - Click através dos olhos;
 - webcam;
 - Técnicas
 - Transformada de Hough;
 - Analise do Histograma;
 - SSR Filter;
 - Reconhecimento com SVM;



- Detecção de fadiga;
- Baseado no Visage (RESTOM, 2006);
- Detecção de olhos fechados;







Desenvolvimento da ferramenta



Principais requisitos

- Captar o vídeo do usuário utilizando uma webcam (RF);
- Adaptar webcam a ser sensível ao espectro infravermelho (RF);
- Desenvolver o dispositivo flash (RF);
- Identificar a pupila (RF);
- Verificar se o usuário realizou o click simples ao piscar uma vez (RF);
- Disponibilizar uma interface para permitir o acompanhamento do processo de reconhecimento e movimentação do olho (RF);
- Implementar a ferramenta utilizando a tecnologia Java (RNF);

Principais requisitos

- Utilizar a biblioteca gráfica ImageJ (IMAGEJ, 2010) (RNF);
- Utilizar a biblioteca de aprendizagem de maquina LibSVM (LIBSVM, 2010) (RNF);
- Utilizar a porta de comunicação USB para comunicação com o dispositovo flash e a biblioteca de comunicação RXTX (RXTX, 2010) (RNF);
- Utilizar ambiente de programação Eclipse (RNF).



Diagrama de casos de uso

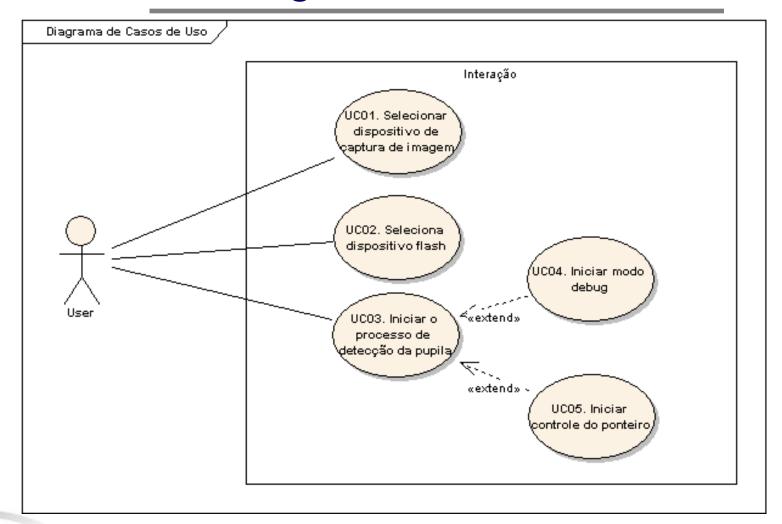




Diagrama de sequência

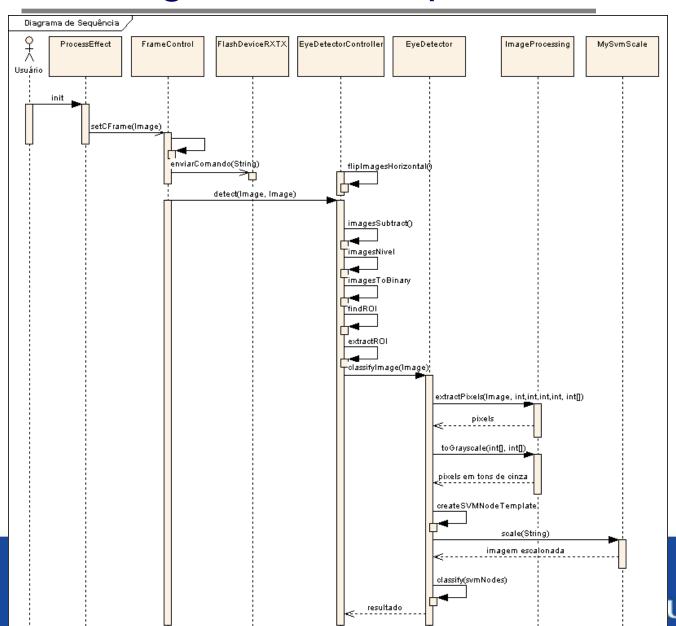
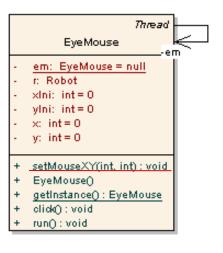




Diagrama de Classes

Pacote control

class control Control nodes: svm_node ([]) model: svm_model = null symFile: File = null svmParScaleFile: File = null serialCom: String = null frame: JFrameTCC x: int = 0y: int = 0maxResults: int = 16 pctClick: int = 35 debugMode: boolean = false eyeAvgWidth: int = 13 eyeAvgHeight: int = 13 screenWidth: int screenHeight: int main(String[]): void setPctClick(int): void getPctClick(): int setDebugMode(boolean) : void isDebugMode(): boolean setXY(int, int) : void getX(): int getY(): int getMaxResults(): int setMaxResults(int): void getEyeAvgHeight(): intgetEveAvgWidth(): intgetScreenHeight(): int getScreenWidth(): int loadSvmModel(): void getModel():svm_model getFrame(): JFrameTCC setSerialCom(String): void getSerialCom(): String



