Ricardo Luiz Cardoso Menezes

Comunicação usando *Eye Tracking* para pessoas com paralisia motora e vocal

São Carlos

2020

Ricardo Luiz Cardoso Menezes

Comunicação usando *Eye Tracking* para pessoas com paralisia motora e vocal

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Universidade de São Paulo

Escola de Engenharia de São Carlos

Orientador: Prof. Dr. Alberto Cliquet Junior

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Menezes, Ricardo Luiz Cardoso
M541c Comunicação usando eye tracking para pessoas com paralisia motora e vocal / Ricardo Luiz Cardoso Menezes; orientador Alberto Cliquet Junior. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. eye tracking. 2. paralisia. 3. comunicação. 4. reabilitação. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Ricardo Luiz Cardoso Menezes

Título: "Comunicação usando Eye Tracking para pessoas com paralisia motora e vocal"

Trabalho de Conclusão de Curso defendido e aprovado em 1 / 12 / 2020,

com NOTA 9,0 (nove , zero), pela Comissão Julgadora:

Prof. Titular Alberto Cliquet Junior - Orientador - SEL/EESC/USP

Dr. Renato Varoto - UNICAMP

Mestre Renata Manzano Maria - Doutoranda - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado aos amigos, colegas e professores que muito me ajudaram durante esta jornada, sem eles, não teria chegado até aqui.

Agradecimentos

Agradeço a todos que possibilitaram que isso acontecesse. Primeiramente agradeço à minha mãe por me dar condições de realizar uma graduação tão longe de casa, criando-me para ter fortitude emocional para aproveitar os bons momentos, e suportar os maus.

Em seguida agradeço aos colegas que me auxiliaram durante toda a graduação, seja ajudando nos estudos ou nos trabalhos. Me ensinaram que juntos com o mesmo propósito, chegamos mais longe.

Finalmente, mas não menos importante, agradeço à todos os professores que, com muito afinco, ensinaram a mim e meus colegas coisas que sozinhos, nunca compreenderíamos, permitindo almejar caminhos anteriormente impensáveis.

"Se vi mais longe foi por estar de pé sobre ombros de gigantes."
Sir Isaac Newton

Resumo

Diferentes origens já foram mapeados para a síndrome de encarceramento, onde o paciente tem seus movimentos extremamente debilitados, muitas vezes restando apenas o movimento dos olhos. Nesse trabalho foi proposto um assistente de comunicação que utiliza apenas os olhos apresentando baixo custo em comparação a outras alternativas do mercado. Para tal foi utilizado *python* somado a *OpenCV*, *dlib* e *pygame* usando uma *webcam* para desenvolver a solução. Com o projeto concluído foi possível verificar que a aplicação base é funcional, entretanto não apresenta alta precisão.

Palavras-chave: *eye tracking*. paralisia. comunicação. reabilitação.

Abstract

Different origins have already been mapped for the locked-in syndrome (LIS), where the patient has his movements extremely restricted, often leaving only the eye movements. In this work, a communication assistant was proposed that uses only the eyes while remaining low cost compared to other alternatives in the market. For this, python was used with to OpenCV, dlib and pygame using a webcam to develop the solution. With the completed project it was possible to map some problems, but the basic application is functional, even if not with high precision.

Keywords: eye tracking. paralysis. communication. rehabilitation.

Lista de ilustrações

Lista de abreviaturas e siglas

API	*Application Programming Interface*
gTTS	*Google Text-to-Speech*
GUI	*Graphical User Interface*
HD	*High Definition*
LIS	*Locked-in Syndrome*
OpenCV	*Open Source Computer Vision Library*
PCCR	*Pupil-Center Cornal-Reflection*
TTS	*Text to Speech*

Sumário

1 Introdução

Atualmente é sabido que existem diversas complicações que limitam as habilidades motoras de pacientes, podendo chegar ao extremo onde não é possível se comunicar por nenhuma outra forma que não o movimento dos olhos, mesmo estando o paciente consciente e com funcionalidades cognitivas normais[1]. Essa condição é conhecida como síndrome de encarceramento ou, do inglês, *locked-in syndrome (LIS)*[2], sendo as causas dessa relacionados a danos causados por diversas fontes, dentre elas contusões, acidentes vasculares cerebrais, entre outros[3].

