



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Áthila Magalhães Santiago

**Estudo de viabilidade de plataforma para rastreamento ocular invariante à
luminosidade utilizando técnicas de processamento digital de imagens e
reconhecimento de padrões**

Araranguá
2022

Áthila Magalhães Santiago

Estudo de viabilidade de plataforma para rastreamento ocular invariante à luminosidade utilizando técnicas de processamento digital de imagens e reconhecimento de padrões

Trabalho de Conclusão de Curso submetida ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Antônio Carlos Sobieranski, Dr.

Araranguá
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Santiago, Áthila Magalhães

Estudo de viabilidade de plataforma para rastreamento ocular invariante à luminosidade utilizando técnicas de processamento digital de imagens e reconhecimento de padrões / Áthila Magalhães Santiago ; orientador, Antônio Carlos Sobieranski, 2022.

29 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Campus Araranguá,
Graduação em Engenharia de Computação, Araranguá, 2022.

Inclui referências.

1. Engenharia de Computação. 2. Rastreamento Ocular. 3. Visão Computacional. 4. Inteligência Artificial. I. Carlos Sobieranski, Antônio. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Áthila Magalhães Santiago

Estudo de viabilidade de plataforma para rastreamento ocular invariante à luminosidade utilizando técnicas de processamento digital de imagens e reconhecimento de padrões

O presente trabalho em nível de bacharel foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof^a. Analucia Schiaffino Morales, Dr^a.
Avaliadora
Universidade Federal de Santa Catarina

Prof. Jim, Dr.
Avaliador
Universidade Federal de Santa Catarina

Prof.(a) Fábio Rodrigues De La Rocha,
Dr(a).
Avaliador Suplente
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão de curso que foi julgado adequado para obtenção do título de Bacharel em Engenharia de Computação.

Araranguá, 25 de março de 2022.

Prof^a. Analucia Schiaffino Morales, Dr^a.
Coordenadora de Curso

Prof. Antônio Carlos Sobieranski, Dr.
Orientador

Estudo de Viabilidade de Plataforma para Rastreamento Ocular Invariante à Luminosidade Utilizando Técnicas de Processamento Digital de Imagem e Reconhecimento de Padrões

Áthila Magalhães Santiago* Antonio Sobieranski †

2022, Março

Resumo

A constante evolução em capacidade de processamento e de armazenamento permitiu o surgimento de programas de computador com algoritmos cada vez mais complexos, capazes de processar uma ampla quantidade de dados em curtas janelas de tempo. Dentre estas tecnologias, é destacável o Aprendizado de Máquina, conceito de Inteligência Artificial amplamente utilizado no reconhecimento de padrões, sobretudo aplicado à imagens, ramo conhecido como Visão Computacional. Dentre as possibilidades de aplicações da visão computacional, destaca-se a área de interação humano-computador (HCI), responsável por elaborar dispositivos e tecnologias capazes de reduzir a distância de comunicação entre humanos e recursos tecnológicos. O presente trabalho tem por enfoque o estudo de viabilidade de uma plataforma de aquisição não-invasiva e invariante à iluminação, com o arcabouço computacional para análise inteligente de imagens. O objetivo é realizar o rastreamento ocular utilizando câmera e iluminação infravermelha, e através de técnicas de processamento de imagem passar comandos para a máquina. Os resultados preliminares obtidos demonstram a viabilidade da solução para um ambiente experimental elaborado especificamente para validação da abordagem proposta.

Palavras-chave: Visão computacional. Inteligência Artificial. Rastreamento ocular.

*athila.santiago@gmail.com

†a.sobieranski@ufsc.com.br

Viability Study of Lightness Invariable Eye-tracking Platform Using Digital Image Processing Techniques and Pattern Recognition

Áthila Magalhães Santiago* Antonio Sobieranski †

2022, Março

Abstract

Constant evolution in processing power and storage capacity allowed the emergence of computer programs with more intelligent algorithms, capable of processing an enormous quantity of data in short time span. Noticeable among these technologies is the concept of Machine Learning, an Artificial Intelligence branch widely used in pattern recognition, specially when applied to images, field known as Computer Vision. One prominent usability in this field is the human-computer interaction (HCI) area, responsible for developing devices and technologies capable of reducing the distance between human communication and technological resources. This study focuses on developing a platform for non-invasive and lightness invariable acquisition of images, with computational support for intelligent analysis of these images. The objective is to perform eye tracking using infra-red camera and lighting, and through image processing techniques give some inputs to the computer. Early results shows the viability of the solution for an experimental environment elaborated specifically for the validation of the proposed approach.

Key-words: Computer Vision. Artificial Intelligence. Eye-tracking.

*athila.santiago@gmail.com

†a.sobieranski@ufsc.com.br

1 Introdução

O aumento significativo da capacidade computacional ao longo das últimas décadas e ampla disponibilidade em diversos contextos de aplicação é notório. Recursos tecnológicos estão embarcados em dispositivos cada vez mais rápidos, e onde não obstante, a capacidade de armazenamento ampliada e interoperabilidade com outros dispositivos tem favorecido uma série de soluções para problemas do mundo real. Tão importante quanto o desempenho em *Gigahertz* ou a capacidade em *Terabytes*, são as novas arquiteturas implementadas, capazes de alavancar ainda mais a capacidade de processamento de computadores.

Computadores pessoais significavam um sistema com um processador que realizavam uma instrução por vez. Quando um programa era executado, era carregado na memória e o processador se dedicava exclusivamente à execução deste programa. Se o sistema operacional precisasse do processador, o programa era interrompido até que o processador fosse liberado novamente. As aplicações atuais consistem de múltiplas *threads*¹ ou processos que podem executar em paralelo, aumentando significativamente a performance (INTEL, 2003)². A combinação do alto poder de processamento e ampla disponibilidade de recursos possibilitam um conjunto de aplicações anteriormente restritas devido à limitação de *hardware*. Um conceito privilegiado pelo avanço tecnológico é o *Machine Learning* (Aprendizado de Máquina), o estudo de algoritmos capazes de se aperfeiçoar interpretando dados. Apesar de sua fundamentação teórica datar de várias décadas atrás, o tópico se revitalizou e se tornou uma das áreas mais estudadas a partir das décadas de noventa e anos dois mil, e continuamente aplicada desde então (HWANG, 2018).

Diversos algoritmos de Inteligência Artificial (IA), anteriormente restritos por questões de performance, puderam ser implementados com heurísticas admissíveis (KALLEM, 2012). Exemplos de aplicações de IA atuais utilizam modelos probabilísticos com otimizações Bayesianas, outras constituem Redes Neurais Artificiais, como as recentes Redes Neurais Convolucionais (RNC). As RNC supracitadas são aplicadas na solução de problemas tipicamente associados a imagem e reconhecimento de padrão (O'SHEA; NASH, 2015), técnicas em alta demanda de aplicação, como reconhecimento facial, reconhecimento de emoções, reconhecimento em linguagem natural, sistemas analíticos de segurança, tutores inteligentes, entre outros. O campo da visão computacional, área de estudo permeia a habilidade de máquinas extrair e interpretar dados de uma imagem ou vídeo, é uma excelente representação desse avanço de *hardware* e *software* (HWANG, 2018). O interesse crescente neste campo em específico não se dá ao acaso. A popularização crescente de smartphones e outros dispositivos móveis, que já tem na câmera uma parte integrante fundamental, fomenta o desenvolvimento em escala de sensores de imageamento para diferentes aplicações, como sensores CCD e CMOS (LITWILLER, 2001).

Uma aplicação promissora do reconhecimento de padrões em visão computacional é o rastreamento facial e ocular, denominado *eye-tracking*, que consiste em um sistema especializado na coleta e interpretação de dados relacionados à visão humana, como movimentação dos olhos, direção do olhar, piscadas, etc., tipicamente utilizando câmeras que capturam imagens no espectro infravermelho (BAZAR; BRIGHAM, 2007). Esta conjuntura de fatores fazem do *eye-tracking* uma tecnologia com alto potencial e baixo custo de implementação, aplicável a várias áreas de estudo como: (i) Robótica, na interação

¹ unidades básicas às quais o sistema operacional aloca tempo de processador, e podem executar qualquer parte do código (BRIDGE, 2021)

² <https://www.intel.com.br/>

de rebôs com seres humanos (BARBUCEANU; ANTONYA, 2009); (ii) *Business Intelligence* avaliando a retenção de atenção das pessoas em relação à uma campanha de marketing (WEDEL; PIETERS, 2007); (iii) Computação no geral, como em sistemas de realidade virtual (CLAY; KÖNIG; KÖNIG, 2019); (iv) Medicina, indo do auxílio ao diagnóstico de doenças (LÉVÊQUE et al., 2018), à reabilitação através do auxílio na interação homem-máquina para pessoas com inabilidades motoras (MAJARANTA; BULLING, 2014).

Neste trabalho, um estudo de viabilidade de plataforma de visão computacional para rastreamento ocular é apresentada, através de uma premissa de invariância à luminosidade e utilização de modelos convolucionais para o reconhecimento. O *hardware* utilizado consiste em uma webcam preparada para receber luz do espectro próximo do infravermelho, emissor infravermelho, um computador incumbido do processamento, e um conjunto de bibliotecas com ferramentas de visão computacional. O objetivo geral é que seja possível enviar comandos de movimentação do cursor do mouse e cliques coerentes com a movimentação do olhar do usuário e suas piscadas. O software desenvolvido aplica transformações na imagem captada, e toma decisão com base nos parâmetros da imagem pós-processamento. Ao final do trabalho os resultados são descritos e a plataforma tida como viável, tendo aproximadamente 85% de precisão.

