EYE-TRACKING PARA O AUXÍLIO DE DEFICIENTES MOTORES NA UTILIZAÇÃO DE INTERFACES-WEB UTILIZANDO CURSOR

Ana Beatriz Freires Ferreira
15/0004851

Eduardo Araujo Batista
15/0008872

FGa - Faculdade do Gama
UnB - Universidade de Brasília

RESUMO

O seguidor de pupila possui diversas aplicações para facilitação de percepção ótica, sendo muito utilizado atualmente como um medidor de interesse e foco, por ser barato e com simples execução pelos usuários.

O projeto visa auxiliar portadores de deficiência motora a utilizar teclados e controles de cursor para uso de computadores pessoais.

Palavras Chaves - Eye Tracking, Curso, Teclado, Detecção de olhos, Pupila.

I. INTRODUCAO

A utilização do seguidor de pupilas se tornou mais empregável nos últimos anos, o eye tracking é uma técnica que vem logo após o rastreamento ocular e este por sua vez permite aferir a posição e o comportamento do movimento dos olhos. Muito utilizado em pesquisas para análise psicológica e linguística.

O mundo moderno está cada vez mais permeado de tecnologias, desde os tradicionais aparelhos eletrodomésticos, passando por gadgets que já são parte fundamental da vida cotidiana, até automatizações de processos tradicionalmente mecânicos, como clappers.

Apesar dessa aplicabilidade da tecnologia no dia a dia, elas são historicamente, aplicadas primeiro em contextos estratégicos, como industrial ou militar, para depois tornar-se acessível em produtos de uso geral. Tendo isso em vista, faz-se necessário manter atenção às tecnologias desenvolvidas e utilizadas nestes contextos específicos, afim de adiantar-se às tendências.

É justamente nesse contexto que aparece o seguidor de pupila (eye tracking), desenvolvido em contexto acadêmico, e com uso já consolidado em determinados setores, como o marketing e propaganda, ainda está nas primeiras aplicações popularizadas, como celulares nichados que já vem com esta funcionalidade embutida. Portanto, percebe-se a tendência da popularização do seguidor de pupila, e a necessidade de desenvolvimentos para esta aplicação.

Agora, se estendeu até o campo de consumo diário. Respondendo a perguntas como “O que chama mais atenção?” ou “O que é visto primeiro?” de maneira mais prática e cientificamente amparada.

O conhecimento com seguidor de pupilas requer o uso de câmeras adaptadas (de média resolução) para que seja possível captar movimentos oculares pequenos e a partir desses dados elaborar uma avaliação do comportamento sobre o movimento dos olhos, percebendo as características mais voluntárias do usuário.

Alguns fatores e variáveis podem ser quantificáveis e são envolvidos em relação ao Eye Tracking como a fixação visual, dilatação das pupilas, foco atencional e etc. Assim como sua abordagem e a utilização se modificou com o passar do tempo. Podendo ser usando para análise de websites, a interface e facilidade de navegação de um determinado site, testar a disposição de produtos dentro de uma loja, as prateleiras e a estética de determinado produtos, usando um mapa de calor é possível determinar a primeira coisa que chama atenção do usuário gradualmente.

II. JUSTIFICATIVA

Atualmente a detecção de partes do rosto está se tornando cada vez mais usual por conta de sua aplicação automatizada e simples.

O seguidor de olhos tem tido inúmeras aplicações no meio publicitário, para análise de interesse dos clientes e confirmação científica quanto ao que chama mais atenção quando exposto. Também tem sido usado para monitoramento no uso de aeronaves, interação do usuário com a interface proposta e etc.

Outro ponto que tem sido muito visto em eye-tracking é no auxílio de pessoas com algum tipo de deficiência motora. A instalação de uma câmera associada a uma programação eye-tracking dá mais autonomia para pessoas que não possuem mais os movimentos de mãos ou/e braços, tornando acessível o uso de um cursor movido pelos olhos do usuário. O software embarcado presente no projeto atenua ajuda de terceiros. Aumentando a independência e liberdade do usuário portador de necessidades.

