



UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO  
CENTRO DE CIÊNCIA E TECNOLOGIA – CCT  
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS – LCMAT  
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Luis Fernando Peixoto Cabral

**RASTREAMENTO OCULAR PARA IDENTIFICAÇÃO DA ATENÇÃO  
VISUAL EM WEBSITES**

Campos dos Goytacazes, RJ  
2023

Luis Fernando Peixoto Cabral

**RASTREAMENTO OCULAR PARA IDENTIFICAÇÃO DA ATENÇÃO  
VISUAL EM WEBSITES**

Trabalho de Conclusão de Curso de Graduação em Ciência da Computação do Centro de Ciência e Tecnologia da Universidade Estadual do Norte Fluminense como requisito para a Cientista da Computação.  
Orientador: Prof. Dr. Luis Antonio Rivera Escriba

Campos dos Goytacazes, RJ  
2023

Luis Fernando Peixoto Cabral

## **RASTREAMENTO OCULAR PARA IDENTIFICAÇÃO DA ATENÇÃO VISUAL EM WEBSITES**

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “Cientista da Computação” e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação.

Campos dos Goytacazes, RJ, 24 de Novembro de 2023.

### **Banca Examinadora:**

---

Prof. Dr. Luis Antonio Rivera Escriba  
Universidade Estadual do Norte Fluminense Darcy Ribeiro

---

Prof. Dr. Fermin Alfredo Tang Montané  
Universidade Estadual do Norte Fluminense Darcy Ribeiro

---

Prof. Dr. Joao Luiz de Almeida Filho  
Universidade Estadual do Norte Fluminense Darcy Ribeiro

Este trabalho é dedicado a minha querida família e aos  
meus colegas de classe

## **AGRADECIMENTOS**

Dedico esse trabalho a minha família, que sempre me apoiou e me incentivou durante essa jornada.

## RESUMO

Este trabalho teve como objetivo desenvolver um sistema de rastreamento ocular utilizando visão computacional e aprendizado de máquina para identificar e fornecer informações de atenção visual durante a navegação de um usuário em um website. Através de uma série de testes realizados, o sistema obteve uma precisão satisfatória em identificar e classificar elementos web, assim como também em identificar a posição do olhar do usuário. Porém, o sistema mostrou a necessidade de melhorias na identificação e classificação de textos, assim como também em identificar a posição do olhar do usuário em ambientes com baixa iluminação. Com o teste geral, o sistema mostrou ser capaz de identificar e fornecer informações, por meio de relatórios que mostram as regiões e elementos do website que obtiveram atenção visual do usuário. Com isso, o sistema se mostrou como uma ferramenta promissora para fornecer informações relacionadas ao comportamento do usuário em um website, que podem ser utilizadas para a criação de interfaces mais eficientes.

**Palavras-chave:** Rastreamento ocular; Atenção Visual; Website; Comportamento do Usuário; Visão computacional; Aprendizado de máquina.

## ABSTRACT

This work aimed to develop an eye tracking system using computer vision and machine learning to identify and provide visual attention information during a user's navigation on a website. Through a series of tests carried out, the system achieved satisfactory accuracy in identifying and classifying web elements, as well as identifying the position of the user's gaze. However, the system showed the need for improvements in the identification and classification of texts, as well as in identifying the position of the user's gaze in environments with low lighting. With the general test, the system proved to be capable of identifying and providing information, through reports that show the regions and elements of the website that received the user's visual attention. As a result, the system proved to be a promising tool for providing information related to user behavior on a website, which can be used to create more efficient interfaces.

**Keywords:** Eye tracking; Visual Attention; Web site; User Behavior; Computer vision; Machine learning.

## LISTA DE FIGURAS

Figura 2.1 – Ilustração do mapa de calor . . . . .	26
Figura 2.2 – Ilustração do mapa de fixação. . . . .	26
Figura 2.3 – Ilustração do scanpath. . . . .	27
Figura 2.4 – Ilustração de uma imagem em escala de cinza. . . . .	28
Figura 2.5 – Ilustração de uma imagem com o filtro de gaussiano. . . . .	29
Figura 2.6 – Ilustração de uma imagem binarizada. . . . .	30
Figura 2.7 – Ilustração de uma imagem após o processo de detecção de bordas. . . . .	30
Figura 2.8 – Processo de dilatação . . . . .	31
Figura 2.9 – Processo de erosão . . . . .	31
Figura 3.1 – Arquitetura do sistema. . . . .	35
Figura 3.2 – Arquitetura da interface do usuário. . . . .	36
Figura 3.3 – Exemplo de capturas realizadas pelo navegador. . . . .	37
Figura 3.4 – Processo de rastreamento ocular. . . . .	38
Figura 3.5 – Processo de detecção dos pontos faciais. . . . .	39
Figura 3.6 – Representação do processo de recorte da região dos olhos. . . . .	40
Figura 3.7 – Representação do processo de detecção da pupila. . . . .	42
Figura 3.8 – Ilustração da imagem binarizada da íris. . . . .	43
Figura 3.9 – Representação do centro da pupila. . . . .	45
Figura 3.10–Etapas de calibração. . . . .	46
Figura 3.11–Representação dos pontos de calibração exibido na tela. . . . .	46
Figura 3.12–Exemplo de pontos do olhar. . . . .	51
Figura 3.13–Representação do mapa de fixação. . . . .	52
Figura 3.14–Exemplo do processo para gerar o mapa de fixação. . . . .	53
Figura 3.15–Representação do Scanpath. . . . .	54
Figura 3.16–Exemplo do processo para gerar o scanpath. . . . .	55
Figura 3.17–Representação do mapa de calor aplicado. . . . .	56
Figura 3.18–Representação da imagem em uma matriz. . . . .	57
Figura 3.19–Processo de criação do mapa de calor. . . . .	59
Figura 3.20–Ilustração do processo de conversão de uma imagem para escala de cinza. . . . .	60
Figura 3.21–Ilustração do processo de binarização de uma imagem. . . . .	60
Figura 3.22–Ilustração do processo do método de Canny aplicado em uma imagem. . . . .	61
Figura 3.23–Ilustração do processo de dilatação aplicado em uma imagem. . . . .	61
Figura 3.24–Ilustração do processo de erosão aplicado em uma imagem. . . . .	62
Figura 3.25–Exemplo do resultado de detecção de elementos em uma captura de tela. . . . .	62
Figura 3.26–Exemplo de elementos classificados. . . . .	63
Figura 3.27–Ilustração do funcionamento do IoU . . . . .	64
Figura 3.28–Exemplo da delimitação de uma fixação. . . . .	64

Figura 3.29–Exemplo de uma sobreposição entre um elemento e uma fixação. . . . .	65
Figura 4.1 – Interface da página inicial. . . . .	67
Figura 4.2 – Interface da página de execução. . . . .	67
Figura 4.3 – Ilustração do processo de extração de pontos faciais. . . . .	70
Figura 4.4 – Ilustração dos pontos faciais. . . . .	72
Figura 4.5 – Representação dos pontos dos olhos. . . . .	73
Figura 4.6 – Ilustração do recorte dos olhos. . . . .	74
Figura 4.7 – Ilustração da conversão da imagem para escala de cinza. . . . .	74
Figura 4.8 – Ilustração da remoção de ruído. . . . .	75
Figura 4.9 – Ilustração da binarização da imagem. . . . .	75
Figura 4.10–Mapa de fixação gerado pelo sistema. . . . .	82
Figura 4.11–Scanpath gerado pelo sistema. . . . .	84
Figura 4.12–Mapa de calor gerado pelo sistema. . . . .	86
Figura 4.13–Imagem convertida para cor cinza. . . . .	88
Figura 4.14–Imagem após o processo de binarização. . . . .	88
Figura 4.15–Imagem após o processo do Método de Canny. . . . .	89
Figura 4.16–Imagem após o processo de dilatação. . . . .	90
Figura 4.17–Imagem após o processo de erosão. . . . .	90
Figura 4.18–Ilustração dos elementos identificados. . . . .	91
Figura 4.19–Elementos classificados pelo sistema. . . . .	93
Figura 5.1 – Usuário olhando para um ponto fixo na tela. . . . .	95
Figura 5.2 – Os pontos que o usuário olhou durante o teste. . . . .	96
Figura 5.3 – Usuário realizando o teste de precisão da estimativa do olhar em um ambiente com boa iluminação e baixa iluminação. . . . .	97
Figura 5.4 – Arquivos CSV com as coordenadas do olhar do usuário para cada ponto fixado na tela no ambiente com boa iluminação. . . . .	98
Figura 5.5 – Arquivos CSV com as coordenadas do olhar do usuário para cada ponto fixado na tela no ambiente com baixa iluminação. . . . .	99
Figura 5.6 – Comparação da média dos resultados obtidos no ambiente com boa iluminação e baixa iluminação. . . . .	100
Figura 5.7 – Capturas de telas dos sites utilizados para os testes de precisão da identificação de elementos web. . . . .	101
Figura 5.8 – Elementos web identificados pelo sistema. . . . .	102
Figura 5.9 – Comparação da média dos resultados por tipo de elemento WEB. . . .	103
Figura 5.10–Usuário selecionando a opção de executar o experimento na interface do sistema. . . . .	104
Figura 5.11–Usuário preenchendo as informações do experimento. . . . .	104
Figura 5.12–Usuário olhando para os pontos de calibração apresentados na tela. . .	105
Figura 5.13–Usuário navegando no site. . . . .	105

Figura 5.14–Capturas de telas feitas pelo sistema durante a navegação no site. . . .	106
Figura 5.15–Coordenadas do olhar do usuário para cada captura de tela realizada. .	106
Figura 5.16–Fixações indentificadas pelo sistema para a primeira captura de tela. .	107
Figura 5.17–Fixações indentificadas pelo sistema para a segunda captura de tela. .	108
Figura 5.18–Fixações indentificadas pelo sistema para a terceira captura de tela. .	109
Figura 5.19–Fixações detectadas para a primeira captura de tela. . . . .	110
Figura 5.20–Fixações detectadas para a segunda captura de tela. . . . .	110
Figura 5.21–Fixações detectadas para a terceira captura de tela. . . . .	111
Figura 5.22–Scanpath detectados para a primeira captura de tela. . . . .	112
Figura 5.23–Scanpath detectados para a segunda captura de tela. . . . .	112
Figura 5.24–Scanpath detectados para a terceira captura de tela. . . . .	113
Figura 5.25–Mapa de calor para a primeira captura de tela. . . . .	114
Figura 5.26–Mapa de calor para a segunda captura de tela. . . . .	114
Figura 5.27–Mapa de calor para a terceira captura de tela. . . . .	115
Figura 5.28–Elementos identificados da primeira captura de tela. . . . .	115
Figura 5.29–Elementos identificados para a segunda captura de tela. . . . .	116
Figura 5.30–Elementos identificados para a terceira captura de tela. . . . .	117