2 Trabalhos Correlatos

Na literatura, é possível encontrar diversos trabalhos que utilizam *eye-tracking* com o objetivo de aprimorar a interação entre humano e computador. São apresentados três trabalhos com finalidades semelhantes, onde diferentes tecnologias são aplicadas para realizar o *eye-tracking*, sendo elas a eletro-oculografia (EOG), a vídeo oculografia (VOG) e a bobina de busca escleral (SSC), exemplificados nas Figuras 1, 2 e 3 respectivamente. A enorme maioria dos trabalhos recentes tomam a via da vídeo oculografia, por ser a técnica menos invasiva ao usuário, e ao mesmo tempo versátil, dadas as diferentes possibilidades de aplicação. Dependendo da tecnologia utilizada, os trabalhos podem ser divididos em dois grupos: o grupo no qual o sistema proposto é acoplado ao usuário através de dispositivos vestíveis, e o grupo no qual o sistema é fixo. EOG e SSC necessitam de partes vestíveis devido à natureza da tecnologia, enquanto que para VOG isso é arbitrário. No contexto de dispositivos vestíveis alguns trabalhos encontrados se destacam:

- (LUPU; UNGUREANU; SIRITEANU, 2013): Neste trabalho foi utilizado um óculos e uma câmera acoplada à cabeça. A imagem capturada pela câmera é então processada utilizando a biblioteca OpenCV³, onde algumas operações são realizadas, como conversão do espaço de cores e ajuste de posição, seguindo para a restrição da área de interesse. Após identificada a área de interesse, utiliza-se segmentação binária para identificar e isolar a pupila. Uma vez que a pupila é identificada, o algoritmo é calibrado para detectar sua movimentação e partir para a tomada de decisão, que consiste em mover o cursor do mouse ou clicar.

³ <https://opencv.org/about/>



Figura 1 – Sistema de *eye-tracking* por VOG vestível montado na cabeça.

- (BULLING; ROGGEN; TRÖSTER, 2008): Neste trabalho também é utilizado um óculos, porém ao invés de câmera, são utilizados eletrodos (EOG). Os olhos formam um campo de potencial elétrico, que pode ser descrito como um dipolo, tendo seu pólo positivo localizado na córnea, e seu pólo negativo localizado na retina, chamado de potencial corneoretinal. Dois pares de eletrodos são colocados, acima e abaixo de cada olho. Desta forma, quando o olho se mexe, cada polo se aproxima ou se afasta, causando variação na tensão lida pelos eletrodos. Este sinal analógico é então convertido para digital e utiliza-se um Processador Digital de Sinais (DSP) para interpretá-lo. Por fim, utiliza-se um Controlador Digital de Sinais com *freeRTOS* (Sistema Operacional de código aberto para sistemas embarcados) para gerar o sinal de saída e controlar a movimentação do cursor.

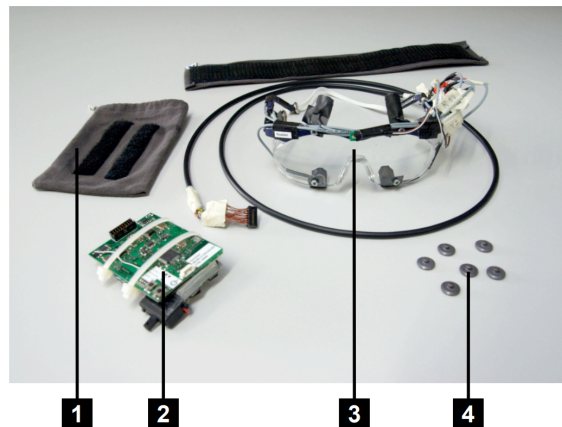


Figura 2 – Componentes do dispositivo vestível baseado em EOG: Bolsa de pano (1), *hardware* de processamento (2), óculos (3), eletrodos (4).

- (WHITMIRE et al., 2016): Neste trabalho o *eye-tracking* é aplicado à realidade virtual utilizando bobinas oculares esclerais (SSC). Esta tecnologia utiliza campos magnéticos alternantes presentes nos olhos do usuário. Corrente alternada é utilizada para transformar a bobina em um eletro-ímã que gera um campo magnético capaz de oscilar em frequências arbitrárias. O fluxo magnético oscilante gerado pelas bobinas intersecciona o contato escleral, induzindo uma corrente de mesma frequência através da bobina escleral. Ao movimentar os olhos, esta tensão induzida varia entre um valor máximo e mínimo. A interação de uma bobina apenas não é suficiente para determinar

a direção do olhar do usuário, sendo necessário utilizar três bobinas ortogonais para cada olho, que podem operar em diferentes frequências. As amplitudes, fases e frequências dos sinais são então processadas com o auxílio do MATLAB para definir a direção do olhar.



Figura 3 – Dispositivo vestível para *eye-tracking* aplicado a realidade virtual utilizando SSC.

No contexto sem dispositivos vestíveis, temos outros exemplos de trabalhos:

- (ZHENG; USAGAWA, 2018): Neste trabalho é utilizada uma *webcam* de baixa resolução para captura da imagem. Utilizando OpenCV, um algoritmo de padrão binário local (LBP) é aplicado para encontrar a face do usuário, em seguida são separadas as áreas de interesse através de características geométricas, como o formato dos olhos e a distância entre eles. Após encontrar as regiões de interesse, filtros são aplicados para suavizar a imagem, devido à baixa resolução da *webcam*, e em seguida utiliza-se a discrepância de cores para identificar a pupila. Com a pupila identificada, são analisadas as movimentações tanto da pupila quanto da cabeça para determinar a direção do olhar.

A literatura encontrada demonstra que há diversas maneiras de se implementar o *eye-tracking*, dependendo principalmente do orçamento disponível e do objetivo a ser atingido. Aplicações que necessitam alta precisão de rastreamento tendem a utilizar EOG ou SSC, dada a natureza dos sinais analógicos. Aplicações que visam reduzir custos, tendem a VOG, devido à ampla variedade e disponibilidade de tipos de câmeras. Cada tecnologia possui seus prós e contras, como a linguagem de programação de alto ou baixo nível e necessidade de hardware especializado, fatores que devem ser levados em consideração na hora da escolha. Na Tabela 1, uma breve análise das tecnologias encontradas é apresentada, elencando seus pontos positivos e negativos.

Tecnologia	Prós	Contras
SSC	Altíssima precisão na detecção de movimentos e baixíssimo tempo de resposta	Necessita a utilização de bobinas aplicadas diretamente aos olhos do paciente através de lentes de contato. Além de possíveis incômodos causados ao usuário pelo uso, a lente pode limitar a movimentação dos olhos, além de precisar ser trocada de tempos em tempos.
EOG	Boa precisão na detecção de movimentos e baixíssimo tempo de resposta	Necessita a aplicação de eletrodos diretamente na face do usuário. Os eletrodos precisam estar ligados por fios até o dispositivo de processamento, limitando a mobilidade. Pode sofrer interferências eletromagnéticas de ruídos externos.
VOG	Pode ser utilizado sem contato direto com usuário e utiliza equipamentos de fácil aquisição	Suscetível a variações de iluminação quando utilizado com câmeras comuns. Como depende da captura de imagens, o usuário necessita estar sempre de frente para a câmera, limitando a mobilidade.

Tabela 1 – Tabela do comparativo das tecnologias mais comuns utilizadas para *eye-tracking*.

3 Fundamentação Teórica

3.1 Transformações

As transformações utilizadas neste trabalho podem ser divididas em duas categorias, transformações de cor e transformações morfológicas.

3.1.1 Transformações de Cor

3.1.1.1 Correção de Gama

Em fotografia, vídeo e computação gráfica, o símbolo gama (γ) representa em um único parâmetro numérico a aproximação da função exponencial que mapeia o valor do pixel para um valor tristímulus (POYNTON; BOOKS24X7; INC, 2003).

O sistema tristímulus é um sistema para parear cores em uma condição definida com as cores primárias — vermelho verde e azul, e seus resultados são expressos nas coordenadas X, Y e Z (NASSAU, 2009). Devido à natureza tricromática da visão humana, uma cor pode ser definida por um conjunto de três valores, que podem ser imaginados como coordenadas de um espaço vetorial de três dimensões. De maneira mais precisa, uma cor pode ser representada por um vetor cuja magnitude é proporcional a iluminância (quantidade de luz que incide sobre uma determinada área de uma superfície plana), e cuja orientação no espaço está relacionada com o tom da cor em si (BRETEAU, 2007).

Algumas representações do sistema tristímulus são demonstradas na Figura 4. Independente de qual cor primária é alocada em cada eixo (na esquerda o eixo X representa a cor verde, ao centro o eixo X representa a cor azul, e à direita o eixo X representa a cor vermelha), as três cores sempre partem do ponto (0,0,0) que representa a cor preto. Na representação à esquerda é vista a diagonal que parte da origem, chamada escala de cinza (*gray scale*), que vai do preto ao branco.

