UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

FÁBIO VICENTE OLIVEIRA

AVALIAÇÃO DE ALTERNATIVAS AO MOUSE UTILIZANDO TÉCNICAS DA VISÃO COMPUTACIONAL

Joinville
2022

FÁBIO VICENTE OLIVEIRA

AVALIAÇÃO DE ALTERNATIVAS AO MOUSE UTILIZANDO TÉCNICAS DA VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Prof. Dr. Benjamin Grando Moreira

Joinville
2022

Dedico este trabalho aos meus pais.

AGRADECIMENTOS

Agradeço aos meus pais e a todos aqueles que conheci durante a minha vida acadêmica, professores e alunos.

RESUMO

Os principais dispositivos de interação humano-computador (mouse e teclado) são limitados em capturar muitos dos comportamentos humanos ao ponto que pessoas com excesso de carga de trabalho pelo mouse podem desenvolver tenossinovite, encontrando dificuldade em interagir com o computador. Sendo assim, este trabalho apresenta uma pesquisa que desenvolveu uma aplicação de interação humano-computador utilizando técnicas de visão computacional que permite uma pessoa controlar o cursor do mouse e os cliques a partir de formas alternativas, como a palma da mão, dedos e olhos. O projeto é implementado na linguagem de programação Python, fazendo uso de uma webcam e bibliotecas de código aberto (OpenCV, Dlib, Mediapipe, Pynput e PyQt5) para implementar a detecção da palma da mão e dos olhos. Os resultados obtidos demonstram que é possível desenvolver aplicações que controlam os principais comandos do computador utilizando uma webcam, embora alguns aprimoramentos precisem ser realizados para uma melhor experiência de uso das soluções apresentadas.

Palavras-chave: Visão computacional. Interação humano-computador. Controle do mouse. Computação perceptiva.

ABSTRACT

The main human-computer interaction devices (mouse and keyboard) are limited in capturing many of the human behaviors to the point that people with excessive mouse workload can develop tenosynovitis, finding it difficult to interact with the computer. Therefore, this work presents a research that developed a human-computer interaction application using computer vision techniques that allows a person to control the mouse cursor and clicks from alternative ways, such as the palm, fingers and eyes. The project is implemented in the Python programming language, making use of a webcam and open source libraries (OpenCV, Dlib, Mediapipe, Pynput and PyQt5) to implement palm and eye detection. The results obtained demonstrate that it is possible to develop applications that control the main commands of the computer using a webcam, although some improvements need to be made for a better experience of using the presented solutions.

Keywords: Computer vision. Human-computer interaction. Mouse Control. Perceptual computing.

LISTA DE FIGURAS

Figura 1 – Rede neural 19
Figura 2 – Resultado do histograma de gradientes orientados 24
Figura 3 – Recursos Haar usado no detector de rosto de Viola-Jones 25
Figura 4 – Máquina de vetores de suporte 27
Figura 5 – Rede neural da técnica YOLO implementada por Redmon et al. (2016) 29
Figura 6 – Marcações da mão feita pela biblioteca Mediapipe 31
Figura 7 – Marcações do rosto feita pela biblioteca Dlib 33
Figura 8 – Interface de usuário do eViacam 35
Figura 9 – Opções de configuração do eViacam 36
Figura 10 – Protótipo dos gestos 38
Figura 11 – Gestos identificados pelo algoritmo 40
Figura 12 – Controle do mouse feito pelo algoritmo 41
Figura 13 – Classes usadas na ferramenta 43
Figura 14 – Aplicação do filtro gaussiano 45
Figura 15 – Menu do aplicativo 48
Figura 16 – Teclado virtual 49
Figura 17 – Pontos do nariz e olhos 49
Figura 18 – Posições do dedinho 50
Figura 19 – Dedos para clicar e arrastar 50
Figura 20 – Rotação da mão 51
Figura 21 – Lista de opções 52
Figura 22 – Pontos da mão utilizados para movimentar o cursor 53
Figura 23 – Região de interesse do olho usada no algoritmo 53
Figura 24 – Olho na escala de cinza e com desfoque gaussiano 54
Figura 25 – Pupila detectada em branco e marcações finais 54

LISTA DE QUADROS

Quadro 1 – Requisitos da interação 16
Quadro 2 – Listas de funções do trabalho de Santos et al. (2015). 37
Quadro 3 – Comandos feito pela aplicação 42

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Objetivos 11
1.1.1 Objetivo geral 11
1.1.2 Objetivos específicos 11
1.2 Estrutura do trabalho 11

2 FUNDAMENTAÇÃO TEÓRICA 12
2.1 Visão Computacional 12
2.2 Interação Humano Computador 13
2.2.1 Características humanas 13
2.2.2 O computador 14
2.2.3 A interação 15
2.2.4 Interação natural 16
2.3 Inteligência artificial 17
2.4 Redes neurais artificiais 18
2.4.1 Rede neural feed-forward 19
2.4.2 Rede neural convolucional 20
2.4.3 Redes neurais de recorrência 20
2.5 Aprendizado de máquina 20
2.5.1 Supervisionado 21
2.5.2 Não supervisionado 21
2.5.3 Semi-supervisionado 22
2.6 Reconhecimento de objetos 22
2.6.1 Pré-processamento de imagens 23
2.6.1.1 Histograma de gradientes orientados 24
2.6.2 Viola-Jones 25
2.6.3 Máquina de vetores de suporte 26
2.6.4 Transformação de recurso invariável de escala 26
2.6.5 Modelos de disparo único 28
2.6.6 You only look once (YOLO) 28
2.6.7 Redes neurais convolucionais baseadas em região 30

3 FERRAMENTAS PARA IMPLEMENTAÇÃO 31
3.1 Mediapipe 31
3.2 OpenCV 32
3.3 Pynput 32

3.4 Dlib 33
3.5 PyQt5 34

4 TRABALHOS CORRELATOS 35
4.1 Enable Viacam (MAURI; SOLANAS; GRANOLLERS, 2009) 35
4.2 Mouse ocular para pessoas com movimentos limitados causados por problemas cervicais e/ou cerebrais (SANTOS et al., 2015) 36
4.3 Sistema Interativo baseado em gestos para utilização de comandos no computador (ALMEIDA, 2013) 37
4.4 Uma Aplicação de Visão Computacional que Utiliza Gestos da Mão para Interagir com o Computador (TRUYENQUE, 2005) 39

5 DESENVOLVIMENTO 42
5.1 Identificação e rastreamento da mão 43
5.2 Identificação e rastreamento dos olhos 44
5.3 Controle dos comandos no computador 45
5.4 Menu 46
5.5 Classe principal 47

6 RESULTADOS 48
6.1 Análise do funcionamento 53

7 CONCLUSÃO 57

REFERÊNCIAS 59

1 INTRODUÇÃO

Com a melhoria das tecnologias de hardware, a interação humano-computador se tornou foco de vários estudos que visam desenvolver interfaces naturais e intuitivas. Soma-se a isso a evolução das tecnologias de redes neurais, aprendizado de máquina e inteligência artificial, que permitiram melhor processamento das informações que o usuário tenta passar para o computador. Em visão computacional por exemplo, existem ferramentas de código aberto que oferecem tecnologias de detecção de objetos, segmentação de pessoas, reconhecimento facial, detecção de pose humana, dentre outras (DIX et al., 2004).

Atualmente as interfaces predominantes de acesso ao Notebook/computador são o teclado e o mouse, porém, as atitudes e comportamentos humanos produzidos vão muito além do que essas ferramentas podem capitar, como gestos, fala e expressões faciais. Em relação ao uso do mouse, a utilização contínua pode ocasionar problemas ao usuário. Segundo Morshed et al. (2020) o uso repetitivo e vigoroso com o punho desviado e o polegar estendido pode levar à tenossinovite, que pode se tratar de um problema de ergonomia devido ao uso excessivo do mouse. Outra questão em relação ao uso do mouse está relacionada com o fato de que pessoas com problemas de coordenação motora podem ter acesso limitado pelas interfaces tradicionais.

Sendo assim, é importante desenvolver aplicações que possibilitam captar as demais expressões humanas e traduzi-las em comandos para o computador a fim de permitir melhor interação. Como alternativa para evitar esses problemas, têm-se a construção de algoritmos que usam técnicas de visão computacional para identificar a palma da mão e os olhos do usuário, nessa detecção pode-se usar as coordenadas da mão para movimentar o cursor do mouse e o status de cada olho (fechado ou aberto) para realizar os cliques.

Considerando isso, este trabalho relata uma pesquisa que desenvolveu uma aplicação de visão computacional que substitui o uso do mouse na interação humano-computador. O controle do ponteiro será feito pelos movimentos da mão em frente à câmera e pela direção do olhar do usuário. Os cliques serão feitos pelo fechamento do olho do usuário: clique do botão esquerdo/direito será feito pelo fechamento do olho esquerdo/direito.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Avaliar recursos de visão computacional como alternativa ao uso do dispositivo mouse.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Identificar formas de interação com o uso da visão computacional;
- Encontrar recursos computacionais para auxílio da implementação;
- Implementar formas de interação;
- Avaliar opções implementadas.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em cinco capítulos: introdução, fundamentação teórica, ferramentas para implementação, trabalhos correlatos, desenvolvimento, resultados e conclusão. Na introdução são discutidas a problemática e objetivos desde trabalho. Na fundamentação teórica são discutidos os conceitos e fundamentos teóricos necessários para o entendimento do projeto. No tópico de ferramentas para implementação são apresentadas as ferramentas que serão usadas na implementação. Nos trabalhos correlatos são analisados trabalhos similares ao tema de visão computacional e interação humano computador. No desenvolvimento são apresentados os métodos utilizados na implementação do projeto. No tópico de resultados são avaliadas as funcionalidades desenvolvidas. E no tópico de conclusão são apresentados os principais pontos depreendidos desse trabalho e perspectivas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este trabalho visa desenvolver uma interface de acesso ao computador que utilize o posicionamento do braço para movimentar o mouse e efetue cliques com o fechamento dos olhos. Para isso, são discutidos inicialmente os principais conceitos da visão computacional e da interação humano-computador. Em seguida são apresentadas as principais técnicas utilizadas por ferramentas que auxiliaram na detecção do braço e do olho, manipulação dos comandos do computador e construção de um menu para o usuário. Por fim, são discutidos trabalhos que desenvolveram interfaces para manipulação dos comandos para computador.

2.1 VISÃO COMPUTACIONAL

Visão computacional é uma área de pesquisa que ajuda os computadores a enxergarem, ou seja, inferir algo a partir de dados de imagens, é considerada uma disciplina multidisciplinar, com ramos da inteligência artificial e aprendizado de máquina, além de usar técnicas e métodos de várias disciplinas da engenharia e ciência da computação (KARN, 2021). Atualmente vem recebendo mais estudos devido a melhoras no desempenho de computadores em geral, que oferecem mais processamento e memória. Dentre as aplicações da visão computacional pode-se citar: esportes, veículos autônomos, entretenimento, medicina, reconhecimento facial, detecção de objetos, robótica, dentre outras. Segundo Rasche (2019) a visão computacional engloba quatro campos: processamento de imagens, visão de máquina, reconhecimento de padrões e computação gráfica.

- Processamento de imagens: está relacionado com a manipulação da imagem a fim de enfatizar certos aspectos da imagem, como bordas e contraste;
- Visão de máquina: tem como objetivo aplicar métodos para identificar os objetos, como fundo uniforme (fundo preto), representação estrutural limitada (bordas) e localização 3D exata;
- Reconhecimento de padrões (ou aprendizado de máquina): se preocupa em métodos para classificar/categorizar os objetos nas imagens, como redes neurais profundas e redes neurais convolucionais;
- Computação gráfica: visa representar os objetos de forma compacta e eficiente.

O reconhecimento de padrões ainda pode ser dividido em classificação, identificação e detecção (RASCHE, 2019). A classificação é a categorização do tipo de objeto que se encontra na imagem e, portanto, lida com as variações internas de uma classe, como a variabilidade de posições e configurações. A identificação é o processo

de reconhecer as partes do objeto em individual. E a detecção é a localização de uma característica específica do objeto na imagem, assim como suas coordenadas.

No processo de desenvolvimento de um algoritmo de aprendizado de máquina tem-se a etapa de treinamento com imagens já inspecionadas e com resposta já calculada. Esse processo pode ter diferentes níveis de dificuldade, dependendo da complexidade do objeto e do tipo de fundo, pois as imagens podem apresentar diferentes graus de ruídos. Quanto mais variedades de fundo e variações do objeto houver, mais imagens de treinamento serão necessárias para evitar erros por ruídos ou detecção.

2.2 INTERAÇÃO HUMANO COMPUTADOR

A interação humano-computador (IHC) pode ser definida como uma disciplina preocupada com a concepção, avaliação e implementação de sistemas de computação ativos para uso humano e o estudo dos principais fenômenos que os cercam (HEWETT et al., 1992), a essa definição se estende as interações entre um ou mais humanos, computadores e ambientes a sua volta. Já os estudos de IHC em sistemas da informação estão preocupados com as formas como os humanos interagem com informações, tecnologias e tarefas, especialmente em negócios, gestão, contextos organizacionais e culturais (ZHANG et al., 2002).

Para Dix et al. (2004), o design de um projeto de IHC tem que levar em conta as características humanas e das máquinas. Os humanos, por exemplo, podem receber informação por meio dos sentidos auditivos, táteis, visual e movimentos; apesar de serem limitados em sua capacidade de processar informação e poderem sofrer influência de emoções. Após receber as informações as pessoas passam a raciocinar, ganhar habilidade ou solucionar problemas.

Os computadores por sua vez, dispõem de vários elementos que interagem com o usuário, dentre os quais estão os dispositivos de entrada do texto, como teclado, mouse, câmera, microfone, scanners e sensores; e de saída de dados, como monitores e som. Os computadores ainda apresentam o hardware, que interfere na interação com o usuário, como memória e processamento.

2.2.1 Características humanas

Os humanos apresentam características determinantes para a concepção e desenvolvimento das interfaces de comunicação, analisá-las é fundamental para o desenvolvimento de interfaces intuitivas e ergonômicas. Segundo Hewett et al. (1992), os principais aspectos humanos utilizados na criação dessas interfaces, são: processamento de informação, linguagem e ergonomia:

- A capacidade humana de processamento de informação pode ser entendida como fenômenos da memória, percepção, atenção, aprendizagem, habilidades e raciocínio;
- A linguagem considera os aspectos de sintaxe, semântica e normas, funcionando como um meio de comunicação e interface do usuário com a máquina;
- A ergonomia compreende as características antropométricas e fisiológicas humanas e o modo como interagem com o espaço de trabalho, sendo assim, um design com boa ergonomia exige segurança, otimização e evitar problemas de fadiga e saúde. Também é preciso levar em consideração as diversidades humanas e populações com deficiência.

Para Dix et al. (2004), em associação com os sistemas computacionais, pode-se dizer que os humanos contam com quatro características principais: canais de entrada e saída, memória, pensamento e emoções:

- Os canais de entrada seriam os sentidos da visão, audição e tato, enquanto os canais de saída seriam os sentidos que podem controlar a interface, podendo ser a voz, o corpo e a visão;
- A memória pode ser dividida em três tipos: sensorial, curto prazo e longo prazo. A memória sensorial compreende o tratamento dado aos estímulos, como visão, audição e tato; a memória de curto prazo representa uma memória rápida e limitada, já a de longo prazo tem maior capacidade, maior tempo de acesso e dura mais;
- O pensamento pode ser dividido em raciocínio e resolução de problemas, o raciocínio é o uso do conhecimento para chegar a conclusões ou deduzir algo, sendo que esse processo pode ser dedutivo, indutivo e abdutivo. A resolução de problemas é o processo de encontrar uma solução adaptando o conhecimento que si tem;
- As emoções, por sua vez, envolveriam estímulos externos que acabariam por impactar na forma como ocorre a interação com o computador.

2.2.2 O computador

Os computadores são uma composição de diferentes hardwares e softwares que juntos se tornaram elementos essenciais e indispensáveis na vida cotidiana atual, isso se deve à grande aplicabilidade e facilidade de uso e acesso, portanto, para o desenvolvimento de interfaces, é importante analisá-los. Segundo Dix et al. (2004) e Hewett et al. (1992), os computadores portam as seguintes características e componentes:

- Dispositivos de entrada: são os teclados alfanuméricos, rastreamento ocular, os algoritmos de reconhecimento de escrita e voz. As técnicas de entrada são

baseadas no: teclado, mouse, canetas e voz;
- Dispositivos de posicionamento: mouse, touchpad (sensor tátil), touch screen, mesa digitalizadora, eye gaze (controla o computador por meio de óculos especiais que rastreiam o olho), dentre outros;
- Telas e monitores: podem ter tecnologias bitmap, CRT, LCDs, projetores, dentre outros;
- Interação 3D: capacete de realidade virtual;
- Sensores e equipamentos especiais: saídas de som, sensores de temperatura, umidade, movimento, giroscópio, dentre outros;
- Impressoras: possuem scanner para digitalização e imprimem dados;
- Memória: cache (atual em conjunto com o processador), RAM (curto prazo e veloz) e disco (magnéticos ou ópticos, mas são lentos);
- Processamento: unidade de processamento central (CPU), placa de vídeo (GPU), microprocessadores e chips especializados;
- Redes (internet): permitem conexões globais com outros computadores e acesso a servidores e banco de dados.

