UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA MECÂNICA
CURSO DE ENGENHARIA MECÂNICA

RAFAEL DE LIMA THOMAZ

TECNOLOGIA DIGITAL ASSISTIVA: UMA ABORDAGEM DE VISÃO COMPUTACIONAL PARA RASTREAMENTO OCULAR

TRABALHO DE CONCLUSÃO DE CURSO

Guarapuava
2017

RAFAEL DE LIMA THOMAZ

TECNOLOGIA DIGITAL ASSISTIVA: UMA ABORDAGEM DE VISÃO COMPUTACIONAL PARA RASTREAMENTO OCULAR

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Engenharia Mecânica ao Departamento de Coordenação de Engenharia Mecânica – COEME – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Bacharel.

Orientador: Prof. Msc. EMERSON A. FEDECHEN.
Coorientador: Prof. Msc. ANDRES J. PORFÍRIO.
Instituição: UTFPR.

Guarapuava
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Coordenação de Engenharia Mecânica
Câmpus Guarapuava

TERMO DE APROVAÇÃO

Trabalho de Conclusão de Curso

TECNOLOGIA DIGITAL ASSISTIVA: UMA ABORDAGEM DE VISÃO COMPUTACIONAL PARA RASTREAMENTO OCULAR

por

RAFAEL DE LIMA THOMAZ

Este **Trabalho de Conclusão de Curso** foi apresentado às **17h:00min** do dia **04/12/2017** como requisito parcial para a obtenção do título de **ENGENHEIRO MECÂNICO**, pela Universidade Tecnológica Federal do Paraná. Após deliberação, a Banca Examinadora considerou o trabalho:

. . . . . .

(aprovado com louvor, aprovado, aprovado com restrições ou reprovado)

Professores Membros que compõe a Banca Examinadora:

Prof. Msc. Emerson A. Fedechen
UTFPR – Orientador

Prof. Dr. Eleandro M. Krynski
UTFPR – Banca

Prof. Msc. Guilherme da Costa Silva
UTFPR – Banca

Prof. Dr. David Lira Nunez
UTFPR GP – Coordenador do Curso de Engenharia Mecânica

A Folha de Aprovação Assinada encontra-se na Coordenação do Curso – COEME

À família, amigos e professores.

AGRADECIMENTOS

Meu sincero agradecimento à minha família, pelo incansável apoio e motivação. À figura do Professor, profissão digna de honra e reconhecimento, indivíduo central do desenvolvimento de uma nação, fonte de inspiração e formador constante de novos filhos da Pátria. A Andres J. Porfírio e Emerson A. Fedechen, tutores que permaneceram incansáveis do início ao fim do projeto. A todos os amigos e colegas que participam direta e indiretamente no trabalho, em especial a Marcos da Rosa Trentin e Rafael Serbay Rodrigues, pela colaboração direta no desenvolvimento das funções e na convergência dos métodos utilizados.

“O primeiro dever da inteligência é desconfiar dela mesma.” – Albert Einstein.

RESUMO

LIMA THOMAZ, Rafael. TECNOLOGIA DIGITAL ASSISTIVA: UMA ABORDAGEM DE VISÃO COMPUTACIONAL PARA RASTREAMENTO OCULAR 2017. 44 f. Trabalho de Conclusão de Curso – Curso de Engenharia Mecânica, Universidade Tecnológica Federal do Paraná. Guarapuava, 2017.

Este trabalho consiste na implementação de uma tecnologia assistiva digital, baseada no rastreamento ocular, destinada a pessoas com disfunções motoras e de comunicação, com o propósito de melhorar suas condições de comunicação e locomoção. Através de uma câmera, de algoritmos de reconhecimento de face e de olhos, juntamente com a utilização de uma função objetivo que identifica o centro da pupila em uma imagem, a direção do olhar do usuário é rastreada, implementando-se um cursor que pode ser utilizado para selecionar ações em uma interface gráfica. Espera-se portanto que a condição de comunicação e autonomia do paciente com quadro de Paralisia Cerebral seja melhorada, oferecendo melhor qualidade de vida ao usuário.

Palavras-Chave: Rastreamento ocular. Visão por computador. Tecnologia digital assistiva. OpenCv. Detecção.

ABSTRACT

LIMA THOMAZ, Rafael. ASSISTIVE DIGITAL TECHNOLOGY: A COMPUTER VISION APPROACH FOR GAZE TRACKING 2017. 44 f. Trabalho de Conclusão de Curso – Curso de Engenharia Mecânica, Universidade Tecnológica Federal do Paraná. Guarapuava, 2017.

This paper presents the implementation of a digital assistive technology, based on eye tracking, aiming people who suffer from motor and communication difficulties, with the purpose of improving their communication skills. Based on a web cam, image processing algorithms for face and eye recognition, together with an objective function which identifies the center of the pupil in an image, it is possible to track the movement of the user's pupil, using it as a cursor which selects commands on a graphical interface on the screen. Thus, it is expected that the condition of communication and autonomy of the patient with Cerebral Palsy will be improved, offering better life quality to the user.

Key-Words: Gaze Tracking. Computer vision. Digital assistive technology. OpenCv. Detection.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

PC	Paralisia Cerebral
IHC	Interface Humano-Computador
OpenCV	*Open Source Computer Vision*
.xml	*Extensible Markup Language*
.wav	*Waveform Audio File*
HD	*High Definition*
GB	*Gigabyte*
DDR3	*Double Data Rate type three*
Hz	Hertz (unidade de medida de frequência s^{-1})
MHz	MegaHertz (unidade de medida de frequência $10^6.s^{-1}$)
GHz	GigaHertz (unidade de medida de frequência $10^9.s^{-1}$)
USB	*Universal Serial Bus*
RAM	*Random Access Memory*
fps	*Frames per second* (taxa de execução de quadros por segundo)
RGB	*Red, Green, Blue* (padrão de descrição de imagem)
LED	*Light-Emitting Diode*

LISTA DE SÍMBOLOS

p_c	Peso do pixel em escala de cinza
g	Gradiente
A^T	Transposta de A
f	Funcão objetivo
$\sum_{i=1}^{N}$	Somatório de $i = 1$ a $i = N$
$\forall i$	Para todo i
$\in$	Pertencente
$\|g_i\|$	Módulo do vetor g_i
W	Watts (unidade de potência) g_i

SUMÁRIO

1 INTRODUÇÃO

De acordo com o Ministério da Saúde do Brasil, a paralisia cerebral (PC) descreve um grupo de desordens permanentes do desenvolvimento do movimento e postura atribuido a um distúrbio não progressivo que ocorre durante o desenvolvimento do cérebro fetal ou infantil, podendo contribuir para limitações no perfil de funcionalidade da pessoa (Diretrizes de atenção à pessoa com Paralisia Cerebral, 2013). Tais limitações normalmente comprometem as funções motoras e muscoesqueléticas, podendo atingir também as funcionalidades visuais, auditivas, cognitivas e de comunicação.

Dentre as causas que levam à condição da PC, pode-se destacar os fatores pré-natais (como a falta de oxigenação no cérebro do feto e infecções congênitas), bem como adversidades durante o nascimento (anóxia neonatal, eclâmpsia, dentre outras), e ainda fatores pós natais (infecções e traumas) (Piovesana et al., 2002). Apesar de serem vários os possíveis distúrbios acarretados por um quadro de PC, não existe correlação direta entre eles, e portanto o comprometimento das funções neuromotoras não é necessariamente acompanhado de desordens cognitivas, comunicativas e neurolinguísticas. Tal observação possibilita a tentativa de uma abordagem por meio de uma tecnologia digital assistiva, que quando adaptada de maneira adequada, pode aprimorar significativamente a qualidade de comunicação do paciente com PC.

Em tempos de barateamento da tecnologia computacional e maior facilidade ao acesso de bens tecnológicos, torna-se viável economicamente o desenvolvimento de plataformas que auxiliem na abordagem educacional de pacientes com PC que preservem as funções envolvidas no processo de comunicação. Essa abordagem pode hoje ser considerada graças ao nível de desenvolvimento alcançado pela indústria de *softwares* e equipamentos eletrônicos, que hoje proporciona o acesso a linguagens de programação que são ao mesmo tempo eficientes, versáteis e inteligíveis. Ademais, os *hardwares* atualmente possuem qualidade superior e preços mais acessíveis, o que estimula a difusão dos recursos tecnológicos, possibilitando o desenvolvimento de produtos inovadores.

Se por um lado o atual grau de desenvolvimento tecnológico gerou novos recursos para cientistas amadores e simpatizantes da engenharia, por outro ele também forneceu subsídeos para que os profissionais da área pudessem repensar e estratificar melhor os diferentes estágios que compõe o desenvolvimento de um produto tecnológico. Desde a concepção da ideia até o lançamento para o mercado existem vários passos intermediários do desenvolvimento. Para o engenheiro é fundamental conhecer detalhadamente cada etapa do processo, a fim de reduzir ao máximo o tempo, o custo e os erros que eventualmente possam tornar o projeto menos eficiente. As novas tecnologias são incisivas quanto a isso, pois permitem a elaboração de protótipos a um custo menor, ao passo que ajudam a reduzir os riscos, os quais são inerentes a todo projeto de engenharia.

Dentre as várias atividades e atribuições da profissão de engenheiro, uma delas merece ser ressaltada: a capacidade do profissional promover o desenvolvimento social e elevar o patamar de qualidade de vida dos indivíduos de uma sociedade. Sempre que algum conceito científico inovador encontra utilidade no quotidiano social, ele traz consigo o impacto na vida dos cidadãos, e a consequência dessa novidade tem o potencial de agregar virtudes na sociedade, sociedade esta que passa a cumprir com sua responsabilidade. Eis pois o seu elixir, a sua habilidade em resolver os impasses que ela mesma ajudou a causar.

Dadas essas considerações, este trabalho apresenta o desenvolvimento de um protótipo que auxilia na comunicação de pacientes com PC, por meio de uma interface humano-computador. Tendo em vista a capacidade de movimento dos olhos, é possível utilizar uma câmera que monitore a direção do olhar do paciente. Por meio de um algoritmo de processamento de imagem escrito com auxílio da biblioteca OpenCV, é possível interpretar a posição do centro da pupila como um mouse, que será utilizado para selecionar palavras num banco de expressões mostrado na tela do computador. Quando a palavra desejada for selecionada, o paciente utiliza o movimento de piscar os olhos para confirmar a escolha, gerando um sinal de áudio que emite o som correspondente àquela palavra, mimetizando um canal de comunicação através da fala.

Comprovada a funcionalidade de tal sistema, este pode ser empregado não apenas para fins de comunicação, mas também como interface para jogos, lazer, atividades de desenvolvimento de raciocínio lógico, ou ainda enviar comandos para uma cadeira de rodas motorizada, conferindo maior independência de locomoção para o usuário.

Este trabalho de pesquisa consiste em uma investigação de caráter qualitativo, já que busca compreender o comportamento de pessoas com PC, a fim de observar quais são as suas reais necessidades, bem como visa entender de que maneira o problema da comunicação pode ser resolvido do modo mais eficiente possível. Por eficiente entende-se: empregando o menor número possível de recursos, com o menor custo possível e que tenha funcionalidade suficiente para ser considerada como uma alternativa na abordagem de pacientes com quadro de PC.

Este projeto tem ainda caráter bibliográfico, pois foi feito a partir do levantamento de referências baseadas em artigos já publicados, *websites* da área de robótica, livros de processamento de imagem, cartilha de diretrizes de atenção à pessoa com PC (Ministério da Saúde), além de jornais e periódicos da área de visão computacional. Essas consultas tem por finalidade recolher informações e conhecimentos prévios a respeito do problema que está sendo tratado.

Por fim, esta pesquisa constituiu-se de caráter exploratório, pois no andamento da sua elaboração foi estabelecido um estudo preliminar do cotidiano de uma pessoa portadora de paralisia cerebral, definindo o problema com maior detalhamento, identificando quais são as ações relevantes a serem realizadas e colhendo dados adicionais para posterior abordagem do problema. Para essa etapa considera-se consultar profissionais especializados na educação de pessoas com deficiência, pois eles também são testemunhas da dificuldade gerada pela ausência de recursos tecnológicos que facilitem a comunicação com seus alunos. Ademais, eles possivelmente dispõem de pistas valiosas na abordagem do problema, pois estão habituados com

essa rotina, possuindo maior experiência nessa área.