Câmeras digitais capturam imagens através de fotosensores. Se uma determinada incidência de fótons que atinge um sensor for dobrada, seu sinal de saída também será

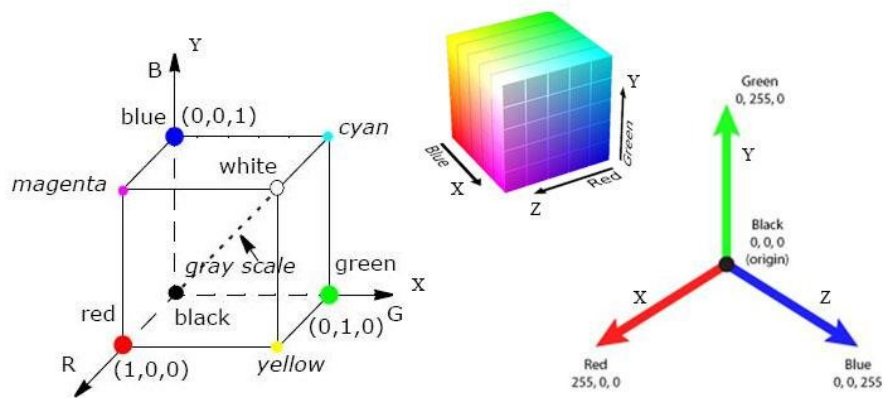


Figura 4 – Distribuição de cores no espaço vetorial tridimensional RGB em formato cúbico, retirado de (BHATTI et al., 2021).

dobrado, representando uma relação linear. Olhos humanos, por sua vez, interpretam a luz de maneira não-linear, ou seja, dependendo das circunstâncias, ao dobrar a quantidade de fótons incidindo sobre a retina, não necessariamente a percepção de brilho será dobrada. A correção de gama serve justamente para diminuir a linearidade da relação de cores de forma a tornar a imagem mais suscetível à percepção subjetiva humana. Na Figura 5 é demonstrado o efeito da variação de gama, onde perceptivelmente um gama maior resulta em uma imagem mais escura, e um gama menor resulta em uma imagem mais clara.



Figura 5 – Representação de imagens utilizando diferentes valores de gama, retirado de (BULL; ZHANG, 2021).

3.1.1.2 Conversão do Espaço de Cores

Câmeras convencionais capturam imagens divididas em um espaço de cores tridimensional, onde cada eixo é representado por uma cor primária (verde, vermelho ou azul). Uma das formas mais comuns de otimizar o processamento de imagem consiste em transformar esse espaço de cores tridimensional em um espaço de cores unidimensional. A representação de cores no espaço unidimensional é a chamada escala de cinza, e pode ser obtida através de uma média das intensidades e das cores de cada pixel. Os valores de intensidade e cor podem ser calculadas a partir de uma média simples, onde o valor final na escala de cinza é a média simples dos eixos, ou escolhendo pesos diferentes para cada eixo. O espaço de cores tridimensional é baseado na percepção humana, que percebe cada cor primária de forma diferente, portanto as conversões de espaço de cores utilizam pesos diferentes para cada cor. A distribuição de pesos utilizada é 0,299 para o vermelho, 0,587 para o verde e 0,114 para o azul, seguindo a recomendação do setor de radiocomunicação

do ITU (*International Telecommunication Union*) (STUDIO..., 2011). A razão para esta distribuição não uniforme se deve ao fato de existirem mais cones sensíveis à cor verde na retina do que à cor vermelho ou azul.

3.1.1.3 Desfoque Gaussiano

Desfocar a imagem é importante para reduzir ruídos e suavizar bordas na imagem. Com este processo, a imagem perde detalhes e torna mais fácil a identificação de objetos. Para desfocar a imagem, cada pixel é misturado com os pixels em volta, e o resultado desta mistura se torna o pixel desfocado. Sendo a forma mais simples uma simples média dos valores dos pixels, diversos métodos podem ser utilizados para desfocar a imagem. Um dos métodos mais utilizados é o Desfoque Gaussiano, que consiste em convolucionar a imagem com uma função Gaussiana, isto é, aplicar um *kernel*⁴ com uma distribuição gaussiana a cada pixel da imagem. A vantagem de utilizar uma distribuição Gaussiana em relação aos modelos uniformes, é permitir a suavização da imagem preservando as bordas presentes. Na Figura 6 são apresentados kerneis de diferentes tamanhos, utilizados no processo de desfoque Gaussiano (BOUTIN, 2016).

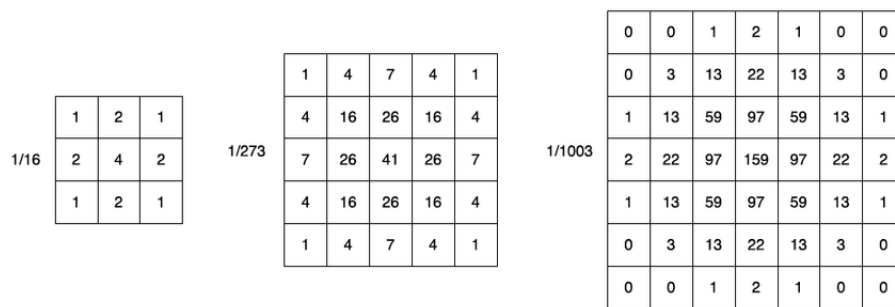


Figura 6 – Aproximação discreta dos *kernel*s Gaussianos 3x3, 5x5 e 7x7, retirado de (SHIPITKO; GRIGORYEV, 2018).

3.1.1.4 Binarização

O processo de binarização de uma imagem consiste em transformar uma imagem da escala de cinza para preto e branco. Um valor limite é definido dentro da escala de cinza, e cada pixel cujo valor ultrapasse este limiar, é atribuída a cor branca ou preta, e a cor oposta caso o valor não atinja o limiar. A imagem obtida ao final, portanto, possui apenas dois valores, 0 ou 1, reduzindo ainda mais a complexidade dos cálculos necessários para o processamento da imagem.

3.1.2 Transformações Morfológicas

Transformações morfológicas são operações baseadas nos formatos das imagens. Além das transformações mais básicas, como rotação e ampliação, existem também transformações mais específicas como dilatação e erosão

⁴ Matriz contendo os valores da transformação a ser aplicada.



Figura 7 – (a) Imagem em RGB; (b) Imagem em escala de cinza; (c) Imagem binária, retirado de (PATEL; PATEL; PATEL, 2018).

3.1.2.1 Rotação

Uma transformação de rotação em duas dimensões consiste em rotacionar um objeto θ graus em torno de um eixo. Para rotacionar um conjunto de pixels, este valor θ deve ser aplicado multiplicando uma matriz de rotação nas posições x e y de cada um deles. A matriz de rotação é uma matriz 2 x 2 na forma:

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (1)$$

Quando uma imagem é rotacionada desta forma, porém, serrilhamentos aparecem, mais perceptivelmente quando linhas retas se tornam diagonais após a rotação. O efeito de serrilhamento faz com que as linhas antes retas fiquem em formato de escada, podendo ocasionar o surgimento de ruído(BERRY, 2013). Um exemplo de rotação rígida é apresentado na Figura 8, onde aplicar uma rotação de $\theta = 15^\circ$ ocasionou no surgimento de ruído.

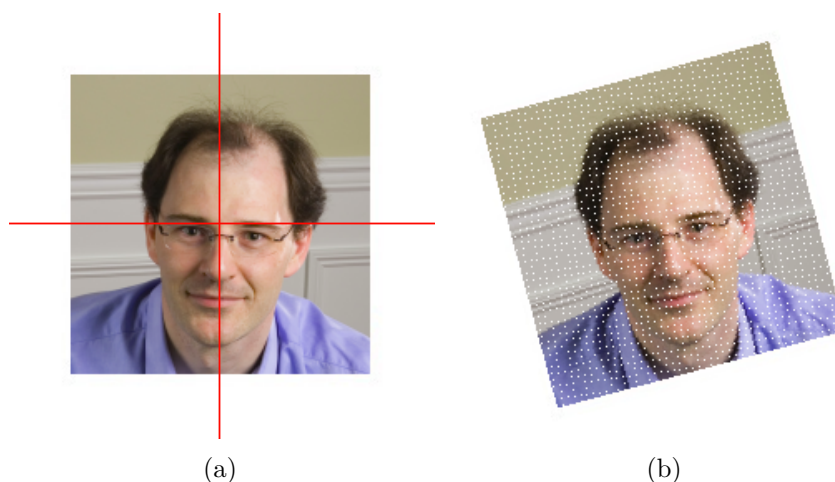


Figura 8 – (a) Imagem original com eixo de rotação (b) Resultado rotacionado com aparecimento de ruído, retirado de (BERRY, 2013).

Este efeito de serrilhamento pode ser reduzido expandindo a matriz de rotação em três matrizes. Estas três matrizes têm características importantes:

- As três são matrizes de cisalhamento

- As três possuem o valor do determinante = 1, ou seja, o resultado rotacionado tem a mesma área inicial

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \quad (2)$$

Com a matriz expandida, cada etapa é aplicada separadamente, e como a área é mantida o resultado final não possui ruído gerado pela rotação. Na Figura 9 é demonstrada a sequência de etapas na rotação utilizando matrizes de cisalhamento, resultando em uma imagem rotacionada e sem ruídos.

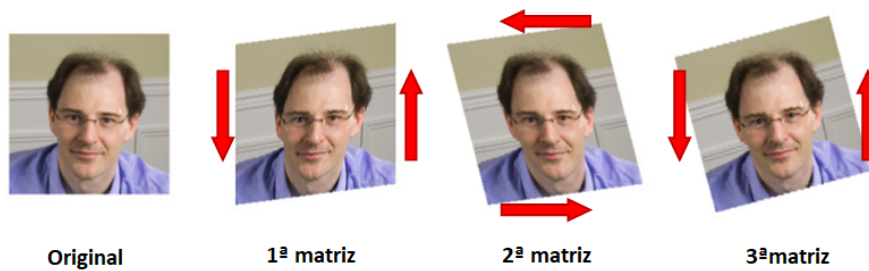


Figura 9 – Etapas de rotação aplicando as matrizes de cisalhamento, retirado de (BERRY, 2013).