Dix et al. (2004) ressaltam que os computadores ainda apresentam alguns fatores limitantes de velocidade no sistema interativo, além do limite de computação, armazenamento, gráficos e rede. Para que o usuário tenha uma experiência adequada é necessário indicar o progresso das tarefas, o consumo atual dos serviços e otimizar as aplicações.

2.2.3 A interação

Segundo Zhang et al. (2002), a interação, no contexto das ciências computacionais, é a preocupação com as maneiras como os humanos interagem com informações, tecnologias e tarefas, dentro dos contextos empresariais, organizacionais e culturais, considerando os aspectos comportamentais, cognitivos, motivacionais e afetivos. Para Baekgaard (2006), a interação tem duas funções em sistemas da informação: relacionar os elementos de um sistema de informação e trocar informações internamente ou entre o sistema e o ambiente, baseado nesse conceito, Baekgaard (2006) cita cinco padrões de interação:

- Sentir: o usuário percebe características de um objeto;
- Modificar: um usuário modifica um objeto;
- Trocar: dois ou mais usuários trocam objetos;
- Mover: move um objeto de uma localização para outra;
- Controlar: um usuário controla(requere) a ação de outro usuário.

Dix et al. (2004) afirmam que a interação é uma forma do usuário se comunicar com o sistema, cujo objetivo é ajudar o mesmo a alcançar resultados de uma tarefa

de algum aplicativo. Essa interação ainda pode ser representada em um ciclo, onde o usuário determina uma tarefa para o computador executar e analisa as respostas para determinar ações futuras. Dix et al. (2004), então, descrevem os seguintes tópicos do ciclo:

1. Estabelecer uma nova meta;
2. Especificar o objetivo;
3. Especificar a sequência de ações;
4. Executar as ações;
5. Perceber o estado do sistema;
6. Interpretar o estado do sistema;
7. Avaliar o estado do sistema em relação aos objetivos.

Para Dix et al. (2004), a interação deve atingir engajamento do usuário e permitir uma interação natural, não exaustiva, intuitiva e otimizada (Quadro 1). Tendo em vista seus objetivos e requisitos, a interface de interação atual predominante pode ser analisada como sendo do tipo janelas, ícones, menus e ponteiros, sendo que a navegação é feita pelo modo apontar e clicar. À extensão desses comandos se somam os menus, interface de linha de comando, diálogos de consulta (alertas e informações) e formulários.

Quadro 1 – Requisitos da interação

Requisito	Funcionalidades
Funcional	os controles são agrupados de acordo com suas funcionalidades
Sequencial	os controles são ordenados de acordo com seu uso mais comum
Frequencial	os controles mais usados podem ser acessados mais facilmente
Adequa a saúde do usuário	posição física, temperatura, iluminação, barulho e tempo

Fonte: Dix et al. (2004, p. 17).

2.2.4 Interação natural

A interação natural (IN), no campo da informática, é o processo mais intuitivo e simplificado de interação humana e se faz por meio de uma interface de usuário, sendo, atualmente, predominantemente feita com o uso do mouse e teclado, mas Garbin (2010) explica que esses métodos podem perder espaços para técnicas de reconhecimento de fala, escrita natural e reconhecimento gestual, pois, esses podem identificar atitudes mais naturais do usuário e usa-las como entrada. Dentre os requisitos de funcionalidade esperados por uma IN estão (GARBIN, 2010):

- Usabilidade: fácil operabilidade, que engloba fácil compreensão e aprendizado das funcionalidades;

- Fatores humanos: exige que o dispositivo ofereça o máximo de segurança e conforto;
- Acessibilidade: exigência de fácil acesso e sem barreiras;
- Tolerância aos erros: a interface permanece estável após erros;
- Facilidade de individualização.

2.3 INTELIGÊNCIA ARTIFICIAL

A inteligência artificial (IA) é a capacidade de um computador digital ou robô controlado por computador para executar tarefas comumente associadas a seres inteligentes e para encontrar soluções para problemas de difícil modelagem matemática, com entradas variadas e que exijam adaptabilidade (MCCUDDEN, 2010). Para Stahl (2020), os principais conceitos relacionados ao campo de IA são aprendizado de máquina, tentativa de replicar as capacidades humanas e sistemas sociotécnicos convergentes.

Máquinas de aprendizagem é a capacidade de adaptação e aprendizado com as experiências que uma IA mantém, isso é alcançado com a implementação de algoritmos de aprendizado, como: supervisionado, não supervisionado, semi-supervisionado, reforço, autoaprendizagem, dentre outras. A tentativa de replicar as capacidades humanas é a ideia de que conhecendo o funcionamento do cérebro humano é possível simular o seu comportamento em um computador (PATEL; JAISWAL, 2021), como os neurônios. Os sistemas sociotécnicos convergentes são aplicações em que a IA não opera de forma autônoma, mas fornece análise de dados e ferramentas para serem incorporadas em outras tecnologias.

Entre os principais componentes de seu funcionamento está o perceptron, um dispositivo que recebe um vetor de entradas, os processa de acordo com pesos e fornece uma saída, o que se assemelha a um neurônio humano. Esse componente forma redes mais complexas que passam a ser denominados agentes inteligentes ou agentes artificiais. As redes mais complexas são um conjunto maior de perceptrons divididos em camadas, sendo que a primeira possui um perceptron por entrada e a última uma para cada saída possível.

Alterando essa arquitetura e aplicando um mecanismo de aprendizado, é possível obter uma matriz de pesos dos neurônios que permite ser aplicada nas tarefas. De acordo com a modelagem feita podem ser produzidos resultados de classificação de dados, reconhecimento de padrões, percepção ou tomada de decisão. Devido a essa flexibilidade proporcionada por suas redes, a inteligência artificial veem encontrando muitas aplicações em áreas como saúde e medicina, esportes, entretenimento, veículos autônomos, design, dentre outros (KARN, 2021).

2.4 REDES NEURAIS ARTIFICIAIS

Redes neurais artificiais são um conjunto de mecanismos que tentam imitar o funcionamento do cérebro animal, a fim de fornecer soluções computacionais flexíveis, rápidas e adaptativas. Essas redes são caracterizadas pela falta de transparência ou habilidade de explicar a decisão, devido a complexidade do mecanismo de detectar padrões por meio de pesos (GABRYS; LEIVISKA; STRACKELJAN, 2005).

Para Braspenning, Thuijsman e Weijters (1995), as redes neurais são dispositivos associativos flexíveis focados em encontrar padrões e fazer reconhecimentos, que são modelados com mais ênfase no treinamento ou aprendizagem (regra *Delta*) do que nas regras de programação. Para Gabrys, Leiviska e Strackeljan (2005) uma rede neural artificial pode ser classificado como um sistema adaptativo inteligente, capaz de se adaptar a um ambiente semelhante ou desconhecido a deriva no tempo e espaço aplicando sua rede para reconhecer as mudanças e reagir de acordo. Dentre os componentes de uma rede neural pode-se citar (AL-JABERI, 2018):

- Unidade de processamento: a menor unidade de processamento é o perceptron, que podem ser colocados em conjuntos, formando camadas, sendo dividas entre camadas de entrada, saída e ocultas. As camadas de entrada possuem um perceptron para cada informação, as de saída um perceptron para cada tipo de saída e as ocultas são camadas intermediárias de processamento;
- Fatores de ponderação: é o peso dado a cada entrada no perceptron para indicar a relevância daquela informação para a saída final;
- Função de soma: presente em cada perceptron, essa função realiza a combinação das entradas no perceptron em uma saída;
- Função de ativação: visa garantir uma resposta controlada e limitada do perceptron.

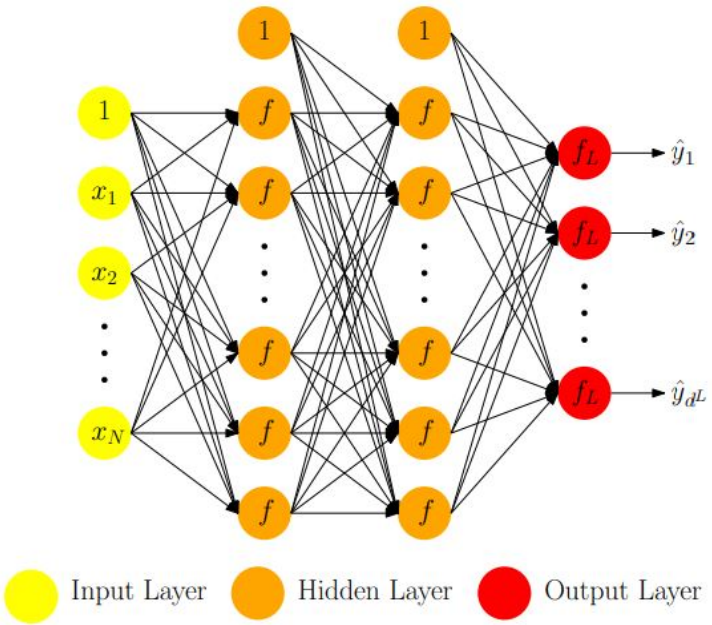
A maioria das arquiteturas de redes neurais faz uso de várias camadas de neurônios artificiais, conectados aos próximos neurônios, os quais detêm um peso que multiplicam as entradas e fornecem uma saída, que pode ser a entrada de uma nova camada ou a saída final (XU et al., 2020). Essas camadas podem ser de três tipos: entrada, ocultas ou de saída (JANIESCH; ZSCHECH; HEINRICH, 2020).

As camadas de entrada capturam todos os dados de entrada, mas o fazem de forma individual possuindo um perceptron para cada entrada, e os repassam às camadas ocultas. Essas, contam com perceptrons que aplicam uma função de ativação e repassam a saída, podendo repassar a todos os perceptrons (camadas totalmente conectadas, onde cada perceptron de uma camada está conectado a todos os outros da camada seguinte), ou a um grupo (camadas agrupas).

As camadas ocultas, no geral, reduzem os dados a serem processados

diminuindo o número de perceptrons a cada camada, resultando em uma maior eficiência. As camadas de saída estão atreladas aos possíveis rótulos da rede, sendo que cada rótulo de saída possui um perceptron (Figura 1).

Figura 1 – Rede neural



Fonte: Ferreira, Medonça, Diniz (2018).

Devido as várias aplicações que as redes neurais encaixam, foram desenvolvidas muitas arquiteturas diferentes a fim de se atender a todos objetivos. A arquitetura de uma rede neural é o arranjo dos neurônios em camadas e os padrões de conexão dentro e entra as camadas (AL-JABERI, 2018). Essas arquiteturas podem ser classificadas em redes recorrentes e lineares. Redes recorrentes ou de retro propagação processam o sinal e os envia novamente para a entrada, como um loop, e redes lineares enviam o sinal para uma camada seguinte, nunca de volta a entrada.

Dentre os vários tipos de redes neurais pode-se citar: neural convolucional, aprendizado profundo, multicamadas de perceptron, camada única, função de base radial, recorrente, autoencoder, dentre outras. Abaixo serão analisados alguns tipos de redes neurais.

2.4.1 Rede neural feed-forward

Uma rede neural feed-forward é uma rede neural em camadas cujos dados passam em sequência pelas camadas, sem voltar em ciclos. Assim, os dados passam por um perceptron, onde entram em uma função de ativação, e seguem para os próximos perceptrons de forma linear. Essa arquitetura é muito comum, pois oferece um fácil treinamento, onde as saídas são comparadas com o valor esperado, resultando em um erro que permite regular os pesos. A rede de camada única, multicamadas de perceptrons, convolucional e função de base radial são exemplos de redes neurais

feed-forward.

2.4.2 Rede neural convolucional

As redes convolucionais apresentam características similares as demais redes neurais, tendo várias camadas intermediárias, porém empregam a convolução como função de ativação. Dentre os seus principais componentes estão as camadas convolucionais, camadas de pooling e camadas totalmente conectadas (ZARGAR, 2021).

As camadas convolucionais são camadas iniciais de entrada que fazem a convolução entre os dados de entrada e os pesos dos perceptrons, resultando em uma matriz, cujos valores indicam a ativação da região. Assim, as camadas funcionam como um filtro, ativando quando há uma região de interesse.

As camadas de pooling são camadas que reduzem os dados ao capturar o valor máximo de uma região e usá-lo da próxima camada, reduzindo progressivamente os dados sem perda de informação, gerando economia computacional. Já as camadas totalmente conectadas efetuam a classificação final como em uma rede neural comum por meio de perceptrons completamente interligados.

2.4.3 Redes neurais de recorrência

Redes neurais recorrentes ou de retro propagação são redes neurais em que as saídas de um perceptron é usada na entrada de um perceptron anterior. Isso implica em uma memória de curto prazo e uma capacidade de prever sequências, pois os dados anteriores podem ser armazenados para a entrada dos dados seguintes.

Em algumas arquiteturas essa rede pode ser ajustada para armazenar padrões por longo prazo, como uma memória de longo prazo para características que não se repetem a muito tempo ou a muitos ciclos. As redes de recorrência são mais utilizadas para processamento de fala e reconhecimento de caligrafia.

2.5 APRENDIZADO DE MÁQUINA

Segundo Karn (2021), o aprendizado de máquina é um subconjunto da inteligência artificial, que faz uso de algoritmos e dados de treinamento para ensinar uma rede a detectar padrões automaticamente ou com pouca intervenção ou a fazer previsões sobre o futuro dos dados. Na sua implementação em redes neurais, os pesos de ativação são ajustados para atender aos dados de treinamento e resposta esperada.

As técnicas de aprendizado de máquina visam encontrar um método de regular rapidamente esses pesos, isso é feito por meio da comparação da resposta esperada com a calculada pelo algoritmo, ou seja, o erro relativo da resposta. Com esse erro

são mapeados os perceptrons que resultam nesse erro e seus pesos são alterados de acordo com uma função de ajuste de treinamento.

A regulação desses pesos deve ser tal que a rede não perca ajustes de treinamentos anteriores, consiga lidar com dados pouco comuns e seja capaz de resolver dados que não apareceram no treinamento. Dentre as técnicas de treinamento mais usadas pode-se citar os tipos: supervisionado, não supervisionado, semi-supervisionado e reforço.

2.5.1 Supervisionado

O treinamento supervisionado está focado em encontrar uma função que descreva como diferentes variáveis medidas podem ser combinadas para fazer uma previsão sobre uma variável de resposta (BAUMER; KAPLAN; HORTON, 2017). Esse método possui dados de entrada e saída prevista, que servem de referência e exemplo para a rede. Com esses dados o algoritmo regula os pesos da rede, de modo que após o treinamento a mesma se torna capaz de prever a saída para dados que estão e não estão na base de dados do treinamento, de modo que a rede passe a generalizar corretamente os dados.

Para Baumer et al., (2017) um treinamento supervisionado a análise pode ter diferentes motivações, como: prever uma saída dada uma entrada, identificar quais variáveis de entrada são úteis, gerar hipóteses (rastrear combinações de entradas antes desconhecidas) ou entender como um sistema funciona. Assim, a construção de um treinamento supervisionado deve levar em conta esses objetivos, adequando os dados e a função de aprendizado a essas metas.

Além disso, para alcançar um bom resultado de treinamento usando esse método, são necessários muitos exemplos de dados para treinamento, pois a rede pode apenas se adequar aos dados de treinamento, sem oferecer generalização para dados um pouco diferentes. Dentre os exemplos de aplicação do aprendizado de máquina supervisionado estão regressão, árvore de decisão, máquinas de suporte de vetores (sigla *SVM* em inglês) e classificador Naive Bayes.

2.5.2 Não supervisionado

No treinamento não supervisionado não há estimativa de saída, então o algoritmo tenta encontrar padrões, anomalias ou grupos nos dados. Para James et al. (2021) a busca não supervisionada tenta descobrir subgrupos homogêneos, fazendo uma análise exploratória dos dados, pois não se pode validar, verificar ou saber a verdadeira resposta.

Essa categoria de aprendizado é caracterizada pela existência apenas de entradas, sem uma saída relacionada, assim, o treinamento não supervisionado foca em

fazer uma análise exploratória tentando encontrar padrões nos dados. Essa investigação resulta em um modelo preditivo capaz de completar valores vazios, localizar a direção das maiores variâncias ou agrupar subgrupos com características homogêneas. Na localização das maiores variâncias o resultado é um vetor com os valores que mais variam em cada dimensão da entrada.

Essa análise de dados encontra aplicações em pesquisas genéticas, sites de compras e mecanismos de busca (JAMES et al., 2021). Nos estudos genéticos sobre o câncer não se pode supervisionar todos os dados do genoma devido ao seu tamanho, então tem-se que usar métodos que conseguem rastrear grupos de genes que podem estar relacionados a essa doença.

Em um site de compras existem inúmeros perfis de consumidor que resultam em diferentes tipos de compras, assim, pode se aplicar aprendizado não supervisionado para descobrir grupos de perfis de consumidor. Em um mecanismo de busca existem inúmeras páginas para se avaliar, mas nem todas serão de interesse do usuário.