III. OBJETIVOS

O objetivo do projeto é facilitar o uso de computadores pessoais para pessoas com algum nível de deficiência ou/e dificuldade motora. Associando um seguidor de pupila (eye-tracking) ao cursor do computador sendo utilizado como mouse e por meio de piscadas, sendo estas temporizadas para serem diferenciadas de piscadas habituais, indicar o objeto em que o cursor deseja que haja clique.

IV. REQUISITOS

Para o auxílio do usuário o sistema deverá adquirir imagem de vídeo de modo contínuo e processá-lo de modo a identificar e rastrear as pupilas, associando seus movimentos ao cursor do computador e quantificando o tempo entre as piscadas para que seja identificada a piscada de clique.

Usando vetores atrelados as pupilas do usuário, utilizando uma câmera para definir a coordenada do cursor e assim determinar o objeto de interesse na interface para que seja clicado e por fim armazenar os dados de imagem na placa eletrônica embarcada (Raspberry Pi 3).

A aquisição das imagens será utilizado um módulo de câmera V2 de 8mp, e seus dados processados no computador de placa única Raspberry Pi 3.

Diagrama 1: Funcionamento do projeto

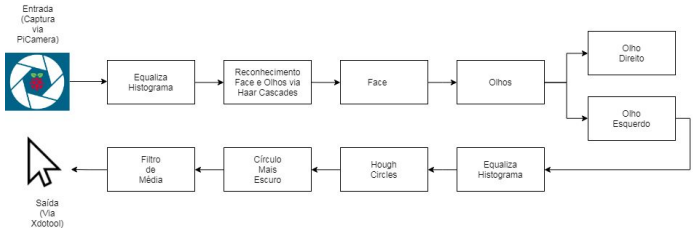

Diagrama 2: Funcionamento do Processamento de Imagem

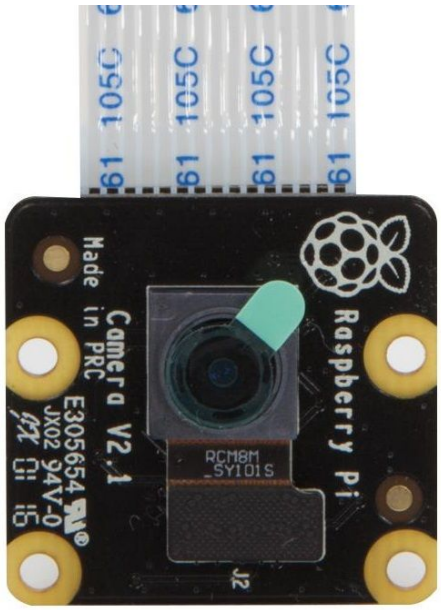

Figura 1: Câmera utilizada para o projeto

V. BENEFÍCIOS

Um dos benefícios no uso de seguidor de olhos está em sua aplicação barata. Apenas o uso de câmeras com resolução alta o suficiente para detectar pupilas não sendo necessário maiores detalhamentos quanto a bordas e

sombras, mas com resolução baixa o bastante para que não custe tanto.

A instalação da câmera em um computador será simples e acessível, o usuário não precisará intervir com nenhuma parte do processamento que será feito inteiramente pelo software embarcado em Raspberry.

Com a câmera instalada no computador o ambiente e a iluminação não serão um problema a ser considerado, já que será um ambiente controlado com uma luz contínua em relação a tela do computador do usuário.

Comparando o projeto de seguidor de pupila para utilização do cursor de um computador com mapa de calor nota-se a diferença de complexidade e velocidade de processamento. Um mapa de calor com alta qualidade requer um tempo de eixo de atenção específico que dure por mais tempo para dar o mesmo resultado que o da primeira aplicação. Se comparado a geometria de duas câmeras (uma voltada para os olhos do usuário e outra para a tela do pc e assim traçar um vetor entre a posição das duas câmeras), esta necessita de um óculos de apoio que ficaria na frente do rosto do usuário causando desconforto e menor aplicabilidade em alguns casos. O projeto possui uma câmera que ficará acima da tela do computador pessoal, logo, não obstruirá a visão do usuário em relação a interface.