## **LISTA DE QUADROS**

Quadro 3.1 – Pontos que delimitam a região dos olhos. . . . .	40
Quadro 4.1 – Pontos dos olhos. . . . .	73

## LISTA DE TABELAS

Tabela 3.1 – Exemplo das informações dos pontos faciais. . . . .	39
Tabela 3.2 – Exemplo dos pontos de calibração associado com as coordenadas das pupilas. . . . .	47
Tabela 3.3 – Resultado dos cálculos para cada ponto de calibração e ponto médio da pupila no eixo x. . . . .	48
Tabela 3.4 – Resultado dos cálculos para cada ponto de calibração e ponto médio da pupila no eixo y. . . . .	49
Tabela 3.5 – Grupo de cores. . . . .	56
Tabela 4.1 – Grupo de cores do mapa de calor. . . . .	86
Tabela 5.1 – Pontos apresentado na tela. . . . .	96
Tabela 5.2 – Tamanho do raio por nível. . . . .	96
Tabela 5.3 – Resultados da precisão do olhar para o ambiente de boa iluminação. . .	98
Tabela 5.4 – Resultados da precisão do olhar para o ambiente de baixa iluminação. .	100
Tabela 5.5 – Quantidade de elementos textuais, figuras e botões identificados manualmente em cada captura de tela. . . . .	102
Tabela 5.6 – Resultados da precisão de identificação de elementos WEB. . . . .	103
Tabela 5.7 – Informações sobre os elementos detectados da primeira captura de tela.	116
Tabela 5.8 – Informações sobre os elementos detectados da segunda captura de tela.	117
Tabela 5.9 – Informações sobre os elementos detectados da terceira captura de tela.	117

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>14</b>
1.1	PROBLEMÁTICA . . . . .	15
1.2	HIPÓTESE . . . . .	15
1.3	OBJETIVO . . . . .	16
1.4	JUSTIFICATIVA . . . . .	16
1.5	MÉTODO . . . . .	16
1.6	ORGANIZAÇÃO DO TRABALHO . . . . .	18
<b>2</b>	<b>ATENÇÃO VISUAL E RASTREAMENTO OCULAR . . . . .</b>	<b>20</b>
2.1	VISÃO HUMANA . . . . .	21
2.2	MOVIMENTOS OCULARES . . . . .	21
2.3	ATENÇÃO VISUAL . . . . .	22
2.4	RASTREAMENTO OCULAR . . . . .	22
<b>2.4.1</b>	<b>HISTÓRIA . . . . .</b>	<b>23</b>
<b>2.4.2</b>	<b>TÉCNICAS . . . . .</b>	<b>24</b>
<b>2.4.3</b>	<b>RASTREIO OCULAR POR SOFTWARE . . . . .</b>	<b>24</b>
2.5	VISUALIZAÇÃO DE DADOS . . . . .	25
2.6	VISÃO COMPUTACIONAL . . . . .	27
<b>2.6.1</b>	<b>PRÉ-PROCESSAMENTO . . . . .</b>	<b>27</b>
<b>2.6.2</b>	<b>SEGMENTAÇÃO . . . . .</b>	<b>29</b>
<b>2.6.3</b>	<b>DETECÇÃO DE BORDAS . . . . .</b>	<b>30</b>
<b>2.6.4</b>	<b>OPERAÇÕES MORFOLÓGICAS . . . . .</b>	<b>30</b>
<b>2.6.5</b>	<b>CLASSIFICAÇÃO DE OBJETOS . . . . .</b>	<b>31</b>
2.7	TRABALHOS RELACIONADOS . . . . .	32
<b>3</b>	<b>MODELAGEM E PROJETO . . . . .</b>	<b>35</b>
3.1	INTERFACE DO USUÁRIO . . . . .	36
3.2	NAVEGADOR WEB . . . . .	36
3.3	RASTREAMENTO OCULAR . . . . .	38
3.4	CALIBRAÇÃO . . . . .	45
3.5	ANÁLISE DE DADOS . . . . .	50
<b>3.5.1</b>	<b>IDENTIFICAÇÃO DA ATENÇÃO VISUAL . . . . .</b>	<b>50</b>
<b>3.5.2</b>	<b>GERAÇÃO DE RELATÓRIOS . . . . .</b>	<b>51</b>
<b>4</b>	<b>DESENVOLVIMENTO . . . . .</b>	<b>66</b>
4.1	FERRAMENTAS E TECNOLOGIAS UTILIZADAS . . . . .	66
4.2	INTERFACE . . . . .	67
4.3	NAVEGADOR . . . . .	67
4.4	RASTREAMENTO OCULAR . . . . .	69
4.5	CALIBRAÇÃO . . . . .	76

4.6	ANÁLISE DE DADOS . . . . .	79
4.6.1	IDENTIFICAÇÃO DE FIXAÇÕES . . . . .	79
4.6.2	GERAÇÃO DE RELATÓRIOS . . . . .	81
5	TESTES E RESULTADOS . . . . .	95
5.1	TESTE DE PRECISÃO DA ESTIMATIVA DO OLHAR . . . . .	95
5.2	TESTE DE PRECISÃO DE IDENTIFICAÇÃO DE ELEMENTOS WEB	101
5.3	TESTE GERAL DO SISTEMA . . . . .	103
5.3.1	MAPA DE FIXAÇÕES . . . . .	109
5.3.2	SCANPATH . . . . .	111
5.3.3	MAPA DE CALOR . . . . .	113
5.3.4	INFORMAÇÕES DOS ELEMENTOS IDENTIFICADOS . .	115
5.4	DISCUSSÃO DOS RESULTADOS . . . . .	118
6	CONCLUSÃO . . . . .	119
	REFERÊNCIAS . . . . .	120

## 1 INTRODUÇÃO

Os sistemas de rastreamento ocular são ferramentas que possuem a capacidade de registrar os movimentos oculares (BERGSTROM; SCHALL, 2014). Os olhos é um órgão fundamental para os seres vivos, pois é através dele que possuímos a capacidade de visualizar e interagir com o ambiente ao nosso redor. As informações coletadas pelos olhos são enviadas para o cérebro, onde são processadas e com isso as características visuais dos objetos, como as cores, formatos e tamanhos, podem influenciar as ações realizadas e o comportamento humano (BURCH, 2021).

Com isso, o rastreamento ocular possui a capacidade de fornecer informações importantes relacionadas ao comportamento humano. Através de estudos dos movimentos oculares é possível extrair informações da região que possui a atenção do olhar de um indivíduo, como também o caminho traçado pelo olhar e o tempo de duração entre um determinado ponto e outro (CHEN *et al.*, 2022). Onde essas informações são utilizadas em diversas áreas, como na medicina para identificar doenças oculares, psicologia no estudo do comportamento humano e no marketing para criação de campanhas publicitárias mais eficientes baseadas na preferência visuais (DONUK *et al.*, 2022).

Uma das formas de identificar os movimentos oculares é por meio de tecnologia de rastreamento ocular. Além das áreas que foram citadas anteriormente, o rastreamento ocular também vem sendo utilizado na web para identificar a atenção visual do usuário e desta forma melhorar a experiência do usuário ao navegar em um site. Onde nesta área a tecnologia de rastreamento ocular pode ser muito útil, pois através dela pode tirar conclusões de quais elementos chamam a atenção dos usuários para cada tipo de site (VERA *et al.*, 2017).

A busca dos websites pelas melhores estratégias na qual proporcionem melhorias no desempenho é cada vez maior. Com o passar dos anos, e cada dia mais a internet está presente no nosso dia a dia, a concorrência pela atenção dos usuários entre os sites também aumentou. Segundo uma pesquisa realizada pelo Sitefy divulgada pelo site Terra (2023), todos os dias são criados em média 252 mil websites na internet. Com isso as ferramentas que auxiliam aos sites a obter mais visitantes, e consequentemente, atrair mais publicidade e aumentar o seu faturamento, ganham cada vez mais espaço no mercado.

As ferramentas de análise de métricas como quantidade de cliques, tempo médio de permanência em uma determinada página e entre outras métricas, já são utilizadas para identificar possíveis pontos de melhorias no site. Porém essas métricas não fornecem informações mais assertivas sobre o comportamento do usuário no website, onde não é possível saber exatamente quais regiões e elementos teve a atenção visual do usuário, onde por exemplo não é possível saber se o usuário leu ou não determinado conteúdo página (VERA *et al.*, 2017).

Apesar de ser muito úteis até o momento, essas métricas não respondem com

precisão se um determinado elemento está cumprido o seu papel visualmente, ou se uma propaganda presente no website está chamando a atenção do usuário conforme o esperado. Devido a isso, a ferramenta de rastreamento ocular possui uma grande importância, uma vez que possui a capacidade de registrar os movimentos oculares e a partir disso identificar a atenção visual do usuário (VERA *et al.*, 2017).

A atenção visual se trata da capacidade humana em selecionar informações visuais importantes no momento atual e desconsiderar todas as outras informações que não são relevantes para o momento (LOCKHOFEN; MULERT, 2021). Com isso, é proposto neste trabalho criar um sistema de rastreamento ocular que permita, através de uma webcam, coletar informações de movimento oculares e identificar atenção visual nos websites e fornecer informações sobre as regiões e os elementos dos websites na qual obteve atenção visual.

## 1.1 PROBLEMÁTICA

Hoje em dia, para verificar o desempenho de um website, um dos métodos mais utilizados é através de métricas como o número de cliques, o tempo de permanência em uma página, quantidade de visitas, entre outros. Através dessas métricas é possível saber se uma determinada página está recebendo a quantidade de visitas esperada, se o tempo que usuário levou para realizar uma ação no site está dentro do esperado e entre outras informações (VERA *et al.*, 2017).

Porém, essas métricas não fornecem informação sobre a atenção visual do usuário, ou seja, não é possível saber quais regiões e elementos do site que o usuário mais olhou. Desta forma, a quantidade de cliques, tempo de permanência e entre outros, não são informações suficientes para determinar se o conteúdo do site foi satisfatório para o usuário. Uma vez que não é possível verificar se o usuário realmente leu o conteúdo da página, ou se ele apenas passou os olhos por ela. Se uma determinada propaganda chamou a atenção do usuário conforme o esperado. A organização dos elementos está fazendo sentido para o usuário, entre outros.