Além de proporcionar uma nova ferramenta para a abordagem de casos específicos de PC, este trabalho tem por objetivo tornar viável o acesso a tal tecnologia, não apenas reduzindo o custo do produto, mas também desenvolvendo o protótipo de maneira a utilizar componentes que sejam comercializados no Brasil, evitando que o usuário necessite realizar os procedimentos alfandegários para importação de tecnologias, reduzindo a burocracia e facilitando o acesso.

1.1 O PROBLEMA

Estima-se que em países em desenvolvimento a incidência de Paralisia Cerebral seja de 7 para cada 1000 nascidos vivos (Zanini et al., p.375-381, 2009). Do total, apenas uma parcela dos diagnosticados preservam as funções neurológicas necessárias para comunicação e uso da linguagem. Apesar de já existirem dispositivos e interfaces de comunicação voltados para quadros de PC, os custos e a burocracia de importação dificultam qualquer tentativa de implantação e difusão de tal tecnologia no Brasil.

Em uma era permeada por *hardwares* com custo acessível e alta capacidade de processamento, soa inconcebível o fato de existirem pessoas que têm o potencial de se comunicar mas que não o fazem por não terem acesso a um sistema dedicado a atender suas necessidades. Ainda que já existam tecnologias que poderiam eventualmente mitigar tamanho empecilho, muitos indivíduos continuariam a padecer diante de tal situação, simplesmente por não gozarem de poder aquisitivo suficiente, ou ainda por desconhecerem as leis de importação que regem os procedimentos de compra de produtos tecnológicos no exterior. Diante de tal obstáculo, cabe indagar-se: É justo que essas pessoas enfrentem tamanha inconveniência? O que tem sido feito para melhorar a qualidade de vida desses indivíduos?

Para a abordagem dessa problemática é proposto um sistema de rastreamento ocular via visão computacional.

1.2 A JUSTIFICATIVA

A responsabilidade social, fruto da observância do princípio de igualdade e equidade entre os homens, evidencia a função categórica de todo cidadão em se compromissar com o desenvolvimento humano e social do seu entorno coletivo. Se nos prelúdios de uma sociedade inexistir a incumbência de todo indivíduo em colaborar com o progresso mútuo, a ordem social por certo padecerá e ruirá o sustentáculo organizacional que permite a convivência harmoniosa. Isto por si só bastaria para legitimar os esforços em se conduzir pesquisa tal como a que está relatada neste documento. Não obstante, julga-se válido salientar que são a ordem e o progresso da nação as razões últimas do senso de irmandade entre os cidadãos, que através do bom uso da ciência são convidados a cooperar para o bem estar dos seus semelhantes.

A introdução de um sistema apropriado para atender indivíduos com paralisia cerebral poderia melhorar substancialmente a qualidade de vida dos que são acometidos com essa disfunção, além de possivelmente proporcionar a aprimoração de métodos computacionais para processamento de imagem. Assim, o motivo maior de tal projeto não compreende apenas o bem estar social, mas tem também objetivo acadêmico de relevância para o desenvolvimento científico e tecnológico regional. Esta pesquisa está alinhada com os objetivos da Universidade Tecnológica Federal do Paraná em fomentar o intuito da cidade de Guarapuava de se consolidar como pólo científico e tecnológico da região centro-sul do estado do Paraná.

Para tanto, propõe-se desenvolver um protótipo que vise a redução do preço frente às tecnologias já existentes, bem como facilitar o acesso no que diz respeito à burocracia atualmente enfrentada no Brasil para importação de aparatos de tecnologia. O uso da visão computacional como tecnologia assistiva é inspirado na área da robótica, e é resultado de um projeto realizado nos Estados Unidos durante o programa Ciência sem Fronteiras, que originalmente empregou tais técnicas para desenvolvimento de um veículo de navegação autônoma, o qual se baseava nas imagens captadas pela câmera para se movimentar e evitar obstáculos de percurso em um lago (Thomaz et al., 2015).

Por fim, este trabalho encontra justificativa no aprimoramento a respeito dos conhecimentos sobre visão computacional, que foram previamente estudados durante o projeto descrito no parágrafo anterior. É importante que essa área seja estudada e desenvolvida se possível ao estado da arte, pois além de ser um tópico bastante recorrente em pesquisas computacionais, o processamento de imagens oferece um rico contexto de aplicações para o cenário industrial, gerando interesse econômico tanto para os que necessitam de soluções tecnológicas quanto para aqueles que as produzem.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver uma IHC que supra a necessidade de comunicação observada em pacientes com Paralisia Cerebral, fornecendo uma alternativa de fácil adaptação e com custo acessível.

2.2 OBJETIVOS ESPECÍFICOS

- Capturar a imagem;
- Implementar um algoritmo que localiza, reconhece e isola a face do usuário;
- Implementar um algoritmo que localiza e identifica a região dos olhos do usuário;
- Aplicar um método de detecção do centro das pupilas através do processamento de imagem;
- Implementar um seletor de posição, que verifica qual a posição do centro das pupilas em relação ao centro da imagem;
- Gerar um cursor que seja operado pelo movimento dos olhos;
- Criar um método de detecção da ação de piscar os olhos, para que o usuário possa assim simular o clique do cursor, selecionando a palavra que desejar na interface gráfica;
- Produzir um banco de palavras, expressões e comandos exibidos em tela para o usuário;
- Estabelecer uma saída de sinal de áudio, para que as palavras e expressões selecionadas pelo usuário sejam reproduzidas em formato de som, a fim de que terceiros compreendam o discurso do usuário.

3 PROCEDIMENTOS METODOLÓGICOS

Os procedimentos metodológicos podem ser assim descritos:

- Fazer o download e instalar a versão mais atualizada do Python;
- Instalação da biblioteca OpenCv para linguagem Python;
- Fazer o download em formato .xml dos classificadores em cascata para face e olhos;
- Implementar o código para detecção da face, que será localizada por um retângulo;
- Implementar o código para detecção dos olhos, os quais serão localizados por dois retângulos;
- Aplicar rotinas de pré-processamento, ou seja, especificar dimensões referentes à posição dos olhos em relação à face, sendo possível assim alimentar o algoritmo com dados pré-conhecidos, aumentando a eficiência da detecção de características;
- Tratar a imagem com filtros para suavizar e reduzir a interferência de brilhos intensos causados por superfícies refletoras, tais como óculos ou outras fontes de ruído;
- Fornecer os pesos para a função objetivo que será posteriormente implementada. Tais pesos consistem em valores de escala de cinza dos pixels que estão dentro da região dos olhos detectada, aumentando assim a probabilidade de que seja identificado corretamente o centro da pupila do usuário;
- Aplicar o gradiente de intensidade luminosa na imagem, podendo assim detectar regiões de contraste significativo, sendo possível assim identificar a pupila;
- Identificar a posição do centro da pupila, rastreando seu movimento. Com base nessa etapa, será possível verificar a funcionalidade e a eficácia do método de detecção;
- Integrar ao código um classificador probabilístico de Bayes, o qual servirá para determinar a direção para onde o usuário está olhando, e assim poder implementar os 4 comandos de direção do cursor (cima, baixo, esquerda e direita);
- Implementar uma função que detecta o clique. Isso será realizado de maneira tal que quando o usuário fechar seus olhos por um determinado período de tempo (que deve ser maior que o tempo de uma piscada involuntária), o código interpretará tal ação como o clique;

- Gerar uma interface gráfica, através da biblioteca pygame. Nela estarão contidos os comandos, palavras, frases e expressões que o usuário necessita para realizar sua comunicação. Essa interface fará o papel intermediário de comunicação entre o usuário e a máquina, necessitando ser intuitiva e eficiente em termos de uso;

- Associar a cada palavra o seu respectivo sinal sonoro, que consistirá de arquivos no formato *wav* previamente salvos, emitindo som quando tal palavra for selecionada.

4 FUNDAMENTAÇÃO TEÓRICA

A visão computacional é um campo interdisciplinar da engenharia que se dedica a estudar maneiras pelas quais os computadores podem adquirir informações relevantes a partir da análise de vídeos e imagens digitais. Tal esforço geralmente tem como objetivo a automatização de tarefas, reconhecimento de padrões e a classificação de características de objetos segundo seus aspectos visuais (BMVA, 2017). Algumas tarefas que são consideradas fáceis de serem executadas por seres humanos (tais como o reconhecimento de uma pessoa em uma fotografia, ou a identificação visual de algum defeito em uma peça) podem se revelar de grande custo computacional, exigindo numerosos recursos da máquina.

Com o advento da digitalização da informação, diferentes técnicas matemáticas e computacionais evoluíram suficientemente a ponto de formarem um campo interdisciplinar que hoje é conhecido por resolver tarefas de alta complexidade. Juntas, algumas técnicas das áreas de processamento de sinais, inteligência artificial e álgebra se refinaram a ponto de serem capazes de viabilizar a implementação de soluções eficientes para problemas considerados de alta complexidade computacional (Fessler, 2017)(Roberts, 2017). Nesta seção do trabalho serão discutidos quais procedimentos, abordagens e algoritmos serão necessários para a implementação de uma solução viável para o problema da comunicação com pacientes de PC.

4.1 ETAPAS DO PROCESSAMENTO DE IMAGEM

O processamento digital de imagens consiste em um método para realização de operações em imagens para diferentes finalidades. Dentre as funções desse método destaca-se a extração de informações úteis que a imagem contém, para posterior processamento e utilização dos dados coletados. Este é um tipo de processamento de sinais no qual o parâmetro de entrada é uma imagem, e a saída pode ser uma imagem, bem como características associadas à ela. O processamento normalmente inclui três passos:

1) Importar (carregar) o arquivo da imagem

2) Analisar, manipular ou modificar o arquivo

3) Exibir a saída com os dados de interesse (Anbarjafari, 2017)

A funcionalidade deste trabalho reside na utilização dos olhos do usuário como um *mouse*, através do qual ele poderá escolher palavras pré-dispostas em um painel mostrado na tela do computador. Para isso, é necessário que o computador capture imagens da face do usuário. Em seguida, é aplicada a abordagem de aproximação multi-estágio (Timm e Barth, 2011), que consiste no reconhecimento progressivo da face e posteriormente dos olhos do usuário. O algoritmo é implementado de maneira tal que quando a face é detectada, seus contornos são imediatamente distinguidos na imagem, através de um retângulo desenhado que evidencia qual é

a região da imagem que corresponde à face do usuário.

Após isso, os olhos precisam ser identificados. Para tanto, um classificador (similar ao que foi utilizado para reconhecimento da face) é empregado. Essa etapa, entretanto, precisa de informações adicionais, correspondentes às medidas antropométricas (Timm e Barth, 2011) da posição dos olhos em relação ao centro da face. Estas informações, juntamente com a utilização do classificador, se revelarão de suma importância para se obter uma estimativa precisa da posição dos olhos.

Uma vez isolados os olhos do usuário, o processamento da imagem se restringirá às regiões identificadas como olhos, reduzindo significativamente a quantidade de informação a ser processada. É chegado, enfim, o momento de se estimar o local mais provável de se encontrar o centro da pupila, que corresponde ao centro dos olhos de um ser humano. Esta informação revelará a direção para a qual o usuário está olhando, e isso permitirá a execução da tarefa de implementação do *mouse*.

O ponto central deste trabalho é certamente o procedimento da detecção do centro da pupila. Para isso, faz-se valer do grande contraste que existe entre a iris e a esclera do olho. Este contraste pode ser detectado com a aplicação de um gradiente de intensidade luminosa, o qual apresenta peculiaridades de valioso conteúdo e informação, já que em objetos circulares (como a esclera) os vetores gradientes se dispoem radialmente, apontando de dentro para fora da região da esclera. A partir disso é possível implementar uma função objetivo que seleciona os pontos que são os melhores candidatos a serem eleitos como centro da pupila. A Figura 1 mostra o fluxograma com as etapas de processamento envolvidas no projeto.