Afim de criar uma forma de contornar o problema da comunicação limitada que este tipo de paciente tem, foi proposto pelo professor Alberto Cliquet Jr durante a disciplina SEL0395 - Introdução à Engenharia de Reabilitação, que eu e meu grupo desenvolvêssemos alguma forma de comunicação para esses pacientes.

Durante a disciplina foi decidido pela utilização de uma câmera ligada a um computador para realizar tal projeto. Pós desenvolvimento não foi possível obter um resultado satisfatório, mas ainda se mostrou um interessante desafio.

Para o desenvolvimento do projeto da disciplina foram utilizados métodos de detecção de círculos, afim de identificar a pupila, somando ao processo um tratamento de cor, mas tendo em vista a alta taxa de falsos positivos gerados no ambiente o processo não se mostrava efetivo.

Somado a isso, mesmo quando a detecção da pupila ocorria com sucesso a imagem era muito dependente da não movimentação do conjunto câmera-paciente[4]. Conseguindo manter o sistema imóvel ainda foi necessário realizar um processo de calibração, o qual tinha em si seus problemas[5].

Em produtos comerciais de *eye tracking* se mostra comum o uso de infravermelho, utilizando o método da reflexão corneal do centro da pupila, do inglês *Pupil-Center Cornal-Reflection (PCCR) Method*[4], sendo esse tipo de tecnologia utilizada também para fins não médicos, com produtos como o *Tobii*, que possui produtos focados para a área de *gaming*, tecnologia assistiva, pesquisa sobre comportamento humano e integração tecnológica[6]. Mas tendo em vista que um dos objetivo do projeto era o baixo custo, buscou-se por outras saídas, que possibilitavam o uso de *webcams*.

Tendo em vista o aprendizado gerado pelo projeto e o resultado final não satisfatório, propus ao professor Alberto Cliquet Júnior a retomada desse projeto, mas dessa vez como um trabalho de conclusão de curso, tentando utilizar-me de um diferente método de detecção a partir de *machine learning* com a biblioteca *dlib*, afim de detectar pontos de

interesse no rosto, que gerou resultados mais satisfatórios.

A fim de haver uma interface de interação com o usuário foi criado também uma janela com sugestões de falas para o usuário, as quais, caso selecionadas seriam ditas pelos alto falantes do computador utilizando uma *api* de *text to speech (TTS)*, conseguindo então quebrar a barreira da comunicação, mesmo que de forma limitada.

2 Objetivos

2.1 Objetivo principal

Este trabalho de conclusão de curso objetiva criar um sistema de assistência para pessoas com limitação motora e vocal de baixo custo, permitindo a seleção de falas por meio do movimento dos olhos, estando essas apresentadas em uma tela e podendo ser emitidas de forma sonora por um sistema de *text to speech*.

2.2 Objetivos específicos

Para alcançar esse objetivo principal, é necessário o cumprimento dos seguintes objetivos específicos:

- Detectar o(s) olho(s) do usuário por meio de uma *webcam*;
- Detectar o movimento do olho;
- Conseguir distinguir diferentes movimentos dos olhos;
- Criar uma *graphical user interface (GUI)* com a lista de falas a serem selecionadas pelo usuário;
- Selecionar falas a partir do *input* do usuário dentre as apresentadas na *GUI*;
- Reproduzir a fala da frase selecionada;

2.3 Objetivo secundário

Com o sistema desenvolvido tem-se como objetivo secundário a avaliação da possibilidade de utilizar o projeto em uma forma mais móvel, usando, por exemplo, uma *raspberry pi*.

3 Desenvolvimento

Ao longo desse capítulo serão apresentados os materiais, ferramentas e métodos utilizados ao longo do desenvolvimento do projeto, sendo em seguida apresentado os resultados obtidos.