Portanto, o problema é que há limitações para saber quais regiões e objetos do website obteve a atenção do usuário.

## 1.2 HIPÓTESE

Através de um sistema de rastreamento ocular para identificação de atenção visual, será possível fornecer informações como a frequência e a duração do olhar para as regiões e elementos de uma página web, que obtiveram a atenção do usuário ao navegar em um website.

### 1.3 OBJETIVO

O objetivo geral deste trabalho foi desenvolver um sistema de rastreamento ocular que fosse capaz de fornecer informações relacionadas à atenção visual do usuário em um website.

#### **Objetivos específicos:**

- Realizar rastreamento ocular e calcular as coordenadas da pupila dos usuários através das imagens fornecida por uma webcam.
- Gerar relatórios que ajudem na interpretação da atenção visual do usuário em um website, como mapas de calor, mapas de fixação e scanpath.
- Identificar elementos como textos, figuras e botões que tiveram atenção visual do usuário

### 1.4 JUSTIFICATIVA

O desenvolvimento de um sistema que fornece informações de atenção visual de um usuário ao navegar em um website é fundamental para compreender o comportamento do usuário e identificar problemas tanto no layout quanto no conteúdo do site. Além disso, um sistema desse tipo possui a capacidade de fornecer informação bastante importante a ser levada em consideração para a construção de interfaces mais eficientes de acordo com as preferências visuais dos usuários(VERA *et al.*, 2017).

Além disso, a informação fornecida por esse sistema sobre a atenção visual do usuário, são importantes não só para os websites, mas também para diversas áreas, por exemplo a área de marketing, onde por meio dessas informações é possível criar campanhas mais eficientes em websites, uma vez que será possível saber se de fato a propaganda chamou atenção do usuário, sendo uma informação adicional além do número de cliques (BURGER; KNOLL, 2018). Uma outra área que pode se beneficiar por meio da informação visual é a área de pesquisa, onde será possível realizar diversos estudos voltados à questão de usabilidade em websites. Com isso, o desenvolvimento de um software de rastreamento ocular para ser usado em website é muito relevante, beneficiando todas as áreas citadas anteriormente, mas principalmente aos usuários web que poderão utilizar os conteúdos baseados na sua preferência visual.

### 1.5 MÉTODO

Inicialmente foi desenvolvido uma interface de usuário para facilitar a interação com os recursos do sistema, como a visualização dos relatórios, realizar a configuração do sistema e por fim a realização do processo de rastreamento ocular.

Em seguida, foi desenvolvido um navegador web com o intuito de permitir a navegação em qualquer site disponível na internet, e além disso, realizar a captura de tela

conforme for realizada a navegação. Inspirado no Fitzpatrick (2021), foi construído utilizando a biblioteca *PyQt5*, que se trata de uma biblioteca em *Python* para construção de interfaces gráficas, onde possui recursos que permite construção de um navegador.

Além disso, para construir o sistema de captura de tela foi utilizado o recurso de capturas de tela da biblioteca *PyQt5*, e baseado no trabalho de Kiruthika *et al.* (2021), utilizando a biblioteca *OpenCV*, foi desenvolvido um processo de comparação de imagens para realizar a captura de telas apenas quando houver de fato alguma mudança no conteúdo exibido na tela.

Também foi desenvolvido um sistema de rastreamento ocular, que será responsável por receber os frames de uma webcam com a face do usuário, e partir disso será feito uma série de identificações que primeiramente baseado no trabalho de Hangaragi *et al.* (2023), foi utilizado a biblioteca *Mediapipe* através do método *Face mesh*, que irá detectar os pontos faciais do usuário, onde será utilizado posteriormente para detectar região dos olhos do usuário.

Inspirado no trabalho dos autores Hange Wang *et al.* (2022), através de processamento de imagem utilizando a biblioteca *OpenCV* será feito o recorte da região dos olhos. Em seguida, baseado no trabalho de Brachter e Gerhardt (2020) será feito o processo de segmentação e detecção de contornos para identificar a pupila do usuário. E por fim será feito o processo de cálculo do centroide da pupila, onde irá obter as coordenadas do centro da pupila, que será utilizado para processo tanto de calibração quanto de estimativa do olhar.

Na sequência foi desenvolvido um processo de calibração e estimativa do olhar, onde o processo de calibração baseado no trabalho de Brachter e Gerhardt (2020). Esse processo é responsável por realizar a associação entre as coordenadas do centro da pupila com coordenadas com tela do computador, onde é feito um modelo de regressão linear, na qual será utilizado no processo de estimativa do olhar que usará esse modelo para prever as coordenadas que o usuário está olhando na tela.

Por fim, foi desenvolvido um processo de análise dados na qual irá processar os dados coletados durante o processo de rastreamento ocular, em que irá identificar atenção visual do usuário, e partir disso, gerar os relatórios que irá mostrar as regiões e os elementos do site que obteve a atenção do usuário durante a navegação. Onde para mostrar as regiões que obtiveram atenção visual, foram utilizadas as bibliotecas *OpenCV* para desenvolver os mapas de fixações e o scanpath e foi utilizado a ferramenta *Pygaze* do trabalho de Dalmaijer *et al.* (2014) para gerar mapa de calor.

Para o desenvolver a detecção dos elementos que obtiveram atenção visual, baseado no trabalhos de Goyal *et al.* (2021) e Bukhari *et al.* (2011), foi utilizado processamento de imagens, através da biblioteca *OpenCV* para realizar a segmentação e detecção de contornos. Após isso, o processo de classificação foi realizado através de SVM que se trata de um algoritmo de aprendizagem de máquina, onde irá classificar os elementos em figuras,

texto e botões. Após esse processo será feita a interseção entre os pontos de atenção visual identificados com os elementos detectados, e desta forma obter-se os elementos que obtiveram atenção visual do usuário.

Ao final, serão realizados três testes para verificar a eficiência do sistema. O primeiro teste irá verificar a eficiência em identificar as coordenadas na qual o usuário está olhando na tela. O segundo teste será responsável por verificar a eficiência na identificação e classificação dos elementos do website. E por fim, o terceiro teste será feito com intuito de verificar a eficiência do sistema como um todo.

#### **A metodologia:**

1. Revisão da literatura
2. Construção de uma interface de usuário para o sistema
3. Implementação de um navegador web
4. Desenvolvimento de um sistema de rastreamento ocular
5. Processo de calibração e estimativa do olhar
6. Análise
  - a) Identificação de atenção visual
  - b) Geração
    - i. Mapa de calor
    - ii. Mapa de fixação
    - iii. Scanpath
    - iv. Elementos dos websites que obtiveram atenção visual do usuário

## 1.6 ORGANIZAÇÃO DO TRABALHO

O presente trabalho é organizado a partir da introdução que apresenta o contexto que envolve o trabalho, além disso, apresenta as problemáticas, objetivos, motivação e metodologia.

O capítulo 2 aborda os conceitos e definições que envolvem o trabalho, como: atenção visual, movimentos oculares, rastreamento ocular, visão computacional e métodos de visualização de dados de rastreamento ocular.

O capítulo 3 apresenta a modelagem proposta para o sistema, detalhando e exemplificando cada elemento do modelo, como por exemplo, rastreamento ocular, calibração, análise de dados e entre outros.

O capítulo 4 apresenta detalhes referentes à implementação do modelo proposto no capítulo 3.

O capítulo 5 aborda os testes realizados no sistema implementado, apresentando os resultados obtidos e a discussão dos mesmos.

Por fim, o capítulo 6 apresenta as conclusões a respeito do trabalho realizado e sugestões para trabalhos futuros.

## 2 ATENÇÃO VISUAL E RASTREAMENTO OCULAR

A atenção visual é um conceito bastante importante para se ter o entendimento do comportamento humano, pois é através desse conceito que o ser humano consegue selecionar as informações que são mais importantes e desconsiderar as que não fazem sentido no momento (LOCKHOFEN; MULERT, 2021). Através da detecção da atenção visual é possível identificar quais os objetos ou pontos de interesse que o usuário está prestando atenção. Uma forma de identificar a atenção visual de um indivíduo é através de uma tecnologia chamada rastreamento ocular.

O rastreamento ocular é responsável por detectar e coletar os movimentos oculares de um indivíduo, e com isso é possível identificar as fixações e sacadas que são dois movimentos oculares fundamentais para se identificar a atenção visual (RUBENSTEIN; RAKIC, 2013). Onde a fixação ocorre quando os olhos ficam estabilizados em um ponto de interesse que pode ser um objeto em específico ou uma região de interesse (DUCHOWSKI, 2017). Onde é através desse movimento que é o momento na qual são extraídas as informações visuais e são enviadas para o cérebro para serem processadas ao redor (SERENO *et al.*, 2009). Já as sacadas são os movimentos que ocorrem quando os olhos se movem entre uma fixação e outra, onde é nesse processo que é feita uma busca por novas informações visuais (DUCHOWSKI, 2017).

Onde através da coleta de dados referentes aos movimentos oculares realizada durante o processo de rastreamento ocular, é possível visualizar as regiões de atenção e como mapa de fixação, mapa de calor e scanpath. Onde o mapa fixação mostra todas as fixações, mostrando a duração e a localização da região (BERGSTROM; SCHALL, 2014). O scanpath é responsável por mostra visualmente a trajetória realizada pelo usuário durante a navegação, na qual possui todas as fixações realizadas juntamente com a duração de cada fixação, além disso a sequência de fixações realizadas conectadas por linhas, indicando a trajetória percorrida pelo usuário (BERGSTROM; SCHALL, 2014). Já o mapa de calor exibe as regiões que obtiveram da atenção visual, onde às regiões que possuem cores mais quente indicam que foram os locais que tiveram uma maior atenção visual, enquanto as regiões que possuem cores mais frias indicam que foram os locais menos visualizados pelo usuário, mas que mesmo assim obtiveram uma certa atenção visual, segundo Bergstrom e Schall (2014).

Além disso, uma outra forma de obter um entendimento da atenção visual e consequentemente do comportamento do usuário, é realizar identificação dos elementos que obtiveram a fixação, por meio de visão computacional e classificação de objetos, conforme sugerido por Barz e Sonntag (2021).