FlashDeviceRXTX sin: FlashDeviceRXTX = null portas: String ([]) listaDePortas: Enumeration pdxctrl: RXTXControl = null LIGAR LEDS EIXO EXTERNO: String = "O" LIGAR_LEDS_EIXO_INTERNO: String = "X" DESLIGAR_LEDS: String = "F" serialIniciada: String = "" FlashDeviceRXTX() getInstance(): FlashDeviceRXTX ObterPortas(): String[] + ListarPortas(): void PortaExiste(String): boolean enviarComando(String): String main(String[]): void



Diagrama de Classes

Pacote eye

class eye

EyeDetector Control Eye Detector mfd: EyeDetector = null f(0) dth: int = 30. con: Control = null fHeight: int = 30 con: int = 0grayPixels: int([]) xy: int(Π). pixels: int(∏). mfd. $lastPosX^{\perp} int = -1$ faces: int lastPosY: int = -1foundEveLastFrame: boolean EyeDetector() ic: ImageCalculator = new ImageCalcul... {readOnly} + setfWidth(int): void mf: MaximumFinder = new MaximumFinder() {readOnly} setfHeight(int) : void <u>classifylmage(lmage) : double</u> EveDetectorControl() createSVMNodeTemplate():svm_node[] - detect(Image, Image) : void classify(svm_node∏) : double desenhaRoi(Image, Color, int, int, int, int): Image

ImageProcessing

- + ImageProcessing()
- + extractPixels(Image, int, int, int, int, int[]): int[]
- + toGrayscale(int[], int[]) : int[]
- toBufferedImage(Image): BufferedImage

FURB

Diagrama de Classes

Pacote video

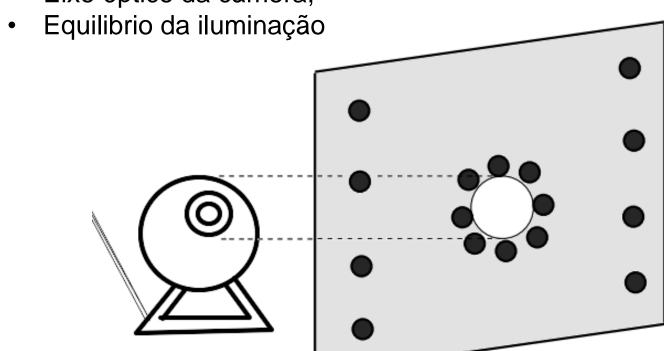
transferData(PushBufferStream): void

class video E#fect Process Effect itb: ImageToBuffer formats: Format([]) outData: byte ([]) bti: BufferTolmage image: Image = null Thread iiPar: Image = null Frame Control inputFormat: Format outputFormat: Format ativo: boolean = true frameControl: FrameControl cframe: Image = null first: boolean = true iiPar: Image -frameControl time: int = 250 + ProcessEffect(CaptureDeviceInfo) edc: EyeDetectorControl = null process(Buffer, Buffer): int getSupportedInputFormats(): Format[] FrameControl() getSupportedOutputFormats(Format): Format[] run(): void + setInputFormat(Format): Format + setAtivo(boolean): void + setOutputFormat(Format): Format + setCframe(Image): void getControl(String): Object getCframe(): Image getControls(): Object[] matches(Format, Format[]): Format getName(): String open(): void close(): void reset(): void desenhaRoi(Image, Color, int, int, int, int): Image BufferTransferHandler DevicesFinder ControllerListener rMask: int = $0 \times 00007 F00$ Process Effect Launcher gMask: int = 0x000003E0 processor: Processor bMask: int = 0x0000001F waitSync: Object = new Object() {readOnly} supportedInputFormats: Format([]) = new Format[[{... {readOnly}} stateTransitionOK: boolean = true supportedOutputFormats: Format([]) = new Format[]{... {readOnly} # supportedins: Format([]) = new Format[[{... ProcessEffectLauncher() open(DataSource, CaptureDeviceInfo, JFrameTCC, svm_model): Processor DevicesFinder() findDevices(JFrameTCC): PushBufferDataSource waitForState(int): boolean controllerUpdate(ControllerEvent): void autoDetect(): void