3.1 Materiais e métodos

3.1.1 *Webcam*

Tendo em vista o desejo por maior acessibilidade do projeto, foi utilizado durante o projeto uma logitech c922 *pro stream*, apresentada na figura 1, por questões de disponibilidade e resolução, além do fato dessa possuir um tripé, o qual facilita o desenvolvimento e os testes.

Figura 1 – *Webcam* logitech c922 *pro stream*

Fonte: Autor(2020)

Durante o desenvolvimento foram testadas diferentes resoluções de entrada a fim de testar se a resolução menor de outras *webcams* seria um limitante, mas não houve uma perda de qualidade perceptível mediante tais ajustes, mantendo a resolução acima de 640x480 pixels, sendo a maior resolução testada 1920x1080 pixels.

3.1.2 Ferramentas computacionais

Tendo em vista que o projeto faz uso de detecção de imagem, GUI e TTS decidiu-se pelo uso da linguagem *Python*, em razão da sua versatilidade e velocidade de desenvolvimento.

A fim de se capturar a imagem da *webcam* e realizar tratamentos na imagem capturada, foi utilizada a biblioteca *Open Source Computer Vision Library (OpenCV)*, que possui em si diversas ferramentas de visão computacional e *machine learning*[7].

Outro motivo da escolha por essa biblioteca vem da sua confiabilidade, tendo em vista seu vasto uso no mercado e por estar presente em produtos de grandes empresas, como *Google*, *Yahoo*, *Microsoft*, *Intel*, entre outras[7].

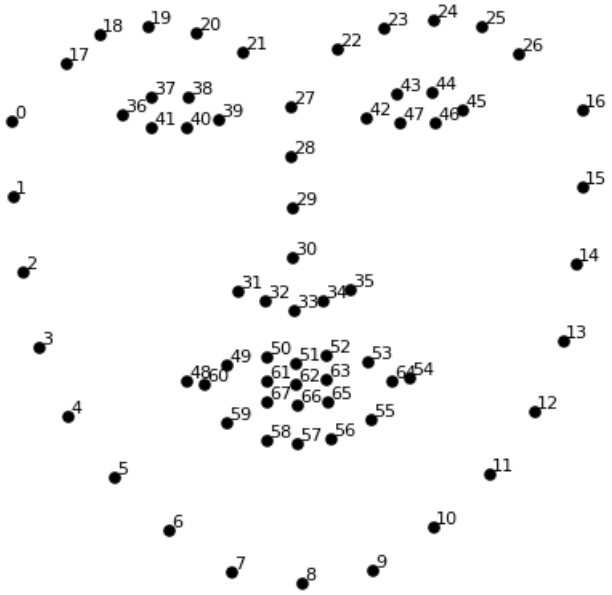
Figura 2 – *OpenCV* logo

Fonte: https://opencv.org/ (2020)

Outra biblioteca utilizada também por sua interação com imagens foi a *dlib*, que é um biblioteca escrita em *C++*, mas compatível com diversos sistemas. Suas principais funcionalidades giram ao entorno de *statistical machine learning*, mas ela também possui ferramentas de *networking*, *theads*, interface gráfica, álgebra linear, processamento de imagem, entre outros[8].

O foco de uso dessa biblioteca deu-se nesse último elemento, o processamento de imagens, mais especificamente na sua capacidade de identificação, feita por meio de um modelo pré-treinado que consegue detectar 68 pontos de interesse na face humana, apresentados na figura 3.

Figura 3 – *68 facial landmarks*



Fonte: Feng et al., 2020. Using Eye Aspect Ratio to Enhance Fast and Objective Assessment of Facial Paralysis.