VI. METODOLOGIA

A. *Bill of materials*

- Notebook Dell
- Raspberry Pi 3B
- Câmera V2.1
- C++

B. *Linguagem*

O código foi programado inteiramente em linguagem C++ com OpenCV instalado no mesmo para utilização de determinadas funções de processamento de imagens.

A utilização da linguagem de mais baixo nível aumentou a rapidez de processamento, proporcionou uma diminuição de *delay* entre o que era captado pela câmera, o que era exibido no terminal de execução e o que era determinado para o mouse a partir da posição da púpila.

Outra vantagem pode ser notada com a utilização de C++, esta é referente a funções como *xdotool.* Este controla mouse e teclado, para sua utilização foi configurado justando a uma função de system para que sua funções fossem preservadas ao longo do código descrito em C++.

C. *Identificação de face*

Para iniciar a interação homem-máquina, faz-se necessária a captura da imagem do usuário, realizada através da webcam do notebook, que é ativada através do algoritmo desenvolvido. Cada frame desta captura é a entrada no algoritmo que realiza o processamento em loop até o comando de interrupção. Foram capturados e identificados 30 frames para est epasso.

Para que a face seja detectada utiliza-se funções da ferramenta OpenCV, que utilizando uma *haar cascade* é capaz de fornecer as coordenadas da face no frame, essas bibliotecas são utilizadas para a definição de um retângulo ao redor da face, determinando a região do rosto. Depois de alguns testes para adaptar o melhor data seet para identificação de rostos foi estabelecido o uso do haar: *haarcascade_frontalface_alt.xml*

Cada frame capturado pela webcam passa por este processamento antes de ser exibido, gerando um vídeo com *face tracking* em tempo real.

Além disso, os parâmetros do tamanho do rosto foram definidos de modo a diminuir os falsos positivos, já que na condição de uso é esperado um tamanho mínimo de face no frame.

O resultado do código para face:

```
for (cv::Rect &face :face)
  {
    rectangle(frame, faces[0].tl() +face.tl(), faces[0].tl() +
face.br(), cv::Scalar(0, 255, 0), 2);
  }
```

rectangle plota um quadrado com os parâmetros definidos . Sendo utilizada uma lógica de for com incremento de i e lida a matriz formada por faces em x, y. Estes referentes ao posicionamento e identificação de rosto presente na entrada da câmera, sobre a largura do retângulo a ser plotado que varia para cada largura de rosto e sobre a altura que este retângulo necessitaria ter para abranger boa parte do rosto do usuário.

D. *Identificação dos olhos*

Ao definir a região do rosto no processamento anterior, executa-se um processo similar dentro dessa região. Com o mesmo processo, utilizando o OpenCV, mas com uma *haar cascade* específica para olhos, a cada frame define-se as coordenadas dos olhos e é traçado um retângulo em cada olho. Depois de alguns testes para adaptar o melhor

data seet para identificação de rostos foi estabelecido o uso do haar: *haarcascade_eye.xml*

O processamento é feito apenas na região dentro do rosto para minimizar os falsos positivos, haja vista que a possibilidade de haver dentro de uma região um falso positivo facial ainda e que também haja um falso positivo de olho é bem pequena, além de, graças ao dinamismo do algoritmo, falsos positivos são rapidamente corrigidos.

Para os olhos:

```
for (cv::Rect &eye :eyes)
  {
    rectangle(frame, faces[0].tl() +eye.tl(), faces[0].tl() +
eye.br(), cv::Scalar(0, 255, 0), 2);
  }
```

A lógica é muito parecida para a identificação de faces, tendo o olho tendo uma diferença em relação aos canais que estão sendo tratados. Enquanto no rosto só utiliza-se a entrada colorida, nos olhos foi feito um tratamento para que fosse obtido a entrada em tons de cinza, isso facilita o processo de identificação a partir dos cascades utilizados, para essa conversão para tons de cinza é utilizada a função *cvtColor*. Para plotar um retângulo foi utilizada a função *rectangle* com os parâmetros definidos. Sendo utilizada uma lógica de for com incremento de j e lida a matriz formada por olhos em x, y. Sendo referentes a coordenada dos olhos quanto a entrada, sobre a largura do olho e sobre a altura do olho na entrada.