2.5.3 Semi-supervisionado

O aprendizado semi-supervisionado faz uso tanto da técnica de treinamento supervisionado quanto do não supervisionado, sendo que há mais dados do último. Essa técnica permite uma economia de custos, pois os dados supervisionados são caros de serem produzidos; assim com o uso dos exemplos supervisionados a rede tenta inferir o resultado dos demais dados. Enquanto isso, os dados sem rótulos devem ter uma diferença suave dos exemplos e devem formar grupos com características semelhantes.

Segundo Triguero, García e Herrera (2015) o aprendizado semi-supervisionado pode ser dividido entre agrupamento e classificação semi-supervisionado. O agrupamento visa obter grupos mais bem definidos do que os modelos não supervisionados. A classificação semi-supervisionada visa categorizar os dados e é dividida em transdutiva e indutiva. A classificação transdutiva é o treinamento em que os dados a serem previstos estão no treinamento, enquanto a classificação indutiva os dados a serem previstos não estão no treinamento.

2.6 RECONHECIMENTO DE OBJETOS

O reconhecimento de objetos é a tarefa de identificar e segmentar a partir de uma imagem o seu conteúdo e sua localização, assim como sua classificação de objetos corretamente. Nesse processo deve haver a captura e o reconhecimento do conteúdo da imagem, sendo que segundo Szeliski (2011), para os computadores, a tarefa mais desafiadora é o reconhecimento de classe ou categoria ao qual a imagem carrega.

A partir dessa problemática várias soluções de reconhecimento de objetos foram surgindo ao longo dos anos, fazendo com que o melhor mecanismo para uma tarefa seja diferente do melhor mecanismo para outra tarefa. Essas realidades vêm do fato de que é mais eficaz construir detectores de propósito específico que exigem rapidez, precisão e baixo processamento para efetuar essas tarefas (SZELISKI, 2011).

Dentre as técnicas de detecção de objetos podem-se citar: Viola-Jones (VIOLA; JONES, 2001), máquina de vetores de suporte Cortes e Vapnik (1995), transformação de recursos invariável de escala (LOWE, 1999), modelo de disparo único (FEI-FEI; FERGUS; PERONA, 2006), you only look once (REDMON et al., 2016) e redes neurais convolucionais baseadas em regiões (GIRSHICK et al., 2014).

2.6.1 Pré-processamento de imagens

No reconhecimento de objetos é muito comum o pré-processamento de imagens para otimizar o funcionamento dos processos de detecção, que demandam mais processamento e são mais complexos de serem desenvolvidos. Segundo Burger e Burge (2008), o processamento digital de imagens é a concepção, projeto, desenvolvimento e aprimoramento de programas de imagens digitais a fim de facilitar a extração de informações significantes sobre seu conteúdo.

O processamento de imagens no reconhecimento de objetos deve ser capaz de lidar com ruídos, valores ausentes e/ou escalas variadas e fornecer um padrão normalizado de imagem a ser processada pelo algoritmo. Dentre as etapas desse processamento pode-se citar (BURGER; BURGE, 2008):

- Aquisição de imagens: feito por meio de lentes que capturam uma imagem invertida em relação ao eixo óptico;
- Amostragem espacial: realizado por sensores em uma placa que capturam a intensidade da luz;
- Amostragem temporal: feito em intervalos iguais de tempo por um mesmo sensor;
- Quantização de valores: cada valor do sensor passa por uma conversão analógica-digital que será armazenado em um pixel;
- Resolução: padronização do tamanho da imagem;
- Conversão do espectro de cores: as imagens são convertidas para tons de cinza para simplificar a ação dos algoritmos.

É partir de imagens padronizadas que os algoritmos de reconhecimento passam a atuar. Isso é feito para simplificar a implementação dos algoritmos de reconhecimentos, pois os dados a serem tratados são menores (além de carregarem corretamente as informações do conteúdo), o que permite um processamento menor e mais veloz. Dentre as técnicas de processamento de imagens podem-se citar (PETERS, 2017):

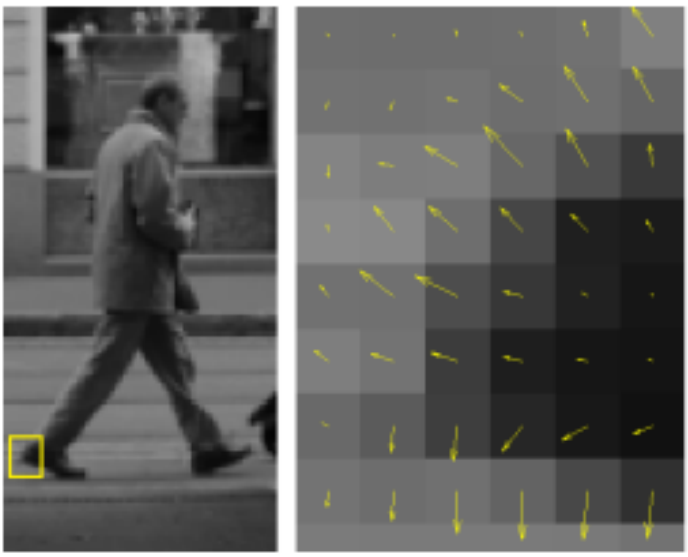
- Coloração falsa: colocar cores visíveis de espectros não visíveis;
- Seleção de pixels: centróide, pixels de canto ou pixels de bordas;
- Manipulação de pixels: converter o espectro de cores;
- Processamento de pixels: alterar o brilho;
- Filtragem: redução de ruído ou alteração de sombras;
- Segmentação: detecção de bordas ou diferenciar objetos sobrepostos.

2.6.1.1 Histograma de gradientes orientados

O histograma de gradientes orientados é um método de processamento que permite representar as características de uma imagem de forma simples e compacta, por meio de vetores indicando a orientação em que os pixels mais aumentam de intensidade. Esse método permite retratar as características de forma invariante a rotações, translações e iluminação.

O método, estudado por Dalal e Triggs (2005), consiste em calcular em cada pixel sua orientação por meio da comparação de seu gradiente com o de seus vizinhos. Assim, passa-se a ter um vetor em cada pixel indicando a orientação em que o gradiente aumenta e proporcional a esse aumento (Figura 2). Essa técnica permite capturar bordas ou gradientes que representam as principais características locais, sendo ainda invariante, o que é imprescindível a maioria dos algoritmos de detecção de objetos.

Figura 2 – Resultado do histograma de gradientes orientados



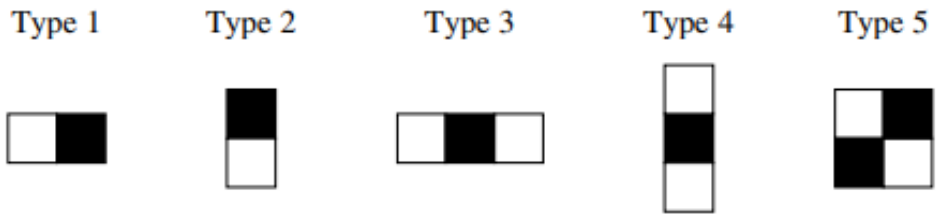
Fonte: Bauer, Brunsmann e Schlotterbeck-Macht (2010).

2.6.2 Viola-Jones

O detector de Viola-Jones (VIOLA; JONES, 2001) foi desenvolvido inicialmente para detecção de rostos, entretanto as suas técnicas podem ser aplicadas para detectar outros objetos. Dentre os procedimentos para desenvolver esse detector estão: transformar a imagem de entrada em uma imagem integral, seleção do algoritmo AdaBoost e elaboração do classificador em cascata (JENSEN, 2008).

A imagem integral é feita tornando cada pixel igual à soma total de todos os pixels acima e à esquerda do pixel em questão. O resultado é uma matriz que possibilita calcular a soma dos pixels de um retângulo com quatro operações. A partir dessa matriz é possível verificar de forma eficaz se há uma semelhança com os recursos Haar da Figura 3, que são escolhidos de acordo com sua semelhança com o rosto humano ou objeto desejado por meio do AdaBoost, além de serem escolhidos retângulos devido a sua eficiência computacional de cálculo da soma dos pixels. Cada recurso resulta em um valor que é obtido subtraindo o valor da soma do retângulo branco da soma do preto.

Figura 3 – Recursos Haar usado no detector de rosto de Viola-Jones



Fonte: Jensen (2008).

A partir da imagem integral são escolhidos vários retângulos para se comparar com os valores dos recursos, caso haja semelhança em um local, pode-se ter um rosto ou objeto. Para escolher os melhores recursos Haar, Viola e Jones usaram o algoritmo AdaBoost para analisar todos os possíveis recursos e selecionar os melhores, que mais contribuem para detecção. O AdaBoost é um algoritmo de aprimoramento de aprendizado de máquina capaz de construir um classificador forte por meio de uma combinação ponderada de classificadores fracos (JENSEN, 2008).

Com os melhores recursos selecionados, foi construído um classificador em cascata com os mesmos, cujo primeiros classificadores são fracos e pequenos para gerar eficiência, cujo funcionamento rejeita a maioria dos negativos e seleciona quase todos os positivos (VIOLA; JONES, 2001). No próximo estágio novos recursos são usados na classificação, sendo que cada estágio é treinado usando o algoritmo AdaBoost. O resultado é um classificador em cascata com vários estágios, onde cada um usa um ou mais recurso Haar.

Dentre as vantagens da abordagem de Viola-Jones estão (YAN, 2019):

computação rápida de recursos, eficiente na seleção de recursos, invariante de escala e localização, em vez de dimensionar a imagem, dimensiona-se os recursos e possibilita a implementação para outros objetos. Como desvantagens tem-se (YAN, 2019): é mais eficaz em imagens frontais, dificilmente suporta rotação em torno dos eixos verticais e horizontais e é sensível às condições de iluminação.

2.6.3 Máquina de vetores de suporte

Máquina de vetor de suporte (do inglês support vector machine, sigla SVM) é uma extensão do classificador de margem máxima, um algoritmo que busca otimizar a distância entre duas classes construindo uma linha ou plano, mas que pode ser aplicado em casos não lineares (JAMES et al., 2021). O algoritmo recebe dados com características em valores numéricos e identifica a qual grupo o mesmo pertence. Inicialmente esse método produzia uma classificação linear para dados sem erros e que fossem separáveis, entretanto, Cortes e Vapnik (1995) aprimoraram essa técnica para que o algoritmo consiga lidar com dados não lineares, que não são separáveis.

Para isso, os vetores de entrada passam por uma expansão de ordem em vetores de suporte, que aplicam a cada valor do vetor de entrada uma função, podendo ser uma função polinomial, gaussiana, radial ou sigmoide, que resulta em dados no hiper plano. Com os novos valores é calculada a convolução do produto escalar de cada vetor, resultando em dados que permitem uma solução não linear. A partir daí, é estabelecida as margens dos dados, que é feito por uma função que minimiza os erros formando uma margem flexível e permite alguns erros para valores que distanciam muito da maioria.

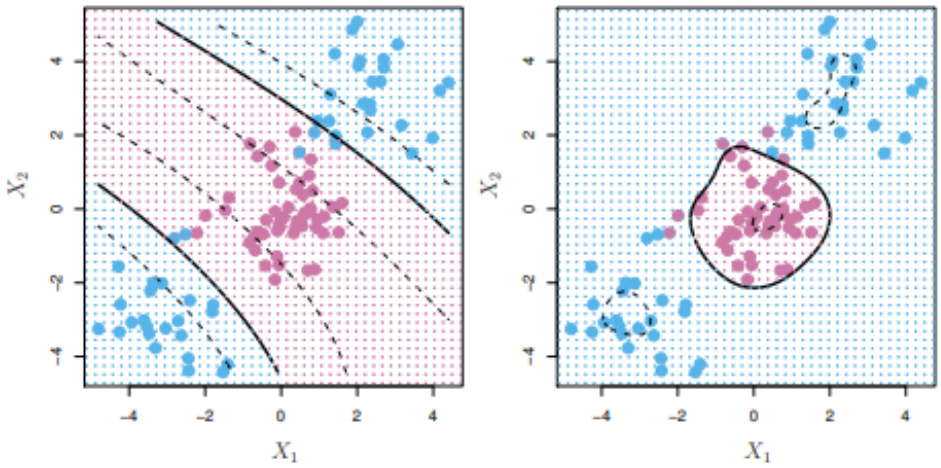
Dentre as vantagens do SVM estão a alta generalização e a capacidade de ser implementado para qualquer problema, pois é possível adaptá-lo para reconhecimento de objetos usando histograma de gradientes orientados. A abordagem de Cortes e Vapnik (1995) ainda permite controle de capacidade e alteração da superfície de decisão. A Figura 4 abaixo mostra o resultado do método polinomial e radial, respectivamente.

2.6.4 Transformação de recurso invariável de escala

A transformação de recursos invariável de escala (do inglês scale invariant feature transform, sigla SIFT), é um método que foi desenvolvido por Lowe (1999) para ser capaz de encontrar semelhanças entre imagens, por meio de quadros de características, mas que fosse robusto as variações de escala, rotação, ruído, distorções, iluminação e ponto de vista. Dentre as aplicações desse método estão reconhecimento de objetos e rastreamento em vídeo.

Primeiro deva-se encontrar pontos chaves de alta variação (chamados

Figura 4 – Máquina de vetores de suporte



Fonte: James et al. (2021).

keypoints), que possam conter regiões de características que possam ser usadas para comparação. Para isso a imagem de entrada é convolucionada com uma função gaussiana de uma dimensão (Equação 1), nas direções verticais e horizontais. Esse processo é feito com $\sigma = \sqrt{2}$ (A). A mesma operação é feita de forma mais suave com $\sigma = 2$ (B).

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{1}$$

Em seguida é reamostrada a imagem B aplicando a interpolação bilinear com espaçamento de 1.5 em cada direção de cada pixel. Com esse resultado, já é possível detectar os máximos e mínimos comparando cada pixel com seus vizinhos, resultando na eliminação de vários pixels, por exemplo, sobram no máximo 1000 pontos chaves de uma figura de 512 por 512.

Com os pontos chave são contruídas características em torno dos mesmos, para isso a imagem A é usada para extrair o gradiente e a orientação de cada pixel e um quadro em torno de cada ponto chave é arranjado contendo os dados de gradiente e orientação, que são usados para comparar as características. Para compensar o efeito da iluminação, uma tolerância de 0.1 vezes a magnitude do gradiente é usada.

A orientação dos pixels possui valores em oito direções, com magnitudes diferentes. Para evitar os efeitos da rotação, a direção é calculada em relação a direção de cada ponto chave, em cada quadro de características. Assim, o vetor descritor, que armazena os quadros de características, passa a ser invariante a rotações. Com essa abordagem, o método SIFT permite encontrar padrões, com alta estabilidade, mesmo em imagens incompletas, com ruídos, distorções, rotações, variação de iluminação,

dentre outras.

2.6.5 Modelos de disparo único

O modelo de disparo único surgiu para atender a problemas de detecção de múltiplas categorias e que o fizesse com poucas imagens de treinamento. Neste método uma nova categoria pode ser aprendida com poucas imagens (1 a 5 imagens), a partir de uma rede neural treinada em outra categoria com milhares de imagens. Essa técnica parte da ideia de que se pode aprender mais com uma imagem e que se pode reaproveitar o aprendizado de outras categorias, mesmo que as classes não apresentem características similares (FEI-FEI et al., 2006).

Uma nova categoria possui aspectos que podem ser assimilados a partir de uma rede já treinada. Esse processo é feito por meio de um algoritmo Bayesiano baseado na representação de categorias de objetos com modelos probabilísticos (FEI-FEI et al., 2006). A Equação 2 mostra essa implementação feita por Fei-Fei et al. (2006), onde *I* é a imagem de entrada, I_t as imagens de treinamento da categoria nova, O_{fg} a categoria nova e O_{bg} a categoria velha. Assim, a probabilidade de a imagem conter um objeto de classe nova (*R*) é a probabilidade do modelo novo aprendido sobre a do modelo de fundo já treinado.

$$R = \frac{p(O_{fg}|I, I_t)}{p(O_{bg}|I, I_t)} = \frac{p(I|I_t, O_{fg})p(O_{fg})}{p(I|I_t, O_{bg})p(O_{bg})} \tag{2}$$

Expandindo a Equação 2 com uma análise bayesiana chega-se na Equação 3,onde θ e θ_{bg} são os modelos paramétricos, para as categorias novas e antigas, respectivamente. A constante $\frac{p(O_{fg})}{p(O_{bg})}$ foi omitida nessa expansão, sendo que para determinar o valor de *R* basta calcular as probabilidades de a imagem conter uma classe nova ou velha e as integrais em relação aos modelos paramétricos.

$$R = \frac{\int p(I|\theta)p(\theta|I_t, O_{fg})d\theta}{\int p(I|\theta_{bg})p(\theta_{bg}|I_t, O_{bg})d\theta_{bg}} \tag{3}$$

Assim, as informações prévias de categorias aprendidas anteriormente são representadas com uma função de densidade de probabilidade adequada nos parâmetros dos modelos (FEI-FEI et al., 2006). Esse processo resulta em uma rede capaz de alcançar resultados de 70 a 95% a partir de poucos exemplos de treinamento e podem ser usados tanto para detecção quanto para diferenciação. Esse método se torna muito útil em tarefas que não dispõem de muitos dados catalogados, oferecendo uma alta flexibilidade e personalização.