## 2.1 VISÃO HUMANA

A visão é um dos cinco sentidos do corpo humano, onde possibilita a capacidade de enxergar o mundo ao redor. Esse processo é realizado através de um órgão chamado olho, que é responsável por transformar a luz captada em impulsos eletroquímicos, que são enviados para o cérebro. Onde esses impulsos são processados pelo cérebro, e assim é obtido uma percepção dos objetos visuais (BURCH, 2021).

Para Burch (2021), o olho possui a função de realizar o ajuste de foco para que um determinado objeto seja visualizado com mais clareza, a córnea é responsável por realizar o processo de entrada de luz no olho, já a íris exerce a função de gerenciar a luminosidade de entrada, como por exemplo em casos que o olho é exposto a uma grande quantidade de luz o tamanho da abertura da Íris diminui para que entre menos luz. Em casos que possua pouca luz, a íris tem seu tamanho aumentado, para que possa entrar mais luz.

O processo de absorção de luz é feito através de duas células, os bastonetes e os cones, onde estão localizadas nas retinas e são responsáveis por capturar a luminosidade através dos fotorreceptores. Onde os bastonetes possuem a função de realizar a captura de luz, e é fundamental para a visão noturna. Enquanto os cones são responsáveis por realizar a captura das cores e os detalhes do objeto, onde tendo sua grande importância em ambientes que possua uma boa iluminação. Através dessas células é possível captar a luz que consequentemente serão transformados em sinais que serão enviados ao cérebro permitindo assim a visão, segundo Burch (2021).

Um outro fator importante para a visão é os movimentos oculares, onde através deles que em situações na qual o corpo ou a cabeça estão em movimento, é possível obter uma imagem estabilizada na retina (STUART, 2022). Além disso, os movimentos oculares também podem fornecer informações importantes para o entendimento do comportamento do usuário.

## 2.2 MOVIMENTOS OCULARES

O movimento ocular de acordo com Bridgeman (2012) se trata de uma mudança que pode ser voluntária ou involuntária da posição e direção dos olhos em relação um determinado ponto de referência, esses movimentos são realizados através de seis músculos tanto no olho esquerdo quanto no direito. Na qual possibilitar o posicionamento ocular para com que o seja capturado a percepção e acompanhamento de um estímulo visual pela retina de forma eficaz.

O entendimento do movimento ocular é essencial para compreender o comportamento humano, pois através dele que as informações chegam ao cérebro e a partir disso possibilitar com que o indivíduo perceba, foque e lembre do que está ao seu redor (SERENO *et al.*, 2009). Para o contexto de rastreamento ocular existem dois tipos principais de movimentos oculares que são eles as fixações e as sacadas.

### (A) Fixações

As fixações é um tipo de movimento ocular que se caracteriza na qual os olhos do indivíduo permanecem fixados por um determinado momento em um objeto ou ponto de interesse (DUCHOWSKI, 2017). Onde esse tempo leva em torno de 200 a 300 milissegundos conforme descrito por Gobbi *et al.* (2017). Além disso, as fixações também são caracterizadas como um conjunto de pontos do olhar que são próximos entre si, e que podem ser medidos em tempo e dispersão (BURCH, 2021).

### (B) Sacadas

A sacada é um tipo de movimento ocular que se caracteriza por realizar o direcionamento do olhar rapidamente entre um ponto de fixação e outro. Esse movimento leva em torno de 10 a 100 milissegundo de duração (DUCHOWSKI, 2017).

## 2.3 ATENÇÃO VISUAL

A atenção visual segundo Lockhofen e Mulert (2021) se trata de um mecanismo cognitivo na qual possibilita realizar a filtragem seletiva das informações que um indivíduo é submetido no dia a dia. Onde de acordo com Haith e Benson (2008) é feito um direcionamento do foco e a concentração em um determinado estímulo visual. A partir deste direcionamento do foco é possível dar destaque às informações que são relevantes para o indivíduo e ignorar as outras que não fazem sentido no momento (LOCKHOFEN; MULERT, 2021).

De acordo com Rubenstein e Rakic (2013) a atenção visual ocorre quando é obtido às fixações e sacadas, onde as sacadas é quando o ocorre um direcionamento do foco de atenção para um novo ponto de interesse, e as fixações são pausas que ocorrem e é onde o ponto de interesse ou objeto é examinado. Com Chen *et al.* (2022) uma pesquisa realizada indicou que fixações mais longas são associadas a uma maior atenção visual, e as fixações que possuem uma menor duração e quantidade é indicativo de uma menor atenção visual. Sendo assim os movimentos oculares possuem um papel fundamental para a atenção visual, pois através de métricas como duração, frequência e localização de fixações é possível determinar o que o indivíduo está prestando atenção (CHEN *et al.*, 2022).

Com isso, a identificação da atenção visual possui uma grande importância para a compreensão do comportamento humano.

## 2.4 RASTREAMENTO OCULAR

Segundo Bergstrom e Schall (2014) o rastreamento ocular se trata de um método na qual permite extrair informações relevantes sobre os movimentos oculares, como por

exemplo saber onde que o usuário está olhando em uma tela, a duração do tempo do olhar para um objeto em específico, e o caminho percorrido pelos olhos. Esse processo é realizado através de um dispositivo chamado de rastreador ocular.

O rastreador se trata de um dispositivo que permite realizar a reflexão corneana, que consiste em emitir uma luz para em direção ao olho do usuário e a partir disso através de uma câmera com alta resolução capturar a reflexão da luz, e desta forma é possível realizar a detecção dos movimentos oculares (BERGSTROM; SCHALL, 2014). A partir disso, por meio dos movimentos oculares é possível identificar dois elementos principais que são as fixações e as sacadas.

Através desses tipos de movimentos oculares é possível entender o comportamento do usuário, por esse motivo a ferramenta de rastreamento colar é utilizada em várias áreas como por exemplo a psicologia que utiliza esta ferramenta para obter a compreensão do comportamento humano. Também é utilizado em pesquisas de experiência de usuário no marketing, no estudo de interação humano computador e entre outras áreas (HOLMQVIST; ANDERSSON, 2017).

#### 2.4.1 HISTÓRIA

Os primeiros estudos relacionados ao rastreamento ocular segundo Bergstrom e Schall (2014) e Burch (2021), foi realizado a partir do século XIX, através dos pesquisadores Wells e Erasmus Darwin, onde tinham o intuito de entender o padrão dos movimentos dos olhos e com isso entender o comportamento humano.

No século XX os pesquisadores para capturar os movimentos dos olhos começaram a utilizar filmes e um quimógrafo para tentar ouvir os movimentos oculares (BURCH, 2021). Através desses estudos obteve os primeiros avanços que foram bastante significativos e importantes para o estudo de rastreamento ocular hoje em dia. Como por exemplo, foi descoberto que durante uma leitura os movimentos dos olhos se alternam entre pequenas pausas e saltos rápidos, onde esses movimentos são conhecidos como fixações e sacadas (BURCH, 2021).

Com a invenção do computador e desenvolvimento de software e hardware a tecnologia de rastreamento teve um grande avanço, na qual possibilitou capturas mais precisas e mais rápidas dos movimentos oculares (BURCH, 2021). Além disso, entre as décadas de 60 e 70 deu início a uma nova era para a tecnologia de rastreamento ocular, pois foi nesta época que surgiu a técnica baseada em vídeo (BERGSTROM; SCHALL, 2014). Em seguida, na década de 90 surgiu uma outra técnica, porém dessa vez baseada em ressonância magnética, que proporcionaram obter informações mais detalhadas sobre os movimentos dos olhos (BURCH, 2021). A partir desses avanços de hardware e software foram criadas muitas possibilidades em relação à aplicação dessa tecnologia (BURCH, 2021).

### 2.4.2 TÉCNICAS

Existem quatro tipos de técnicas principais para detecção dos movimentos oculares no processo de rastreamento ocular, que são o Eletro-Oculografia, Lente escleral, Videografia-Oculografia e por fim a técnica Baseada em Vídeo. A seguir serão detalhados cada uma dessas técnicas.

#### (A) ELETRO-OCULOGRAFIA

A Eletro-Oculografia se trata de uma técnica que consiste em detectar os movimentos oculares por meio de registros de potencial elétrico diferencial na região da pele ao redor do olho. Essa técnica era utilizada na década de 70, porém com o avanço das técnicas de rastreamento ocular ao passar dos anos essa técnica foi deixando de ser utilizada e trocada por outras técnicas menos invasivas e com maior precisão da detecção de movimentos oculares (DUCHOWSKI, 2017).

#### (B) LENTE ESCLERAL

A técnica da lente escleral consiste em utilizar uma lente de contato nos olhos do usuário para realizar a detecção dos movimentos oculares. Essa técnica é extremamente sensível e precisa. Porém por utilizar uma lente de contato nos olhos essa técnica acaba sendo invasiva, o que acaba sendo uma desvantagem na utilização dessa técnica (DUCHOWSKI, 2017).

#### (C) VIDEOGRAFIA-OCULOGRAFIA

A técnica de Videografia-Oculografia consiste em detectar os movimentos oculares por meio da extração de características dos olhos, onde é obtido informações como, forma da pupila reflexão corneana, movimentos tanto de rotação quanto de translação e entre outros (DUCHOWSKI, 2017).

#### (D) BASEADA EM VÍDEO

A técnica baseada em vídeo se caracteriza por utilizar webcams comuns para realizar o processo de rastreamento ocular. Onde é utilizado algoritmos de processamento de imagens para detectar a posição das pupilas (DUCHOWSKI, 2017).

### 2.4.3 RASTREIO OCULAR POR SOFTWARE

O processo de rastreamento ocular realizado através de software é baseado na utilização de webcam comum para realizar o processo de rastreamento ocular. Onde é utilizado para capturar as imagens da face do usuário, e partir disso por meio de processamento e análise de imagem juntamente com algoritmos de visão computacional é obtido informações na qual podem ser utilizadas para identificar os movimentos oculares. Existem algumas opções de softwares que realizam essa função de identificação dos movimentos oculares, como por exemplo *GazePointer*, *WebGazer* e o *ITU GAZE*.

O *GazePointer* se trata de um software que é baseado em um projeto chamado *GazeFlow WebCam Eye-Tracking Engine* que é um projeto de código aberto de rastreamento ocular. Onde essa ferramenta utiliza uma webcam comum, essa ferramenta possui a capacidade de possibilitar controlar o ponteiro do mouse através dos movimentos dos olhos. Dessa forma, esse software possui a utilidade para pessoas com algum tipo de deficiência física onde não consegue utilizar os mouses convencionais (GAZEPOINTER, 2023).