3.1.2.2 Redimensionamento

Para redimensionar uma imagem deve-se remover ou acrescentar pixels em uma imagem. Para encontrar os valores destes novos pixels é realizado o processo de interpolação, que consiste em determinar os novos valores com base nos valores disponíveis. A interpolação pode ocorrer de várias maneiras, a depender dos valores disponíveis originalmente e do objetivo a ser atingido. O método mais básico de interpolação aplicado no redimensionamento de imagens é a interpolação por vizinho mais próximo, onde o valor do novo pixel, seja ele uma combinação de diversos pixels numa redução de tamanho, ou um novo pixel de fato adicionado na imagem no aumento do tamanho, é obtido com base no valor do(s) pixel(eis) mais próximos de sua posição na imagem original (ANGEL, 2017). Quando há mais de um pode-se escolher favorecer uma direção em detrimento das outras ou até mesmo realizar uma média entre os valores. Outros exemplos comuns são interpolação linear, interpolação bilinear e interpolação cúbica, que produzem resultados melhores ao custo de maior complexidade de algoritmo.

3.1.2.3 Dilatação e Erosão

Os processos de dilatação e erosão podem ser obtidos aplicando um *kernel*, chamado de elemento estruturante, a cada pixel da imagem. Este elemento estruturante pode ter vários tamanhos e formatos, normalmente quadrados, círculos e retas, normalmente utilizando aquele que mais se aproxima do formato do elemento que se pretende dilatar ou erodir. Na Figura 10 os elementos estruturantes mais comuns são representados em

matrizes 5x5. O kernel retangular é o formato mais simples, e dilata ou erode em todas as direções de maneira uniforme. O kernel elíptico tem zeros nas extremidades da matriz, causando erosão e dilatação levemente arredondadas, enquanto o kernel em cruz possui valor um apenas na linha e coluna centrais, resultando em erosão e dilatação nas direções horizontais e verticais.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Retangular 5x5 Elíptico 5x5 Cruz 5x5

Figura 10 – Exemplos de elementos estruturantes

O processo de dilatação ocorre ao aplicar um *kernel* em um determinado pixel, onde ocorre a substituição do valor do pixel pelo valor máximo encontrado no kernel, causando um efeito aparente de dilatação das regiões claras da imagem. Já o processo de erosão de uma imagem ocorre ao aplicar um *kernel* em um determinado pixel, mas substituir o valor do pixel pelo valor mínimo encontrado no kernel, causando o efeito aparente de encolhimento das regiões claras da imagem. Apesar de serem processos análogos, dilatação e erosão não são processos inversos, ou seja, aplicar uma erosão seguida de uma dilatação, ou vice-versa, não necessariamente fará com que sua imagem volte ao estado original(?).



Figura 11 – Exemplo de processo de dilatação, retirado de (REDA; MATEEVITSI; OFFORD, 2013).

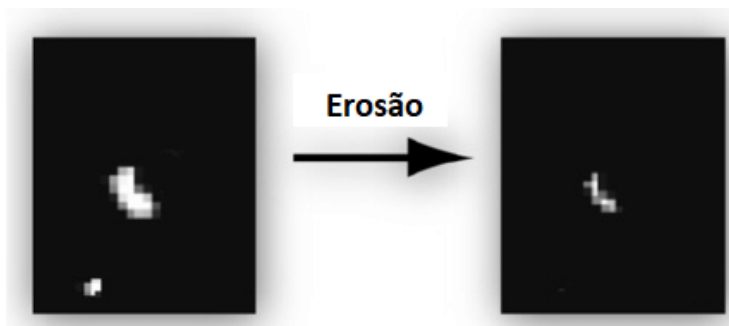


Figura 12 – Exemplo de processo de erosão, retirado de (REDA; MATEEVITSI; OFFORD, 2013).

3.2 Aprendizado de Máquina

3.2.1 Redes Neurais Artificiais

Redes Neurais Artificiais (ANN) são sistemas de processamento computacional altamente inspirados nos sistemas nervosos centrais biológicos, como por exemplo o cérebro humano. Uma ANN é composta por vários nodos interconectados, chamados neurônios, cujo trabalho proporciona um aprendizado distribuído coletivo, visando otimizar o resultado da saída (O'SHEA; NASH, 2015).

Uma tipologia comum de ANN é em forma de camadas. Uma primeira camada, chamada camada de entrada, é onde os dados para treinamento são inseridos, normalmente um vetor. Estes dados são então enviados para uma segunda camada, chamada de camada oculta, responsável por tomar decisões sobre os dados obtidos da camada de entrada, realizando um cálculo em cima destes valores. Na camada de saída ocorre a avaliação do quanto as mudanças realizadas por cada neurônio da camada oculta favoreceu ou desfavoreceu o resultado esperado pela rede. É na camada oculta onde ocorre o aprendizado do algoritmo, e várias camadas ocultas podem ser concatenadas, processo conhecido como aprendizagem profunda (*deep learning*). Um modelo convencional de ANN é demonstrado na Figura 13, com uma camada de entrada, uma camada oculta e uma camada de saída.

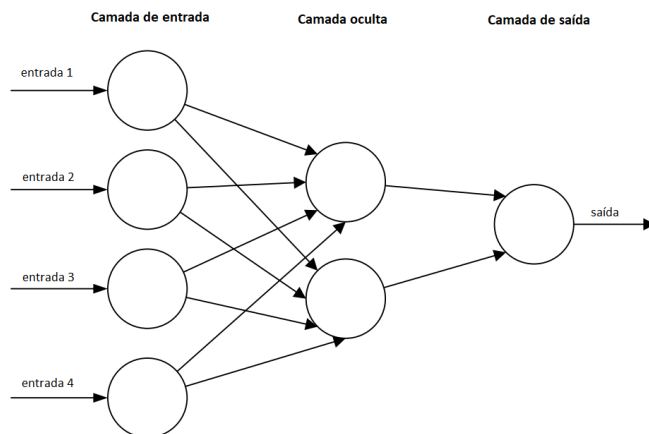


Figura 13 – Modelo de ANN de três camadas, retirado e adaptado de (O'SHEA; NASH, 2015).

3.2.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais, Convolutional Neural Network (CNN), em inglês, são redes neurais de aprendizagem profunda, que como sugere o nome, utilizam convoluções entre suas camadas ocultas. CNNs podem solucionar tarefas mais complexas que ANNs, pois são capazes de reduzir o número de parâmetros de entrada para uma ANN. CNNs são extremamente eficientes em resolver problemas de reconhecimento de padrões, uma característica muito importante desse tipo de rede é a capacidade de obter aspectos abstratos quando os dados são propagados pelas camadas mais profundas. Em uma CNN de reconhecimento de rostos, por exemplo, uma camada fica responsável por identificar os contornos, a camada seguinte responsável por identificar as formas mais simples, a camada

seguinte identifica os elementos mais específicos da face, e assim por diante(ALBAWI; MOHAMMED; AL-ZAWI, 2017).

Uma típica CNN tem camadas de convolução, responsáveis por extrair e mapear dados específicos da entrada, e camadas de *pooling*, que simplificam as informações obtidas na camada de convolução anterior antes de enviar os dados pra a camada à frente, chamada de camada totalmente conectada, que é onde ocorre a classificação dos dados processados, conforme demonstrado na Figura 14. Ao final, a saída da CNN é uma probabilidade de a entrada ser ou não o que era esperado pela rede.

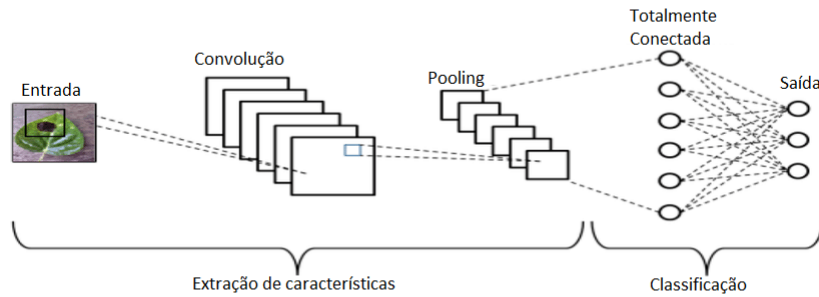


Figura 14 – Representação de uma CNN, retirado e adaptado de (GANATRA; PATEL, 2020).

4 Metodologia

A metodologia utilizada pela plataforma desenvolvida neste trabalho é ilustrada no fluxograma da Figura 10, demonstrando as etapas desde a aquisição de imagens, até a obtenção dos resultados propostos nos objetivos através de técnicas de processamento de imagem e auxílio de algoritmos de inteligência artificial.

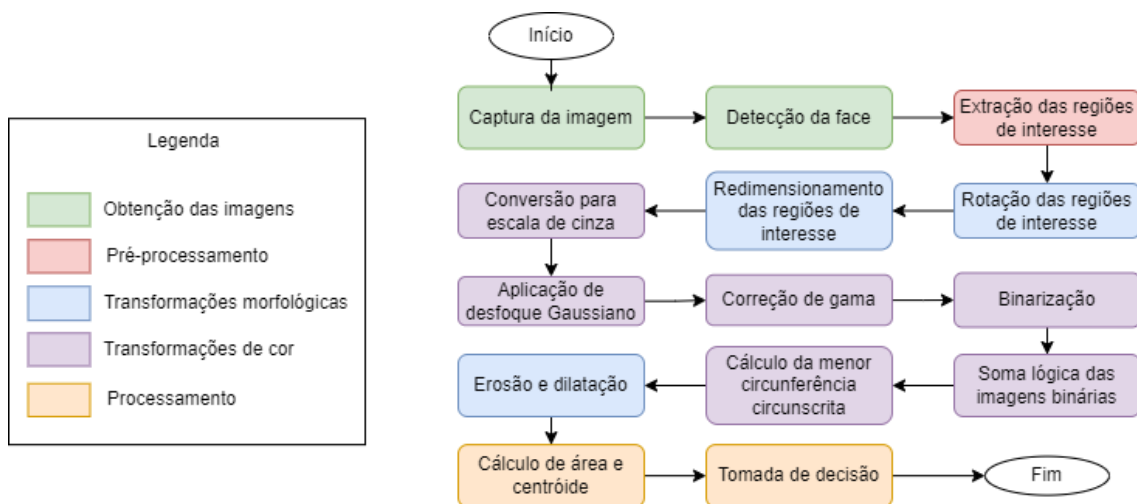


Figura 15 – Fluxograma das etapas de processamento de imagem.