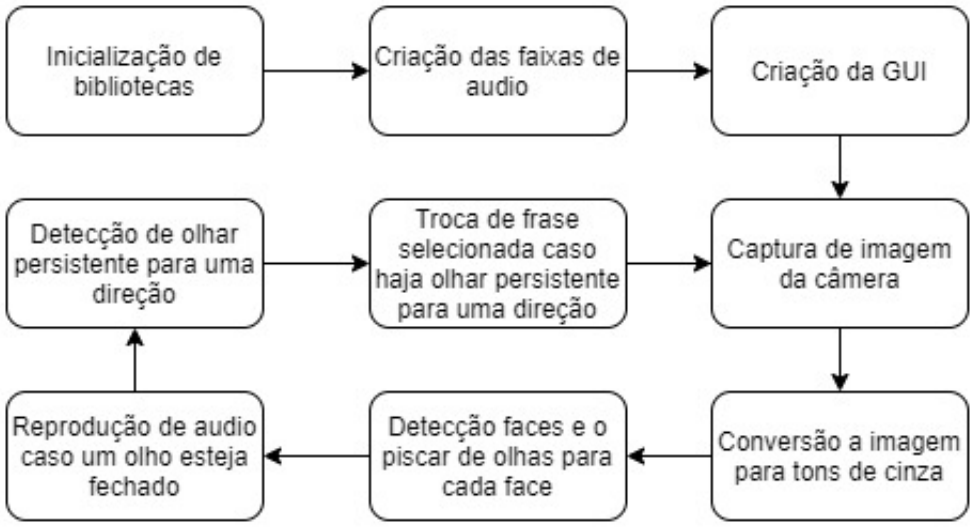
Para se desenvolver uma GUI foi utilizada a biblioteca *Pygame*, principalmente por seu fácil desenvolvimento, tendo em vista o design imaginado, além da rápida configuração de diferentes elementos de temporização e portabilidade para diversas plataformas[9].

Finalmente, a fim de se gerar versões faladas das frases foi utilizada a *Application Programming Interface (API)* da Google, a *Google Text-to-Speech (gTTS)*, que cria áudios no formato *.mp3* a partir de *strings*.

3.1.3 Aplicação

A aplicação em si é separada em duas partes, a inicialização e o laço de aplicação, seguindo o diagrama apresentado na figura 4. Durante a inicialização é realizada a configuração dos elementos a serem utilizados. As frases a serem apresentadas e lidas são separadas e, a partir delas utilizando o gTTS, são gerados os áudios a serem reproduzidos.

Figura 4 – Diagrama de funcionamento



Fonte: Autor (2020)

Em seguida é criada uma captura a partir da entrada da *webcam* e são inicializados os processos de detecção de face com os preditores dos 68 pontos, já apresentados na figura 3.

Finalmente é realizada a inicialização da biblioteca do *pygame*, que nos permite a criação de uma janela e as primeiras frases a serem apresentadas, sendo essas escritas em conjuntos de três, como apresentado na figura 5.

Estando o estado inicial devidamente inicializado, é prosseguido então para o seguimento que será executado em *loop* durante a aplicação. No começo do *loop* é capturado uma imagem da câmera (um *frame*), sendo essa imagem convertida para a uma escala de tons de cinza, uma vez que não são necessárias as diferentes cores para realizar a identificação.

Em seguida é utilizado o detector de faces da biblioteca dlib na imagem, sendo possível listar múltiplas faces. Para cada face identificada é utilizado mais uma vez a biblioteca dlib para identificar os 68 pontos de cada face.

Utilizando os pontos entre 36 e 47 é possível calcular o tamanho de cada olho identificado, calculando a largura do olho pela distância entre os pares de pontos 36 e 39 e 42 e 45. Para se calcular a altura de cada olho, ou a abertura de cada olho, inicialmente são calculados os pontos médios dos pontos altos de cada olho (que seriam os pares 37 e 38 para o olho direito e 43 e 44 para o olho esquerdo), realizando esse processo também para os pontos baixos de cada olho (40 e 41 para o olho direito e 46 e 47 para o olho esquerdo).

Com o ponto alto e ponto baixo médios de cada olho é possível medir a abertura dos mesmos, podendo então aferir se esses estão abertos ou fechados ao se observar a relação entre abertura e largura de cada olho.