O *WebGazer.js* se trata de uma biblioteca desenvolvida em *JavaScript* que possibilita ser integrado em qualquer site, na qual através de uma webcam comum juntamente com um processo de calibração permitindo assim realizar a detecção dos movimentos dos olhos e consequentemente estimar a localização do olhar do usuário em relação a uma página web. Onde esse processo é utilizado um algoritmo de regressão Ridge para realizar o processo de estimativa do Olhar (WEBGAZER, 2023).

O *ITU GAZE Tracke* é um software no qual utiliza uma câmera de visão noturna ou infravermelha para a realização do processo de rastreamento ocular. Onde essa ferramenta é uma alternativa de baixo custo monetário em relação aos sistemas comerciais (DEVINBARRY, 2014).

Dessa forma, essas ferramentas de rastreamento ocular baseada em software é uma boa alternativa na qual possui um baixo custo monetário além disso não são invasivas, uma vez que não é necessário ter se um contato direto com os olhos do usuário.

## 2.5 VISUALIZAÇÃO DE DADOS

Segundo Bergstrom e Schall (2014), o processo de análise de dados em rastreamento ocular consiste em transformar os dados que foram coletados em informações que sejam úteis na qual será utilizada em seguida para entender o comportamento do usuário durante a interação com o sistema. Como por exemplo, quanto tempo o usuário direciona o seu olhar para uma determinada região e como os olhos se movem durante uma navegação de um ponto ao outro da interface (PANETTA *et al.*, 2020). Dessa forma, por meio da análise das informações e a visualização é possível realizar tanto uma análise qualitativa quanto exploratória dos dados, possibilitando assim uma elaboração de hipótese em relação ao comportamento do usuário (BLASCHECK *et al.*, 2014). Para a visualização desses dados é utilizado três principais tipos de representações gráficas, que são o mapa de calor, mapa de fixação e o scanpath.

### (A) MAPA DE CALOR

O mapa de calor é um tipo de visualização de dados onde mostra as distribuições dos pontos do olhar, ou seja, as regiões na qual o usuário mais focou o seu olhar. Com isso, esse tipo de visualização se trata de uma ferramenta útil para entender as áreas de atenção visual do usuário. O mapa de calor possui as cores quentes como por exemplo vermelho e amarelo para representar as regiões

mais visualizadas pelo usuário, e as cores frias como por exemplo azul e ciano para representar as regiões na qual o usuário visualizou com menos frequência (BERGSTROM; SCHALL, 2014). A Figura 2.1 apresenta um exemplo de um mapa de calor.



Figura 2.1 – Ilustração do mapa de calor.

## (B) MAPA DE FIXAÇÕES

O mapa fixação se trata de um tipo de visualização que mostra as fixações realizadas pelo usuário, onde é representado por um círculo, com o tamanho é variável de acordo com a duração da fixação, ou seja, quanto maior a concentração de atenção em um determinado ponto maior será o tamanho do círculo da fixação (BERGSTROM; SCHALL, 2014). A Figura 2.2 apresenta um exemplo de um mapa de fixação.



Figura 2.2 – Ilustração do mapa de fixação.

### (C) SCANPATH

Os mapas de scanpath assim como os gráficos de fixações eles mostram graficamente os pontos de fixação, porém possui uma diferença que ele mostra de forma sequencial a ordem, e uma linha conectada entre as fixações representando o processo de navegação. Onde cada ponto é numerado com a ordem da fixação, além disso o círculo assim como no mapa de fixação, eles variam de tamanho, onde quanto maior o tamanho do círculo maior será a duração da fixação (BERGSTROM; SCHALL, 2014). A Figura 2.3 apresenta um exemplo de um scanpath.



Figura 2.3 – Ilustração do scanpath.

## 2.6 VISÃO COMPUTACIONAL

A visão computacional segundo Bhuyan (2019), se trata de um ramo da ciência da computação que consiste em fazer com que uma máquina consiga visualizar o mundo, de forma semelhante aos seres vivos. Onde desta forma possui objetivo de extrair as informações a partir de imagens ou vídeos, como cores, formas, movimentos, condições de iluminação e entre outras. Onde desta forma, essas informações possam ser utilizadas para interpretação e tomadas de decisões (STOCKMAN; SHAPIRO, 2001).

### 2.6.1 PRÉ-PROCESSAMENTO

O pré-processamento de imagem consiste em aplicar um conjunto de técnicas para realizar a preparação da imagem para com que possa destacar as informações de interesse da imagem. Onde umas das etapas é o processo de conversão para escala cinza e processo de suavização da imagem.

#### (A) CONVERSÃO PARA ESCALA DE CINZA

A conversão para escala de cinza consiste em um processo de transformação de uma imagem com cores pré-definidas, para uma imagem com apenas dois

tons de cores, preto e branco. Esse processo é realizado para os casos em que as cores na imagem não é uma informação relevante e, possui o intuito de facilitar o processo de extração de informações da imagem além de diminuir o processamento computacional (KANAN; COTTRELL, 2012). A Figura 2.4 apresenta um exemplo de uma imagem na escala de cinza.



Figura 2.4 – Ilustração de uma imagem em escala de cinza.

Para realização desse processo é feito uma operação na imagem que esteja na escala Red, Green, Blue (RGB), que irá converter a imagem para a escala de cinza. Na qual para cada pixel pertencente à imagem, será multiplicado um valor peso para cada canal de cor. Para a função "rgb2gray", segundo Kanan e Cottrell (2012) é utilizado por diversos softwares como MATLAB, GIMP e entre outros. E que também é utilizado para este trabalho juntamente com o OpenCV, os pesos segundo OpenCv (2023) são 0.299, 0.587 e 0.114 para os canais das cores do RED, GREEN e BLUE respectivamente do RGB. Portanto, a fórmula para conversão de uma imagem com escala do RGB para escala de cinza de acordo com OpenCv (2023) é:

$$Y = 0.299R + 0.587G + 0.114B$$

## (B) FILTRO DE SUAVIZAÇÃO

Segundo os filtros gaussianos são pertencentes aos filtros lineares, onde são utilizados no processo de suavização e remoção de ruídos presentes na imagem na qual possam atrapalhar o processo de extração de informações. Onde o filtro gaussiano consiste em utilizar a função gaussiana para realizar o processo de suavização (BHUYAN, 2019). A Figura 2.5 apresenta um exemplo de uma imagem aplicada ao filtro gaussiano.



Figura 2.5 – Ilustração de uma imagem com o filtro de gaussiano.

O filtro gaussiano realiza uma modificação nos pixels central, onde para isso os pixels vizinhos são multiplicados por um valor na matriz gaussiana, na qual o resultado dessa operação é então somado e dividido pelos coeficientes da matriz gaussiana. O resultado é uma imagem com menos ruídos e suavizada e quanto maior o tamanho da matriz gaussiana, maior será a suavização da imagem (BHUYAN, 2019).

### 2.6.2 SEGMENTAÇÃO

O processo de segmentação de imagens consiste em dividir uma determinada imagem em segmentos baseadas em características como, cor, textura, intensidade e entre outras características (BHUYAN, 2019). Com a divisão da divisão da imagem em segmentos, é possível identificar componentes na imagem, como objetos, fundo e entre outros (Szeliski, 2014). Desta forma possibilitando a extração de informações de interesse da imagem.

Um dos métodos de segmentação de imagem é a limiarização, esse processo consiste em dividir a imagem em dois grupos, onde cada grupo é definido por um critério de limiar baseado na intensidade dos pixels presente na imagem. Onde os pixels que for menor que o limiar será atribuído a um grupo e os pixels que possuir uma intensidade maior que o limiar será atribuído a outro grupo. Resultando em uma imagem binária, na qual um grupo irá possuir a cor branca e outro irá possuir a cor preta (BHUYAN, 2019). A Figura 2.6 apresenta um exemplo de uma imagem binarizada.



Figura 2.6 – Ilustração de uma imagem binarizada.

### 2.6.3 DETECÇÃO DE BORDAS

O processo de detecção de bordas é fundamental para o processo de identificação de elementos em uma imagem, pois através das bordas identificadas é possível detectar os contornos que compõem os objetos presentes na imagem. Para realização desse processo, uma das metodologias mais utilizadas na realização desse processo, é o método de Canny, que consiste em localizar na imagem variações de intensidade de pixels, onde essas variações identificadas são consideradas como bordas (BHUYAN, 2019). A Figura 2.7 apresenta o resultado da aplicação do método de Canny em uma imagem.



Figura 2.7 – Ilustração de uma imagem após o processo de detecção de bordas.

### 2.6.4 OPERAÇÕES MORFOLÓGICAS

As operações morfológicas são técnicas que possuem objeto de realçar ou diminuir as características de uma imagem, em que acabam ajudando no processo de detecção de

contorno, remoção de ruídos, preenchimento de lacunas e entre outras aplicações. Onde os principais operadores morfológicos são a dilatação e a erosão (SRISHA; KHAN, 2013).

A dilatação consiste em realizar um processo que aumenta as regiões branca da imagem, proporcionando dessa forma os preenchimentos de lacunas entre os pixels, possibilitando com isso a união entre os objetos que estão separados. Esse processo é ilustrado na Figura 2.8, em que os elementos da imagem da cor branca são expandidos (SRISHA; KHAN, 2013).

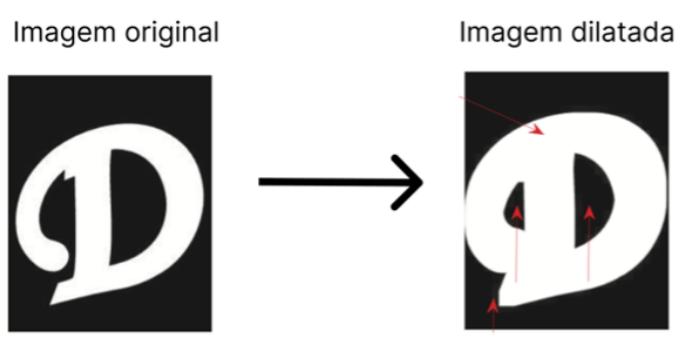


Figura 2.8 – Processo de dilatação (Modificado de Srisha e Khan (2013)).

A erosão faz o processo inverso da dilatação onde é realizado a diminuição das regiões brancas da imagem, fazendo com os objetos fiquem mais separados entre si, além de remover pequenos ruídos da imagem. Esse processo é apresentado na Figura 2.9, que mostra a diminuição dos elementos da cor branca (SRISHA; KHAN, 2013).