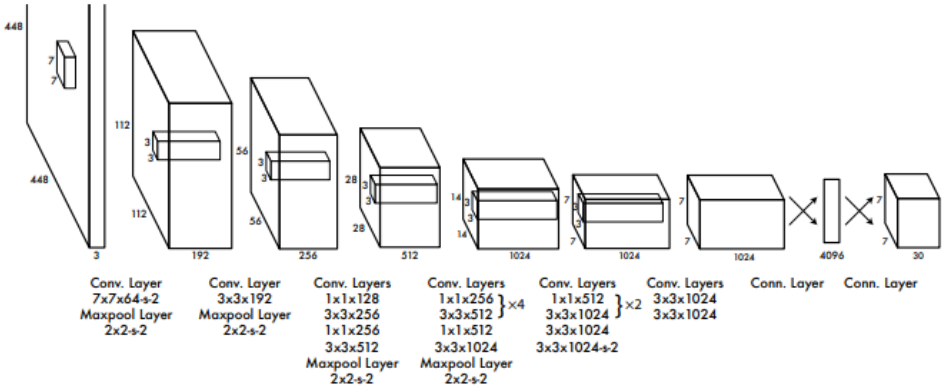
2.6.6 You only look once (YOLO)

Você só olha uma vez (em inglês, you only look once (YOLO)) é um algoritmo de regressão para detectar objetos separados espacialmente baseado em redes neurais

convolucionais. Desenvolvido inicialmente por Redmon et al. (2016), o algoritmo possui uma única rede neural convolucional que constrói caixas delimitadoras e determina a probabilidade da classe dos objetos contidos na imagem, a partir da imagem inteira em uma única estimativa.

A imagem inicial passa por uma divisão em grades, onde em cada célula é executada uma detecção, que determina se há um objeto e sua pontuação de confiança. Essa detecção é feita por uma rede neural convolucional que possui uma camada inicial capturando os dados, seguido por 24 camadas convolucionais, as quais também reduzem os dados, e por fim, duas camadas convolucionais totalmente conectadas que preveem as coordenadas e o tipo de objeto detectado (Figura 5).

Figura 5 – Rede neural da técnica YOLO implementada por Redmon et al. (2016)



Fonte: Redmon et al. (2016).

Uma versão mais rápida do algoritmo YOLO também foi desenvolvido, chamada *Fast YOLO*, possuindo 9 camadas convolucionais ao invés de 24 e alguns filtros. Assim, o YOLO consegue entregar taxas de quadros mais rápido, passando de 45 para 155 quadros por segundo. Ambas redes foram treinadas em um conjunto de dados com várias imagens usando uma função de ativação linear.

Dentre as vantagens desse algoritmo estão a simplicidade de implementar, o treinamento é feito direto com a imagem total, boa generalização para novos domínios, rápido e robusto. Dentre as desvantagens do YOLO tem-se as limitações das caixas delimitadoras que encontram dificuldades em detectar objetos pequenos, desempenho afetado por escalas e proporções incomuns, recursos limitados para caixas e função de perda é a mesma para caixas pequenas e grandes. Assim, o erro de uma caixa grande é o mesmo que de uma caixa pequena, porém mais grave, gerando a maior concentração de erros do algoritmo.

2.6.7 Redes neurais convolucionais baseadas em região

Redes neurais convolucionais baseadas em regiões, sigla R-CNN, é um método de detecção de objetos para propostas de região que atua de baixo para cima, para localizar e segmentar objetos (GIRSHICK et al., 2014). Uma R-CNN é dividida em três módulos: gerador de propostas de regiões, grande rede neural convolucional e um conjunto de SVMs lineares específicos de cada classe.

Inicialmente o algoritmo executa um gerador de propostas de regiões candidatas na imagem inteira para selecionar regiões que possam conter um objeto. Isso é feito por meio do algoritmo de pesquisa seletiva desenvolvido por Uijlings et al. (2013), que adapta a segmentação para uma busca seletiva, usando um conjunto diverso de estratégias de agrupamentos complementares e hierárquicas. Dentre as vantagens da busca seletiva estão sua estabilidade, robustez, velocidade e independência de classe de objetos, podendo identificar objetos rígidos, não rígidos e amorfos (UIJLINGS et al., 2013).

Em seguida o R-CNN aplica em cada proposta de região uma rede neural convolucional de alta capacidade, capaz de extrair um vetor de características de comprimento fixo. Nessa tarefa é usado o framework *Caffe*, que seleciona um vetor de características de dimensão 4096 por meio da propagação da imagem em RGB de 227x227, em cinco camadas convolucionais e duas totalmente conectadas.

Por fim, com os recursos selecionados, propaga-se os mesmos em um SVM de cada classe, que é treinado individualmente, resultando em uma pontuação para a classe. Cada classe possui um SVM que passou por um treinamento supervisionado e cada SVM é utilizada na etapa final processando os recursos coletados. Com todas as regiões pontuadas de uma imagem, aplica-se uma supressão não máxima para cada classe, que rejeita uma região se a mesma tiver uma sobreposição de união de interseção com uma região selecionada de pontuação mais alta maior que um limite aprendido (GIRSHICK et al., 2014).

O R-CNN alcança uma performance alta e robusta devido a dois fatores: a aplicação de CNNs de alta capacidade a propostas de regiões de baixo para cima para localizar e segmentar objetos e o treinamento de grandes CNNs. O massivo pré-treinamento supervisionado para uma tarefa auxiliar, seguido por um ajuste fino específico do domínio, resulta em um aumento no desempenho, que pode ser usado com eficácia para outros problemas de visão computacional.

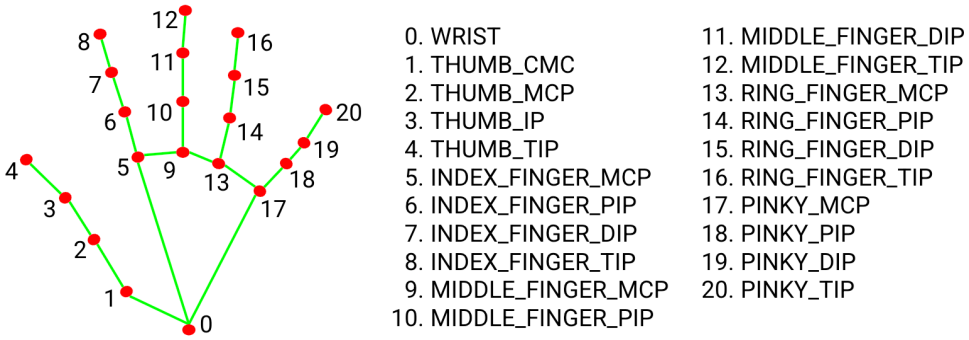
3 FERRAMENTAS PARA IMPLEMENTAÇÃO

Para o desenvolvimento do projeto é utilizada a linguagem de programação Python e as ferramentas Mediapipe, OpenCV, Pynput, Dlib e PyQt5. Cada uma dessas ferramentas são abordadas nos tópicos abaixo.

3.1 MEDIAPIPE

Mediapipe é uma ferramenta de código aberto que permite customizar redes neurais para vídeos (MEDIAPIPE, 2020b). Dentre as detecções que a Mediapipe apresenta suporte, está a detecção de mãos, que usa um primeiro filtro que detecta uma palma da mão na imagem e um segundo filtro com 21 marcações da mão (Figura 6). Neste trabalho a Mediapipe é usado para detectar esses pontos da mão e suas posições.

Figura 6 – Marcações da mão feita pela biblioteca Mediapipe



Fonte: Mediapipe (2020a).

O detector de palma opera na imagem inteira e retorna uma caixa delimitadora com a mão orientada, isso é feito usando um detector de disparo único junto a um algoritmo de supressão não máxima (método da secção 2.6.5). Esse processo restringe o processamento da próxima rede neural a uma área de interesse que contém os dados da mão, permitindo otimizações de rotação de imagens e área a ser analisada.

O modelo de referência identifica as 21 marcações da mão a partir da imagem segmentada, esse processo é feito em uma rede neural convolucional que passou por um processo de aprendizado supervisionado com imagens da mão em diferentes ângulos e movimentos. O modelo de referência usa um método com três eixos na imagem para detectar melhor os movimentos de abrir ou fechar a mão. Essa classe é construída a partir das técnicas de aprendizado de máquina e redes neurais convolucionais, faz uso de modularização para aumentar a otimização e possui várias camadas de processamento.

O Mediapipe emprega pipelines de aprendizado de máquina, que funciona orquestrando o fluxo de dados de entrada, recursos de processamento, modelo de aprendizado, parâmetros e saída prevista. O primeiro modelo faz a detecção e delimitação da palma da mão (modelo de disparo único) para que o segundo modelo (modelo de regressão) opere sobre uma região já recortada de menor dimensão, facilitando o processamento.

Dentre os conceitos do framework estão os *Calculators* e *Graphs*. Os Calculators são uma subclasse da classe *CalculatorBase* possuindo uma inicialização, abertura, processamento e fechamento. O processamento é feito repetidamente, podendo ser em paralelo, e é responsável pelo reconhecimento e rastreamento das marcações da mão em si. Os Graphs são uma especificação da topologia e das funcionalidades de uma classe do Mediapipe, possuindo uma série de nós, onde a saída de um é a entrada de outro (comunicação em pacotes), sendo que cada nó de um gráfico possui uma calculadora. Essa modularização permite um alto desempenho e reúso. A ferramenta ainda faz uso de técnicas de detecção em vídeo, que aproveitam o resultado anterior para resolver o frame atual.

3.2 OPENCV

Open source computer vision library (OpenCV) é uma biblioteca de código aberto com estrutura modular que possui vários algoritmos de visão computacional (BRADSKI, 2021). Dentre os recursos disponibilizados pela OpenCV estão funções de processamento de imagens, controle de interface, manipulação de vídeo, recursos para imagens 3D e 2D, redes neurais, aprendizado de máquina e gráficos.

Neste trabalho foram utilizados do OpenCV os recursos de aplicar filtro nas imagens capturadas, converter o espectro de cores de uma imagem, desenhar nas imagens (usado na implementação) e fazer a captura de vídeo da câmera. A abertura da câmera e a captura das imagens foi realizado com funções fornecidas pelo OpenCV.

O OpenCV fornece funções que permitem desenhar pontos, contornos e retas na imagem, que facilitam o processo de implementação das funcionalidades da aplicação. O OpenCV também fornece conversões de espectro de cores para RGB, cinza e outras que são necessárias para a imagem de entrada de algumas ferramentas de detecção. Além disso, o OpenCV fornece filtros que podem ser aplicados nas imagens, como o filtro gaussiano, que reduz o ruído e bordas falsas da imagem.

3.3 PYNPUT

Pynput é uma biblioteca que permite controlar e monitorar as funções do mouse e teclado (PYNPUT, 2021). A ferramenta Pynput é utilizada para controlar o ponteiro do mouse, efetuar os cliques do botão direito e esquerdo e abrir o teclado.

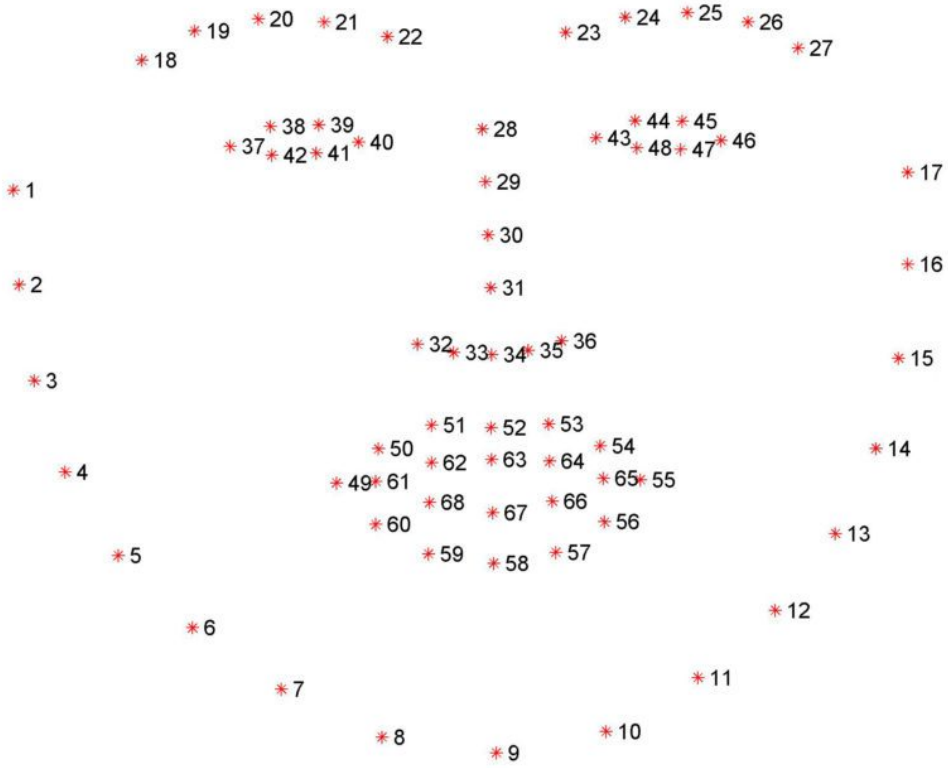
3.4 DLIB

Dlib é um kit de ferramentas contendo algoritmos de aprendizado de máquina e ferramentas para a criação de software complexo para resolver problemas do mundo real (KING, 2009). Neste projeto o Dlib é utilizado para a detecção dos olhos e nariz, além de fornecer um arquivo já treinado que realiza a identificação de 68 pontos no rosto (Figura 7).

Para detectar o rosto e rastrear os 68 pontos, o Dlib utiliza histograma de gradientes orientados (DALAL; TRIGGS, 2005) e máquina de vetores de suporte linear (CORTES; VAPNIK, 1995). O uso de gradientes orientados resulta em uma rede invariante em escala, iluminação e rotações, tornando a mesma robusta e eficiente em representar as características da imagem.

Já a máquina de vetores de suporte é um dos principais métodos de classificação de dados, podendo categorizar dados não separáveis usando aumento de dimensão. Assim, essa rede neural consegue lidar com grande diversidade de dados e valores, mantendo a detecção resiliente.

Figura 7 – Marcações do rosto feita pela biblioteca Dlib



Fonte: Rosebrok (2017).

3.5 PYQT5

PyQt5 é uma ligação para Python do Qt v5, o qual é uma biblioteca em C++ que implementa APIs de localização, posicionamento, multimídia, Bluetooth, interface de usuário, dentre outras (PYQT, 2012), assim, o PyQt5 permite que a linguagem Python suporte a biblioteca em C++ Qt. Neste projeto o PyQt5 será utilizado para construir a interface de usuário, que exibirá alertas e várias caixas que permitem ao usuário selecionar quais interações deseja ativar, além de ter um botão fechar.

4 TRABALHOS CORRELATOS

Neste item são apresentados trabalhos cujo tema aborda a visão computacional e interação humano computador, objetivando fundamentar a discussão teórica da pesquisa. A exposição dos trabalhos de Mauri, Solanas e Granollers (2009), Santos et al. (2015)), Almeida (2013) e Truyenque (2005) implica, entre outros aspectos, na distinção do uso, para interação com o computador, de gestos, do corpo inteiro e da mão. Embora este trabalho não objetive atender a pessoas com movimentos limitados, como realizado pelo trabalho de Santos et al. (2015), o controle do mouse pelos olhos é similar ao proposto.