Para garantir que o olho está de fato fechado em contraste de ser apenas uma medição com algum erro é estabelecido um contador de permanência, sendo necessário a detecção de um olho fechado se manter por ao menos 5 ciclos de repetição do *loop* para ser validado. Esse comportamento ocorre individualmente para cada olho.

Em seguida é analisado se a iris mais a pupila está na esquerda, no centro ou na direita do olho. para tal é traçada uma região ligando os 6 pontos que contém um olho (esse calculo é feito individualmente para cada olho).

Com a região traçada, é utilizada a mesma como máscara, tornando preto tudo que estiver fora dela nessa nova imagem criada, como mostrado nas figuras 7 e 8 em comparação com a figura 6.

Em seguida, a fim de analisar com maior precisão cada olho, são medidos os pontos extremos da região, realizando um recorte da imagem original tendo como resultado a imagem apenas dos olhos, como apresentados na figura 9.

Tem-se então uma imagem de um olho em preto e braco estando preto também toda a área fora do olho. Com isso é calculado a média dos valores apresentados na imagem, uma vez que ao estar em tons de cinza cada *pixel* é apresentado com um valor indo de 0 a 255, sendo 0 preto e 255 branco. Com o valor médio é traçado o *treshold* da imagem, sendo valores menores do que a média atualizados como 0 e valores igual ou maior que a média atualizados como 255, como mostrado na figura 10.

Finalmente, a imagem é divida ao meio entre a região a esquerda e a região a direita, sendo possível com isso somar os valores de cada região e, com a razão entre a soma do lado esquerdo e do lado direito, observar a direção em que o usuário está olhando.

Idealmente, caso o usuário estivesse olhando para um lado a razão seria maior que 1, para o outro menor que 1 e exatamente 1 caso estivesse olhando para frente, mas isso não ocorre por diferentes fatores, dentre eles a não simetria do olho humano, imprecisão da medição, questões fisionômicas individuais de cada pessoa, entre outros.

Para lidar com a não simetria dos olhos é realizada a média entre a razão calculada no olho esquerdo e no olho direito, mas para os outros elementos não há uma saída tão trivial para se contornar, resultando em *thresholds* diferentes a serem considerados para identificar a direção que diferentes pessoas estariam olhando.

Com a direção do olhar do usuário é realizado um procedimento semelhante ao do piscar de olhos para se garantir a direção que a pessoa está observando. Estando o usuário olhando para a esquerda, é selecionada a primeira frase dentre as listas, movendo-a para a posição central, levando antiga frase da posição central para baixo e apagando a antiga frase inferior. Uma nova frase é apresentada na primeira posição.

Isso ocorre por se considerar as frases apresentadas no arquivo de texto como *inputs*

cíclicos, então a última frase é posicionada antes da primeira fechando um *loop*. Caso seja identificado que o usuário esteja persistentemente olhando para a direita o mesmo ciclo de ações ocorre, mas no sentido inverso.

Caso seja detectado que o usuário esteja piscando com apenas um dos olhos (independente se é o olho esquerdo ou direito) é reproduzido o áudio da frase apresentada no meio da tela, alcançando então o objetivo da aplicação.