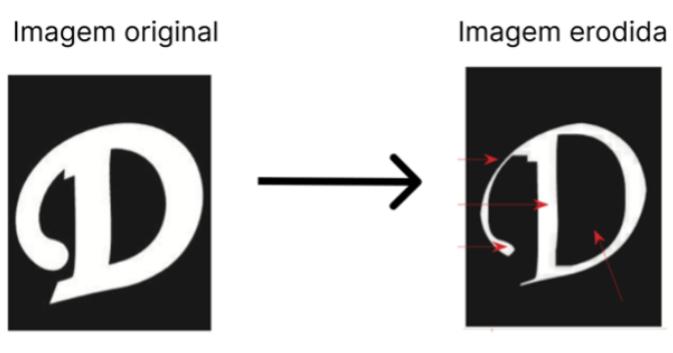


Figura 2.9 – Processo de erosão (Modificado de Srisha e Khan (2013)).

### 2.6.5 CLASSIFICAÇÃO DE OBJETOS

O processo de classificação de elementos consiste em extrair as características de um objeto, e a partir disso realizar um processo de classificação, em que define um grupo no qual esse objeto pertence baseado em suas características. Onde essas características podem ser cor, textura, bordas, forma e entre outras. Um dos algoritmos de aprendizado de máquinas mais utilizados para isso, é o *Support Vector Machine* (SVM), que possui a funcionalidade de realizar o processo de classificação de elementos, onde partir de uma base

de dados de objetos, é feito processo de extração de características em que são definidas classes. E a classificação é realizada comparando as características de um objeto com as características dos objetos das classes (AWAD; KHANNA, 2015).

## 2.7 TRABALHOS RELACIONADOS

O rastreamento ocular é uma das tecnologias que possui a capacidade de identificar a atenção visual e com isso, compreender o comportamento do usuário, pensando nisso o Donuk *et al.* (2022) desenvolveu uma ferramenta de rastreamento ocular utilizando uma webcam comum para identificar atenção visual e ser uma alternativa aos rastreamentos oculares tradicionais que possuem um valor monetário maior. Onde o autor construiu essa ferramenta para ser utilizada em páginas na web para detectar as regiões de atenção visual. Onde para isso foi utilizado aprendizado de máquina para detectar os pontos faciais do usuário, onde posteriormente é identificado e feito o recorte das regiões dos olhos, através do processamento de imagens. Que posteriormente é utilizado essa imagem recortada para identificar a íris. Para isso foi realizado uma série de etapas de processamento de imagem, como a conversão para escala de cinza, segmentação e operações morfológicas como dilatação e erosão. Onde posteriormente é feito o processo de detecção da pupila e com isso é obtido coordenada da pupila. Onde posteriormente é feito processo calibração em que é associado às coordenadas das pupilas com os pontos de calibração, e com isso é feito o processo de estimativa do olhar. Como resultado a ferramenta obteve uma boa taxa de sucesso, porém o autor ressalta que devido a problemas em que o ambiente seja de pouca iluminação a eficácia deste sistema pode ser comprometida.

Assim como o estudo realizado pelo donuk2022cnn o autor papoutsaki2016webgazer apresentou uma ferramenta chamada WEBGAZE, que é um sistema de rastreamento ocular que é integrada a websites, utilizando webcam de baixo valor monetário. Onde o processo de calibração implementado pela ferramenta é orientado a cliques, onde o usuário deve clicar em pontos de referências na tela para que seja feita a associação entre o ponto de calibração e coordenada do olhar do usuário. E com isso a ferramenta consegue estimar o olhar do usuário em tempo real. Os resultados obtidos pelo estudo mostraram que a ferramenta possui uma boa precisão para estimar o olhar do usuário em websites, onde foi obtido um erro médio de 104 Pixel, porém assim como no trabalho de donuk2022cnn em ambientes com baixa de iluminação a precisão é afetada.

Pensando em uma forma de visualizar os dados obtidos através de uma ferramenta de rastreamento ocular, e assim obter o entendimento do comportamento do usuário ao navegar em páginas na web, o Menges *et al.* (2020) apresenta uma ferramenta para mostrar visualmente esses dados. Essa ferramenta fornece os tipos de visualização como mapa de fixações, mapa de calor e scanpath, além disso o agrupamento de dados que possui uma análise mais detalhada como fluxo de atenção. Onde foi realizado uma série de testes para analisar a eficácia da ferramenta. Foi realizado um experimento com 20 usuários navegando

em diferentes sites onde foi solicitado para que cada participante realizasse a navegação conforme é feito no cotidiano deles. Os dados coletados durante esse experimento foram analisados por seis analistas de dados onde foi concluído que a através dos relatórios gerados pela ferramenta obteve uma nota de quatro de cinco para a utilidade. Mostrando que uma ferramenta que fornece informações visuais sobre o comportamento do usuário ao navegar em um website é útil.

Já Vera *et al.* (2017) com o intuito de criar uma alternativa a métricas de desempenho tradicionais para os websites, como número de cliques, duração média em uma determinada página e entre outros. Desenvolveu um sistema chamado AKORI, na qual possui o objetivo de, através de um processo de rastreamento ocular, coletar dados para identificar a atenção visual do usuário em websites. O sistema após realizar o processo de rastreamento ocular, coleta de dados e identificação da atenção visual, é gerado três tipos de relatórios: o mapa de calor, mostrando as regiões predominantes de atenção. Mapa de foco visual que mostra apenas a região de atenção visual, e por fim, o último é mapa de elementos do website, onde é feita uma identificação dos elementos do layout e é feita uma combinação entre a atenção visual identificada com os elementos identificados do website, possibilitando desta forma detectar os elementos que teve a atenção do usuário.

Pensando em uma forma de automatizar o processo de detecção e classificação dos elementos que obtiveram atenção visual os autores Barz e Sonntag (2021) desenvolveram um sistema capaz automatizar o processo de identificação de objetos que o usuário visualizou, onde desenvolveu um sistema, que através de dados coletados de uma ferramenta de rastreamento ocular, identificar automaticamente os objetos na qual o usuário obteve a atenção visual. Onde a partir de uma fixação identificada é realizado o processo de detecção e classificação do objeto. Onde esse processo foi feito através de visão computacional e aprendizado de máquina. E com isso o sistema é capaz de fornecer informações relevantes relacionadas aos objetos que teve a atenção visual do usuário.

Com o objetivo de extrair informações para melhorar a performance de um site, os autores Boardman *et al.* (2022), Hung e Chun-Chia Wang (2021) e Schröter *et al.* (2021), utilizaram a ferramenta de rastreamento ocular para realizar identificação de atenção visual em sites de e-commerce, onde é analisado o comportamento do usuário em relação a mudanças específicas na interface. Onde o autor Boardman *et al.* (2022) usou a ferramenta em um site varejista para detectar as regiões da página inicial que mais chama a atenção do usuário, onde para analisar os dados coletados foi utilizado, mapa de calor, scanpath, e foi delimitado algumas áreas de interesses no layout do site para compreender o comportamento do usuário em relação a essas áreas. Já o autor Hung e Chun-Chia Wang (2021) utilizou a ferramenta para verificar o desempenho de uma interface responsiva de um comércio, onde as informações de atenção visual obtidas foram realizadas para organizar os elementos da interface de acordo com a preferência visual do usuário. E por fim o autor Schröter *et al.* (2021) utilizou a ferramenta para avaliar o

desempenho de um site de compras online, para analisar a atenção visual em situações em que um produto possui foto com e sem um modelo humano.

### 3 MODELAGEM E PROJETO

A arquitetura proposta para a ferramenta de rastreamento ocular que permita a realização da detecção da atenção visual em website consiste em vários módulos responsáveis de funcionalidades específicas, como captura da imagem, rastreamento ocular, calibração, estimativa de coordenadas, análise de dados, gestão do navegador web, dados armazenados, e relatórios. Dessa forma, a arquitetura desenvolvida irá proporcionar uma maior flexibilidade e facilidade, uma vez que cada módulo recebe uma entrada e gerará uma saída.

A Figura 3.1 ilustra a arquitetura proposta, inspirada nos trabalhos de Donuk *et al.* (2022) e Vera *et al.* (2017). A arquitetura é composta a partir da interface do usuário, responsável por realizar a comunicação entre o usuário e o sistema, possibilitando assim, com que seja possível utilizar os recursos da ferramenta de maneira mais simplificada, proporcionando visualizar experimentos já realizados e os relatórios desenvolvidos, assim como, iniciar um novo experimento e realizar configurações da ferramenta.

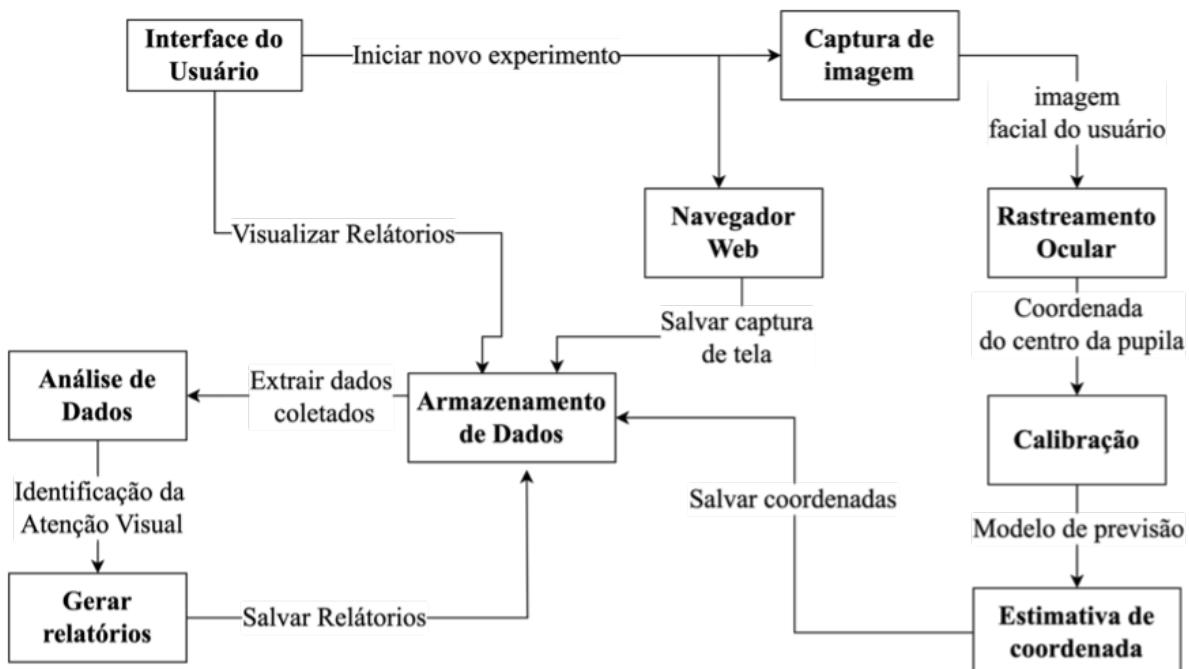


Figura 3.1 – Arquitetura do sistema (Inspirado nos trabalhos de Donuk *et al.* (2022) e Vera *et al.* (2017)).