4.1 Equipamento de Aquisição de imagem

A câmera utilizada captura imagens na resolução 640 x 480 e teve seu filtro infravermelho removido. Para proporcionar a iluminação da imagem foi utilizado um refletor de leds infravermelhos. O comprimento de onda infravermelho é captado pela câmera mesmo em ambientes escuros e não gera desconforto e perigo ao usuário mesmo em doses moderadas.

4.2 Software de Processamento de imagem

4.2.1 Algoritmo de Reconhecimento Facial com Dlib

A biblioteca Dlib é um conjunto de ferramentas que utilizam algoritmos de *Machine Learning* para solucionar problemas complexos da vida real. Uma das ferramentas é detector de pontos de referência da face (*face landmark detection*, algoritmo capaz de identificar rostos em imagens e vídeos (KING,). A ferramenta é pré-treinada utilizando o conjunto de dados iBUG 300-W, e é capaz de fornecer 68 pontos de referência para identificação das faces na imagem (ZAFEIRIOU; TZIMIROPOULOS; PANTIC, 2016). Para realizar o *eye-tracking* através de VOG, o primeiro passo deve ser a identificação das regiões de interesse, especificamente os olhos do usuário. Uma vez obtidas as regiões de interesse, uma sorte de operações podem ser realizadas para identificar as pupilas e determinar a direção do olhar do usuário.

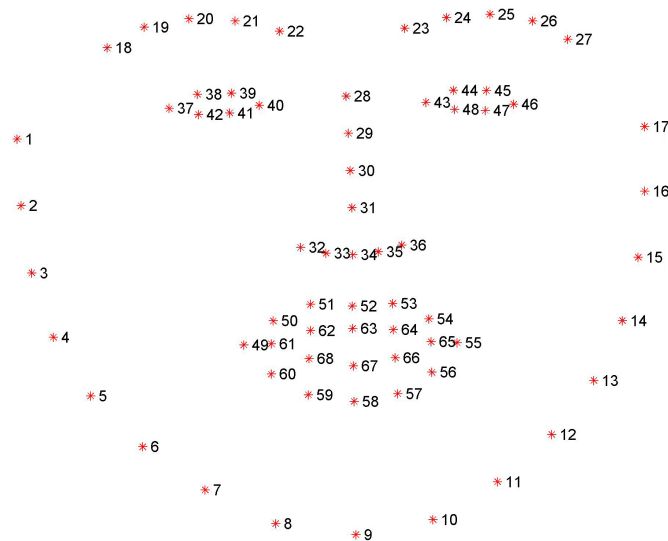


Figura 16 – Os 68 pontos de referência identificados pelo algoritmo, retirado de (ZAFEIRIOU; TZIMIROPOULOS; PANTIC, 2016).

Utilizando este algoritmo, é possível identificar com facilidade as duas regiões de interesse referentes aos olhos do usuário. Na Figura 16 é apresentado o modelo com os 68 pontos de referência da face gerado pelo algoritmo.

4.3 Biblioteca OpenCV

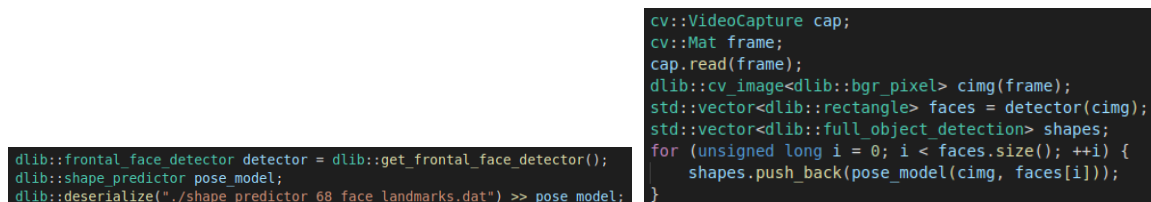
O OpenCV é uma biblioteca de código aberto de visão computacional e *Machine Learning*, construída para fomentar a criação de produtos comerciais que utilizem a tecnologia. A biblioteca contém milhares de algoritmos otimizados para processamento de imagens e vídeo, sendo utilizada por diversas empresas, como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, etc (BRADSKI, 2000).

O software foi desenvolvido utilizando a linguagem C++ em conjunto as bibliotecas descritas.

4.4 Pré-processamento: Detecção da Face e Regiões de Interesse

Conforme descrito no fluxograma, os primeiros passos para o rastreamento ocular do usuário é identificar seus olhos. A partir dos pontos detectados pelo algoritmo de pontos de referência da face, podemos extrair duas imagens, uma para cada olho, do menor retângulo circunscrito aos pontos de referência dos olhos.

Na Figura 17(a) é instanciado o algoritmo detector de faces e o arquivo com o resultado dos treinamentos para a identificação dos pontos de referência da face é carregado no detector de formas. Em seguida, na Figura 17(b), é instanciada a leitura da câmera, que armazena o frame lido na variável frame. A imagem da variável frame é convertida do formato utilizado pela biblioteca OpenCV para o formato utilizado pela biblioteca Dlib, para então ser utilizado como entrada para o algoritmo detector de faces instanciado anteriormente. Iterando sobre cada face encontrada, o algoritmo de detecção dos pontos de referência é aplicado.



```
dlib::frontal_face_detector detector = dlib::get_frontal_face_detector();
dlib::shape_predictor pose_model;
dlib::deserialize("./shape_predictor_68_face_landmarks.dat") >> pose_model;
```

```
cv::VideoCapture cap;
cv::Mat frame;
cap.read(frame);
dlib::cv_image<dlib::bgr_pixel> cimg(frame);
std::vector<dlib::rectangle> faces = detector(cimg);
std::vector<dlib::full_object_detection> shapes;
for (unsigned long i = 0; i < faces.size(); ++i) {
    shapes.push_back(pose_model(cimg, faces[i]));
}
```

(a) (b)

Figura 17 – (a) Inicialização do detector de faces (b) Leitura da imagem e detecção das faces e pontos de referência.

Na Figura 18 são extraídos os pontos de referência que delimitam cada um dos olhos, conforme visto na Figura 16. Estes pontos são então utilizados para encontrar o menor retângulo circunscrito à eles, utilizando a função `minAreaRect`, ilustrado na Figura 19. Os centros e ângulos dos retângulos encontrados são utilizados para criar a matriz de rotação de cada olho, utilizando a função `getRotationMatrix2D`, mostrado na Figura 20. Estas matrizes de rotação são aplicadas à imagem original através da função `warpAffine`, e com a função `getRectSubPix` são extraídos o retângulos das regiões de interesse das imagens rotacionadas.

```

std::vector<cv::Point2f> left_eye_points;
std::vector<cv::Point2f> right_eye_points;

left_eye_points.push_back(cv::Point2f(shapes.at(s).part(36).x(), shapes.at(s).part(36).y()));
left_eye_points.push_back(cv::Point2f(shapes.at(s).part(37).x(), shapes.at(s).part(37).y()));
left_eye_points.push_back(cv::Point2f(shapes.at(s).part(38).x(), shapes.at(s).part(38).y()));
left_eye_points.push_back(cv::Point2f(shapes.at(s).part(39).x(), shapes.at(s).part(39).y()));
left_eye_points.push_back(cv::Point2f(shapes.at(s).part(40).x(), shapes.at(s).part(40).y()));
left_eye_points.push_back(cv::Point2f(shapes.at(s).part(41).x(), shapes.at(s).part(41).y()));

right_eye_points.push_back(cv::Point2f(shapes.at(s).part(42).x(), shapes.at(s).part(42).y()));
right_eye_points.push_back(cv::Point2f(shapes.at(s).part(43).x(), shapes.at(s).part(43).y()));
right_eye_points.push_back(cv::Point2f(shapes.at(s).part(44).x(), shapes.at(s).part(44).y()));
right_eye_points.push_back(cv::Point2f(shapes.at(s).part(45).x(), shapes.at(s).part(45).y()));
right_eye_points.push_back(cv::Point2f(shapes.at(s).part(46).x(), shapes.at(s).part(46).y()));
right_eye_points.push_back(cv::Point2f(shapes.at(s).part(47).x(), shapes.at(s).part(47).y()));

```

Figura 18 – Inicialização de arrays com os pontos de referência que delimitam os olhos.

```

cv::RotatedRect left_eye_rect = cv::minAreaRect(left_eye_points);
cv::RotatedRect right_eye_rect = cv::minAreaRect(right_eye_points);

```

Figura 19 – Inicialização do menor retângulo circunscrito aos pontos de cada olho.

```

cv::Mat left_rot_mat = cv::getRotationMatrix2D(left_eye_rect.center, left_eye_rect.angle, 1.0);
cv::Mat right_rot_mat = cv::getRotationMatrix2D(right_eye_rect.center, right_eye_rect.angle, 1.0);
cv::warpAffine(frame, left_rotated, left_rot_mat, frame.size());
cv::warpAffine(frame, right_rotated, right_rot_mat, frame.size());
cv::getRectSubPix(left_rotated, left_size, left_center, left_bounding_crop);
cv::getRectSubPix(right_rotated, right_size, right_center, right_bounding_crop);

```

Figura 20 – Inicialização da matriz de rotação a partir do retângulo encontrado e transformação de rotação em torno do ponto central de cada retângulo.