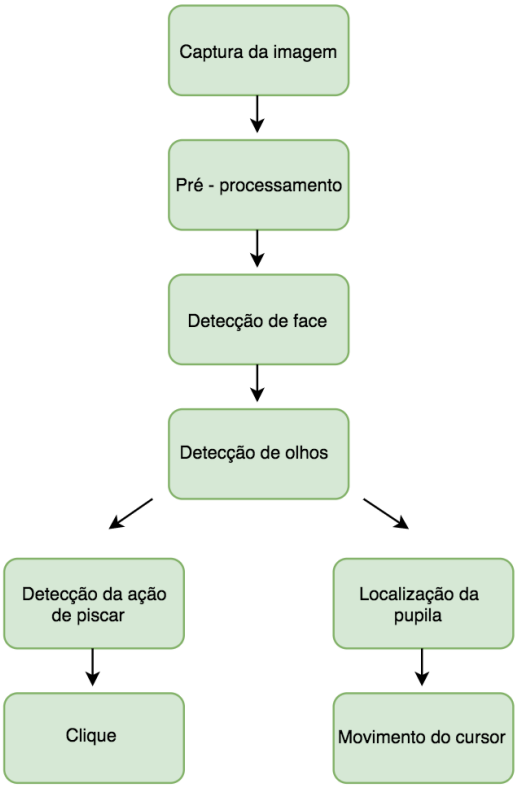

Figura 1: Fluxograma descrevendo as etapas de operação
Fonte: Autoria própria

4.1.1 Sistema de captura da imagem

Antes que qualquer procedimento possa ser executado, é necessário primeiramente obter a imagem de entrada. O algoritmo utilizado captura a imagem proveniente da *web cam* do computador (testado em um MacBook Pro OS X Yosemite, câmera 720p FaceTime HD, processador 3.1 GHz Intel Core i7, memória de 8 GB 1867 MHz DDR3). A utilização de câmeras com maior resolução de *pixels* é encorajada pelo fato de a função objetivo ser capaz de distinguir melhor as características da pupila conforme se aumenta a quantidade de *pixels* que descrevem a mesma imagem. Idealmente quanto maior a resolução, melhor o funcionamento da função objetivo e do algoritmo como um todo. A Figura 2 mostra a comparação de uma mesma imagem com diferentes resoluções espaciais (à esquerda com maior resolução, à direita com menor resolução).

Entretanto, ainda que haja um eventual ganho de precisão do método com o aumento da resolução, haverá também inevitavelmente maior custo computacional associado ao processamento da imagem. Um maior número de *pixels* significa que mais elementos de imagem deverão ser manipulados, aumentando o tempo envolvido na aplicação de filtros, na detecção de características e na localização do ponto de interesse.

Existe portanto um *trade-off* (balanço) entre precisão de detecção e a rapidez do

algoritmo. Aumentar demasiadamente os requisitos de precisão pode acarretar em baixa taxa de execução de *frames* por segundo, levando à perda de exibição de informações em tempo real. No outro extremo também há inconveniências, ou seja, reduzir exageradamente a precisão em favor de maior velocidade de execução pode acarretar na ineficiência do método em geral, inviabilizando o objetivo do projeto.

Disso se verifica a importância em se utilizar os métodos da maneira mais otimizada possível, valendo-se de artifícios matemáticos que sejam eficientes e rápidos, para que assim seja viável aumentar a resolução da câmera de captura sem comprometimento da taxa de *fps*. Afirmações mais significativas apenas poderão ser dirigidas com a execução de testes, avaliando-se câmeras (ou imagens estáticas) com maior ou menor resolucão espacial.

Figura 2: Comparação de uma mesma imagem com diferentes resoluções espaciais
Fonte: Autoria própria

4.1.2 Detecção de face e olhos

Para detecção da face e dos olhos é utilizado um classificador em cascata Haar. Este é um tipo de classificador para detecção de objetos, que após treinado com imagens que contenham o objeto procurado, se torna capaz de analisar características em uma região de uma imagem, retornando *true* se a região potencialmente contiver o objeto, e *false* caso não o contenha. Esta abordagem funciona tanto para a identificação da face como para estimar a da região dos olhos (OpenCV, 2017). A figura 3 mostra os etágios de classificação do filtro Haar.

Figura 3: Estágios de detecção de um filtro em cascata Haar.
Fonte: CodeProject

Essa etapa é fundamental para o bom funcionamento do algoritmo. Esse tipo de classificador retorna uma lista de possíveis objetos detectados, sendo esta lista composta por retângulos, os quais são identificados pela posição do seu vértice superior esquerdo, pela sua largura e espessura. Ao identificar um possível objeto que corresponda à face, o classificador desenhará em cada *frame* um retângulo azul, explicitando assim qual a porção da imagem que contém a face. Esse é o primeiro passo para que haja a delimitação da região de interesse, que corresponderá à porção da imagem que contém os olhos do usuário.

Ao localizar a região de interesse dos olhos, o classificador desenha um retângulo verde na tela e assim delimita a área de atuação da função objetivo, evitando que o programa procure pela pupila em regiões onde ela certamente não estaria localizada. A função objetivo atuará com um considerável custo computacional, e portanto é essencial que sua área de atuação seja a menor possível. Após realizada essa etapa do processamento, inicia-se a rotina de busca pelo ponto que melhor se enquadra como sendo o centro da pupila de cada olho do usuário.

4.1.3 Localização do centro da pupila

Após localizadas as regiões dos olhos procura-se onde se encontra o centro da pupila, ainda que de maneira estimada. Kothari e Mitchel (Kothari e Mitchel, p. 519–522, 1996) propuseram um método que se baseia na análise do campo de vetores para explorar características da imagem . Eles se aproveitaram do fato de haver um grande contraste luminoso entre a íris e a esclera do olho, além do fato geométrico de existir aproximadamente uma simetria circular na forma da esclera. Isto sugere que, através de objetos matemáticos como o gradiente, é possível extrair informações a respeito da posição do olho de um ser humano em uma imagem.

O presente trabalho também se valerá de ferramentas matemáticas para extração de informações, baseado em análises do campo vetorial da imagem, como descrito por Timm e Barth em seu trabalho com localização precisa do centro do olho por meio de gradientes (Timm e Barth, 2011). Neste procedimento os autores estabelecem uma relação matemática entre um ponto candidato a centro da pupila e todos os vetores gradientes da imagem.

Seja λ um círculo. Sendo **c** um possível centro, g_i o vetor gradiente na posição x_i da circunferência de λ e d_i o versor que aponta de **c** para x_i, então **c** será de fato centro de λ se o versor d_i apontar na mesma direção de g_i, para todos os pontos x_i pertencentes à cinrcunferência de λ. A Figura 4 mostra três possíveis candidatos a centro da pupila em diferentes situações: a) Para um candidato fora da região da pupila, b) para um candidato dentro da região da pupila, c) para um candidato que coincide com o centro real da pupila. Note como d_i e g_i alinham-se perfeitamente na situação descrita em c), não apenas para um ponto x_i, mas para todos os pontos x_i's pertecentes à circunferencia λ.

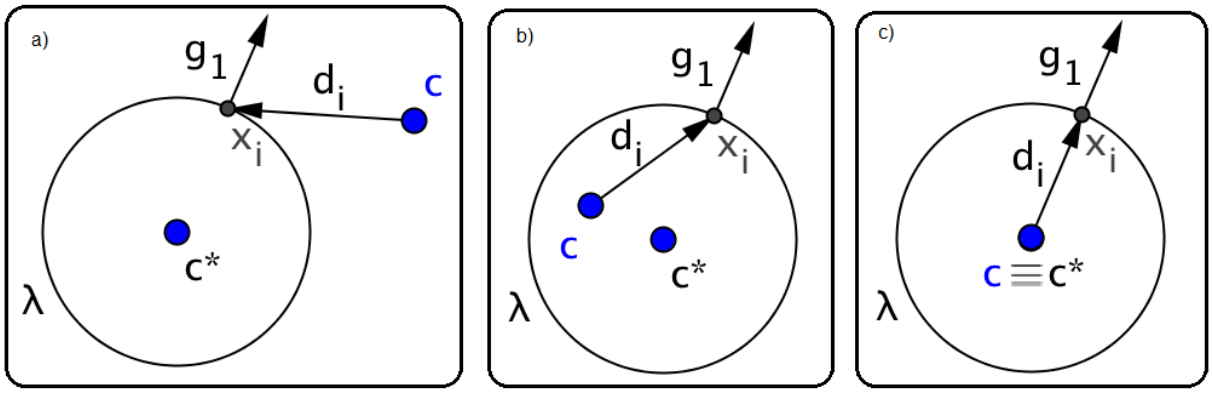

Figura 4: Possível centro avaliado em três casos: a) para um ponto c fora da região da pupila, b) para um ponto c dentro da da região pupila, c) para c coincidente com c*
Fonte: Autoria própria

Em álgebra, dois vetores possuem mesma direção e sentido quando o produto escalar entre os dois é maximizado. Sendo assim, em uma imagem descrita por pixels com posição x_i, $i \in \{1,2,...N\}$, para encontrar o centro ótimo **c*** de um objeto circular basta que se maximize a seguinte função objetivo:

$$f = \frac{1}{N}\sum_{i=1}^{N}(d_i^T g_i)^2 \tag{1}$$

Sendo:

$$d_i = \frac{x_i - c}{\|x_i - c\|} \quad , \quad e \quad \|g_i\| = 1 \quad \forall i \in \{1,2,...N\} \tag{2}$$

Segundo os autores, o vetor d_i deve ser unitário para que se obtenha pesos iguais para todas as posições de pixels. Além disso, os vetores gradientes são normalizados para melhorar a robustez às variações nas condições de luminosidade e contraste.

O vetor gradiente pode ser calculado segundo a equação abaixo:

$$g_i = \left(\frac{\partial I(x_i,y_i)}{\partial x_i}, \frac{\partial I(x_i,y_i)}{\partial y_i} \right)^T \tag{3}$$

O gradiente pode ser calculado de diversas maneiras. É possível uma implementação própria através da criação de uma função que toma como entrada a posição do pixel e retorna o gradiente naquele ponto. É também possível a utilização da função *np.gradient*, a qual é nativa da biblioteca *numpy*, e portanto é otimizada para cálculos com matrizes, executando sua rotina com economia de tempo. A opção escolhida para este trabalho é a função Sobel, própria da biblioteca OpenCV, cuja utilização é voltada para processamento de imagens, calculando não apenas a matriz do gradiente de todos os pixels, como também aplicando uma suavização (borramento) à imagem de entrada.

4.1.4 Etapas de pré-processamento

A etapa de localização da região dos olhos é a primeira a empregar o pré-processamento. Além da estimativa gerada pelo classificador, o algoritmo é nutrido com informações prévias a respeito das distâncias típicas e médias entre os olhos e as bordas do detector de face. Isso porque a face humana mantém certas proporções de distâncias entre olhos, sendo possível estimar de antemão onde estaria a região dos olhos em relação à face detectada, aumentando a eficiência de detecção de olhos.

Dentro da região dos olhos podem existir interferências geradas pelo contraste entre a pele e as sobrancelhas, pálpebras e cílios. Tais contrastes não são bem vindos, pois não trazem informações a respeito do centro do olho, podendo levar a falsas maximizações da função objetivo. A intenção do algoritmo é detectar exclusivamente o contraste que leve até o centro dos olhos. Como a pupila é geralmente escura quando comparada à esclera e à pele, sugere-se a aplicação de pesos p_c, de maneira que possíveis centros ótimos que sejam escuros sejam favorecidos em relação àqueles que são claros.