Para a realização da rotina de um novo experimento, o sistema executa paralelamente os módulos de navegador web, captura de imagem e rastreamento ocular. Onde posteriormente será realizado o processo de calibração e em seguida a estimativa do olhar.

O módulo de navegador web é responsável por realizar a navegação no website e por realizar a captura de tela do que está sendo exibido para o usuário conforme o conteúdo é alterado.

Paralelo a isso, o módulo de captura de imagem tem a funcionalidade de registrar os frames do usuário e fornecer para o módulo de rastreamento ocular, onde os movimentos oculares do usuário são obtidos. A partir disso, obtém-se uma coordenada na qual será utilizado pelo módulo de calibração, onde é feito uma associação entre a coordenada da pupila do usuário com as coordenadas da tela. O resultado deste módulo é um modelo de regressão linear, que através de uma coordenada da posição da pupila é estimado uma posição que o usuário está olhando na tela.

Esse processo é realizado até que o usuário finalize o experimento e ao final é armazenado na base de dados, as informações coletadas. Em seguida o módulo de análise de dados acessa as informações coletadas na base de dados e em seguida, realiza um processamento de dados, na qual vai identificar a atenção visual e a partir disso gerar relatórios para visualização dos elementos que obtiveram a atenção visual do usuário no website.

### 3.1 INTERFACE DO USUÁRIO

A interface do usuário tem o objetivo de facilitar a comunicação entre o usuário e a ferramenta. Essa interface possui cinco telas principais: a página inicial, página de visualização, página de execução, página de configuração e por fim a página de informações.

A Figura 3.2 ilustra a arquitetura da interface do usuário, onde a página inicial exibe as principais funcionalidades do sistema, como realização de experimentos, visualização dos relatórios, configurações e as informações do projeto. Na tela de visualização de relatórios, o usuário pode acessar todos os resultados dos experimentos já realizados. Como, por exemplo, o mapa de calor, mapa de fixações e entre outros. Já na página de configurações o usuário poderá selecionar a webcam desejada para a realização dos experimentos. E por fim, a página de informações, contém todos os detalhes sobre a ferramenta e o projeto realizado.



Figura 3.2 – Arquitetura da interface do usuário.

### 3.2 NAVEGADOR WEB

O módulo de navegador web é responsável por exibir o conteúdo dos sites para o usuário durante o processo de rastreamento ocular, além disso possui a funcionalidade de

realizar capturas de telas conforme o conteúdo exibido é alterado, onde essas imagens são armazenadas para serem utilizadas posteriormente nos processos de geração de relatórios.

Para exibir o conteúdo web, foi desenvolvido um navegador inspirado em Fitzpatrick (2021), na qual é utilizado a biblioteca *PyQt5*, que possibilita a criação de interfaces gráficas, onde é utilizado o módulo *QWebEngineView* para a criação de um componente gráfico do navegador, na qual possibilitar acessar e visualizar o conteúdo dos websites.

A função de captura de tela é realizada de acordo com a alteração do conteúdo exibido pelo navegador, baseado no trabalho de Kiruthika *et al.* (2021), por meio de uma comparação entre as capturas de tela, é verificado se houve alguma alteração, caso exista, essa captura de tela é armazenada, caso contrário é descartado. Essa verificação de imagens, assim como descrito por Kiruthika *et al.* (2021) é realizada através da diferença absoluta entre as imagens, ou seja, esse processo calcula a diferença absoluta das intensidades dos pixels entre as imagens. Onde para que a imagem seja armazenada é necessário que a matriz resultante seja diferente de uma matriz composta de zeros. Onde o cálculo da diferença de intensidade dos pixels entre as imagens é dado pela seguinte equação:

$$C = |A - B|$$

Onde,  $A$  e  $B$  são as matrizes das intensidades dos pixels das imagens,  $C$  é a matriz resultante da diferença entre as duas matrizes.

Por exemplo, considerando duas imagens  $A$  e  $B$  representado na Figura 3.3, ambos com o tamanho de 3 pixels de largura e 3 pixels de altura.



Figura 3.3 – Exemplo de capturas realizadas pelo navegador.

Supondo que a imagem  $A$  e  $B$  possuem os seguintes valores de intensidade dos pixels representado nas matrizes abaixo:

$$A = \begin{bmatrix} 40 & 23 & 11 \\ 232 & 12 & 123 \\ 47 & 13 & 64 \end{bmatrix} \quad B = \begin{bmatrix} 21 & 15 & 5 \\ 85 & 4 & 34 \\ 26 & 11 & 15 \end{bmatrix}$$

Com isso, realizando o cálculo da diferença entre as imagens  $A$  e  $B$ , temos que a matriz resultante  $C$  é dada por:

$$C = \begin{bmatrix} 40 & 23 & 11 \\ 232 & 12 & 123 \\ 47 & 13 & 64 \end{bmatrix} - \begin{bmatrix} 21 & 15 & 5 \\ 85 & 4 & 34 \\ 26 & 11 & 15 \end{bmatrix} = \begin{bmatrix} 19 & 8 & 6 \\ 147 & 8 & 89 \\ 21 & 2 & 49 \end{bmatrix}$$

Portanto, através do cálculo realizado temos que a matriz resultante  $C$ , não é uma matriz de zeros, isso significa que as imagens não são iguais e com isso a imagem  $B$  é armazenada na base de dados.

### 3.3 RASTREAMENTO OCULAR

O rastreamento ocular consiste em uma técnica que a partir de uma entrada de vídeo será realizado uma série de identificações com o intuito de obter-se os movimentos oculares do usuário a partir da imagem de vídeo. A Figura 3.4 ilustra este processo.



Figura 3.4 – Processo de rastreamento ocular.

O processo de rastreamento ocular consiste em receber como entrada uma imagem com a captura facial do usuário e a partir disso realizar as técnicas de rastreio ocular. Inicialmente o processo é feita a detecção da face do usuário, em seguida a detecção da região dos olhos, na qual será realizado o recorte da imagem com a área de interesse, e posteriormente a partir da imagem recortada através de técnicas de processamento de imagem, é feita a identificação da região da íris. Por fim, é realizado o cálculo do centroide da pupila detectada. Por meio desse processo, é obtido uma coordenada da pupila na qual será utilizada para os processos posteriores.

## A. DETECÇÃO DA FACE DO USUÁRIO

A detecção de face é um processo importante para o rastreamento ocular. Uma vez que esse processo irá facilitar os procedimentos posteriores, como detecção da região dos olhos e a detecção da região central da pupila. Esse processo assim como no trabalho Hangaragi *et al.* (2023) é realizado através da biblioteca *Mediapipe* disponibilizada pela google, que possui o recurso de identificar o rosto do usuário a partir de uma imagem.

Para isso, é utilizado a função *Face mesh* que utiliza o modelo de aprendizado de máquina *Blazeface* apresentado no trabalho de Bazarevsky *et al.* (2019), onde a partir de um frame de vídeo como entrada o modelo retorna a região face delimitada. Na qual essa informação é utilizada por um outro modelo apresentado no trabalho de Kartynnik *et al.* (2019) que utiliza redes neurais que identificam os pontos faciais a partir da região delimitada da face, conforme representado na Figura 3.5.

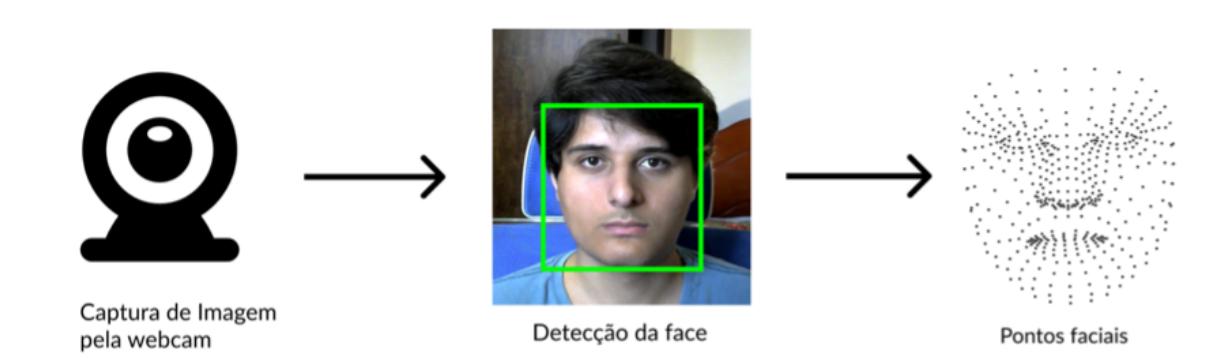


Figura 3.5 – Processo de detecção dos pontos faciais.

Com a identificação da face na imagem, de acordo com Hangaragi *et al.* (2023) é retornado 468 pontos que compõem a malha facial do usuário, onde cada um desses pontos possui uma coordenada ( $x,y,z$ ), onde  $x$  e  $y$  se trata de posição do ponto na imagem, e  $z$  se trata da distância do ponto em relação a webcam. Onde esses pontos são representados na Tabela 3.1 de exemplo.

Tabela 3.1 – Exemplo das informações dos pontos faciais.

Ponto facial	X	Y	Z
1	10	2	3
2	11	3	4
3	14	5	3
4	17	3	4
...	...	...	...
468	342	125	5

De acordo com os dados apresentados na Tabela 3.1, o ponto facial 1, possui a coordenada ( $x, y$ ) de posição na imagem (10, 2) e distância desse ponto para câmera de 3. Para o ponto facial 2, tem coordenada de posição (11, 3) e possui a distância 4 para a

câmera. E assim por diante para todos 468 pontos faciais

## B. DETECÇÃO DOS OLHOS

A execução do processo de detecção dos olhos é realizada mediante a ubiquação dos pontos do contorno do olho, na malha facial do rosto. Onde dos 468 pontos identificados no processo de detecção face, de acordo com Hange Wang *et al.* (2022) 32 pontos correspondem à região dos olhos, sendo 16 pontos para cada um dos olhos. Segundo Aiphile (2022) os pontos que delimitam a região dos olhos representados no Quadro 3.1.