Figura 21 – Detecção de face e extração das áreas de interesse.

Nesta etapa do processo duas transformações morfológicas básicas são aplicadas: rotação e redimensionamento. A correção de angulação é aplicada individualmente em cada imagem com base no ângulo entre a linha que une os pontos mais distantes de cada olho e a horizontal. Dessa forma, a imagem será sempre disposta com o seu eixo maior

paralelo à horizontal, exibido na Figura 21. Em seguida, as imagens são redimensionadas para um tamanho fixo, neste caso 320 x 120.

4.5 Processamento

Nesta etapa transformações de cor são aplicadas em cada uma das regiões de interesse, a fim de identificar as pupilas do usuário. Primeiramente é feita uma conversão do espaço de cores para escala de cinza, utilizando a função `cvtColor` com o modo de conversão `COLOR_BGR2GRAY`, e depois é aplicado um desfoque Gaussiano 5x5, mostrado na Figura 22.

```
cv::cvtColor(left_bounding_crop, left_gray_crop, cv::COLOR_BGR2GRAY);
cv::cvtColor(right_bounding_crop, right_gray_crop, cv::COLOR_BGR2GRAY);

cv::GaussianBlur(left_gray_crop, left_gray_crop, cv::Size(5,5), 0);
cv::GaussianBlur(right_gray_crop, right_gray_crop, cv::Size(5,5), 0);
```

Figura 22 – Conversão do espaço de cor para escala de cinza e aplicação do desfoque Gaussiano.

Na Figura 23 é representada a correção de gama aplicada em cada uma das imagens. Iterando sobre as imagens, a função `saturate_cast` é aplicada a cada pixel, alterando seu valor. Na Figura 24 é apresentado o resultado das transformações aplicadas até o momento. A última transformação de cor, o processo de binarização, é aplicado nas imagens através da função `threshold`. Utilizando a função `threshold` com o modo `THRESH_BINARY_INV`, o resultado final é invertido, o que pode ser visto na Figura 26, onde as regiões brancas representam as partes escuras da imagem pré-transformação.

```
for( int y = 0; y < left_gray_crop.rows; y++ ) {
    for( int x = 0; x < left_gray_crop.cols; x++ ) {
        for( int c = 0; c < left_gray_crop.channels(); c++ ) {
            left_gray_crop.at<cv::Vec3b>(y,x)[c] =
                cv::saturate_cast<uchar>(
                    (alpha_slider/100.0)*left_gray_crop.at<cv::Vec3b>(y,x)[c]+ betha_slider
                );
        }
    }
}

for( int y = 0; y < right_gray_crop.rows; y++ ) {
    for( int x = 0; x < right_gray_crop.cols; x++ ) {
        for( int c = 0; c < right_gray_crop.channels(); c++ ) {
            right_gray_crop.at<cv::Vec3b>(y,x)[c] =
                cv::saturate_cast<uchar>(
                    (alpha_slider/100.0)*right_gray_crop.at<cv::Vec3b>(y,x)[c] + betha_slider
                );
        }
    }
}
```

Figura 23 – Correção de gama.



Figura 24 – Resultado após a transformação do espaço de cor para escala de cinza, aplicação do desfoque Gaussiano e correção de gama. Elaborado pelo autor.

O valor limite definido para o processo de binarização é aplicado individualmente em cada região de interesse, e deve ser ajustado para que identifique a pupila em qualquer posição do olhar enquanto mantém o mínimo de ruído na imagem.

```
cv::threshold(left_gray_crop, left_bin_crop, left_bin_slider, 255, cv::THRESH_BINARY_INV);
cv::threshold(right_gray_crop, right_bin_crop, right_bin_slider, 255, cv::THRESH_BINARY_INV);
```

Figura 25 – Binarização.

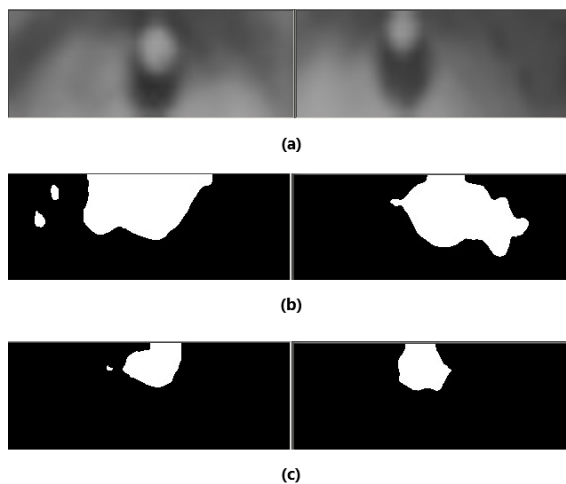


Figura 26 – Resultado da binarização da imagem. (a) Imagem em escala de cinza (b) Binarização com muito ruído (c) Binarização com ruído reduzido.

Depois de obter as imagens binárias, um produto lógico entre as duas imagens é realizado utilizando a função *bitwise_and*, demonstrado na Figura 27. Este produto atua como um segundo filtro de ruído, eliminando das imagens toda informação que não for comum a ambas. Em seguida são identificados os contornos presentes na imagem utilizando a função *findContours*, visto na Figura 28. Então, na Figura 29, é calculado e desenhado o menor círculo circunscrito aos contornos encontrados na imagem resultante do produto lógico.


```
cv::Mat bin_sum;
cv::bitwise_and(left_bin_crop, right_bin_crop, bin_sum);
```

Figura 27 – Soma lógica das imagens binárias.

```
std::vector<std::vector<cv::Point> > sum_contours;
std::vector<cv::Vec4i> sum_hierarchy;
cv::findContours(bin_sum, sum_contours, sum_hierarchy, cv::RETR_EXTERNAL, cv::CHAIN_APPROX_SIMPLE );
```

Figura 28 – Detecção de contornos da imagem.

```
for( size_t i = 0; i < sum_contours.size(); i++ )
{
    approxPolyDP( sum_contours[i], sum_contours_poly[i], circledist, true );
    minEnclosingCircle( sum_contours_poly[i], sum_centers[i], sum_radius[i] );
    cv::circle( bin_sum, sum_centers[i], (int)sum_radius[i], cv::Scalar(255, 0, 0), -1);
}
```

Figura 29 – Cálculo e desenho do menor círculo circunscrito às regiões encontradas.

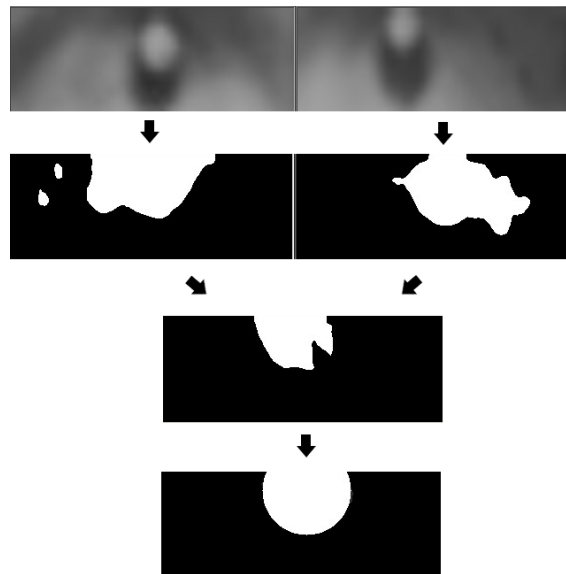


Figura 30 – Soma lógica das regiões de interesse e desenho de círculo circunscrito à região comum.

O círculo obtido representa a pupila do usuário, tendo como parâmetros relevantes para a tomada de decisão a área e o seu centróide. Ao movimentar os olhos, a área e posição do centróide se alteram, permitindo identificar a direção do olhar.

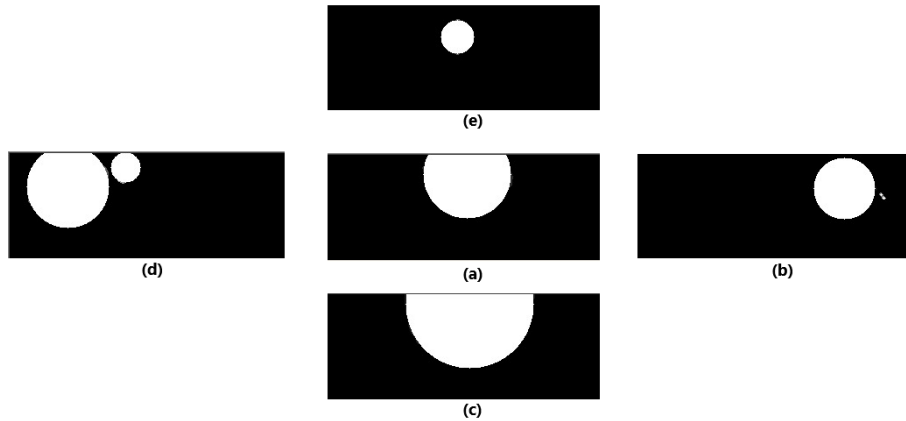


Figura 31 – Exemplos de variação da área e posição do círculo obtido (a) usuário olhando para o centro (b) usuário olhando para a direita (c) usuário olhando para baixo, (d) usuário olhando para a esquerda (e) usuário olhando para cima.