Assim, integrando esse pré-conhecimento na função objetivo tem-se:

$$f = \frac{1}{N} \sum_{i=1}^{N} p_c (d_i^T g_i)^2 \tag{4}$$

O peso $p_c = I^*(c_x,c_y)$ é o valor da escala de cinza da Imagem I^*, que corresponde à imagem de entrada I após ser invertida e suavizada, suavização esta realizada por meio de um filtro de Gauss (borramento). Além de desconsiderar os falsos centros, este procedimento evita problemas que se originam devido a brilhos intensos, como por exemplo o reflexo de óculos mostrado na Figura 5 .

Figura 5: Ruído introduzido por superfície altamente reflexiva
Fonte: Autoria própria

4.1.5 Conversão da informação digital em uma ação

A partir da localização precisa do centro da pupila, pode-se utilizar a posição dos olhos para movimentar um cursor na tela. Para isso, a região do olho é divida em 4 quadrantes, delimitados por duas diagonais que se cruzam no centro geométrico do retângulo. Para isso, faz-se uso do classificador probabilístico bayesiano, que consiste em uma técnica derivada do teorema de Bayes, para cálculo de probabilidades condicionais (Rish e Irina, 2001):

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{5}$$

Tal abordagem permite inferir probabilisticamente para qual direção o usuário está

olhando: para cima, para baixo, para esquerda ou para direita. Com esses 4 comandos, o usuário pode navegar seu cursor pela tela do computador, e quando desejar clicar, ele fecha seus olhos e os mantém fechados por um período maior do que quando pisca o olho, diferenciando assim um comando de clique de uma piscada involuntária. O comando de clique pode ser implementado pela condição de mudança no padrão de gradientes dentro da região dos olhos, sendo detectada a variação na escala de cinza devido ao ato de se fechar os olhos.

Na tela, o usuário terá um painel de palavras, ações ou comandos a serem selecionados. Através da biblioteca Pygame é possível implementar uma interface gráfica de maneira bastante amigável e eficiente, já que ela é leve e é otimizada para uso em jogos, os quais manipulam imagens e sons em tempo real. Sendo assim, cada palavra terá associada a si um arquivo de som no formato *wav*, que emitirá o som correspondente àquela palavra que foi selecionada pelo usuário. Desta maneira, o paciente poderá se comunicar com outras pessoas, ou até desfrutar de atividades lúdicas e jogos.

4.2 ALGORITMOS DE PROCESSAMENTO DE IMAGEM

Para que os objetivos deste projeto se cumpram, será necessário empregar técnicas e algoritmos que foram previamente desenvolvidos por outros autores, evitando assim que estas rotinas tivessem que ser implementadas desde o início, permitindo então que o foco deste trabalho seja na utilização e aplicação destes conceitos.

O primeiro conceito fundamental para a execução deste projeto é o classificador em cascata do tipo Haar. Este método de detecção de objetos foi inicialmente proposto por Paul Viola (Viola e Jones, p. 137–154, 2004), sendo melhorado posteriormente por Rainer Lienhart e Jochen Maydt (Lienhart e Maydt, p. 900-903, 2002).

Primeiramente, o classificador é treinado com amostras do objeto que se pretende detectar. No caso particular deste trabalho, o classificador foi alimentado com vários exemplos de imagens de faces e olhos, informando o classificador de que as características buscadas são similares às que estão presentes nas amostras. Estes são os chamados exemplos positivos, e a informação repassada é de que o classificador retorne *true* sempre que encontrar um objeto similar ao que está sendo mostrado. A base de testes é própria, sendo composta por cerca de 40 imagens estáticas com exemplo de olhos em diferentes tonalidades, com variados tamanhos e resoluções.

O termo cascata significa que o resultado do procedimento consiste de vários classificadores mais simples, os quais são aplicados sequencialmente em uma região da imagem até que os vários estágios de classificação sejam aceitos totalmente (caso em que a região contém o objeto procurado), ou até que um estágio da classificação seja rejeitado, caso em que o objeto procurado não foi encontrado dentro da região de interesse. Os resutados de cada estágio estão organizados segundo uma árvore de decisão, culminando em um retorno do tipo falso ou verdadeiro.

Após ser treinado, o classificador pode atuar de fato em imagens e regiões de interesse de aplicação. É válido salientar que este método de detecção não é único, sendo entretanto escolhido para esta pesquisa devido à disponibilidade de códigos de livre acesso na Internet. Paul Viola e Michael Jones (Viola e Jones, p. 137–154, 2004) sugeriram em 2004 um método de detecção de objetos para aplicação competitiva em tempo real, inspirados no problema da detecção de face em imagens.

4.3 A BIBLIOTECA OPENCV

O OpenCV (Open Source Computer Vision) é uma biblioteca destinada à implementação de algoritmos de visão computacional e aprendizagem de máquina, otimizada para o processamento de imagens em tempo real. Desenvolvida pela empresa Intel, ela foi originalmente escrita para a linguagem C/C++, possuindo hoje interfaces também para as linguagens Java, Python e para MATLAB/OCTAVE. É bastante versátil, suportando diversos Sistemas Operacionais, como Windows, Linux e macOS entre vários outros SO para desktop. Opera também em dispositivos móveis, sendo o Android, iOS e BlackBerry 10, alguns dos principais.

Esta biblioteca contém mais de 2500 algoritmos otimizados para o uso da visão computacional, que podem ser usados para rastrear objetos em movimento, detecção e reconhecimento de face, identificação e descrição de objetos, reconhecimento de cenários e realidade aumentada, dentre muitas outras aplicações. Dada sua característica de possuir acesso livre, houve uma grande difusão entre os programadores, sendo que sua comunidade *online* hoje conta com mais de 47 mil membros. Isso é de grande valia, pois, juntamente com sua documentação, os fóruns de discussão online contém muitas informações importantes, questões e dúvidas resolvidas, fornecendo um ótimo recurso de pesquisa para quem não possui ampla experiência na área.

No endereço *online* do OpenCv estão relatados alguns casos bem sucedidos de implementações que utilizaram esta ferramenta. Dentre outros, é válido ressaltar sua utilização para vídeos de vigilância em Israel, deteção de acidentes em piscinas na Europa, intervenções de arte interativa na Espanha e em Nova Iorque, além da ampla utilização em ambientes industriais ao redor do mundo, com o propósito de inspecionar produtos em uma linha de produção (OpenCV, 2017).

4.4 A BIBLIOTECA PYGAME

A Pygame é uma biblioteca de acesso livre destinada à linguagem Python, voltada para o desenvolvimento de aplicações multimídia, sendo os jogos o foco do sua utilização pelos usuários (Pygame, 2017). Entre os vários motivos que levam à escolha da Pygame como provedora da interface gráfica deste trabalho, destacam-se a sua portabilidade, simplicidade e alta eficiência. Suas funções principais são escritas utilizando C otimizado e Assembly, ambas linguagens de baixo nível, o que acelera consideravelmente a execução e exibição de componentes gráficos.

Além disso, ela suporta uma variedade de sistemas operacionais e pode operar de maneira bastante otimizada em CPUs com vários núcleos. Sua habilidade de aceitar códigos sem o emprego de interfaces gráficas a torna bastante adequada para aplicações que mesclem diversas funcionalidades, como execução de arquivos de som, processamento de imagem ou apenas para captar comandos de um controlador externo.

Dadas as suas características, é válido destacar o grande número de usuários e fóruns online de discussão, bem como exemplos de códigos, vídeos e tutoriais que circulam pela rede mundial de computadores, onde os autores descrevem o passo a passo da criação de eventos, como a captura do clique do *mouse*, captura de entradas de comando via teclado, interface com periféricos serial, entre outros. A otimização para execução rápida da biblioteca fornece suporte para que posteriormente seja feita a tentativa do embarque em um *single-board*. Estes dependem fortemente de algoritmos e bibliotecas otimizadas para que executem a aplicação em uma taxa aceitável de quadros por segundo, haja vista sua limitação quanto ao *clock* de processamento e espaço livre de memória.

4.5 TECNOLOGIAS JÁ EXISTENTES

O desenvolvimento de uma tecnologia de assistência por rastreamento dos olhos não é uma novidade. Existem opções disponíveis no mercado que se propõe a realizar essa tarefa. O PCEye Plus, desenvolvido pela marca Tobii® é um exemplo de tecnologia que pode ser utilizado para atender necessidades de usuários portadores de paralisia cerebral. O dispositivo possui a habilidade de rastrear os olhos utilizando inclusive capatação em infra vermelho, com taxa de atualização de quadros de 60 Hz, 1.5 W de consumo típico e interface USB (Tobiidynavox, 2017). O fabricante sugere através do *website* que o produto pode ser utilizado para independência no lazer, possuindo acesso ergonômico, possibilidade de interação social e expressão artística. No presente momento este dispositivo custa 1599 euros, que convertidos na taxa de câmbio do dia 14/06/2017 equivalem a 5919,02 reais.

Outra tecnologia também jesenvolvida e disponível no mercado é desenvolvida pela gazepoint®. O produto GP3 Eye tracker disponibiliza apenas o hardware, não sendo ofertado o display que mostra as imagens. A sua taxa de atualização de quadros é de 60 Hz, com interface amigável e compatível com displays de 24 polegadas ou menor. Este dispositivo requer 8GB de memória RAM, processador core i7 ou melhor, suportando Windows 7, 8.1 ou 10, e não suportando os sistemas operacionais macOS e Linux (gazepoint, 2017). No presente momento (14/06/2017) esse aparelho custa 695 dólares americanos, que convertidos na taxa de câmbio atual equivalem a 2279,46 reais.

Esses produtos certamente podem ser aplicados para o caso estudado por esse trabalho, surgindo então a questão: Por que simplesmente não utilizar algo já existente no mercado? Basicamente, são três os principais motivos que justificam o desenvolvimento da tecnologia ao invés da compra. O primeiro e também mais evidente diz respeito ao preço bastante elevado

dos produtos mencionados. São custos inacessíveis para muitos brasileiros, que poderiam ser beneficiados pela possibilidade de aquisição de uma tecnologia similar no Brasil, a um preço muito menos significativo. A proposta desta pesquisa é de implementar tal procedimento a custo reduzido para o usuário.

O segundo motivo pelo qual se torna inviável adquirir estes produtos é relativo à burocracia de importação necessária para aquisição destes aparelhos. Além do custo próprio do produto, é bastante provável que haja custos adicionais de taxação de impostos pela Receita Federal, o que tornaria tais tecnologias ainda mais caras para utilização no Brasil.

O terceiro motivo pelo qual não se considera a compra destes aparelhos é a portabilidade. O segundo produto descrito não suporta os sistemais operacionais Mac e Linux, que certamente tem seu espaço entre os usuários. Deste modo a utilização estaria restrita aos usuários de Windows, que apesar de serem maioria, não representam a totalidade de pessoas que poderiam ser beneficiadas por uma tecnologia assistiva. Deste modo torna-se necessário o desenvolvimento de uma tecnologia que possa ser adquirida facilmente em território nacional, evitando maiores constrangimentos e preocupações por parte dos usuários.

5 RESULTADOS

5.1 PRÉ - PROCESSAMENTO

5.1.1 Conversão para escala de cinza

Em imagens obtidas pelo padrão RGB, cada *pixel* de uma imagem tem sua cor representada por três valores que variam de 0 a 255: (*Red, Green, Blue*). Portanto a imagem de entrada possui informações a respeito das três componentes: azul, verde e vermelho. Entretanto para os fins desejados nesse trabalho (como cálculo do gradiente de luminosidade), apenas é interessante conhecer o valor dos *pixels* em uma única escala de cinza. Assim, *pixels* com valores de cinza mais próximos a 255 são mais brancos (intensa luminosidade), bem como *pixels* com valores mais próximos a 0 tendem ao preto (ausência de luminosidade). Essa coversão se faz necessária porque o gradiente envolvido na função objetivo leva em consideração apenas a variação da intensidade luminosa ao longo da imagem. A Figura 6 mostra a mesma imagem antes e após a conversão para escala de cinza.