4.1 ENABLE VIACAM (MAURI; SOLANAS; GRANOLLERS, 2009)

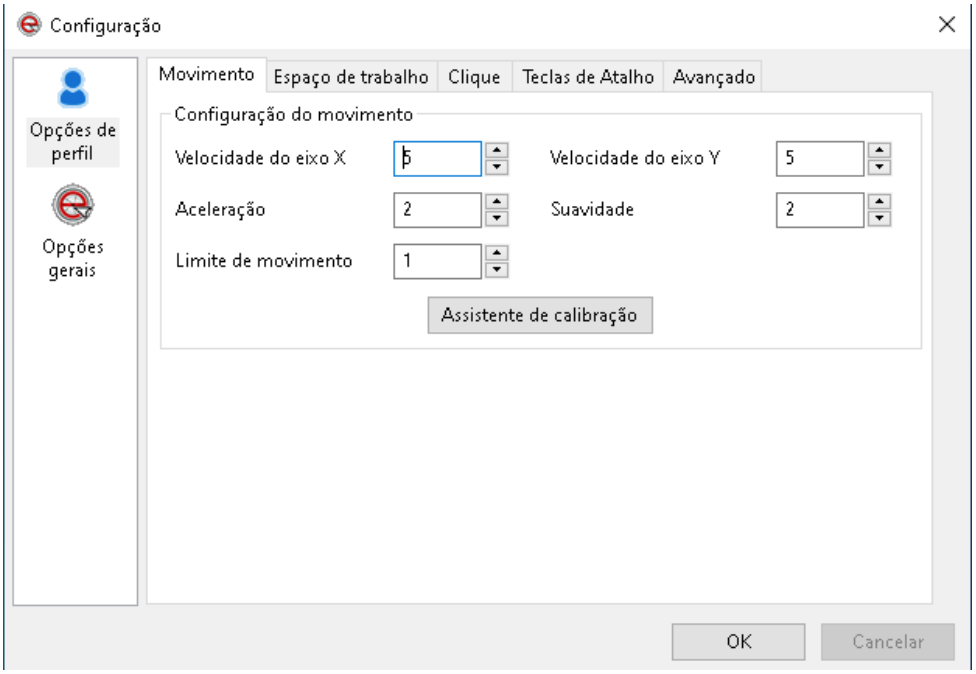
ENABLE VIACAM (eViacam) é um aplicativo de interação humano computador desenvolvido por Mauri, Solanas e Granollers (2009). Essa aplicação permite controlar o movimento do mouse utilizando o rosto, por meio de imagens da WEB CAM, e efetuar um clique após ficar com o rosto imóvel por alguns segundos. O clique a ser efetuado pode ser escolhido no menu da aplicação, podendo ser o clique esquerdo, direito, rolagem, arrastar ou duplo clique (Figura 8). Após executar algum desses cliques selecionados uma vez, a opção padrão, botão direito, volta a ser habilitada. Após a instalação, o eViacam permite habilitar uma opção que faz a aplicação inicializar junto ao sistema operacional. O eViacam tem como principal objetivo tornar a WEB CAM um dispositivo de entrada "independente das mãos".

Figura 8 – Interface de usuário do eViacam

Fonte: (MAURI; SOLANAS; GRANOLLERS, 2009)

O eViacam permite efetuar uma calibragem personalizada que captura os movimentos horizontais e verticais máximos que o usuário deseja se mover e os utiliza para gerar os parâmetros de espaço de trabalho da aplicação, que também podem ser configuradas no menu. Ele também permite configurar a sensibilidade do movimento, como velocidade nos eixos x e y, suavização e aceleração (Figura 9). Os cliques podem ser personalizados, como tipo, tempo de espera para executar, cliques consecutivos e deslocamento máximo para efetuar um novo clique. O eViacam ainda permite configurar algumas teclas de atalho para interagir com a aplicação, podendo ser secelionado o tipo de atalho e a funcionalidade a ser executada.

Figura 9 – Opções de configuração do eViacam



Fonte: (MAURI; SOLANAS; GRANOLLERS, 2009)

4.2 MOUSE OCULAR PARA PESSOAS COM MOVIMENTOS LIMITADOS CAUSADOS POR PROBLEMAS CERVICAIS E/OU CEREBRAIS (SANTOS et al., 2015)

Santos et al. (2015) desenvolveram uma aplicação de visão computacional que usa os movimentos dos olhos e das pálpebras para controlar o ponteiro do mouse. O mouse se movimenta de acordo com o posicionamento da íris no olho e os cliques são feitos com o fechamento dos olhos. Essa aplicação possibilita que pessoas com movimentos limitados por problemas cervicais, ou seja, que não tem movimentos normais das mãos, interajam com o computador por meio dos olhos do usuário e uma câmera externa de 720p de resolução. A substituição da câmera embutida no Notebook por uma externa foi feita para conseguir maior resolução. Para a interação foram implementadas as funções do Quadro 2.

Para o desenvolvimento do projeto foi utilizado a linguagem c# e funções da biblioteca OpenCV. No processamento das imagens são feitos o reconhecimento da face, suavização, detecção de bordas e localização do centro da circunferência (olhos) e para o reconhecimento facial foi usado a técnica proposta por Paul Viola e Michael Jones, que utiliza um algoritmo de aprendizado a partir das características extraídas do filtro de Haar.

A suavização consiste em desaparecer bordas irrelevantes devido a ruídos, e é feita com a convolução com um filtro predeterminado. A detecção de bordas é feita com o algoritmo de John Canny e a localização do centro da circunferência da

Quadro 2 – Listas de funções do trabalho de Santos et al. (2015).

Função	Movimento
Realizar o clique do botão esquerdo ou acionar a tecla selecionada no teclado virtual	Fechar olho Esquerdo (2 segundos)
Reiniciar o ponteiro do mouse para canto inferior central	Fechar olho direito (2 segundos)
Ativar ou desativar a rolagem	Fechar os dois olhos (3 segundos)
Movimentar o ponteiro do mouse ou a barra lateral para direção desejada e navegar entre as teclas do teclado virtual.	Olhar para cima, para baixo, para esquerda ou para direita.

Fonte: Santos et al. (2015, p. 97).

imagem é feito para localizar o centro do olho, tarefa feita por meio da transformada de Hough-Circles, que produz uma matriz de mesmo tamanho da imagem e depois percorre a mesma para incrementar os pixels em destaque e todos em um raio R do mesmo, sendo o centro do olho localizado nos maiores valores da matriz.

Para a implementação ainda foi desenvolvido um processo de calibragem do algoritmo para que as direções do olhar para cima/baixo/esquerda e direita fossem adequadas aos usuários, o centro é acionado quando as outras direções não são detectadas, funcionando como uma área neutra que não move o mouse. A ferramenta ainda conta com uma barra de ferramentas que não minimiza após a abertura de novas abas.

Os autores concluíram que foi possível movimentar o mouse por toda a tela e fazer uso do teclado virtual, mesmo que sem alta precisão e não tendo a mesma agilidade que um mouse. Os autores recomendaram que o posicionamento da câmera não seja inclinado e que a distância aproximada de 70cm da câmera é ideal. Por fim, notaram que alta ou baixa iluminação influenciaria no desempenho do algoritmo, assim como o uso de óculos pelo usuário, pois esse uso gera reflexos e perdas de resolução.

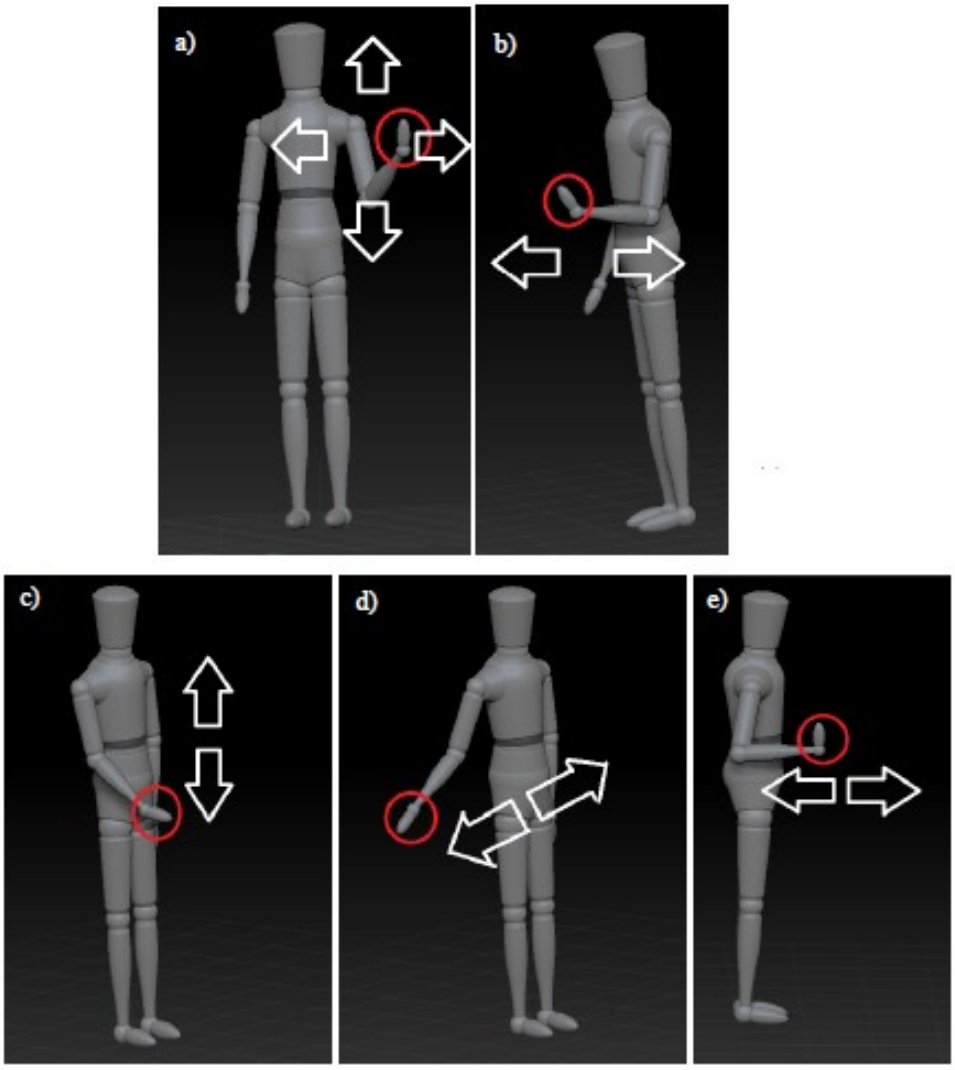
4.3 SISTEMA INTERATIVO BASEADO EM GESTOS PARA UTILIZAÇÃO DE COMANDOS NO COMPUTADOR (ALMEIDA, 2013)

Almeida (2013) desenvolveu uma aplicação interativa que detecta gestos tridimensionais, por meio do dispositivo microsoft kinect, e realiza comandos no computador. O microsoft kinect é um sensor de movimentos que possui reconhecimento de voz e detecção do esqueleto corporal.

O autor desenvolveu quatro telas para interface de usuário que descrevem os gestos e as ações que a aplicação suporta, além de permitir algumas configurações. Três telas descrevem os tipos de gestos e suas respectivas ações no sistema operacional e a última permite algumas configurações. Dentre os gestos que o projeto

consegue detectar estão os da Figura 10, onde o gesto da Figura 10-a representa as ações de arrastar e selecionar, o da figura 10-b clicar, o da Figura 10-c rolar, o da Figura 10-d rotacionar e o da Figura 10-e aplicar zoom.

Figura 10 – Protótipo dos gestos



Fonte: Almeida (2013, p. 54).

A implementação do projeto é feita usando a *Application Programming Interface* (API) do kinect, que fornece a captura e segmentação dos principais pontos do esqueleto da pessoa e várias outras classes para captura e manipulação dos dados do sensor. A partir dos dados do kinect o autor implementou a detecção de gestos de acordo com a posição relativa dos pontos do esqueleto e em seguida, converteu o gestos em um comando para o computador.

O autor concluiu que o trabalho desenvolvido se mostrou usual e prático, porém é feito para um sistema operacional e contexto de uso específico, apesar de poder ser adaptado. O projeto possui como limitações a precisão do sensor e do algoritmo de reconhecimento de gestos desenvolvido. O autor ainda observa que a computação perceptiva é uma área promissora para interação humano-computador, pois há um

avanço crescente das tecnologias disponíveis.

4.4 UMA APLICAÇÃO DE VISÃO COMPUTACIONAL QUE UTILIZA GESTOS DA MÃO PARA INTERAGIR COM O COMPUTADOR (TRUYENQUE, 2005)

Truyenque (2005) desenvolveu uma aplicação que permite ao usuário interagir com o computador por meio de gestos com a mão, dispensando o uso dos demais dispositivos, sendo necessário apenas uma câmera, o que torna o projeto não intrusivo e de baixo custo. O projeto é dividido em segmentação por subtração de fundo, detecção de silhueta e reconhecimento de gestos.

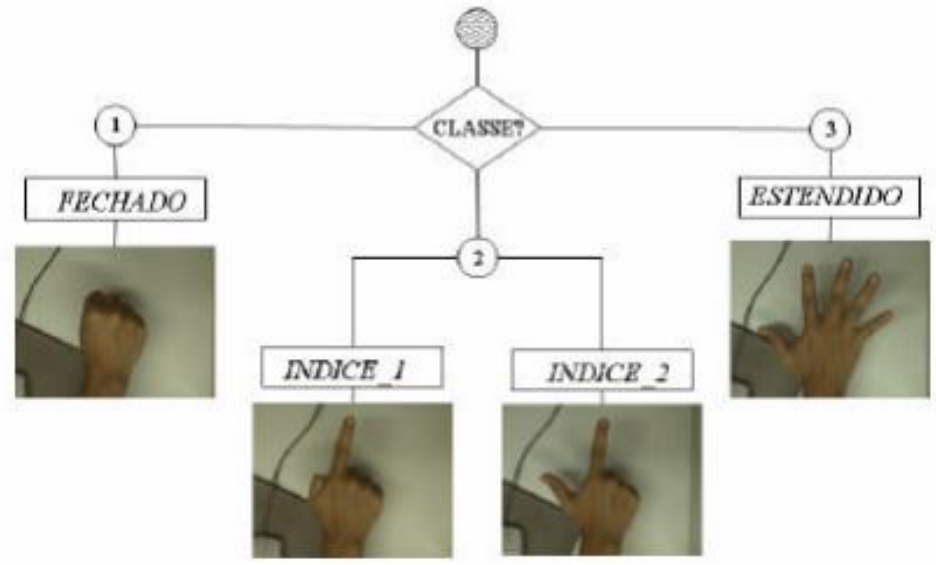
Na segmentação por subtração de fundo é feita a detecção de um fundo estático, que é usado para comparar com as imagens atuais e detectar uma mão e o fundo. O modelo de cor usado no algoritmo é feito usando o vetor *chromaticity line* para que se possa comparar cores iguais com intensidades luminosas diferentes e assim detectar sombras. A modelagem da imagem média de fundo é feita pelo algoritmo de Horprasert, Harwood e Davis (2000), que usa conjunto de imagens de treino. Por fim é feita a subtração de fundo, em que é calculada a diferença entre a imagem de referência e a imagem atual, diferença que é identificada pela distorção de cor entre as duas imagens.

Na detecção de silhueta são detectados pontos que fazem parte da morfologia da mão, de forma que esses dados possam ser usados posteriormente. Esta detecção foi desenvolvida em três passos: melhoria da segmentação, detecção de bordas e estruturação dos pontos que fazem parte da silhueta da mão. Na primeira etapa foram aplicados filtros morfológicos de dilatação e erosão, que completam pequenos buracos e removem regiões com poucos pixels. Na segunda etapa são feitas a detecção de bordas, que utilizou os resultados do filtro de dilatação menos o de erosão para encontrar a borda, o que gera economia computacional; e o armazenamento dos contornos dos objetos a partir de um algoritmo que se inicia em um ponto que verifica as vizinhanças e se direciona para um que está preenchido. No último passo foi feita a seleção da maior silhueta encontrada na imagem, para evitar ruídos, e uma subamostragem, para reduzir a quantidade de pontos a serem usados.

No reconhecimento de gestos foram realizados o reconhecimento de gestos, posição e orientação dos dedos a partir da silhueta. Para alcançar esses objetivos foram identificados os conjuntos de pontos que formam picos (pontas dos dedos) e vales (fundo do dedo). A partir desses dados foram usados os picos para detectar os dedos e a direção foi calculada a partir da posição pontos vizinhos e a direção de sua curvatura. Os tipos de gestos identificados pelo algoritmo são mostrados na Figura 11. A identificação é feita contabilizando o número de dedos detectados.

Na aplicação dos gestos coletados foi desenvolvido um sistema de controle para

Figura 11 – Gestos identificados pelo algoritmo

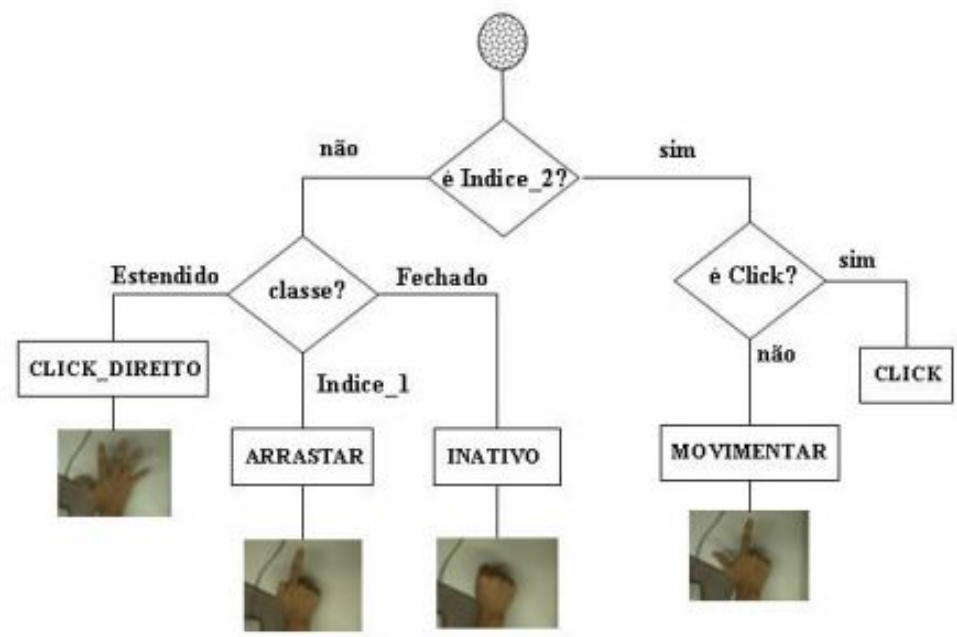


Fonte: Truyenque (2005, p. 76).

o mouse e um sistema de controle para o jogo AirStrike II. As funções implementadas para o mouse de acordo com o gesto da mão, podem ser vistos na Figura 12. Para o jogo foram implementadas as funções ESQUERDA, DIREITA, PARA_FRENTE, PARA_TRAS, ATIRAR, MUDAR_ARMA e SOLTAR_EXPLOSIVO. Os movimentos foram implementados usando o gesto índice 1 e o ângulo que o mesmo forma de acordo com a referência. A função ATIRAR é feita pelo gesto índice 1, a função MUDAR_ARMA é feita quando a sequência índice 2 e índice 1 for identificada e a função SOLTAR_EXPLOSIVO foi feita sempre que o gesto mão estendida for feito.