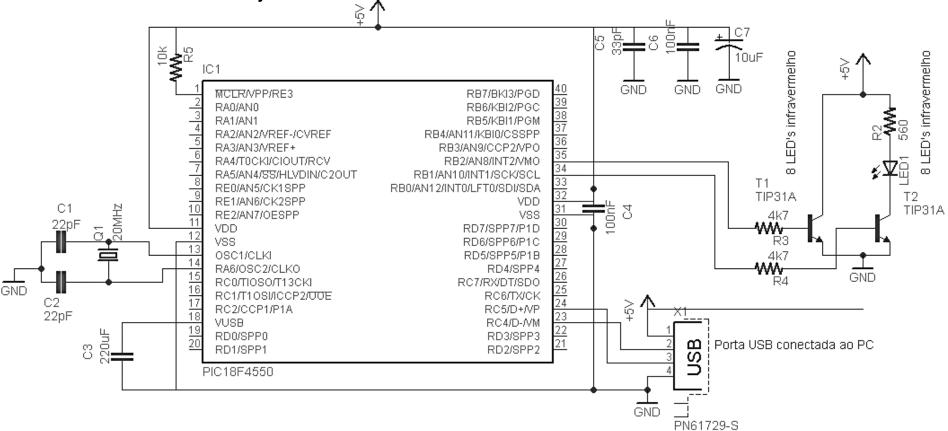
- Sistema flash
 - 16 LEDs infravermelhos;
 - Eixo óptico da câmera;





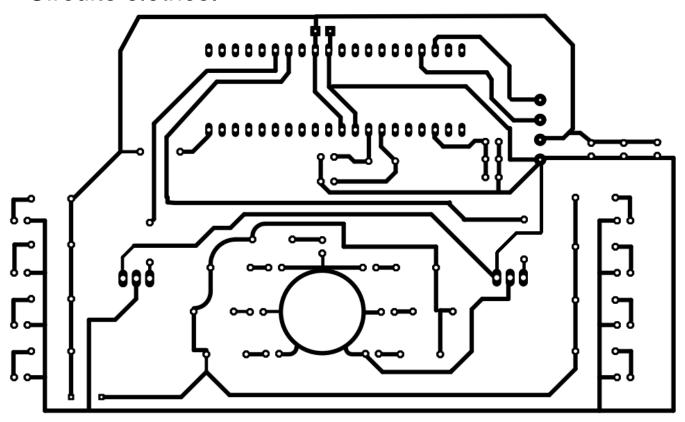
Dispositivo flash

Projeto do circuito.





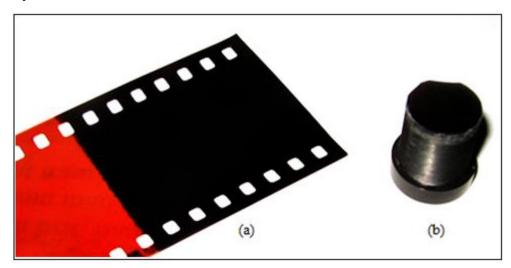
- Dispositivo flash
 - Circuito elétrico.



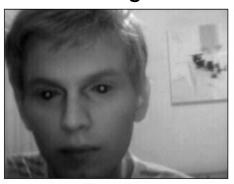


- Adaptação da webcam
 - Sensores CCD são sensíveis a luz visível e infravermelha;
 - Remoção do filtro anti-infravermelho;
 - Inclusão do filtro pró-infravermelho.



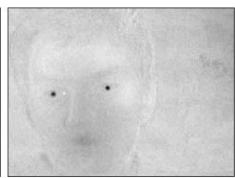


- Processamento de imagens
 - Subtração de quadros;
 - Image Calculator IMAGEJ;



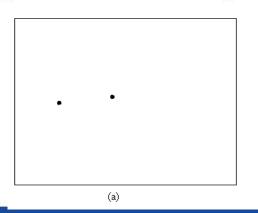


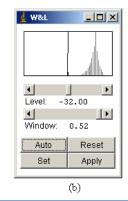
(b)



(c)

- Nivelamento;
- Binarização;
- ROI Find Maxima;

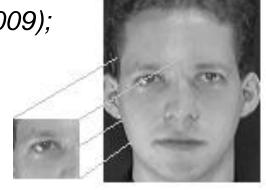






- Reconhecimento de padrões
 - LibSVM;
 - Database of Faces (CAMBRIDGE, 2009);
 - Extração da região dos olhos;





Padrão de entrada da LibSVM

```
-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
```



- Reconhecimento de padrões
 - Classificação da imagem de entrada.

```
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; }
```



Resultados e discussão

- Taxa de acerto se mostrou muito favorável;
- Escalonamento;
- Velocidade comunicação com dispositivo flash;
- Velocidade de acionamento dos LEDs;
- Posicionamento do usuário



Condições adequadas

- Fonte de luz posicionada superior, a frente ou traseira;
- Ângulo máximo de 30 graus;
- Mínimo de 20 frames por segundo;
- Distância do dispositivo ente 40 e 90 cm;
- Atenção para reflexo na utilização de óculos;



Conclusão

- Possibilitou desenvolver uma ferramenta capaz de monitorar a região dos olhos;
- Tempo de resposta viável, com exceção aos LEDs;
- Ferramenta eficaz quando as condições mínimas de operação são atendidas;
- Mais testes necessários para o aprimoramento;
- Objetivos em geral foram alcançados.



Extensões

- Melhora no desempenho no algoritmo svm-scale;
- Utilização de LED de altíssima velocidade;
- Melhora no precisão do movimento e na acurácia da detecção do click;
- Utilização do Filtro de Kalman;



Obrigado!

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

Provérbio Chinês