Em posse do círculo que representa a pupila do usuário, novas transformações morfológicas podem ser aplicadas para facilitar a determinação da direção do olhar. O excesso de ruído pode prejudicar a interpretação, para isso, é realizado um processo de erosão e dilatação, que ao diminuir e aumentar as áreas encontradas, além de ajudar a reduzir o ruído, melhoram a precisão do cálculo de área diminuindo sua amplitude de variação. Na figura 32, utilizando a função *getStructuringElement*, é instanciado o elemento estruturante que será utilizado nas funções *erode* e *dilate*, erosão e dilatação, respectivamente. Com o parâmetro *MORPH_ELLIPSE*, o elemento estruturante gerado é de formato elíptico, e foi determinado para ter tamanho 8x3, porque é o menor tamanho possível para representar a mesma taxa de proporção das regiões de interesse.

```
cv::Mat erode_mat = cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(16, 6));
cv::erode(bin_sum, bin_sum, erode_mat, cv::Point(-1, -1), iterations_erode);
cv::dilate(bin_sum, bin_sum, erode_mat, cv::Point(-1, -1), iterations_dilate);
```

Figura 32 – Criação do elemento estruturante, erosão e dilatação dos círculos encontrados.



Figura 33 – Região encontrado após processo de erosão.

Para identificar as piscadas de cada olho é realizado um cálculo de proporção ocular utilizando as distâncias Euclidianas entre os pontos de referência gerados que delimitam os olhos. Esta proporção é direta em relação a área interna dos pontos de referência, da forma que uma redução brusca deste valor representa uma piscada do usuário (KUWAHARA et al., 2022). Uma vez que a piscada de um olho é detectada, o sistema atrasa em alguns frames a análise de novas piscadas a fim de eliminar possíveis leituras duplicadas.

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Figura 34 – Fórmula para cálculo da proporção ocular.

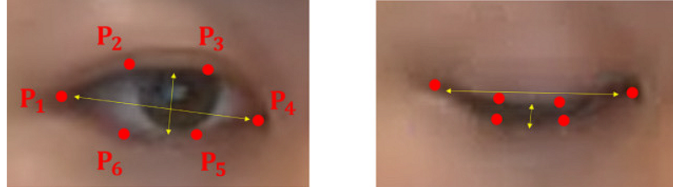


Figura 35 – Representação da diferença de proporção em um olho aberto e um olho fechado. Retirado de (KUWAHARA et al., 2022).

Após todas as transformações descritas, é feita uma classificação do resultado a partir da média dos últimos parâmetros obtidos com a imagem pós-processamento. Os parâmetros relevantes para a análise são a área da região encontrada e a posição do seu centróide.

5 Resultados Preliminares

5.1 Ambiente Experimental

Para a execução deste trabalho foi utilizado uma máquina equipada com um processador Intel Core i5 2,5 GHz com 8 GB de memória RAM no sistema operacional Ubuntu 20.04. Para testar a metodologia proposta, a câmera foi disposta imediatamente abaixo de um monitor LCD de 24 polegadas e o refletor com leds infravermelhos imediatamente acima do mesmo monitor. Tanto a câmera quanto o refletor foram posicionados com auxílio de tripés. Durante os experimentos, os usuários ficaram a cerca de 70 centímetros da câmera e do refletor.

Os primeiros experimentos foram realizados com a câmera acima do monitor e o refletor acima da câmera. Essa disposição do hardware ocasionou na dificuldade da identificação da pupila quando o usuário olhava para baixo, devido à cobertura por parte da pálpebra. Para melhorar a identificação do olhar para baixo, a câmera foi disposta abaixo do monitor, tornando possível enxergar a pupila mesmo com a pálpebra cobrindo parcialmente os olhos. Na Figura 36 é representado como o ambiente experimental foi organizado.

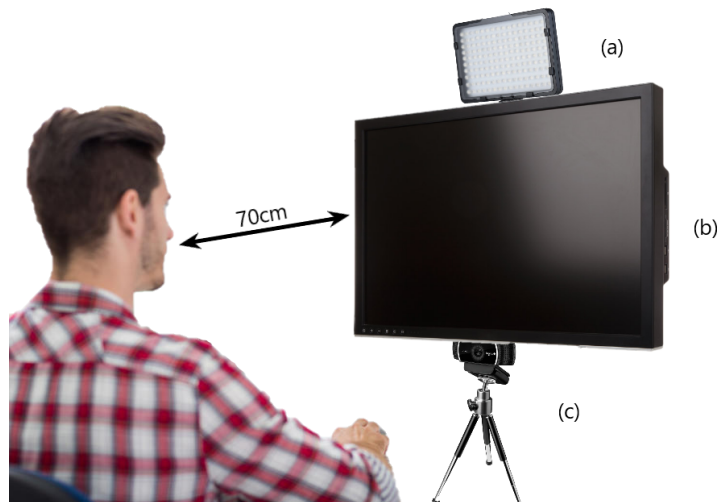


Figura 36 – Diagrama esquemático do ambiente experimental. (a) Refletor de LEDs infravermelhos posicionados imediatamente acima do monitor (b) Monitor (c) Câmera disposta imediatamente abaixo do monitor.

5.2 Resultados da Aquisição e Dataset Utilizado

Utilizando o conjunto de *hardware* e *software* descritos anteriormente, foram gravados três vídeos, cada um contendo um usuário realizando um protocolo. No protocolo definido, os usuários foram instruídos a olhar para cinco pontos fixos dispostos na tela em uma sequência específica, podendo piscar livremente durante todo o processo. Ao final, 3430 frames foram obtidos e analisados.

5.3 Resultados dos Métodos de Análise de Imagem

Para determinar quantitativamente a precisão da detecção e identificação do direcionamento do olhar, foi contabilizado o número de frames em que o resultado apresentado pelo software era condizente com a direção real do olhar do usuário, e a quantidade de piscadas identificadas pelo software foi comparada com o número de piscadas reais dos usuários.

Para avaliar o direcionamento do olhar, o software é capaz de identificar a direção no eixo vertical e horizontal separadamente, conforme demonstrado nas Figuras 37 e 38 . É contabilizado como um frame errado, portanto, qualquer frame onde a direção apontada pelo software em qualquer um dos eixos não corresponda à direção real do olhar do usuário.



Figura 37 – Demonstração da detecção de direção do olhar no eixo horizontal.



Figura 38 – Demonstração da detecção de direção do olhar no eixo vertical.

Frames	Corretos	Errados	Total	Precisão
Usuário 1	986	45	1031	95,64%
Usuário 2	871	313	1184	73,56%
Usuário 3	1163	52	1215	95,72%
Totais	3020	410	3430	88,31%

Tabela 2 – Tabela dos resultados obtidos na identificação da direção do olhar do usuário.

Para avaliar a identificação de piscadas do usuário, o software é capaz de identificar quando cada olho pisca separadamente, demonstrado na Figura 39. É contabilizado como um frame errado, portanto, qualquer frame em que o usuário pisca qualquer um dos olhos e o software não reconhece a piscada.

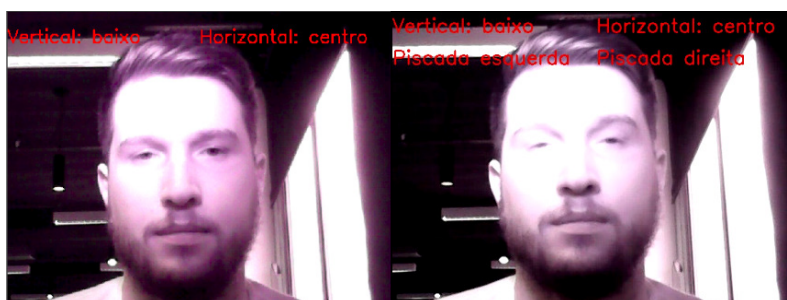


Figura 39 – Demonstração da detecção de piscada.

Ao final, a resposta de saída do programa foi utilizada como a entrada de comandos para o mouse, efetivamente deslocando o cursor para as direções detectadas, e efetuando cliques de acordo com a piscada detectada. O tempo médio de processamento de cada frame foi de aproximadamente 66 milissegundos, valor considerado instantâneo para um

Piscadas	Corretas (esq.)	Erradas (esq.)	Corretas (dir.)	Erradas (dir.)	Precisão (esq.)	Precisão (dir.)
Usuário 1	2	0	2	1	100%	66,67%
Usuário 2	9	2	8	3	81,82%	72,73%
Usuário 3	2	1	2	0	66,67%	100%
Totais	13	3	12	4	82,83%	79,80%

Tabela 3 – Tabela dos resultados obtidos na contagem de piscadas do usuário.

ser humano e bastante baixo, o que sugere que o processamento poderia ser realizado de maneira satisfatória por computadores portáteis de baixo custo, em vez de um notebook, o que pode reduzir ainda mais os custos. Na figura 40 está demonstrada a execução do software desenvolvido.

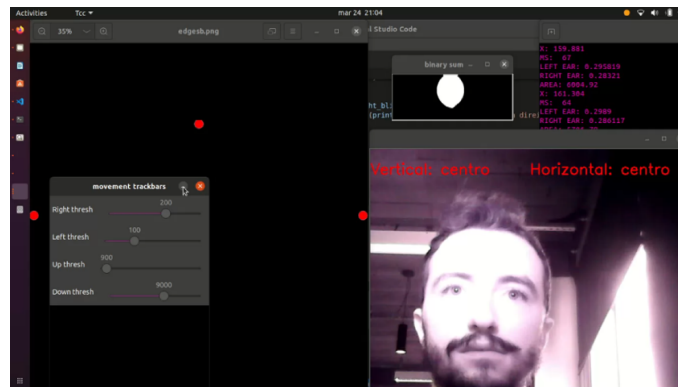


Figura 40 – Representação da plataforma em execução. Elaborado pelo autor

Combinando as informações apresentados nas Tabelas 2 e 3, a precisão média geral dos resultados obtidos foi de 84,81%, identificando corretamente 4 de cada 5 comandos, ao , valor que pode ser considerado instantâneo para um ser humano.