O autor concluiu que o projeto detectou adequadamente os gestos e os dedos, com alta precisão, além de operar em tempo real a uma taxa de 20 quadros por segundo. Para o autor é possível desenvolver aplicações de visão computacional para interação, que dispensam marcadores, outros equipamentos ou adaptação do ambiente. O projeto ainda lida bem com alterações de iluminação, brilho ou reflexos.

Figura 12 – Controle do mouse feito pelo algoritmo



Fonte: Truyenque (2005, p. 85).

5 DESENVOLVIMENTO

Assim, este trabalho consiste em capturar as imagens feitas pela câmera, determinar se há um braço ou um rosto, identificar a posição do braço, identificar o status dos olhos (aberto ou fechado) e converter essas informações em comandos para o computador, sendo que cada uma dessas tarefas pode ser selecionada pelo usuário. Para isso foram utilizadas bibliotecas de código aberto para efetuar a detecção do braço e do olho. O Quadro 3 apresenta os comandos e funcionalidades implementadas neste trabalho. Mais adiante, cada elemento desse quadro será detalhado.

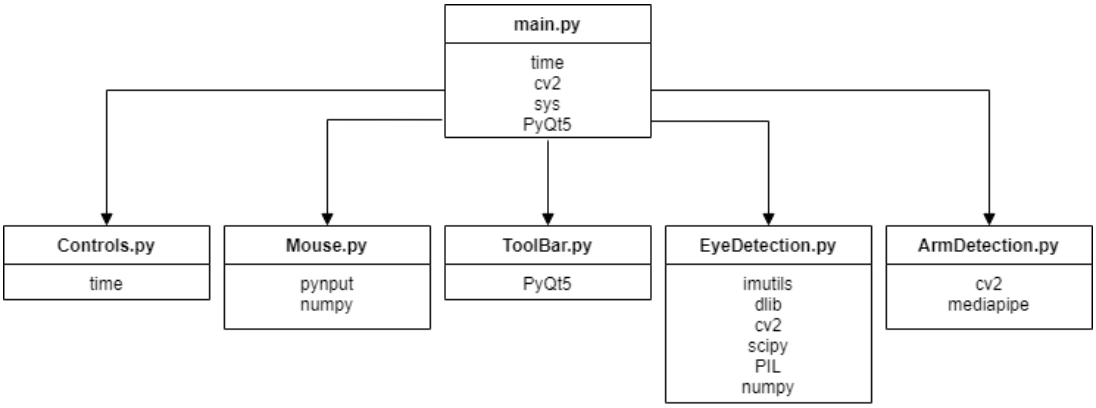
Quadro 3 – Comandos feito pela aplicação

Comando	Funcionalidade
Abrir o teclado virtual	dois olhos fechados por 3s
Efetuar o clique esquerdo	olho esquerdo fechado por 1.5s ou 3s(duplo)
Efetuar o clique direito	olho direito fechado por 1.5s
Rolagem	dedinho (cima, baixo ou meio)
Duplo clique e arrastar	dois dedos para cima (Duplo clique) e três dedos para cima (Arrastar)
Cliques com a orientação da palma da mão	Efetua cliques de acordo com a rotação da mão
Movimentar o mouse	Opções abaixo
Mover o ponteiro do mouse	palma da mão
	rosto
	dedo indicador
	dedo polegar
	dedo médio
	dedo anelar
	dedinho
	direção do olhar
	direção do olhar e dedo indicador

Fonte: Autor (2021).

A fim de facilitar a implementação e organização do trabalho foram feitas classes para encapsular as principais tarefas do projeto: obter coordenada da mão (classe *ArmDetection.py*), obter status do olho (classe *EyeDetection.py*), construção da interface (classe *ToolBar.py*) e executar os comandos (classe *Mouse.py*). A integração dessas classes é feita nos arquivos *main.py* e *Controls.py*, que contêm o loop do processo de captura e execução das tarefas selecionadas (Figura 13, onde *cv2* representa a bilioteca OpenCV). Cada um desses arquivos são explicados nos itens 5.1 a 5.5.

Figura 13 – Classes usadas na ferramenta



Fonte: Autor (2021).

5.1 IDENTIFICAÇÃO E RASTREAMENTO DA MÃO

Para encapsular as funções relacionadas a obtenção de coordenadas de pontos da mão, foi feita uma classe chamada *ArmDetection.py*. Essa classe é inicializada com o coeficiente de confiança de detecção, apesar de possuir um valor padrão de 0.7, e possui cinco funções: *convert_to_rgb*, *find_arm*, *two_or_three_fingers*, *finger_up_or_down*, *click_arm* e *find_all_positions*.

A função *convert_to_rgb* é usada para converter uma imagem de RGB para BGR utilizando a função *cv2.cvtColor* da biblioteca OpenCV. A função *find_arm* é usada para determinar se há ou não uma mão na imagem, caso positivo, pode-se então encontrar a posição da palma da mão. O trabalho de detecção da mão é feito pela biblioteca Mediapipe, por meio da classe *Hands*, disponível em *mediapipe.solutions.hands.Hands*. Essa classe recebe os parâmetros modo da imagem (*static_image_mode*), número máximo de mãos (*max_num_hands*), coeficiente mínimo de detecção da mão (*min_detection_confidence*) e coeficiente mínimo de detecção das marcações da mão (*min_tracking_confidence*). A função da classe *Hands* do Mediapipe que efetua a deteccão é chamada *process* e retorna um array contendo as coordenadas e as laterais de cada mão encontrada.

A partir da resposta da função *find_arm* é feito o rastreamento da posição do ponto 0 na palma da mão (Figura 6) pela função *find_all_positions*. O rastreamento apenas faz a multiplicação das taxas de altura e largura, encontrada pela função *find_arm*, pelas dimensões da imagem, retornando um array com as coordenadas de cada uma das 21 marcações da mão.

A função *click_arm* identifica a rotação da mão, comparando o valor das coordenadas y de dois pontos na palma da mão (pontos 0 e 1). Caso a coordenada y do ponto um esteja no intervalo de y0 + 25 e y0 + 40, então há o clique com o botão esquerdo. Caso a coordenada y do ponto zero da palma da mão esteja 45 pixels acima

do ponto um, então há o clique do botão direito. Esses comandos são acionados após 1.5 segundos atendendo a cada regra, sendo possível efetuar duplo clique após três segundos na posição horizontal.

A função *two_or_three_fingers* identifica se os dedos indicador, médio e anelar estão em conjunto levantados (retorna *THREE*), ou os dedos indicador e médio estão ambos levantados (retorna *TWO*) ou não (retorna *DOWN*). Já a função *finger_up_or_down* identifica se o dedo está levantado (retorna *UP*), abaixado (retorna *DOWN*) ou não (retorna *DOWN*).

5.2 IDENTIFICAÇÃO E RASTREAMENTO DOS OLHOS

Para identificar o status de cada olho é utilizado a biblioteca Dlib e uma rede neural já treinada disponível no github[1], que detecta 68 pontos no rosto da pessoa, incluindo seis pontos em cada olho. Esse processo, encapsulado na classe *EyeDetection.py*, inicia recebendo as variáveis *both_eye_ar_thresh* e *single_eye_thresh*, que por padrão tem os valores 0.19 e 0.2, respectivamente. Essa classe tem oito funções: *convert_to_rgb*, *find_face*, *left_eye*, *right_eye*, *both_eye*, *find_iris*, *find_position_nose* e *find_position_look*.

A função *convert_to_rgb*, assim como na classe ArmDetection.py, é responsável por converter o espectro de cores da imagem. A função *find_face* é usada para detectar se há um ou mais rostos por meio da função *dlib.get_frontal_face_detector* da biblioteca Dlib, caso positivo pode-se executar as funções que analisam os status dos olhos, entretanto essa função é utilizada apenas internamente na classe.

As funções *right_eye*, *left_eye* e *both_eye* são responsáveis por determinar se os olhos direito, esquerdo e ambos estão fechados ou abertos. Essas funções recebem um array com as coordenadas do rosto, separam os pontos que correspondem a cada olho, calculam a taxa relativa de abertura de cada olho e determinam se estão fechados ou abertos.

A função *find_iris* é responsável pela identificação da presença ou não do olho, isso é feito verificando se há ou não um rosto, caso positivo, então há dois olhos; a biblioteca usada neste processo é a Dlib. Em seguida, caso haja um rosto, a classe *dlib.shape_predictor* é usada para converter a imagem com rosto, retornada da função *dlib.get_frontal_face_detector*, para 68 pontos com posições do rosto.

A função *find_position_nose* calcula e retorna a posição da ponta do nariz do rosto detectado na imagem. A função *find_position_look* é a mais complexa e começa extraindo da imagem da câmera uma região de interesse que engloba o olho esquerdo. Em seguida essa região passa por uma ampliação, conversão para o espectro de cores cinza e suavização de bordas com a função *GaussianBlur* da biblioteca OpenCV.

[1] https://github.com/davisking/dlib-models/blob/master/shape_predictor_68_face_landmarks.dat.bz2

Esse processo reduz os efeitos do ruído e minimiza bordas falsas, aplicando uma função gaussiana (Equação 4) nos pixels da imagem. Essa função funciona aplicando uma transformação bidimensional, ou seja, uma convolução da imagem com um filtro de valores gaussianos. A Figura 14 representa uma imagem original e o seu resultado aplicando a função gaussiana ($\sigma = 2$).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4}$$

Figura 14 – Aplicação do filtro gaussiano

Fonte: Yap, Bister e Ewe (2003).

Em seguida, é feito a identificação do limiar da pupila, onde é usada a função *cv2.threshold* com parâmetros que resultam em bordas 1 e fundo 0. Essa função funciona verificando se cada pixel é maior que um limiar, se sim então o mesmo passa a valer 1, senão 0. Por fim, é determinado a borda da pupila, usando a função *cv2.findContours*, que constrói uma curva na fronteira do objeto onde zeros e uns se encontram, o que permite encontrar a área do mesmo com a função *cv2.contourArea* e as coordenadas do centro do centro da pupila, usada para mover o ponteiro.

5.3 CONTROLE DOS COMANDOS NO COMPUTADOR

Para executar os comandos de cliques, abrir teclado virtual e mover o mouse, foi feita uma classe de nome *Mouse.py* que encapsula essas tarefas em dez funções: *click*, *keyboard*, *scroll_down*, *scroll_up*, *drag_press*, *drag_release*, *move*, *move_mouse*, *move_mouse_eye_direction* e *move_finger_eye*. O teclado virtual (função *keyboard*) é aberto pressionando as teclas *Ctrl*, *windows* e *o*, para isso ser feito no algoritmo foi usada a biblioteca *pynput*, que permite controlar e monitorar comandos do computador, por meio da função *pynput.keyboard.Controller().press(TECLA)*, substituindo o valor de *TECLA* pelas teclas usadas.

A função *click* recebe como parâmetro o tipo de click a ser feito (direito, esquerdo, duplo), e executa o clique com a função do pynput *pynput.mouse.Controller().press(pynput.mouse.Button.TYPE)*. O valor de *TYPE* é substituído pelos valores de *right* ou *left*, e a função recebe a variável adicional 2, caso seja um duplo clique.

As funções *scroll_down* e *scroll_up* realizam a rolagem da tela para baixo e para cima, respectivamente. As funções *drag_press* e *drag_release* realizam as funções de arrastar. A função *move* move o ponteiro do mouse direto para uma posição recebida.

A função *move_mouse* movimenta o mouse a partir de pontos relativos à câmera e da posição anterior. Essa função possui um mecanismo de suavização do movimento do mouse, para evitar tremedeiras na tela, que usa a posição anterior e movimenta para a atual a uma taxa de 0.167 por padrão, mas que pode ser configurada. Para evitar perder a detecção do braço nas extremidades da tela, a função coloca uma borda relativa menor que a da imagem da câmera, que reduz a altura em 250 pixels e a largura em 100 pixels, permitindo que o ponteiro atinja a borda antes do braço, dedo ou rosto chegar nos cantos. O movimento do cursor do mouse é feito pela alteração da variável *pynput.mouse.Controller().position*, que recebe as coordenadas de largura e altura para se mover.

A função *move_mouse_eye_direction* também movimenta o ponteiro do mouse, porém o faz com as coordenadas da pupila. Por último tem-se a função *move_finger_eye*, que faz pequenos movimentos no mouse em qualquer que seja sua localização e é usada como ajuste fino para as coordenadas do olho.

5.4 MENU

Para que o usuário consiga selecionar quais interações deseja fazer e saber se o seu braço ou olhos estão sendo detectados, foi construído um menu com checkboxs para o usuário selecionar quais formas de interação aplicar que também exibe dois alertas, um se o rosto não está sendo detectado e outro se o braço não está sendo detectado. O menu foi desenvolvido com o auxílio da ferramenta PyQt5, que fornece uma interface de programação de aplicações que permite desenvolver interface de usuário.

A classe responsável por esse menu é a classe *ToolBar.py*, que possui uma função que altera a mensagem de alerta exibida (*printf_msg*), caso o algoritmo volte a detectar o braço ou rosto. Essa função recebe como parâmetro uma mensagem e altera o alerta do menu. O menu pode ser minimizado e possui um botão que fecha a aplicação.

Esse menu ainda possui seis checkboxs para o usuário selecionar quais formas

de interação deseja, sendo que a última diz respeito ao controle do cursor do mouse, que habilita uma lista suspensa permitindo ao usuário selecionar apenas um tipo de controle que deseja usar no ponteiro do mouse. As opções de checkbox podem ser vistas no Quadro 3, sendo que as opções de mover o ponteiro do mouse fazem parte da lista suspensa e as demais são checkbox.

Para monitorar as opções do menu foram feitas funções que retornam o status de cada checkbox e da opção selecionada na lista de opções. Essas funções são chamadas a cada ciclo do programa, e determinam se alguma tarefa deva ser executada.

5.5 CLASSE PRINCIPAL

Para desenvolver o loop principal do projeto foi feita uma classe *main.py* que também faz uso das demais classes citadas nos itens anteriores. Essa classe inicia o menu, captura as imagens da câmera, efetua a contagem de tempo dos olhos fechados e conecta as respostas das classes *ArmDetection.py* e *EyeDetection.py* com a classe que efetua comandos no sistema operacional *Mouse.py*. A classe *main* ainda faz uso de uma classe *Controls.py* para encapsular as funções relacionadas à classe *EyeDetection.py* e a classe *ArmDetection.py*.

Para fazer a captura de vídeo da câmera é usada a classe *cv2.VideoCapture* junto com a função *read()* da bilioteca OpenCV. Essa função é implementada a partir de uma API do sistema operacional, que fornece acesso aos dispositivos do computador. Em seguida a função *iris_control* da classe *Controls* é chamada, onde verifica a presença ou não de um rosto. Caso positivo são verificados os status de cada olho e se alguma das tarefas de abrir o teclado, clicar com o olho esquerdo, clicar com o olho direito, movimentar o mouse com o rosto, olhar ou olhar e dedo, estiver selecionada; então as mesmas são executadas. Nessa função ainda são coletados os tempos de cada olho fechado.

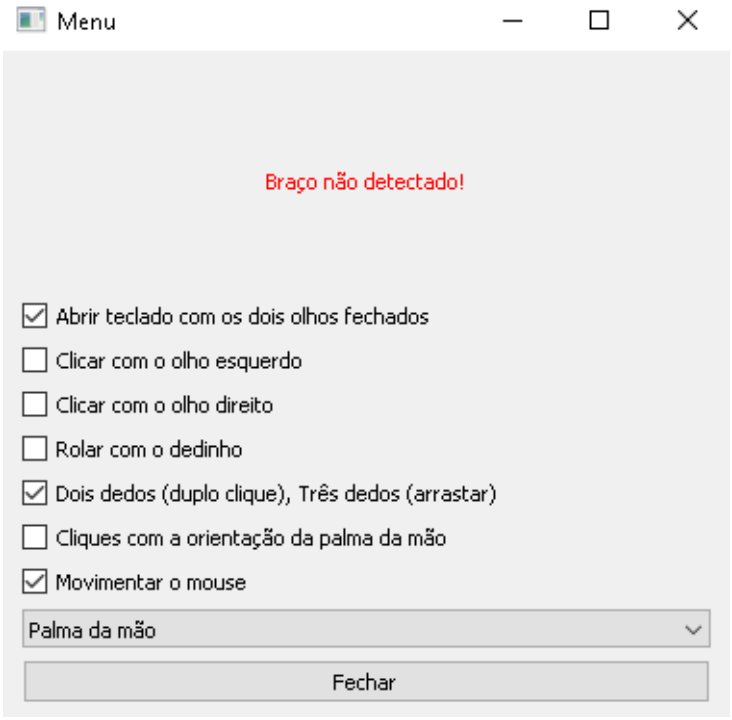
Na sequência a função *arm_control* da classe *Controls* efetua o processo de detecção da mão e caso positivo, verifica se alguma das tarefas de rolagem, duplo clique/arrastar, movimentar o mouse com algum dedo ou palma da mão, está selecionada. Se alguma dessas tarefas estiver selecionada então a mesma é executada.