3.2 Resultados e discussão

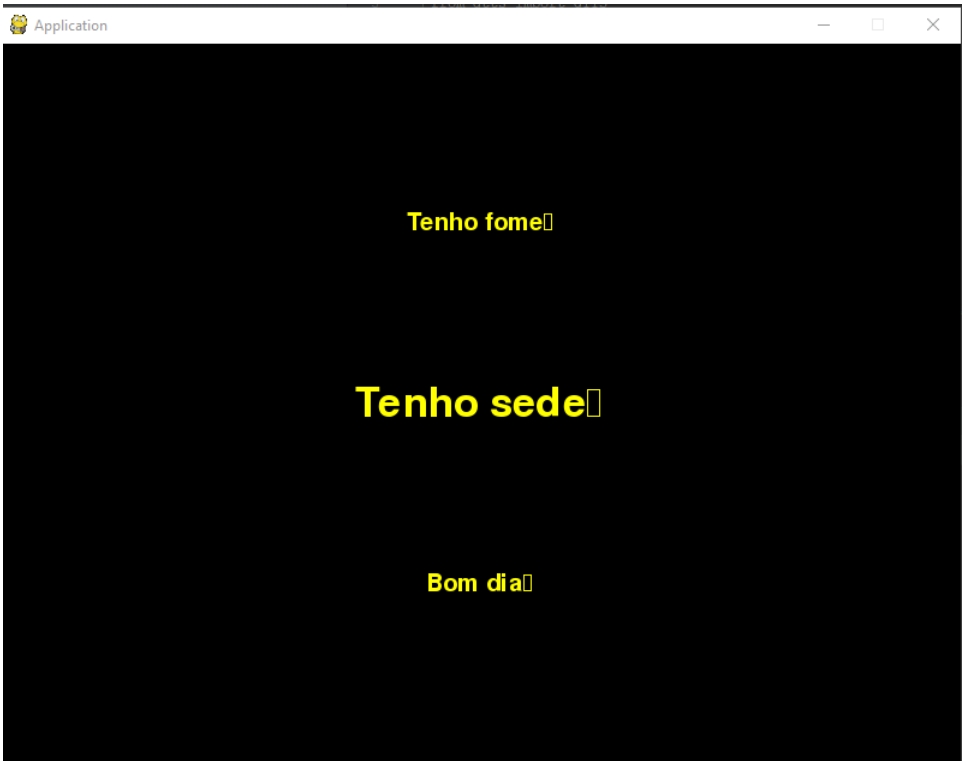
Nessa sessão será discutido o funcionamento final da aplicação e os problemas detectados.

3.2.1 Funcionamento

Com toda a aplicação feita é possível alcançar a funcionalidade desejada, conseguindo realizar o controle da ferramenta, a seleção e reprodução de falas utilizando apenas os olhos, mas sua detecção falta precisão, podendo o uso ser frustrante.

Durante o funcionamento é possível perceber a tela de aplicação, como mostrado na figura 5. Outras capturas possíveis são das de comparação da imagem obtida pela câmera e as imagens geradas após a aplicação das máscaras, como mostrado nas figuras 6, 7 e 8.

Figura 5 – Aplicação no *pygame*

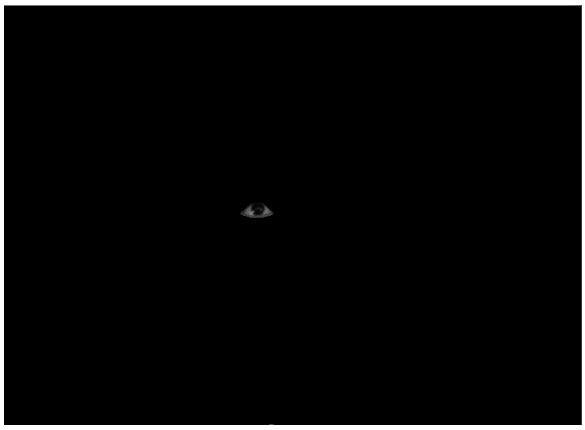


Fonte: Autor (2020)

Figura 6 – Imagem capturada antes da aplicação da máscara

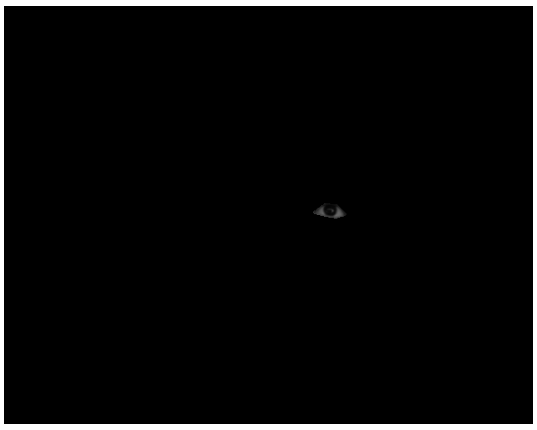
Fonte: Autor (2020)