D. Identificação de íris e pupila

Utilizando os parâmetros definidos no passo anterior foi utilizada a função *HoughCircles* presente no Open CV, essa função é responsável por detectar objetos circulares ao redor de uma imagem digital, utilizando características determinadas pela Transformada de Hough que utiliza de técnicas de extração de aspectos especificados. Para o Hough Circles elegesse por votação os objetos circulares dentro de imagens digitais imperfeitas e assim determina os círculos e os parâmetros destes com a essa função. Com isso foi utilizada a função gradiente para determinar as bordas de interesse dentro do parâmetro de olhos, que no caso serão as bordas da íris e da pupila. Dois parâmetros foram definidos para determinar o tamanho da íris e neste também incluía o raio do círculo determinado previamente pela função Hough Circles que é plotado para definir o tamanho da íris do usuário. Esses parâmetros foram convertidos em coordenadas polares para que o círculo definido fosse plotado corretamente.

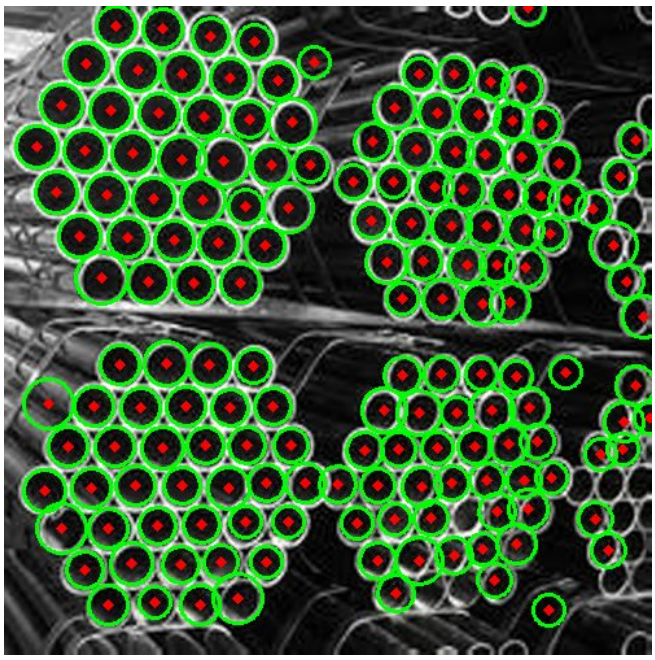

Figura 2: Hough Circles

Dentro do círculo localizado pela função de Hough foi utilizada uma função para detectar a pupila. Encontrando a pupila com maior acurácia, sendo a o centro da íris, o processamento de encontrar e estabelecer a região da íris (região de maior interesse) tornaria-se também mais eficaz e com maior precisão. Essa função identificava o círculo com menos energia, ou seja, com o maior número de pixels pretos presentes na área dos olhos anteriormente processados e demarcados, se esse círculo também tivesse uma forma mais regular circular delimita-se a pupila. Esse segundo processo foi necessário para que a função de Hough para encontrar a íris se torna-se mais estável e centralizada. Isso diminui a quantidade de erros relacionados a encontrar círculos dentro da parte recortada da imagem imagem referente aos olhos.

E. Controle do Mouse

Os princípios utilizados para movimentação do cursor seguem o básico de *gaze direction*, que é a definição da direção do olhar de uma pessoa através de métodos de processamento de imagens. Os resultados poderiam ser melhorados com a implementação de um algoritmo de gaze mais robusto, como os que utilizam filtro de Kalman, porém, ainda não houve esta implementação.