Os alertas também são feitos na classe *main.py* de acordo com a resposta da detecção do braço/olho. A função *close* fecha a aplicação e é acionada após um duplo clique no botão fechar do menu.

6 RESULTADOS

As funcionalidades implementadas no projeto são mostradas no Quadro 3 e o menu desenvolvido, na Figura 15. Observa-se que as seis primeiras opções são checkboxes que permitem ao usuário selecionar quais interações deseja usar, a última é uma lista de seleção, onde apenas uma opção pode ser selecionada.

Figura 15 – Menu do aplicativo



Fonte: Autor (2021).

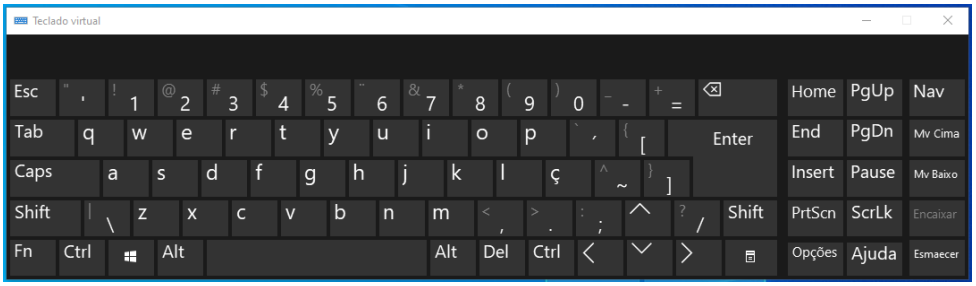
Para iniciar a aplicação é preciso usar os meios tradicionais de acesso para rodar o programa uma primeira vez, sendo que na sua inicialização, por padrão, as opções de abrir o teclado com os dois olhos fechados, duplo clique e arrasto com os dedos e movimentar o cursor com a palma da mão estarão selecionadas. Assim, em seguida, o usuário passa a ter métodos adicionais de interação com o computador, podendo usar o meio que lhe é mais conveniente para personalizar o acesso.

A opção *Abrir teclado com os dois olhos fechados* permite abrir o teclado virtual (Figura 16) fechando os dois olhos por 3 segundos continuamente. A opção *Clicar com o olho esquerdo* permite efetuar cliques do botão esquerdo fechando-se o olho esquerdo por 1.5s (um clique) ou 3s (dois cliques). A opção *Clicar com o olho direito* permite efetuar cliques com o botão direito do mouse fechando o olho direito por 1.5s.

A detecção do status dos olhos funciona de forma aceitável, com um tempo de resposta bom e alta precisão, sendo necessário um pouco de treino por parte do

usuário e adaptação de alguns parâmetros para o contexto de uso do usuário. Nota-se que essas opções só são executadas caso haja um rosto na webcam. A Figura 17 mostra as marcações (pontos em vermelho) do olho usada para detectar a curvatura do olho e seu status.

Figura 16 – Teclado virtual



Fonte: Autor (2021).

Figura 17 – Pontos do nariz e olhos

Fonte: Autor (2021).

A quarta opção (*Rolar com o dedinho*) permite rolar a página de acordo com a posição do dedinho, caso a ponta do dedinho esteja a mais de 20 pixels de altura da posição de sua base por mais de dois segundos, a página rola para baixo. Caso esteja a menos de 20 pixels de altura da posição de sua base por dois segundos, a página rola para cima, senão fica sem rolar. A Figura 18 mostra essas três posições: rolar para cima, não rolar e rolar para baixo; sendo que em vermelho está destacado a base do

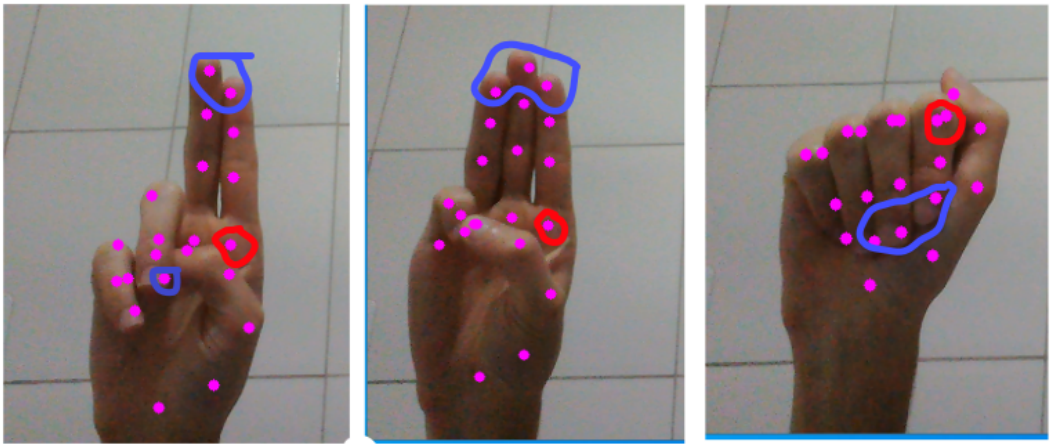
dedinho e em azul a ponta, cujas coordenadas são utilizadas no algoritmo.

Figura 18 – Posições do dedinho

Fonte: Autor (2021).

A opção *Dois dedos (duplo clique), três dedos (arrastar)* permite efetuar um duplo clique com o botão direito se os dedos indicador e médio estiverem a menos de 20 pixels de altura da base do indicador. Ainda nessa opção, caso os três dedos (indicador, médio e anelar) estejam menos de 20 pixels de altura da base do indicador, então o botão esquerdo é acionado e só desativa após todos os dedos abaixarem, permitindo selecionar um conteúdo. A Figura 19 mostra três posições possíveis, a primeira mostra dois dedos para cima (duplo clique), a segunda três dedos (arrastar) e a terceira não faz nada; sendo a marcação em vermelho a base do indicador e em azul as pontas dos dedos.

Figura 19 – Dedos para clicar e arrastar



Fonte: Autor (2021).

A sexta opção (*Cliques com a orientação da palma da mão*) permite efetuar cliques rotacionando a mão e mantendo-a em uma mesma posição por 1.5 segundos. Essa funcionalidade apresenta funcionamento robusto e de fácil uso, permitindo uma interação ergonômica e dispensando a localização dos dedos. Ambas opções só

funcionam quando uma mão aparece na tela. A Figura 20 mostra os pontos utilizados e as possíveis posições, clique esquerdo e direito, respectivamente, sendo em vermelho o ponto 0 e em azul o ponto 1.

Figura 20 – Rotação da mão

Fonte: Autor (2021).

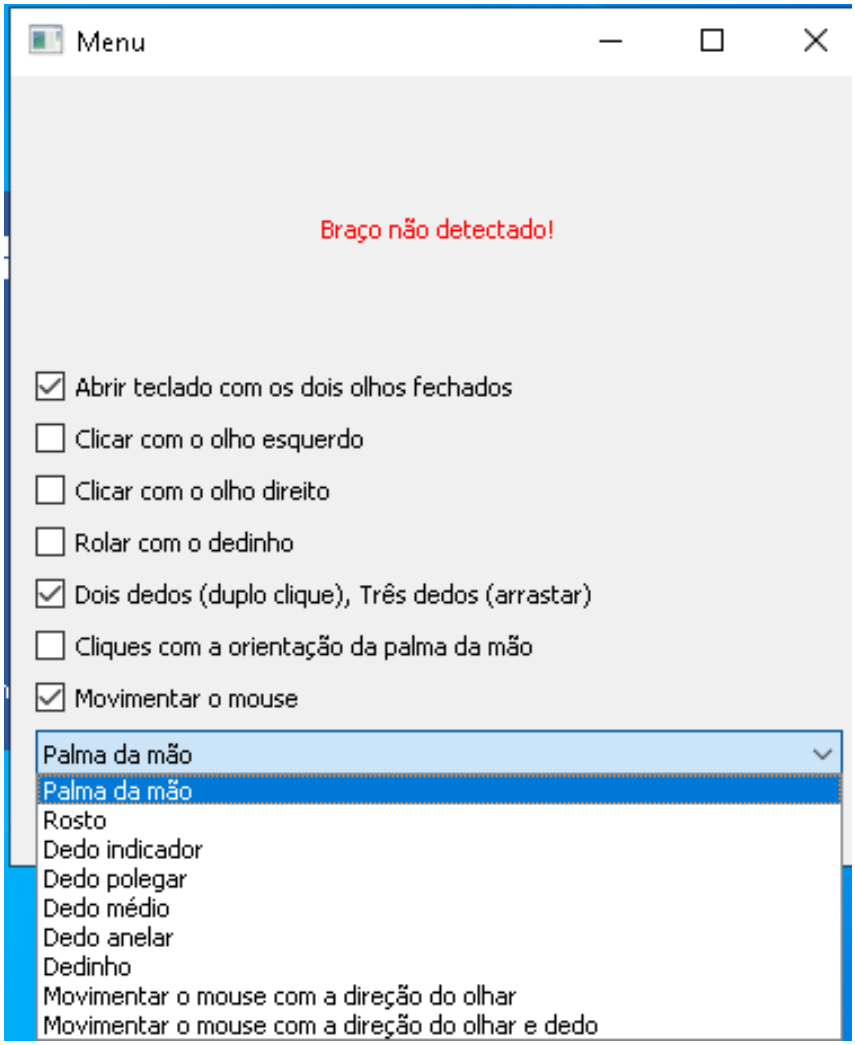
O último checkbox permite habilitar uma lista de opções para movimentar o mouse (Figura 21), cada uma dessas opções permite movimentar o cursor do mouse de uma forma diferente. As opções *Palma da mão*, *Dedo indicador*, *Dedo polegar*, *Dedo médio*, *Dedo anelar* e *Dedinho* permitem controlar o ponteiro por meio dos pontos 0, 8, 4, 12, 16 e 20 da mão da figura 6, respectivamente. A Figura 22 mostra esses pontos destacados em vermelho, observa-se que cinco deles são pontas dos dedos e um fica na palma da mão.

Essas funcionalidades funcionam de maneira similar, alterando a coordenada que é enviada a função que movimenta o cursor, sendo cada uma da respectiva seleção do usuário. Na função que move o ponteiro (função *move_mouse* da classe Mouse.py), a coordenada do ponteiro é proporcional a coordenada recebida em relação às dimensões da imagem com 80 pixels a menos da largura e 70 pixels a menos da altura. Assim o usuário poderá movimentar até os cantos da tela sem sair do campo de visão da câmera.

A opção *Rosto* move o cursor de acordo com as coordenadas do ponto 34 da Figura 7 do nariz da pessoa (Figura 17, ponto azul); essa opção captura a coordenada do nariz e envia para a função *move_mouse* da classe Mouse.py, que é a mesma utilizada nas funcionalidades anteriores. Mover o cursor com o rosto é pouco ergonômico, apesar dessa opção operar corretamente, pois é preciso movimentar muito a cabeça e o corpo. A opção *Movimentar o mouse com a direção do olhar* movimenta o ponteiro de acordo com a posição da pupila do olho. Com um retângulo determinando as possíveis posições da pupila na íris, é calculada uma posição proporcional na tela relativa a da pupila na íris.

Assim como a escolha anterior, a opção *Movimentar o mouse com a direção do olhar e dedo* também movimenta o cursor de acordo com a direção da pupila, usando a mesma função anterior, mas também permite um ajuste fino com o dedo indicador da mão da pessoa. Essas duas últimas opções são as menos estáveis da aplicação,

Figura 21 – Lista de opções



Fonte: Autor (2021).

possuindo uma alta taxa de erro acerca da detecção da pupila, o que faz com que o ponteiro fique instável e não se direcione para posição desejada.

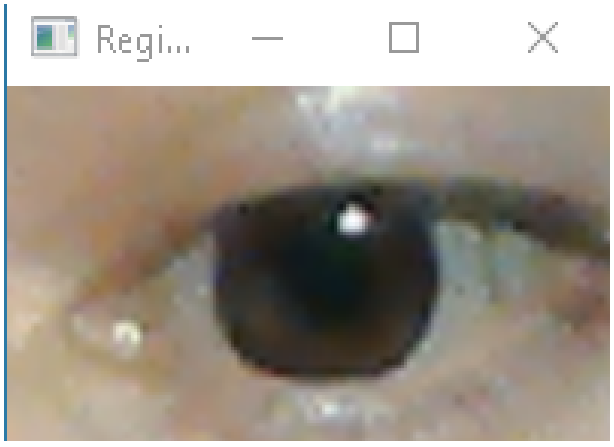
Em quase toda a aplicação o funcionamento de movimentar o ponteiro é feito a partir da posição relativa do ponto na imagem em questão, sendo que há um retângulo delimitando uma borda interna menor que a total, para evitar perdas de detecção quando o usuário tenta atingir as bordas da tela. Nas opções de movimentar o cursor com os olhos há uma delimitação de uma região de interesse no olho esquerdo (Figura 23), que permite capturar uma imagem estabilizada do olho e efetuar cálculos melhores sobre a direção da pupila.

A partir dessa região, a imagem é convertida para o espectro de cinza e um desfoque gaussiano é aplicado. Isso é feito para minimizar o efeito dos ruídos e das bordas irrelevantes. Em seguida é aplicado um limiar que torna os pixels mais escuros brancos e mais claros pretos. Assim a iris é identificada em branco, Figura 25, sendo

Figura 22 – Pontos da mão utilizados para movimentar o cursor

Fonte: Autor (2021).

Figura 23 – Região de interesse do olho usada no algoritmo



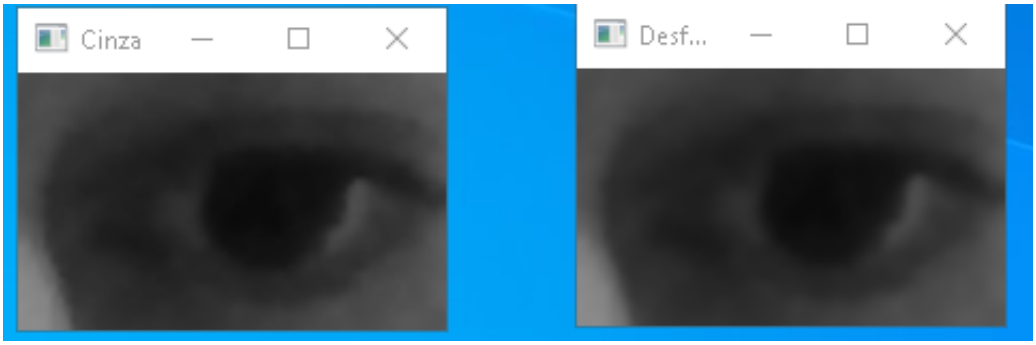
Fonte: Autor (2021).

que esse mecanismo é sensível a variações de iluminação, necessitando de alteração para operar em outros ambientes e alcançar bons resultados.

6.1 ANÁLISE DO FUNCIONAMENTO

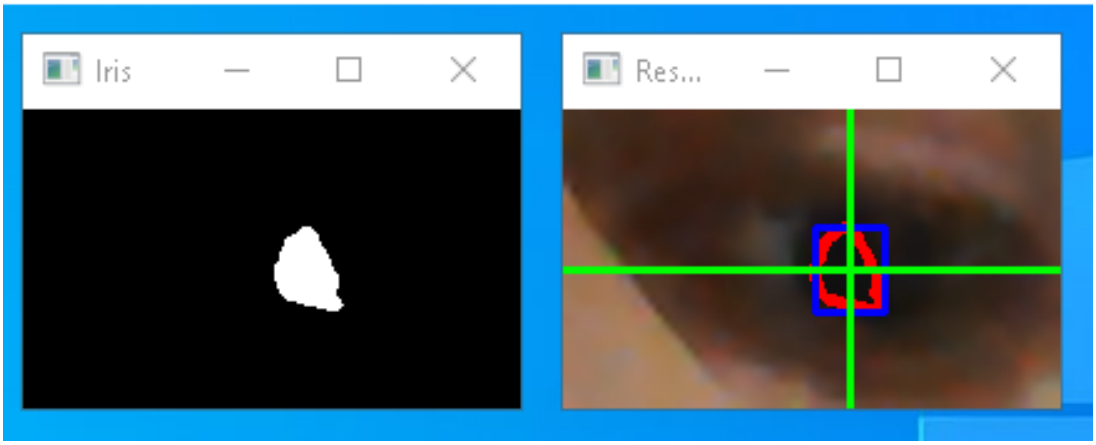
Abaixo serão analisadas as funcionalidades implementadas na aplicação, todas as opções precisariam ter os parâmetros adaptados caso fosse executada utilizando

Figura 24 – Olho na escala de cinza e com desfoque gaussiano



Fonte: Autor (2021).