Figura 7 – Máscara do olho direito



Fonte: Autor (2020)

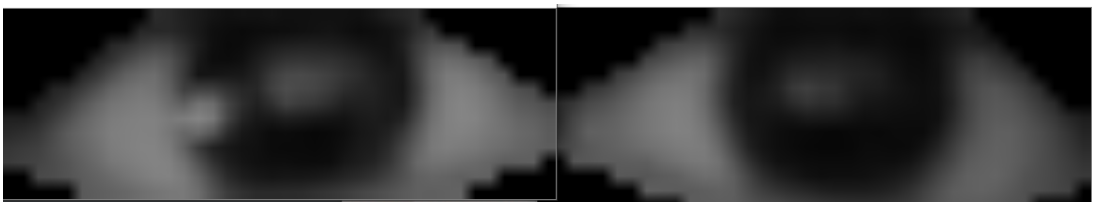
Figura 8 – Máscara do olho esquerdo



Fonte: Autor (2020)

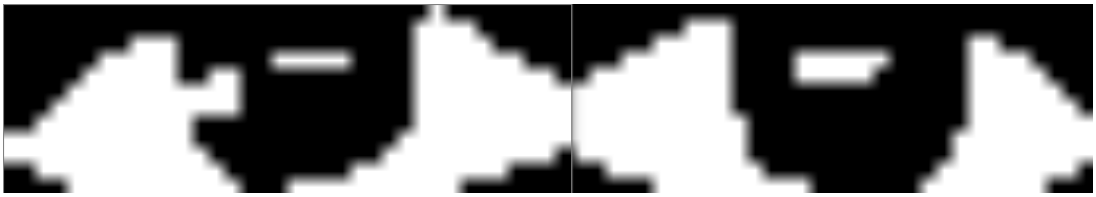
Ao retirar a área nula da máscara é possível ver os olhos em foco, como apresentado na figura 9, podendo ver o resultado do processo de *thresholding* na figura 10.

Figura 9 – Imagem formada pela região de cada olho lado a lado



Fonte: Autor (2020)

Figura 10 – Imagem formada pela região de cada olho lado a lado pós threshold



Fonte: Autor (2020)

Uma exemplificação do funcionamento comparando a imagem capturada e a *GUI* pode ser observada em video *hosteado* no *YouTube* no *link* a seguir: https://youtu.be/M5Ws3U6tQ9E.

3.2.2 Problemas detectados

Todos os problemas giram ao entorno de falhas de detecção, sendo essa advinda de diversas fontes.

3.2.2.1 Luz

Um problema presente em todo processo de *gaze detection* ou *eye detection* é a luminosidade no rosto do usuário[4]. Esse problema se mostra ainda mais impactante ao se utilizar uma *webcam* e trabalhar no espectro visível de luz.

Outro problema relacionado a luz que se mostrou muito impactante foi o reflexo da mesma nos olhos. Ao tentar suprir o problema da baixa luz ambiente utilizando iluminação artificial (tanto por meio de lâmpadas direcionadas diretamente no rosto ou utilizando o brilho de monitores) a detecção sofria uma grande perda de precisão, uma vez que o reflexo em muitas vezes se confundia com a esclera, dificultando a detecção da iris e da pupila.

3.2.2.2 Acessórios e fisionomia

Tendo em vista que para realizar a detecção é utilizada um rede treinada com faces apenas, uso de acessórios como óculos faz com que a detecção não ocorra corretamente, muitas vezes considerando a armação do óculos como o olho em si.

Finalmente, um dos elementos de mais complexa correção é o fato que para diferentes faces são necessários diferentes *thresholds*, tanto para a identificação do piscar de olhos como da identificação da direção do olhar. Para solucionar esse tipo de problema seria necessário a implementação de um processo de calibração e a validação da mesma, tendo em vista que o processo para de determinação dos *thresholds* atualmente usados foram diversas iterações de diferentes testes.