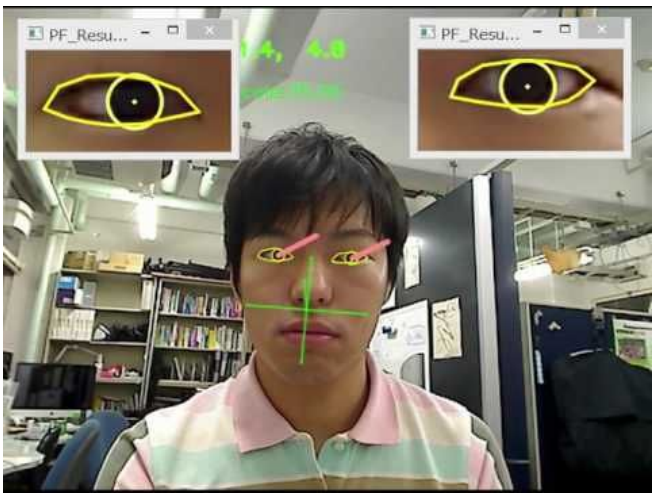

Figura 3: Gaze Direction

Alguns funções foram testadas para que o cursor do mouse pudesse ser controlado a partir dos processamento e identificação da íris. A que se adequou ao projeto foi a função *xdotool*.

Xdotool é uma ferramenta de funções que permite a emulação o clicar de um mouse ou o apertar botões de um teclado. Para o escopo do projeto foi utilizado para emular o cursor de um mouse a partir de variáveis. Variáveis estas que estão no eixo x e eixo y em relação a tela do computador.

No projeto foi aplicada com as variáveis responsáveis pela detecção de irís. O formato circular tem uma altura e largura posicionada sobre a tela -também possui um raio. Essa altura e largura foi utilizada para o xdotool a fim de que o cursor do mouse fosse controlado pelo posicionamento da íris. O comando para a função é:

```
system(("xdotool mousemove " + std::to_string(location.x) + " " + std::to_string(location.y)).c_str());
```

location é sobre o posicionamento do cursor, sendo x sobre eixo largura da posição da íris e y sobre eixo altura da posição da íris.

F. Resultados

Toda a programação utilizada foi efetuada na Raspberry Pi 3B, usando a IDLE para C++: Sublime.

Para que a identificação de pupilas se tornasse mais simples foi utilizado uma equalização de histograma na região dos olhos (dentro do retângulo amarelo). Com a função *equalizeHist*, esta função uniformiza a intensidade das cores presentes na entrada. Tornando o que é escuro mais escuro e o que é claro mais claro. Agindo com um auto contraste de processamento instântaneo. Alguns testes também foram feitos com *tresholding,* essa função elimina bit a bit da imagem de entrada, diminuindo a quantidade de tons presentes na imagem, chegando até a imagem binária e quando tratando de encontrar a pupila via câmera é mais eficaz que se utilize ela binária para que seja feita uma diferenciação entre a pupila (preto - 0) e o globo ocular (branco -255).

Um filtro de média foi aplicado dentro do processamento da identificação dos olhos para diminuir os ruídos e estabilizar a captura da imagem. Um filtro de média é feito a partir de uma média de um número de pontos do sinal da entrada, nesse caso a imagem capturada pela PiCamera, para produzir cada ponto do sinal de saída, o processamento final da imagem de entrada. Possui uma implementação recursiva e rápida ao contrário de outras alternativas que diminuiria o processamento de imagem, como a convolução.