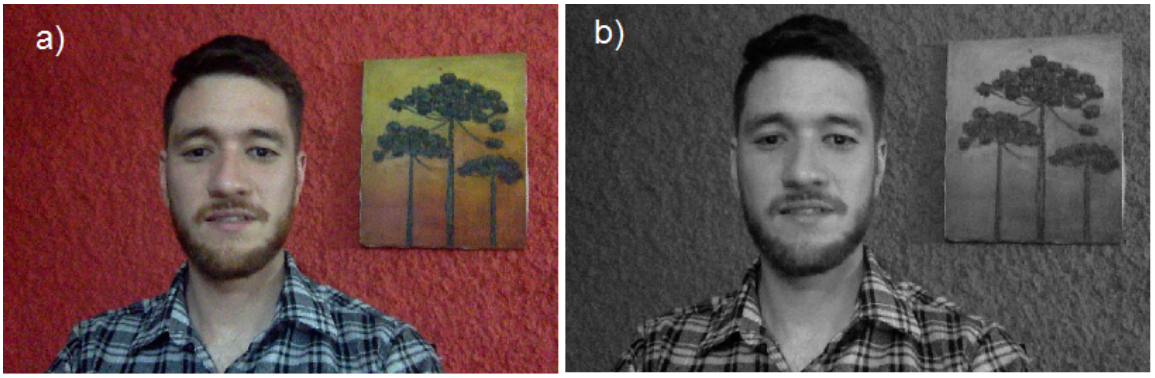

Figura 6: a) Imagem sem conversão b) Imagem após a conversão para escala de cinza
Fonte: Autoria própria

5.1.2 Suavização via filtro de Gauss

A presença de superfícies reflexivas acrescenta ruído à imagem, sendo que os pontos que refletem a luz aparecem na imagem como áreas com alto valor na escala de cinza (valor próximo de 255, tendendo ao branco). Essas alterações podem ser atenuadas utilizando-se a suavização da imagem através do filtro gaussiano, o qual adiciona um efeito de borramento na imagem. A Figura 7 compara a mesma imagem antes e após a aplicação do filtro gaussiano.

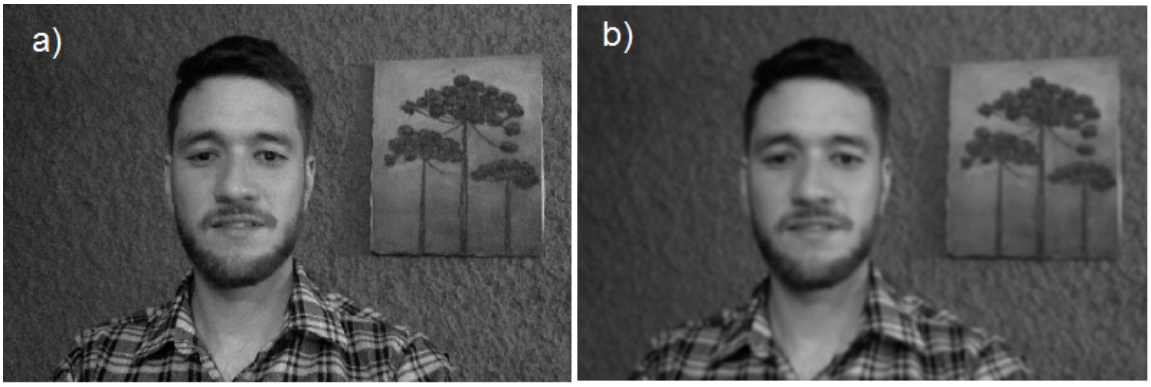

Figura 7: a) Imagem sem aplicação do filtro b) Imagem após aplicação do filtro de Gauss
Fonte: Autoria própria

5.2 APROXIMAÇÃO MULTI - ESTÁGIO

5.2.1 Classificador em cascata de face

A primeira etapa após o pré-processamento consiste na identificação da região da face. O classificador é aplicado para cada quadro capturado pela câmera, e através de uma estrutura de decisão (árvore de decisão) a região é eleita (ou não) como uma porção da imagem que contém a face. O resultado desta etapa está mostrado na figura 8 .

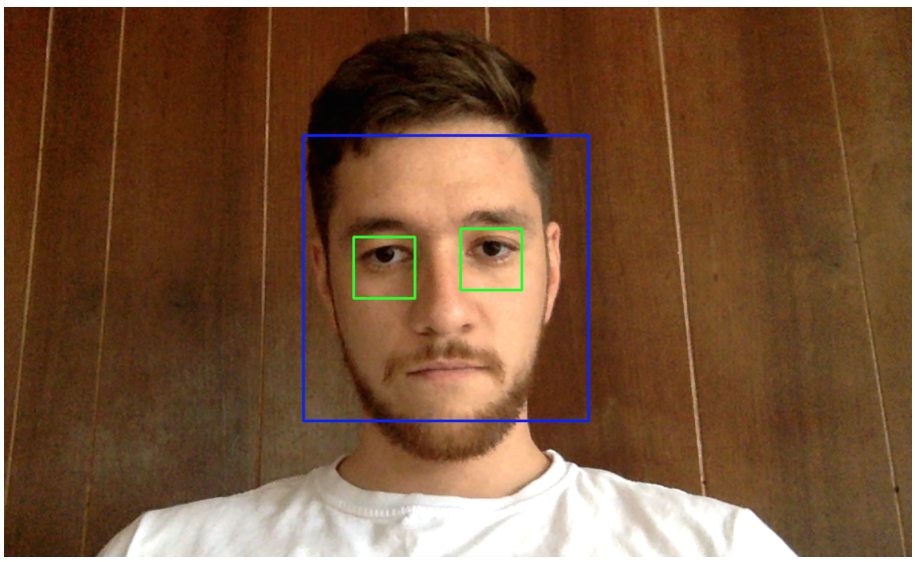

Figura 8: Resultado da detecção de face e olhos através dos classificadores
Fonte: Autoria própria

5.2.2 Classificador em cascata para olhos

Após a identificação facial, o algoritmo realiza uma busca pela região da imagem que

contenha os olhos. Essa etapa é fundamental para o bom funcionamento do programa, pois ela delimita a região sobre a qual será aplicada a função objetivo, a qual busca o centro da pupila. Para aumentar a precisão do método, deve-se informar no código qual a região da face que tem maior chance de conter os olhos. Dado o fato de que os olhos geralmente se encontram na parte superior da face, é possível acrescentar ao código a instrução para que os olhos sejam procurados apenas da metade da face para cima, evitando detecções equivocadas. A Figura 8 mostra os resultados obtidos nessa etapa do processamento.

5.3 A FUNÇÃO OBJETIVO

5.3.1 O descritor da pupila

O sucesso da implementação deste projeto está fortemente ligado a uma descrição coerente e precisa do objeto que se deseja identificar. O fato de a pupila apresentar simetria circular favorece a programação baseada em artifícios oriundos da álgebra linear, como a descrição via campo de vetores e a operação do produto escalar. O propósito dessa função é receber um *pixel* qualquer da região dos olhos como entrada, e rotornar um valor que será tanto maior quanto mais próximo o *pixel* estiver do centro da pupila. Sendo assim, a função objetivo é pensada de maneira que o centro da pupila seja também o *pixel* que maximiza seu valor no domínio da região dos olhos. A Figura 9 mostra a plotagem gráfica bidimensional da função objetivo, evidenciando seu ponto de máximo local.

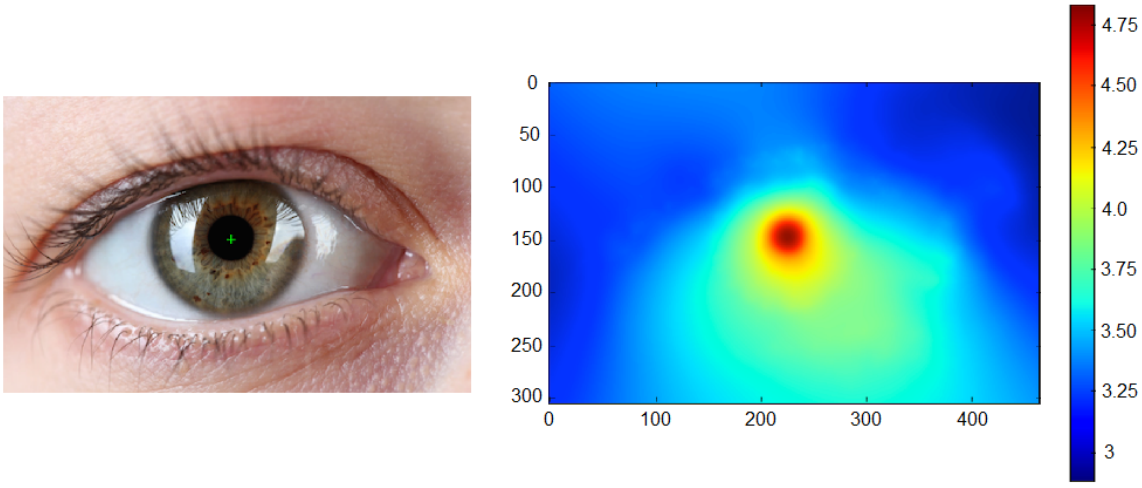

Figura 9: Plotagem bidimensional da função objetivo e sua imagem correspondente
Fonte: Autoria própria

5.3.2 Otimização com operações matriciais

Durante a implementação e teste do algoritmo, notou-se uma significativa demora para retornar os valores da função objetivo. Em uma imagem com 160x320 *pixels* testada, o algoritmo levou cerca de 25 minutos para retornar o ponto central da pupila. A razão para isso se deu devido ao fato de que a função buscava o centro da pupila acessando ponto a ponto de uma matriz, percorrendo duas estruturas de repetição aninhadas (um laço *for* dentro de outro). Isso se mostrou ser de extrema ineficácia para o processamento, e a solução proposta foi transformar a função objetivo em uma rotina de operações com matrizes, tendo em vista o fato de que a biblioteca *numpy* é otimizada para cálculos matriciais. Atentando-se para essa simples observação, foi possível executar os mesmos cálculos 160 vezes mais rápido (em média) do que anteriormente. Com isso, a taxa de *frames* por segundo (fps) aumentou consideravelmente, a ponto de ser possível uma deteção bastante precisa e ao mesmo tempo rápida o suficiente para funcionar em tempo real.

É válido ressaltar que nem toda instância de problema computacional admite a troca de estruturas de repetição aninhadas por operações matriciais. Algumas vezes, o acesso de uma matriz elemento a elemento se faz impreterível, e para esses casos o custo temporal de processamento é dispendioso.

5.4 O MÉTODO DE BUSCA

5.4.1 A busca pelo máximo local

Após ser implementada, a função objetivo foi testada separadamente em um conjunto de 40 imagens diferentes de olhos que compunham a base de imagens teste. Isso permitiu que grande parte dos erros provenientes da detecção da pupila fossem isolados de outros possíveis erros originados nos demais trechos do código. Mesmo para casos em que a íris aparece parcialmente oclusa, a função mostrou ser capaz de detectar o centro aproximado da pupila, conforme mostrado na Figura 10 .

Figura 10: Imagens com oclusão de íris e pupila
Fonte: Autoria própria

O ponto central consiste não apenas em calcular o valor da função para um *pixel* específico, mas também em descobrir qual é o *pixel* que atinge o maior valor quando aplicado na função objetivo. Isso é matematicamente equivalente a buscar o máximo local de uma função $f : A_{mxn} \rightarrow \mathbb{R}$, onde A_{mxn} é a matriz com m linhas e n colunas, cujos elementos estão definidos no conjunto dos números reais. Vários métodos estão disponíveis para encontrar o máximo local de uma função.

5.4.2 A necessidade de otimizar a busca

A maneira trivial de se encontrar o máximo local da função objetivo seria simplesmente percorrer *pixel* a *pixel* da imagem da região dos olhos, calculando o valor da função para cada ponto da imagem e posteriormente retornando o maior valor encontrado. Foi verificado através dos testes que essa imagem é composta por uma matriz cujas dimensões (linhas e colunas) são menores que 70 *pixels*. Assim, no pior caso, seria necessário avaliar o valor da função um total de $70\text{x}70 = 4900$ vezes para ter certeza de qual seria o maior valor, e consequentemente retornar a posição do centro da pupila. Esta seria uma maneira pouco eficiente de se buscar o máximo local, pois o número de verificações aumentaria com o quadrado da ordem da matriz, caracterizando uma complexidade $O(n^2)$.