3.2.2.3 Limitações de hardware

Durante todo o desenvolvimento foi utilizada uma *logitech c922 pro stream*, que captura imagens em *high definition (HD)* (1920 por 1080 *pixels*) a fim de se ter uma melhor resolução na pequena região que é cada olho, havendo perda de precisão ao utilizar outras *webcams* com resolução inferior na identificação do piscar de olhos e da direção do olhar.

Elementos como a resolução espacial da *webcam* e a lente da mesma são as fontes das limitações, uma vez que para o melhoramento da imagem o objetivo seria obter o maior número de *pixels* por olho, sendo então relevante a resolução, a lente utilizada e a relação espacial entre a câmera e a face.

Uma vez que é utilizada a API gTTS, é necessário que o equipamento esteja conectado à internet, uma vez que a mesma utiliza funções já preparadas em seu *site* de tradução.

4 Conclusão

Analisando os resultados obtidos é possível dizer que, como *proof of concept*, o projeto foi um sucesso, tendo em vista que alcançou a aplicação final, já sendo possível mapear alguns problemas que uma possível versão comercial viria a enfrentar.

Tendo em vista que toda operação é realizada em *python* mostra-se possível portar o projeto para um microprocessador, como uma *raspberry pi*, sendo recomendado o uso de uma versão mais potente tendo em vista a captura e tratamento de imagem.

Infelizmente dado o cenário mundial de difícil mobilidade e acesso a alguns recursos não foi possível realizar durante o tempo de projeto o porte para uma versão móvel, mas já foi possível perceber pontos a serem adaptados.

Creio que em uma versão portátil seria uma ideia retirar a biblioteca gTTS, retirando a possibilidade de se customizar diferentes frases, mas permitindo o sistema operar sem a necessidade de acesso à rede, o que facilitaria com a mobilidade e segurança do equipamento.

Outra possível melhoria a ser implementada seria um estudo de uma possível calibração, tornado a aplicação mais inclusiva e garantindo o funcionamento em um maior grupo de pessoas. Além disso poderia ser estudado uma forma de melhor representar as falas disponíveis e a fala selecionada, sendo realizado um melhor estudo de usabilidade.

Em geral esse projeto foi de um grande aprendizado de uma aplicação direta de uma rede pré-treinada e como isso em muito acelera e melhora o desenvolvimento, ao comparar o resultado dessa versão com a anterior, em que tentou-se lidar com identificação de formas ou padrões de cores, sendo na época implementado um processamento mais complexo, mas menos efetivo.

Referências

[1] J. Duffy, *motor speech disorders substrates, differential diagnosis, and management*. Elsevier, 1995. Citado na página 21.

[2] C. H. Hawkes, ""locked-in"syndrome: Report of seven cases," *Br Med J.*, 1974 Nov 16. Citado na página 21.

[3] G. Bauer, F. Gerstenbrand, and E. Rumpl, "Varieties of the locked-in syndrome.," *J Neurol*, pp. 77–91, 1979 Aug. Citado na página 21.

[4] D. Cleveland, "Theory, practice, and standardization of eye-tracking technology." https://www.youtube.com/watch?v=wi19uS4JFJ4. Citado 2 vezes nas páginas 21 e 32.

[5] T. Santini, "Fast and unassisted eye tracker calibration for gaze-based pervasive human-computer interaction." https://www.youtube.com/watch?v=gYtF3j2Hdl4. Citado na página 21.

[6] Tobii, "Business units fields of use." https://www.tobii.com/group/about/business-units-and-fields-of-use/, 2020. Citado na página 21.

[7] "About opencv." https://opencv.org/about/. Acessado: 28/09/2020. Citado na página 26.

[8] "Dlib overview." http://dlib.net/intro.html. Acessado: 30/09/2020. Citado na página 26.

[9] "About pygame." https://www.pygame.org/wiki/about. Acessado: 02/10/2020. Citado na página 27.