6 Conclusões e Trabalhos Futuros

A plataforma apresentada foi capaz de captar as imagens com pouca interferência do espectro de luz visível, e realizar transformações na imagem de forma a obter parâmetros que puderam ser utilizados para classificar a imagem e gerar um resultado coerente. Esse resultado foi utilizado para gerar comando de movimentação e cliques em um computador.

Os resultados obtidos podem ser considerados ótimos, levando-se em consideração a velocidade de processamento de cada frame, o custo e facilidade de acesso aos equipamentos, e principalmente a qualidade da imagem captada pela câmera, frente à precisão média de aproximadamente 85% de acerto na detecção da direção dos olhares e piscadas.

Diferente dos trabalhos correlatos apresentados anteriormente, a plataforma apresentada não utiliza dispositivos vestíveis, sendo totalmente não invasiva, e pode ser uma alternativa para facilitar a interação humano-computador por parte de pessoas com incapacidades motoras.

A calibragem do sistema depende do tamanho da tela a ser utilizada, sendo mais fácil de atingir um ponto satisfatório com telas maiores. A mesma solução pode se mostrar inviável para monitores pequenos, já que o olho se movimenta pouco para ir de uma extremidade à outra da imagem. A distância entre o usuário e a câmera também possui

limitações. O usuário não pode estar muito próximo da câmera já que seu rosto precisa estar inteiramente visível para que a detecção aconteça. Além disso, o usuário também não pode estar distante ao ponto de que as regiões de interesse sejam muito pequenas para que qualquer diferença expressiva nas direções do olhar sejam identificáveis.

Algumas mudanças na plataforma podem ser feitas tanto em níveis de *hardware* quando em níveis de *software* para melhorar a classificação do resultado. Uma câmera com maior resolução, ou até mesmo a utilização de múltiplas câmeras podem ser opções de melhoria de *hardware*. A utilização de computadores portáteis podem ser uma boa alternativa para baratear custos e aumentar a portabilidade da plataforma.

Já em *software*, uma aplicação auxiliar para realizar comandos mais complexos, como segurar e arrastar um arquivo, seria essencial para uma utilização plena, e possíveis melhorias no processamento poderiam ser obtidas testando outros kernels nas transformações por convolução.

Referências

- ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 2017. Citado na página 19.
- ANGEL, A. *Nearest neighbor interpolation*. 2017. Disponível em: <<https://www.imageprocessing.com/2017/11/nearest-neighbor-interpolation.html>>. Citado na página 16.
- BARBUCEANU, F.; ANTONYA, C. Eye tracking applications. *Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I*, Transilvania University of Brasov, v. 2, p. 17, 2009. Citado na página 9.
- BAZAR, N. S.; BRIGHAM, F. J. Eye-tracking technology: an introduction. *The Telecommunications Review*, Citeseer, v. 18, p. 46–48, 2007. Citado na página 8.
- BERRY, N. *How to rotate an image*. 2013. Disponível em: <<https://datagenetics.com/blog/august32013/index.html>>. Citado 2 vezes nas páginas 15 e 16.
- BHATTI, U. et al. Advanced color edge detection using clifford algebra in satellite images. *IEEE Photonics Journal*, PP, p. 1–1, 02 2021. Citado na página 13.
- BOUTIN, M. *Edge detection with Gaussian Blur*. 2016. Disponível em: <https://www.projectrhea.org/rhea/index.php/Edge_Detection_with_Gaussian_Blur>. Citado na página 14.
- BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. Citado na página 21.
- BRETEAU, J.-M. *Color systems RGB and CIE XYZ*. 2007. Disponível em: <http://www.optique-ingenieur.org/en/courses/OPI_ang_M07_C02/co/Contenu_07.html>. Citado na página 12.
- BRIDGE, K. *Processes and threads*. 2021. Disponível em: <<https://docs.microsoft.com/en-us/windows/win32/procthread/processes-and-threads>>. Citado na página 8.
- BULL, D. R.; ZHANG, F. Chapter 4 - digital picture formats and representations. In: BULL, D. R.; ZHANG, F. (Ed.). *Intelligent Image and Video Compression (Second Edition)*. Second edition. Oxford: Academic Press, 2021. p. 107–142. ISBN 978-0-12-820353-8. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B978012820353800013X>>. Citado na página 13.
- BULLING, A.; ROGGEN, D.; TRÖSTER, G. It's in your eyes: Towards context-awareness and mobile hci using wearable eog goggles. In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. New York, NY, USA: Association for Computing Machinery, 2008. (UbiComp '08), p. 84–93. ISBN 9781605581361. Disponível em: <<https://doi.org/10.1145/1409635.1409647>>. Citado na página 10.
- CLAY, V.; KÖNIG, P.; KÖNIG, S. *Eye tracking in virtual reality*. Bern Open Publishing, 2019. Disponível em: <<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7903250/>>. Citado na página 9.

- GANATRA, N.; PATEL, A. Performance analysis of fine-tuned convolutional neural network models for plant disease classification. p. 293–305, 01 2020. Citado na página 19.
- HWANG, T. *Computational Power and the Social Impact of Artificial Intelligence*. 2018. Citado na página 8.
- INTEL. *Intel® Hyper-Threading Technology Technical User's Guide*. [S.l.], 2003. Disponível em: <<http://www.cslab.ece.ntua.gr/courses/advcomparch/2007/material/readings/Intel%20Hyper-Threading%20Technology.pdf>>. Citado na página 8.
- KALLEM, S. R. Artificial intelligence algorithms. *IOSR Journal of Computer Engineering*, v. 6, n. 4, p. 01–08, 2012. Citado na página 8.
- KING, D. E. *Dlib C++ library*. Disponível em: <<http://dlib.net/>>. Citado na página 20.
- KUWAHARA, A. et al. Eye fatigue estimation using blink detection based on eye aspect ratio mapping(earn). *Cognitive Robotics*, v. 2, p. 50–59, 2022. ISSN 2667-2413. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2667241322000039>>. Citado 2 vezes nas páginas 26 e 27.
- LÉVÊQUE, L. et al. State of the art: Eye-tracking studies in medical imaging. *Ieee Access*, IEEE, v. 6, p. 37023–37034, 2018. Citado na página 9.
- LITWILLER, D. Ccd vs. cmos: Facts and fiction. *PHOTONICS SPECTRA* ©, Laurin Publishing Co. Inc., Jan 2001. Citado na página 8.
- LUPU, R. G.; UNGUREANU, F.; SIRITEANU, V. Eye tracking mouse for human computer interaction. In: *2013 E-Health and Bioengineering Conference (EHB)*. [S.l.: s.n.], 2013. p. 1–4. Citado na página 9.
- MAJARANTA, P.; BULLING, A. Eye tracking and eye-based human–computer interaction. In: _____. *Advances in Physiological Computing*. London: Springer London, 2014. p. 39–65. ISBN 978-1-4471-6392-3. Disponível em: <https://doi.org/10.1007/978-1-4471-6392-3_3>. Citado na página 9.
- NASSAU, K. *Tristimulus System*. Encyclopædia Britannica, inc., 2009. Disponível em: <<https://www.britannica.com/science/tristimulus-system>>. Citado na página 12.
- O'SHEA, K.; NASH, R. *An Introduction to Convolutional Neural Networks*. 2015. Citado 2 vezes nas páginas 8 e 18.
- PATEL, I.; PATEL, S.; PATEL, A. Analysis of various image preprocessing techniques for denoising of flower images. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*, v. 6, p. 1111–1117, 05 2018. Citado na página 15.
- POYNTON, C.; BOOKS24X7, I.; INC, E. I. *Digital Video and HD: Algorithms and Interfaces*. Elsevier Science, 2003. (Computer Graphics). ISBN 9781558607927. Disponível em: <<https://books.google.com.br/books?id=ra1lcAwgvq4C>>. Citado na página 12.
- REDA, K.; MATEEVITSI, V.; OFFORD, C. A human-computer collaborative workflow for the acquisition and analysis of terrestrial insect movement in behavioral field studies. *EURASIP Journal on Image and Video Processing*, v. 2013, p. 48, 12 2013. Citado na página 17.

SHIPITKO, O.; GRIGORYEV, A. Gaussian filtering for fpga based image processing with high-level synthesis tools. In: . [S.l.: s.n.], 2018. Citado na página 14.

STUDIO encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. Geneva, CH, 2011. v. 2017. Citado na página 14.

WEDEL, M.; PIETERS, R. A review of eye-tracking research in marketing. In: _____. *Review of Marketing Research, Volume 4*. United States: M.E. Sharpe Inc., 2007. p. 123–146. Citado na página 9.

WHITMIRE, E. et al. Eyecontact: Scleral coil eye tracking for virtual reality. In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. New York, NY, USA: Association for Computing Machinery, 2016. (ISWC '16), p. 184–191. ISBN 9781450344609. Disponível em: <<https://doi.org/10.1145/2971763.2971771>>. Citado na página 10.

ZAFEIRIOU, S.; TZIMIROPOULOS, G.; PANTIC, M. 300 w: Special issue on facial landmark localisation “in-the-wild”. *Image and Vision Computing*, v. 47, p. 1–2, 2016. Citado na página 20.

ZHENG, C.; USAGAWA, T. A rapid webcam-based eye tracking method for human computer interaction. In: *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*. [S.l.: s.n.], 2018. p. 133–136. Citado na página 11.