Quando se trata de aplicações com detecção em tempo real, a complexidade temporal dos algoritmos envolvidos influi significativamente no desempenho e na taxa de *fps* do programa. A otimização da busca se torna crucial por duas principais razões: manter a precisão do método sem reduzir significativamente a taxa de *fps*, e também pelo fato de que métodos menos custosos podem ser executados satisfatoriamente em computadores com menor capacidade de processamento, aumentando assim a abrangência de dispositivos que suportam o programa. Dentre vários possíveis métodos de busca, optou-se a princípio pelo método do gradiente descendente, sendo que futuramente serão testados os métodos de Newton e o método do gradiente conjugado.

5.4.3 O método do gradiente descendente

O método do gradiente descendente é um algoritmo de otimização iterativo de primeira ordem para encontrar o mínimo local de uma função. Para isso, são efetuados passos proporcionais ao negativo do gradiente da função calculado no ponto atual de cada iteração. Se o objetivo for encontrar o máximo local de uma função, basta que os passos efetuados sejam proporcionais ao valor positivo do gradiente da função calculado no ponto atual de cada iteração.

O método é baseado na seguinte observação: se uma função de múltiplas variáveis $F(\mathbf{x})$ estiver definida e for diferenciável na vizinhança de um ponto $\mathbf{a}$, então $F(\mathbf{x})$ decresce mais rápido quando se parte de $\mathbf{a}$ em direção ao negativo do gradiente de F calculado no ponto $\mathbf{a}$, $-\nabla F(\mathbf{a})$. Analogamente, $F(\mathbf{x})$ cresce mais rápido quando se parte de $\mathbf{a}$ em direção ao valor positivo do gradiente de F calculado no ponto $\mathbf{a}$, $\nabla F(\mathbf{a})$.

Assim, ocorre que a cada nova iteração, o algoritmo irá calcular o valor da função em um novo ponto $\mathbf{a}_{n+1} = \mathbf{a}_n + \gamma \nabla F(\mathbf{a}_n)$, aproximando-se gradativamente do máximo local atingido pela função objetivo. O valor de γ (passo) pode variar em cada iteração, mas no presente tabalho ele foi escolhido fixo como sendo igual a 1, pois assim se observa melhor precisão do método. Como a função objetivo apresenta um bom comportamento no domínio da região dos olhos (possui um máximo local bem definido, com variações suaves), é esperado que o método convirja em poucas iterações, expondo o máximo local da função onde deverá estar também o centro da pupila.

O ponto de partida é o centro da imagem da região dos olhos, e a cada iteração o algoritmo verifica a variação entre do valor da função no ponto atual e o valor no ponto anterior. Se essa variação for menor que a precisão pré-determinada no código $(2\text{x}10^{-4})$, o algoritmo finaliza a busca e retorna o ponto atual como sendo o máximo local da função. Caso contrário, a busca continua e o próximo ponto a ser avaliado estará na direção do gradiente da função objetivo.

Os testes com o algoritmo mostram que a busca é finalizada com um número menor ou igual a 17 iterações. Como em cada iteração a função objetivo é calculada três vezes (no ponto, no vizinho da coluna e no vizinho da linha), a função objetivo é acessada um total de 51 vezes, quantidade muito inferior (98,96% menos acessos) quando comparada às 4900 iterações necessárias quando a busca é efetuada de maneira exaustiva. A Figura 11 mostra a aproximação sucessiva do método do gradiente descendente através das curvas de nível de uma função.

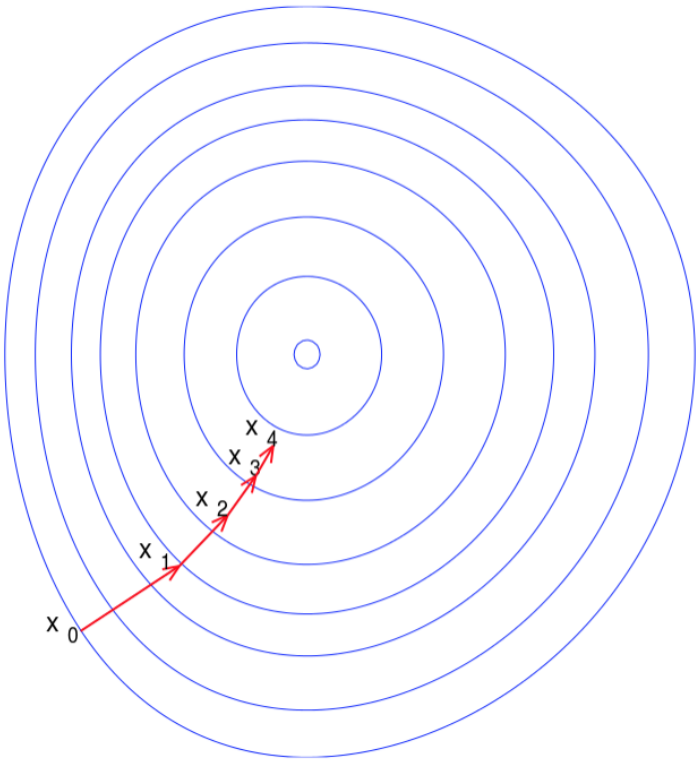

Figura 11: O método do gradiente descendente: Aproximação sucessiva do máximo local
Fonte: Autoria própria

5.4.4 Erros de detecção

Apesar da eficiência de ambos, função objetivo e método de busca, nem sempre a localização do centro da pupila se dá de forma exata ou até mesmo aproximada. Em alguns casos particulares o algoritmo erra significativamente ao retornar pontos equivocados como sendo o centro estimado da pupila. Nestes casos há dois tipos de erros possíveis: equívoco por parte da função objetivo, bem como erro por parte do método do gradiente descendente. Combinados, esses erros podem (em alguns casos e para alguns *frames*) perder o rastreamento ocular, e assim prejudicar o funcionamento do *mouse* que é implementado através do movimento dos olhos.

Ainda que inevitavelmente existam falhas, os testes mostram que estes erros ocorrem raramente, e quando ocorrem perduram durante a exibição de um reduzido número de *frames* (de 30 a 50 *frames*, num universo de aproximadamente 100 *fps*). As falhas na função objetivo ocorrem principalmente devido a condições não uniformes de luminosidade (um lado do rosto mais iluminado que outro), sendo que durante os testes também foram registrados erros devido à uma região de vermelhidão (irritação da esclera) nos olhos. A Figura 12 mostra uma situação de erro de detecção na qual a irritação da esclera (área em vermelho) provoca o surgimento de um máximo local incompatível com o real centro da pupila.

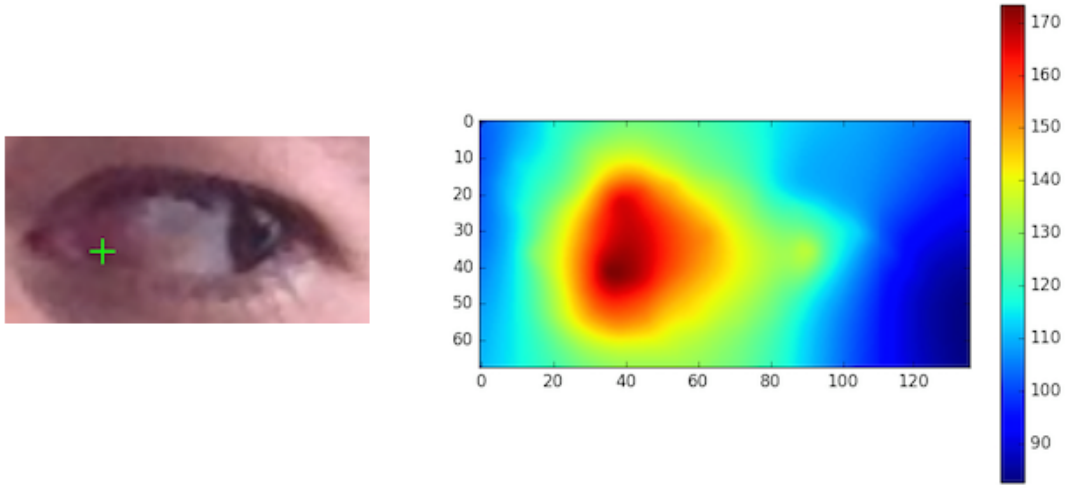

Figura 12: Erro na função objetivo ocasionado pela irritação da esclera.
Fonte: Autoria própria

Para os casos nos quais a função objetivo captura corretamente o centro da pupila, ainda podem existir erros que surgem devido ao mau funcionamento do método de busca. De acordo com o que foi discutido na seção 5.4.2, seria inviável encontrar o valor máximo da função objetivo através da busca exaustiva. Para tanto, utiliza-se um método de busca (aplicado posteriormente à função objetivo) que realiza saltos orientados pelos *pixels* da imagem, evitando o percorrimento de todos os *pixels*, buscando seguir sempre a direção na qual há maior aumento dos valores da função objetivo. Sendo assim, o método se baseia no gradiente da função, assumindo que no domínio da imagem haverá apenas um máximo local. Se houver mais de um máximo local na imagem, ocorrerá a situação na qual o gradiente possivelmente apontará para um máximo local, mas não para o máximo global, e isso acarretará erro de detecção por parte do método. A Figura 13 mostra este cenário, dividido em três eventos: a) detecção correta por parte da função objetivo, b) função objetivo com dois máximos (um local e outro global), c) detecção errônea por parte do método, que se desorientou devido à presença de dois máximos da função objetivo, escolhendo o máximo local (e não o global) como sendo o centro estimado da pupila.

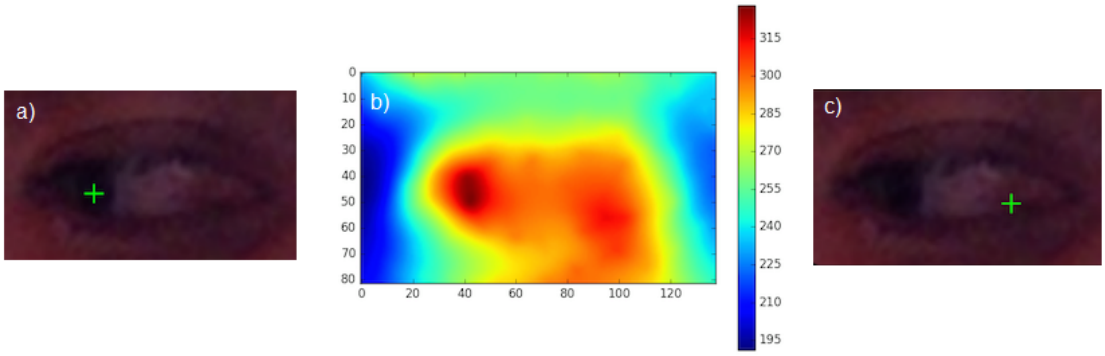

Figura 13: Erro originado pelo método do gradiente descendente, causado pela existência de dois máximos da função objetivo.

Fonte: Autoria própria

Este é um inconveniente por parte do método do gradiente descendente. Ele não é o método mais indicado quando há ocorrência de mais de um máximo local. A solução nesse caso é iluminar o rosto do usuário com uma luz de led, que tornará a luminosidade mais uniforme na região dos olhos, evitando assim o surgimento de mais de um máximo local. É possível também a utilização do método de Newton, a fim de tornar a convergência mais rápida, exigindo menor número de iterações para encontrar o máximo da função objetivo. Apesar da rapidez, o método de Newton não está imune aos erros que surgem quando há mais de um máximo local na função objetivo. Como sugestão para trabalhos futuros inclui-se a utilização de equalizações na imagem capturada, para que o algoritmo funcione em ambientes diversos, evitando a redução da eficiência devido ao ruído de iluminação. Ademais, notou-se que a tecnologia desenvolvida não aplica-se exclusivamente a pessoas com paralisia cerebral, mas também pode ser empregada em jogos em primeira pessoa, na qual a direção do olhar do personagem no jogo pode ser controlada pela direção do olhar do usuário. Testes são necessários, pois o atual programa foi testado com apenas uma pessoa (autor do trabalho), e isso não garante robustez ao sistema, visto que diferentes tonalidades de olhos e pele podem interferir no resultado do algoritmo.