Figura 25 – Pupila detectada em branco e marcações finais



Fonte: Autor (2021).

outra câmera. Além disso, cada funcionalidade para ser executada precisa ser seleciona no checkbox, caso seja uma opção que necessite de dados do rosto ou mão, o usuário terá que dispor o mesmo na direção da câmera e dentro de sua região de abrangência.

- Abrir o teclado com os dois olhos fechados: após selecionar essa opção, o teclado abre se o usuário fechar o olho por pelo menos três segundos, porém a distância e o ângulo do rosto da pessoa da câmera podem interferir na detecção. Essa funcionalidade exige um pouco de treino do usuário para fechar o olho apropriadamente, tendo o mesmo que manter o rosto na direção da câmera e a uma distância de 30cm para minimizar os erros de ruídos e limitações de detecção em rotação dos frameworks utilizados. Em operação, essa opção resulta em algumas tentativas falhas e um tempo maior com os olhos fechados, pois assim é mais fácil do usuário adequar o fechamento do olho;
- Clicar com o olho esquerdo: essa funcionalidade permite clicar o botão esquerdo do mouse fechando o olho esquerdo por 1.5 segundos (um clique) e 3 segundos (duplo clique). Nessa opção o usuário também precisa manter o rosto na direção

da câmera e a uma distância de 30cm, apresentando as mesmas observações, quando em operação, do item anterior;

- Clicar com o olho direito: essa opção permite clicar o botão direito do mouse fechando o olho direito por 1.5 segundos. Essa funcionalidade apresenta as mesmas observações do item anterior. Quando em uso conjunto com pelo menos uma das opções acima, observa-se que há uma interferência na detecção do status do outro olho, pois, fechar apenas um olho sem interferir na taxa de abertura do outro é uma tarefa fatigante;
- Rolar com o dedinho: essa função permite rolar a página posicionando a ponta do dedinho 20 pixels abaixo de sua base ou 20 pixels acima. Essa função é robusta, de fácil aprendizado e adaptação;
- Dois dedos (duplo clique), três dedos (arrastar): essa funcionalidade possibilita efetuar um duplo clique caso apenas os dedos indicador e médio estejam com sua ponta 20 pixels acima de sua base. Caso os três dedos (indicador, médio e anelar) tenham suas pontas 20 pixels acima da base, então a seleção do mouse fica acionada e só libera quando todos os dedos estiverem para baixo;
- Cliques com a orientação da palma da mão: essa função permite clicar rotacionando a mão, sendo que para acionar os cliques é preciso se manter em uma das posições por 1.5 segundos ou três segundos, no caso do duplo clique. Essa opção permite uma interação fácil, sem muito esforço e é robusta a ruídos;
- Movimentar o mouse: para movimentar o ponteiro do mouse foram desenvolvidas nove opções, que serão abordadas abaixo:
  - Palma da mão: essa opção é estável e robusta permitindo atingir todos os pontos da tela com bastante facilidade, apesar de ter que posicionar a mão na frente do próprio rosto para atingir alguns pontos.
  - Rosto: essa função usa um ponto no nariz e também permite manusear o ponteiro com facilidade, porém o usuário tem que movimentar significativamente o rosto, podendo gerar fadiga se usado por muito tempo;
  - Dedo indicador, polegar, médio, anelar e dedinho: cada um desses dedos é uma opção disponível para o usuário escolher. Todos empregam o mesmo mecanismo de funcionamento e apresentam desempenho similar, sendo de fácil uso e robusto, mas é preciso colocar a mão na frente do rosto para atingir alguns pontos da tela, caso a câmera utilizada seja a do Notebook;
  - Direção do olhar: essa opção captura a posição relativa da pupila do usuário no olho e converte em um ponto proporcional na tela. Devido as limitações da rede neural usada, o rastreamento da pupila é instável, variando o tamanho e a localização da pupila, além de estar sujeito a mais erros caso a pupila esteja nos cantos. Além disso, caso o usuário rotacione o rosto, mesmo que

levemente, esse método se torna ineficaz, exigindo que o mesmo posicione o rosto sem rotação na direção da câmera, o que gera desconforto. Assim, essa funcionalidade não é utilizável;

- Direção do olhar e dedo: esse método adiciona à funcionalidade anterior a possibilidade de efetuar pequenas movimentações no cursor através do dedo indicador. Essa opção foi adicionada na tentativa de minimizar os grandes erros obtidos na implementação anterior, porém não foi suficiente, mantendo erros altos que também inviabilizaram o uso dessa opção.

Assim, esse trabalho atingiu os objetivos esperados, sendo as funcionalidades de mover o ponteiro do mouse com a direção do olhar as únicas não funcionais. Os demais métodos são utilizáveis, sendo que as funcionalidades que usam a mão para mover o cursor, cobrem a visão do usuário para atingir alguns pontos da tela. As funcionalidades de clicar fechando os olhos não são muito rápidas e pode haver interferência se o usuário rotacionar o rosto ou se distanciar da câmera. Ademais, esse trabalho oferece ao usuário meios adicionais de interagir com o computador, permitindo escolhas personalizadas e métodos variados.

7 CONCLUSÃO

Neste trabalho a interação humano computador, com aplicação de recursos da visão computacional, foi o tema central dos estudos, resultando em contribuições na revisão de conceitos da visão computacional e interação humano-computador, analise de trabalhos similares ao tema interação humano computador e no desenvolvimento de uma interface de interação que captura imagens da câmera do Notebook e converte os dados em comandos para o Notebook. Caso sejam detectados e as opções estejam selecionadas, a mão servirá para mover o cursor do mouse ou clicar, os olhos para mover o ponteiro ou para efetuar cliques.

A aplicação desenvolvida precisa ser iniciada por meios tradicionais, porém, após se iniciar, vem com opções selecionadas que ampliam os métodos de interação. Idealmente, o algoritmo deveria ser projetado para se iniciar junto com o sistema operacional, mas esse mecanismo não foi implementado, sendo recomendada como trabalho futuro. Observa-se que a distância ideal para o funcionamento do algoritmo é com o rosto a 30cm-40cm da câmera, para que as curvaturas formadas pelos pontos do olho possam representar devidamente o status dos olhos. A iluminação pode impactar o rastreamento da pupila usado nas funcionalidades de mover o mouse, sendo que mesmo corretamente ajustado pode resultar em falhas. Essa funcionalidade não alcançou um desempenho satisfatório e robusto, porém a aplicação não foi testada em equipamentos mais potentes (câmera), nem em ambientes melhor iluminados.

As demais funcionalidades (clicar com o olho e manipular o mouse com a mão), alcançaram os requisitos de estabilidade desejados, sendo de simples interação e fácil aprendizado. Na aplicação é possível selecionar as interações desejadas, assim o usuário é capaz de personalizar o uso da ferramenta. O manuseio do programa demandaria ajustes nos parâmetros para o ambiente do usuário, já que há múltiplas variações de iluminação, resolução da câmera e distâncias operacionais. Os arquivos desenvolvidos no projeto foram disponibilizados no link https://github.com/fabiovicenteoliveira/MouseControl.git.

Assim, esse trabalho contribui com o desenvolvimento de uma interface de acesso ao computador, que oferece meios alternativos ao uso do mouse e do teclado. Entretanto, existem possibilidades inexploradas na interação humano computador, que podem ser foco de trabalhos futuros. Muitas pesquisas podem ser feitas sobre os possíveis métodos com que uma pessoa consegue interagir com a máquina, podendo atender a diferentes usuários e a diferentes propósitos.

O desenvolvimento de interfaces para essas técnicas também são indispensáveis nos estudos futuros. Ter interfaces eficazes, robustas, ergonômicas,

adaptativas e de fácil aprendizado é essencial e definitiva na escolha dos usuários. A técnica de mover o ponteiro do mouse com a direção do olhar apresentada nesse trabalho, por não apresentar os resultados esperados, pode ser atualizada e aprimorada para atingir maior estabilidade. O fato da ferramenta precisar ser iniciada por meios tradicionais e oferecer poucas opções de movimentar o cursor no início, também pode ser objeto de estudos futuros, a fim de implementar o funcionamento junto ao início do sistema operacional e troca de opções por outras técnicas.

Por fim, é sugerido como trabalhos futuros avaliar a solução com diversos equipamentos e também com usuários, para determinar os melhores parâmetros para o funcionamento ou gerar uma sintonização de operação com o ajuste personalizado dos parâmetros de acordo com o usuário. A avaliação com outros usuários poderia ser útil também para determinar outras necessidades de interação com o mouse e teclado, como o uso de atalhos do teclado. Podem ser desenvolvidos atalhos únicos nessas novas soluções que simplificam o acesso do usuário, tornando a interação mais intuitiva e ágil. Recomenda-se também uma análise ergonômica das interações desenvolvidas neste trabalho que busca fazer uma avaliação antropométrica indicando a aplicabilidade deste projeto. Ademais, pode-se desenvolver interfaces focadas em atender pessoas que possuem deficiência, como paralisia nos dedos.

REFERÊNCIAS

AL-JABERI, L. A. Carbon fiber strengthening of geopolymer concrete wall panels with iron fillings. **International Journal of Engineering and Technology**, v. 7, p. 399–402, 2018.

ALMEIDA, F. B. de. **Sistema interativo baseado em gestos para utilização de comandos no computador**. 2013. Monografia (Curso de Engenharia de Software) - Faculdade UnB Gama, Universidade de Brasília, Brasília, 2013.

BAEKGAARD, L. Interaction in information systems - beyond human computer interaction. **Action in Language, Organisations and Information Systems**, v. 1, n. 1, p. 257–269, 2006.

BAUER, S.; BRUNSMANN, U.; SCHLOTTERBECK-MACHT, S. Fpga implementation of a hog-based pedestrian recognition system. **MPC Workshop**, v. 1, n. 1, p. 1–10, 2010.

BAUMER, B. S.; KAPLAN, D. T.; HORTON, N. J. **Modern data science with R**. 6000 Broken Sound Parkway NW, Suite 300: CRC Press, 2017.

BRADSKI, G. **Open source computer vision library**. 2021. Disponível em: https://github.com/opencv/opencv. Acesso em: 19 jul. 2021.

BRASPENNING, P. J.; THUIJSMAN, F.; WEIJTERS, A. J. M. M. **Artificial neural networks**: An introduction to ann theory and practice. Verlag Berlin Heidelberg: Springer, 1995.

BURGER, W.; BURGE, M. J. **Digital image processing**: an algorithmic introduction using java. Verlag Berlin Heidelberg: Springer, 2008. v. 10.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, p. 273–297, 1995.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: **Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. San Diego, CA, USA: IEEE, 2005. v. 1, p. 886–893.

DIX, A. et al. **Human computer interaction**. 3. ed. New Jersey: Prentice Hall, 2004.

FEI-FEI, L.; FERGUS, R.; PERONA, P. One-shot learning of object categories. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 28, n. 4, p. 594–611, 2006.

GABRYS, B.; LEIVISKA, K.; STRACKELJAN, J. **Do smart adaptive systems exist?**: Best practice for selection and combination of intelligent methods. Verlag Berlin Heidelberg: Springer, 2005. v. 173.

GARBIN, S. M. **Estudos da evolução das interfaces homem-computador**. 2010. Trabalho de Conclusão de Curso (Curso de Engenharia Elétrica com ênfase em Eletrônica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

GIRSHICK, R. B. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. **IEEE Conference on Computer Vision and Pattern Recognition**, n. 1, p. 580–587, 2014.

HEWETT, T. T. et al. **ACM SIGCHI curricula for human-computer interaction**. New York: Association for Computing Machinery, Inc., 1992.

HORPRASERT, T.; HARWOOD, D.; DAVIS, L. S. A robust background subtraction and shadow detection. **Computer Vision Laboratory**, v. 1, n. 1, p. 2–7, 2000.

JAMES, G. et al. **An introduction to statistical learning**: with applications in r. 2. ed. Verlag Berlin Heidelberg: Springer, 2021.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. **Electronic Markets**, v. 1, n. 1, p. 2–12, 2020.

JENSEN, O. H. **Implementing the Viola-Jones face detection algorithm**. Kongens Lyngby: Technical University of Denmark, 2008.

KARN, A. Artificial intelligence in computer vision. **International Journal of Engineering Applied Sciences and Technology**, v. 6, n. 1, p. 249–254, 2021.

KING, D. E. Dlib-ml: a machine learning toolkit. **Journal of Machine Learning Research**, v. 10, n. 1, p. 1755–1758, 2009.

LOWE, D. G. Object recognition from local scale-invariant features. In: **Proceedings of the SEVENTH IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION**. Kerkyra, Greece: IEE, 1999. v. 2, p. 1–8.

MAURI, C.; SOLANAS, A.; GRANOLLERS, T. Interactive therapeutic multi-sensory environment for cerebral palsy people. **International Federation for Information Processing**, v. 5727, p. 696–699, 2009.

MCCUDDEN, M. R. **Britannica student encyclopedia**. Berne, Switzerland: International copyright union, 2010.

MEDIAPIPE. **Mediapipe Hands**. 2020. Disponível em: https://google.github.io/mediapipe/images/mobile/hand_landmarks.png. Acesso em: 9 set. 2021.

MEDIAPIPE. **Mediapipe in python**. 2020. Disponível em: https://google.github.io/mediapipe/. Acesso em: 19 jul. 2021.

MORSHED, T. et al. Outcome of surgical resection of de quervain's stenosing tenosynovitis. **Clinical Practice**, v. 17, n. 3, p. 1482–1485, 2020.

PATEL, M.; JAISWAL, M. **Introduction to artificial intelligence**. Verlag Berlin Heidelberg: Red'Mac international press, 2021.

PETERS, J. F. **Foundations of computer vision**: computational geometry, visual image structures and object shape detection. Verlag Berlin Heidelberg: Springer, 2017. v. 124.

PYNPUT. **PYNPUT 1.7.3**. 2021. Disponível em: https://pypi.org/project/pynput/. Acesso em: 23 ago. 2021.

PYQT. **PyQt reference guide**. 2012. Disponível em: https://pypi.org/project/PyQt5/. Acesso em: 19 jul. 2021.

RASCHE, C. **Computer vision**: an overview for enthusiasts. 2019. Bucharest: Politechnic University. oct. 2019. Disponível em: https://www.researchgate.net/publication/336460083_Computer_Vision. Acesso em: 11 set. 2021.

REDMON, J. et al. You only look once: unified, real-time object detection. **ArXiv**, v. 02640, n. 1506, p. 1–10, 2016.

ROSEBROK, A. **Eye blink detection with OpenCV, Python, and Dlib**. 2017. Disponível em: https://www.pyimagesearch.com/wp-content/uploads/2017/04/facial_landmarks_68markup-768x619.jpg. Acesso em: 9 set. 2021.

SANTOS, V. et al. Mouse ocular para pessoas com movimentos limitados causados por problemas cervicais e/ou cerebrais. **Colloquium Exactarum**, v. 7, n. 2, p. 89–101, Jun 2015.

STAHL, B. C. **Artificial intelligence for a better future**: An ecosystem perspective on the ethics of ai and emerging digital technologies. Verlag Berlin Heidelberg: Springer, 2020.

SZELISKI, R. **Computer vision**: Algorithms and applications. Verlag Berlin Heidelberg: Springer, 2011. v. 7.

TRIGUERO, I.; GARCíA, S.; HERRERA, F. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. **Knowledge and Information Systems**, v. 1, n. 42, p. 245–284, 2015.

TRUYENQUE, M. A. Q. **Uma aplicação de visão computacional que utiliza gestos da mão para interagir com o computador**. 2005. Dissertação (Mestrado em Informática) - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

UIJLINGS, J. et al. Selective search for object recognition. **International Journal of Computer Vision**, v. 2, n. 104, p. 154–171, 2013.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. **Computer Society Conference on Computer Vision and Pattern Recognition**, v. 1, n. 1, p. 511–518, 2001.

XU, W. et al. **Advances and applications in deep learning**: Advances in convolutional neural networks. London: IntechOpen, 2020.

YAN, W. Q. **Introduction to intelligent surveillance**: surveillance data capture, transmission, and analytics. 3. ed. Verlag Berlin Heidelberg: Springer, 2019.

YAP, M. H.; BISTER, M.; EWE, H. T. Gaussian blurring-deblurring for improved image compression. DICTA, Macquarie University, Sydney, Australia, v. 7, p. 165–174, 2003.

ZARGAR, S. A. Introduction to convolutional neural networks. **Department of Mechanical and Aerospace Engineering,**, v. 1, n. 1, p. 1–7, 2021.

ZHANG, P. et al. Amcis 2002 panels and workshops i: human-computer interaction research in the mis discipline. **Communications of the Association for Information Systems**, v. 9, n. 1, p. 334–355, 2002.