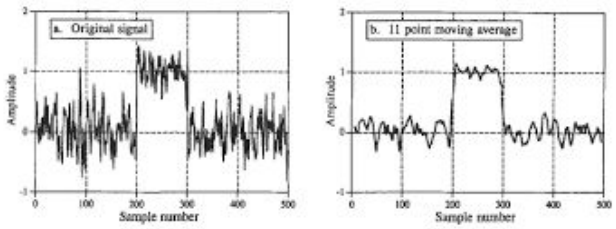

Figura 4: Filtro de Média Móvel

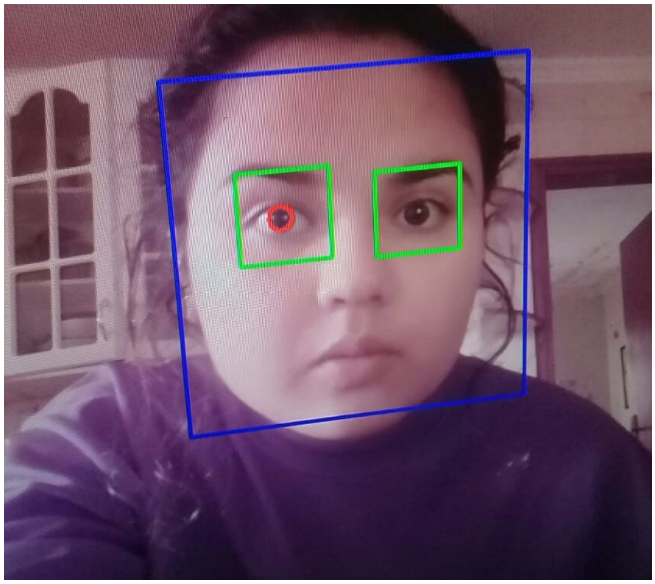
Figura 5: Identificação de face, olhos, íris e pupila.

Em relação a estrutura esperava-se utilizar a Raspberry Pi 0, no dia da apresentação a peça que segurava o cabo flat para câmera quebrou, não havendo como fazer a troca desse equipamento a tempo da apresentação final. As expectativas eram que ficasse como a seguir:

Figura 6: Raspberry Pi 0 na case oficial

Com esse contratempo uma adaptação foi feita, utilizando a case oficial para Raspberry Pi 0 foi conectada o cabo flat da Raspberry Pi 3B e por uma abertura da case foi conectada a mesma. Com isso foi possível apresentar uma câmera mais estável para o controle do mouse a partir de eyetracking.

Figura 7: Protótipo final

VII. REVISÃO BIBLIOGRÁFICA

O rastreamento de pupila é uma ferramenta muito útil quando tratado de estudos e processo cognitivos e transferência e análise de dados em tempo real.

Algumas aplicações estão sendo feitas quando trata-se de auxílio para deficientes motores, algumas doenças impossibilita os movimentos dos membros inferiores e superiores, porém, algumas dessas ainda tornam possível a utilização dos olhos para a comunicação, como ELA: Esclerose Lateral Amiotrófica (Stephen Hawking possui essa doença).

Com o sucesso e praticidade deste, foi aumentando o uso de rastreamento ocular, abrangendo os mais diversos usuários. Na publicidade foi utilizado para análise de atenção em propagandas, testes feitos com três ou mais propaganda operando o seguidor de pupilas para averiguar os focos de atenção dos clientes para com o anúncio, usando um mapa de calor para aferir a zona de maior interesse do usuário.

Depois, associado a outros mecanismos, o seguidor de pupila foi utilizado como detector de sonolência em motoristas de longas viagens para que junto de outros componentes fosse avisado ao funcionário seu grau de sono, quantidade de horas dormidas e o risco de dirigir por longas distâncias nessas condições.

Com todo esse estudo, foi determinado que dois aspectos são necessários para o projeto: a localização do olho em relação a imagem e a identificação do olho para onde está voltado. Deve-se averiguar a existências dos olhos, determinar a posição dos olhos com relação a imagem e depois aferir seu rastreamento frame a frame.

VIII. REFERÊNCIAS

[1] A. Duchowski, Eye Tracking Methodology: Theory and Practice. Springer-Verlag, 2003.

[2] R.S. Feris, T.E. de Campos, and R.M. Cesar Jr., “Detection and Tracking of Facial Features in Video Sequences,” Proc. Conf. Medical Image Computing and Computer-Assisted Intervention, 2000.

[3] Oslosky J., Itoh Y., Ranchet M., Kiyokawa K., Morgan J., Devos H. “Emulation of Physician Tasks in Eye-Tracked Virtual Reality for Remote Diagnosis of Neurodegenerative Disease”.

IEEE Transactions on Visualization and Computer Graphics, vol. 23, 2017.
[4] Timm F., Erhardt B., “Accurate Eye Centre Localisation by Gradients“.