5.5 COMANDOS E INTERFACE

5.5.1 Os comandos

Após verificado o funcionamento adequado da localização do centro da pupila, foram implementados os cinco comandos de ação previamente planejados: mover cursor para esquerda, mover cursor para direita, mover cursor para baixo, mover cursor para cima e finalmente a ação que simula um clique na tela, através do piscar dos olhos do usuário. Foi observada relativa dificuldade em implementar o comando para mover o cursor para baixo, pois quando o usuário realiza o movimento, sua pupila fica parcialmente obstruída pela pálpebra, e a função objetivo

pode falhar esporadicamente devido a tal fato.

Durante os testes, foi possível observar a importância e a influência das condições de luminosidade do ambiente onde se pretende utilizar este software. Foi verificado que condições não homogêneas de intensidade luminosa (um lado do rosto mais iluminado que o outro) podem acarretar perda de eficiência e precisão por parte da função objetivo. Para mitigar esses efeitos aconselha-se que uma luz seja focada na direção do rosto do usuário, e que a fonte de luz estaja preferencialmente logo acima da câmera, para que assim se disponha de maior uniformidade na iluminação dos olhos. A Figura 14 mostra os cinco comandos implementados, bem como a região neutra, que corresponde à área de posição do centro da pupila para a qual nenhum comando é executado (região central).

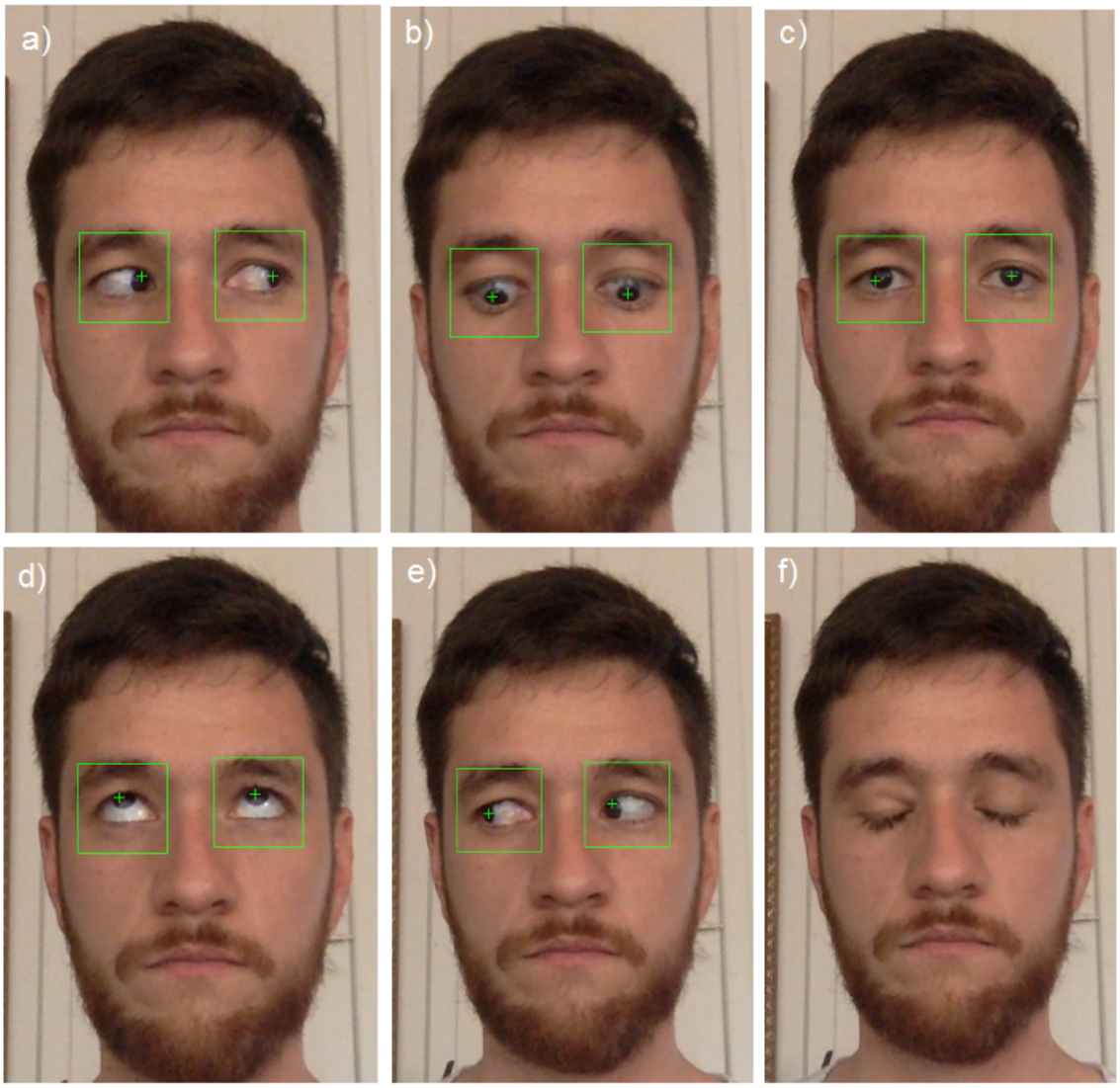

Figura 14: Comandos: a) para direita b) para baixo c) neutro d) para cima e) para esquerda f) clique

Fonte: Autoria própria

Os comandos mostrados acima visam simular as entradas equivalentes às setas do teclado, e o clique simula a tecla enter (*return*). A Figura 15 mostra as entradas do teclado que são simuladas pelo movimento dos olhos.

Figura 15: Teclas do teclado simuladas pelos comandos realizados com os olhos do usuário
Fonte: Autoria própria

5.5.2 Interface com o usuário

A última etapa testada foi a interface gráfica com o usuário, inteiramente desenvolvida a partir da biblioteca Pygame, cuja utilização é própria para jogos e aplicações em tempo real. A princípio uma interface simplificada foi desenvolvida, apenas para que os testes pudessem ser realizados. Inicialmente seis ações foram disponibilizadas para o usuário, todas elas representadas por figuras que associam o significado da ação às imagens correspondentes.

O seletor das ações foi implementado utilizando-se um retângulo vazado com bordas vermelhas, para que o usuário saiba em que posição o cursor está localizado. A partir deles, constatou-se que é possível de fato selecionar as ações no painel apenas com o auxílio do movimento dos olhos, concretizando o principal objetivo deste trabalho, que é disponibilizar uma plataforma de interação entre pessoas com PC e um computador. É notório o fato de que, após implementados os comandos do cursor a partir do movimentos dos olhos, a interface pode ser adaptada da maneira que melhor convier, ou seja, pode ser modificada, incrementada, adaptada e expandida para outras modalidades de acessibilidade, como por exemplo, acesso a jogos e até mesmo à Internet. A Figura 16 mostra a implementação da interface gráfica.

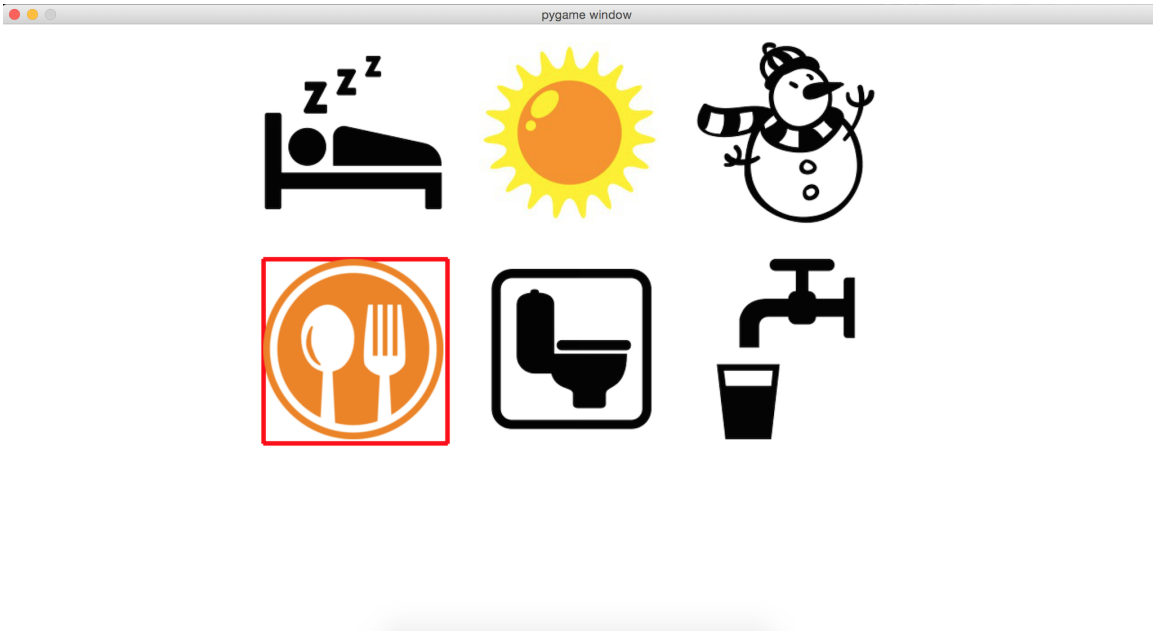

Figura 16: Interface gráfica disponibilizada para o usuário
Fonte: Autoria própria

6 CONSIDERAÇÕES FINAIS

Após a realização de vários testes com o método descrito neste trabalho, verificou-se que de fato é possível cumprir com as intenções previamente pretendidas, sendo que a motivação deste trabalho foi garantida pela oportunidade de ajudar pessoas com disfunções motoras e de comunicação. Os métodos de detecção de olhos e face foram implementados com sucesso, graças aos recentes estudos de Paul Viola e Michael Jones, que obtiveram bons resultados em descrever características faciais em tempo real, valendo-se dos classificadores em cascata. Com isso foi possível delimitar a região de interesse dos olhos, reduzindo a área de atuação da função objetivo, a qual se mostrou bastante confiável em ambientes controlados e com condições homogêneas de luminosidade.

Implementadas estas etapas, tornou-se possível o rastreamento da direção do olhar do usuário, ponto chave para o desenvolvimento desta interface humano-computador. Verificou-se que o seletor de comandos funciona adequadamente desde que hajam condições favoráveis de luminosidade, e ainda que a iluminação se dê de forma heterogênea (uma face do rosto mais iluminada que outra), é possível contornar o problema através da utilização de uma fonte de luz direcionada para o rosto do usuário, e assim homogeneizar a luminosidade na face. Quando o usuário realiza o ato de piscar, ficou comprovado pelos testes que o algoritmo é capaz de detectar tal ação, reconhecendo a ausência do retângulo que identifica a posição dos olhos, e assim interpretando o comando como um clique (ou pressionamento da tecla *enter*).

Finalmente, com todas as etapas de processamento de imagem concluídas, implementou-se a interface gráfica via biblioteca Pygame, exibindo comandos básicos em tela, comandos estes que expressam as necessidades mais básicas de todo ser humano, tais como ir ao banheiro, sensação de fome, sensação de sede, sensação de calor, sensação de frio e sensação de sono. Todos estes comandos são acompanhados pelo seu respectivo sinal de audio, que são pronunciados quando selecionados, alertando outras pessoas sobre o atual estado e necessidades do usuário.

Dentre as oportunidades deixadas para trabalhos futuros se destacam: A tentativa de implementar o mesmo algoritmo utilizando-se métodos mais robustos de busca, tal como método de Newton e método do gradiente conjugado; a ampliação da interface gráfica para atender mais comandos advindos do usuário (cadeira de rodas motorizada, jogos, etc.); uma possível melhoria na função objetivo, para que seja capaz de resistir às mudanças nas condições de iluminação do ambiente; e finalmente a tentativa de implementação deste algoritmo em *hardwares* embarcados, *tablets*, celulares e *Raspberry Pi*, todos estes equipados com menor capacidade de processamento e memória, sendo que assim será possível utilizar este *software* em um conjunto maior de dispositivos, aumentando a abrangência de usuários que possam se beneficiar de tal tecnologia.

Referências

Ministério da Saúde. Diretrizes de atenção à pessoa com Paralisia Cerebral. 1ª edição. Brasília - DF. 2013

Piovesana, A.M.S.G. Encefalopatia crônica, paralisia cerebral. In: Fonseca, L.F.; Pianetti G.; Xavier, C.C. Compêndio de neurologia infantil. Ed. Medci, 2002.

Zanini, G.; Cemin, N.F.; Peralles, S.N. Paralisia Cerebral: causas e prevalências. Fisioter Mov. v.22, n.3. p.375-381, jul/set. 2009.

Thomaz, Rafael L.; Peixoto, Augusto; Reis, Ronaldo. Autonomous Surface Vehicle. Roboboat. CUA, 2015.

Disponível em <https://www.bmva.org/visionoverview> acesso: 10 de junho de 2017.

Disponível em <https://web.eecs.umich.edu/ fessler/course/451/l/pdf/c5.pdf> acesso: 01 de junho de 2017.

Disponível em <https://www.robots.ox.ac.uk/ sjrob/Teaching/SP/l7.pdf> acesso: 03 de junho de 2017.

Disponível em <https://www.tobiidynavox.com/devices/eye-gaze-devices/pceye-plus-access-windows-control> acesso: 23 de maio de 2017.

Disponível em <https://www.gazept.com/product/gazepoint-gp3-eye-tracker> acesso: 13 de junho de 2017.

Timm, Fabian; Barth, Erhardt. Accurate eye centre localisation by means of gradients. Institute for Neuro and Bioinformatics. 2011.

Kothari, R. and Mitchell, J. Detection of eye locations in unconstrained visual images. In Proceedings of the IEEE ICIP, volume 3, pages 519–522. IEEE. 1996.

Viola, P. and Jones, M. (2004). Robust real-time face detection. IJCV, 57(2):137–154.

Lienhart, Rainer. Maydt, Jochen . An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

Disponível em <https://www.opencv.org> acesso: 21 de maio de 2017.

Disponível em <https://www.pygame.org> acesso: 19 de maio de 2017.

Rish, Irina. An empirical study of the naive Bayes classifier (PDF). IJCAI Workshop on Empirical Methods in AI, 2001.

Apêndices

APÊNDICE A - CÓDIGO

```
import numpy as np
import cv2
import pygame
import time
import sys

# multiple cascades: https://github.com/Itseez/opencv/tree/
master/data/haarcascades

#https://github.com/Itseez/opencv/blob/master/data/haarcascades/
haarcascade_frontalface_default.xml
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_
default.xml')
#https://github.com/Itseez/opencv/blob/master/data/haarcascades/
haarcascade_eye.xml
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

# inicializar os monitores de direcao

delta_X = 0
delta_Y = 0
right = 0
left = 0
up = 0
down = 0
center = 0
eye_blink = 0
X_position = 50
Y_position = 500
select = False

#Captura video da camera0 (Capture video from camera0)
cap = cv2.VideoCapture(0)

# Inicializa a pygame
```

```
pygame.init()

# Seta o status "rodando" como True, pra poder entrar no while
infinito
running = True

# Seta o vetor que define o tamanho da tela
size = width, height = 1300,800

# Cria um "objeto" chamado Screen (tela), cujo parametro e o
vetor que define o tamanho da tela
screen = pygame.display.set_mode(size)

# Valores RGB para cores
black = (0, 0, 0)
white = (255,255,255)
red = (255,0,0)

#carregar os aquivos de som .wav

sound1 = pygame.mixer.Sound('cima.wav')
sound2 = pygame.mixer.Sound('baixo.wav')
sound3 = pygame.mixer.Sound('esquerda.wav')
sound4 = pygame.mixer.Sound('direita.wav')
sound5 = pygame.mixer.Sound('bomdia.wav')
sound6 = pygame.mixer.Sound('fome.wav')
sound7 = pygame.mixer.Sound('banheiro.wav')
sound8 = pygame.mixer.Sound('sono.wav')
sound9 = pygame.mixer.Sound('frio.wav')
sound10 = pygame.mixer.Sound('calor.wav')
sound11 = pygame.mixer.Sound('sede.wav')

# carregar imagens

frio = pygame.image.load('frio.png')
fome = pygame.image.load('fome.png')
sono = pygame.image.load('sono.png')
calor = pygame.image.load('calor.jpg')
agua = pygame.image.load('sede.png')
```

```
banheiro = pygame.image.load('banheiro.png')


def PossibleCenter(Xcenter, Ycenter, coluna_final,linha_final,
Xgradient,Ygradient):

# inicializo a funcao objetivo
objFunction = 0
#n = 0

# inicializo as matrizes x e y
indices = np.indices((linha_final,coluna_final))
X = indices[1]
Y = indices[0]
# defino a matriz dos vetores displacement
dx = X - Xcenter
dy = Y - Ycenter

# Normalizo a matriz dos vetores displacement (d)
magnitude = pow((dx*dx) + (dy*dy),0.5)
dx = dx / (magnitude + 0.001)
dy = dy / (magnitude + 0.001)

#Normalizo o gradiente
gradMagnitude = pow((Xgradient**2) + (Ygradient**2),0.5)

Xgradient = (Xgradient) / (gradMagnitude + 0.001)

Ygradient = (Ygradient) / (gradMagnitude + 0.001)

#calculo o produto escalar
dotProduct = dx*Xgradient + dy*Ygradient
dotProduct = np.maximum(dotProduct,np.zeros((linha_final,
coluna_final)))
objFunction = np.sum(((255.0 - roi_new)**3)*(dotProduct**2))
#n = n + 1

# retorno o valor da funcao objetivo normalizado pelo numero de
pixels ao quadrado
```

```
objFunction = (objFunction) / ((linha_final*coluna_final)**2)
return objFunction

while (running == True):
# Verifico o clique do mouse no x vermelho para fechar
for event in pygame.event.get():
# Se encontrar um evento do tipo "sair":
if event.type == pygame.QUIT:
# seto status para falso e caio fora do while infinito
running = False

# Preencho a tela com a cor branca
screen.fill(white)

#Captura quadro a quadro (Capture frame-by-frame)
ret, img = cap.read()
# Colher apenas uma amostra da imagem de entrada (Crop Input image)
#img = img0[40:520, 380:920]
#Faz gaussian blur [borramento] (Apply Gaussian blurring)
blur = cv2.GaussianBlur(img,(5,5),0)
#Muda o espaco de cor de BGR para escala de Cinza (Change color-
space from BGR to Gray)
gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)

#Aplica o classificador em cascata de olhos (Apply cascade face

classifier)

eyes = eye_cascade.detectMultiScale(gray,1.3,9,
0,(60, 60),(90, 90))
eye_blink = eye_blink + 1

for (ex,ey,ew,eh) in eyes:
#ey <  h/2 and ey > h/4
if(1==1):

roi_new = gray[ey:ey+eh, ex:ex+ew]
#roi_color = img[ey:ey+eh, ex:ex+ew]
cv2.rectangle(img,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)
```

```
#Calcula os gradientes
Xsobel = cv2.Sobel(roi_new,cv2.CV_64F,1,0,ksize=5)
Ysobel = cv2.Sobel(roi_new,cv2.CV_64F,0,1,ksize=5)

# Keep track of iterations
n = 0

# initialize X_0 and Y_0 for gradient "ascent"
current_X =  ew/2
current_Y =  eh/2

########### Main Code ##########
#------- Gradient "Ascent"  --------#

gamma = 1
precision = 0.00002
previous_step_size = 100

while (abs(previous_step_size) > precision and n < 15):

n = n + 1

previous_X = current_X
previous_Y = current_Y
centralValue = PossibleCenter(previous_X,previous_Y,ew,eh,Xsobel,
Ysobel)
neighboor_X = PossibleCenter(previous_X+1,previous_Y,ew,eh,Xsobel,
Ysobel)
neighboor_Y = PossibleCenter(previous_X,previous_Y+1,ew,eh,Xsobel,
Ysobel)
gradX = neighboor_X - centralValue
gradY = neighboor_Y - centralValue

current_X = int(current_X + gamma*(gradX/abs(gradX)) )
current_Y = int(current_Y + gamma*(gradY/abs(gradY)) )

previous_step_size = (PossibleCenter(current_X,current_Y,ew,eh,
Xsobel,Ysobel) - centralValue)
```

```
XcenterMax = current_X
YcenterMax = current_Y
delta_X = XcenterMax – ew/2
delta_Y = YcenterMax – eh/2
#print delta_X
#print delta_Y
cv2.line(img,(ex + XcenterMax – 4,ey + YcenterMax),
(ex + XcenterMax + 4,ey + YcenterMax),(0,255,0),1)

cv2.line(img,(ex + XcenterMax ,ey + YcenterMax – 4),
(ex + XcenterMax ,ey + YcenterMax + 4),(0,255,0),1)

if(delta_X > ew/9 and abs(delta_Y) < eh/14):
right = right + 1

if(delta_X < –ew/10 and abs(delta_Y) < eh/14):
left = left + 1

if(delta_Y > eh/20 and abs(delta_X) < ew/10):
down = down + 1

if(delta_Y < –eh/18 and abs(delta_X) < ew/10):
up = up + 1

if(abs(delta_X) < 3 and abs(delta_Y) < 3):
center = center + 1

select = False
if(eye_blink > 1):
print 'XXXXxxxXXXX'
select = True

print eye_blink
eye_blink = 0

if(up > 3):
sound1.play()
print 'cima'
```

```
up = 0
if(Y_position > 240):
Y_position = Y_position - 240

if(down > 3):
sound2.play()
print 'baixo'
down = 0

if(Y_position < 460):
Y_position = Y_position + 240


if(left > 5):
sound4.play()

print 'direita'
left = 0
if(X_position < 1000):
X_position = X_position + 240

if(right > 3):
sound3.play()
if(X_position > 240):
X_position = X_position - 240

print 'esquerda'
right = 0

if (center > 4):
right = 0
left = 0
center = 0
up = 0
down = 0
print 'center'

# posiciona o retangulo vermelho de selecao
print "+++++"
```

```
print eyes

if (len(eyes) > 0):

print "###############"
np.delete(eyes,0,0)

if (len(eyes) > 1):
np.delete(eyes,1,0)
eyes = []
print eyes

pygame.draw.rect(screen,red,(X_position,Y_position,205,205), 5)

if(X_position == 290 and Y_position == 20 and select == True):
sound8.play()
time.sleep(.600)

if(X_position == 290 and Y_position == 260 and select == True):
sound6.play()
time.sleep(.600)

if(X_position == 530 and Y_position == 20 and select == True):
sound10.play()
time.sleep(.600)

if(X_position == 530 and Y_position == 260 and select == True):
sound7.play()
time.sleep(.600)

if(X_position == 770 and Y_position == 20 and select == True):
sound9.play()
time.sleep(.600)

if(X_position == 770 and Y_position == 260 and select == True):
sound11.play()
time.sleep(.600)

# pisca e mostra as imagens na tela
```

```
screen.blit(sono, (290,20))
screen.blit(fome, (290,260))
screen.blit(calor, (530,20))
screen.blit(banheiro, (530,260))
screen.blit(frio, (770,20))
screen.blit(agua, (770,260))
#print "-----------------"


#print eyes
# Nunca esquecer de Atualizar a tela (update), para mostrar as coisas evoluindo no tempo
pygame.display.update()

cv2.imshow('img',img)
k = cv2.waitKey(30) & 0xff
if k == 27:
break


#Quando cair fora do while infinito, sair da Pygame e fechar o systema
pygame.quit()
sys.exit()
# Matar as janelas do OpenCV
cap.release()
cv2.destroyAllWindows()
```