Quadro 3.1 – Pontos que delimitam a região dos olhos.

<b>Pontos do olho esquerdo</b>	362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385, 384, 398
<b>Pontos do olho direito</b>	33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161, 246

Através desses pontos, é realizado o processamento da imagem na qual é feito o recorte da região. Onde esse processo é ilustrado na Figura 3.6.

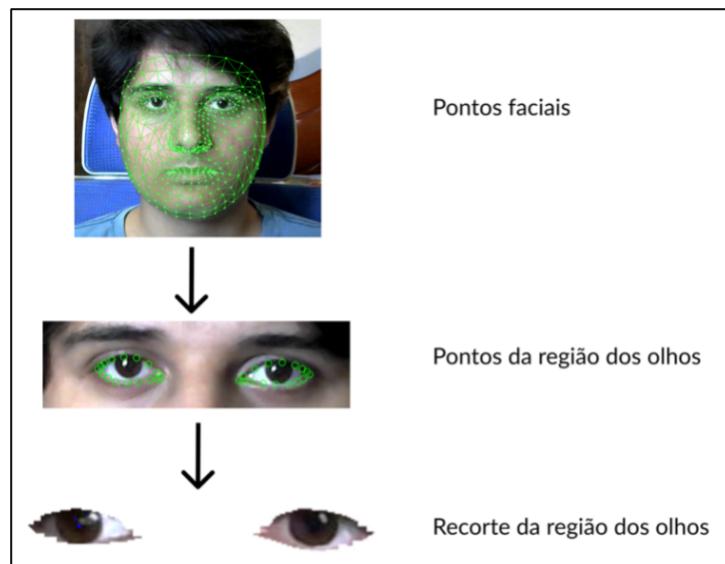


Figura 3.6 – Representação do processo de recorte da região dos olhos.

Na conclusão desse processo são obtidas duas imagens, uma referente a região do olho esquerdo e a outra a região do olho direito.

## C. DETECÇÃO DA PUPILA

O processo de detecção de pupila consiste em pegar a região dos olhos e assim como no trabalho de Brachter e Gerhardt (2020), aplicar técnicas de processamento de imagem, com objetivo de identificar a região central da pupila, que é a região de interesse para o processo de rastreamento ocular.

Inicialmente é feito o processo de conversão da imagem para escala de cinza, esse processo irá facilitar os processos posteriores, uma vez que a cor não é uma informação relevante para o processo (KANAN; COTTRELL, 2012).

Em seguida, é feito o processo que remove qualquer tipo ruído da imagem que possa atrapalhar a detecção da íris, onde é aplicado um filtro gaussiano que realiza uma suavização na imagem resultando em menos ruido (BHUYAN, 2019).

Posteriormente, é feito a binarização da imagem, que é um processo de segmentação que transforma os pixels da imagem em preto ou branco, onde cada pixel presente na imagem, é comparado com um limiar, onde se o valor for maior que esse limiar, então esse pixel é convertido para branco, caso o valor de intensidade seja menor que o limiar, então será convertido para preto. Resultando assim em uma imagem com apenas tons preto e branco destacando as regiões de interesse (BHUYAN, 2019).

Posteriormente, é feito o processo de delimitar a partir da imagem binarizada a região da íris, para isso é feito um processo de detecção de contornos, que fornecer todos os contornos presente na imagem, onde para esse caso é considerado apenas o maior contorno, que se trata da íris.

Por fim, com a região da íris delimitada, é realizado a identificação da posição em coordenadas do centro da pupila. Onde esses processos são representados na Figura 3.7.

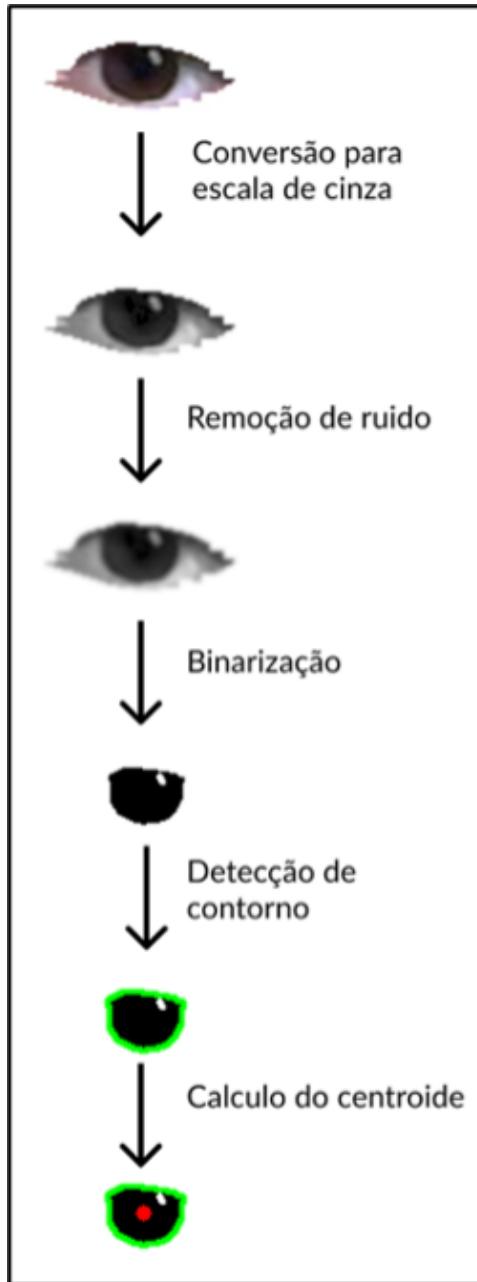


Figura 3.7 – Representação do processo de detecção da pupila.

A realização do cálculo do centroide da pupila, é realizada através de "momentos da imagem", que consiste segundo Nguyn (2019) de um cálculo que fornece a métricas como o raio, área e o centroide de uma imagem binarizada. Onde segundo Spiliotis e Mertzios (1998) pode ser expressa pela seguinte equação:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Onde,  $M_{ij}$  é o momento de ordem  $(i + j)$ , onde  $i$  é a ordem do momento no eixo  $x$  e  $j$  é ordem do momento no eixo  $y$ ,  $(x, y)$  é a intensidade do pixel na posição  $(x, y)$ . Para a realização do cálculo do centroide de acordo com Nguyn (2019) é utilizado os momentos  $M_{00}$ ,  $M_{01}$ , e  $M_{10}$ , em que ambos são calculados a partir das seguintes equações:

$$M_{00} = \sum_x \sum_y I(x, y)$$

$$M_{10} = \sum_x \sum_y x \cdot I(x, y)$$

$$M_{01} = \sum_x \sum_y y \cdot I(x, y)$$

Onde o  $M_{00}$  é o momento de ordem 0 que realizar o somatório de todas as intensidades dos pixels da imagem, o  $M_{10}$  e  $M_{01}$  são o momento de ordem 1 que faz o somatório das intensidades dos pixels na imagem, onde o  $M_{10}$  multiplica o somatório das intensidades dos pixels pela posição do pixel no eixo  $x$ , e o  $M_{01}$  faz multiplicação do somatório pela posição do pixel no eixo  $y$ .

Ainda de acordo com Nguyn (2019), as coordenadas do centroide são dadas pela seguinte equação:

$$C_x = \frac{M_{10}}{M_{00}}$$

$$C_y = \frac{M_{01}}{M_{00}}$$

Onde,  $C_x$  é a coordenada do centroide no eixo  $x$ ,  $C_y$  é a coordenada  $y$  do centroide. Por exemplo, considerando uma imagem binarizada de uma íris, de um tamanho de 4 pixels de altura e 4 pixels de largura. Conforme mostrado na Figura 3.8, uma imagem binarizada possui apenas dois valores de intensidade, o 0 que representa as cores brancas, e o 1 que representa a cor preta.

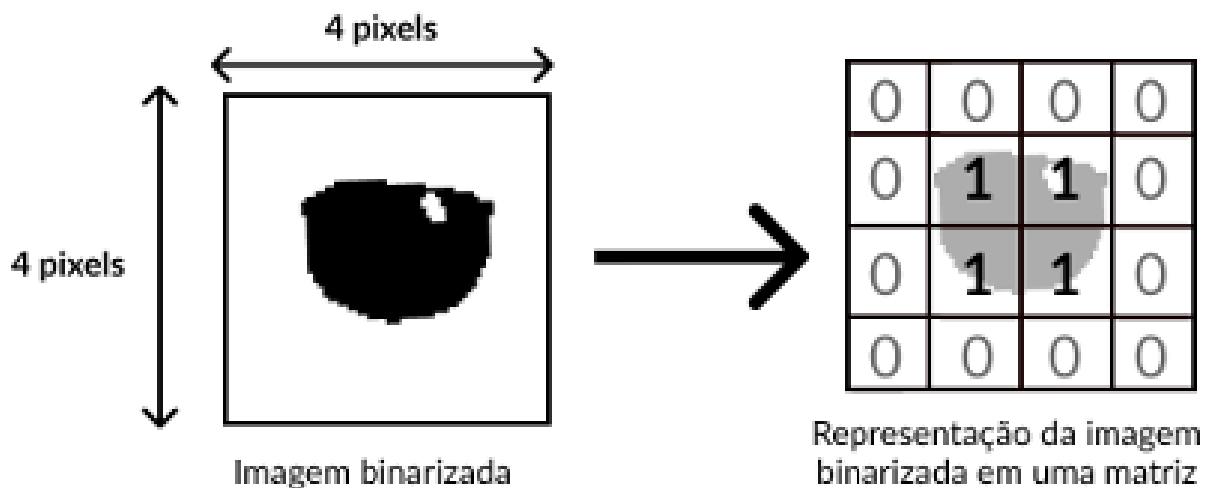


Figura 3.8 – Ilustração da imagem binarizada da íris.

Desta forma, calculando inicialmente o  $M_{00}$ ,  $M_{10}$ ,  $M_{01}$ , a partir dos dados da imagem binarizada da Figura 3.8, temos que:

$$\begin{aligned}
M_{00} &= \sum_x \sum_y I(x, y) \\
&= I(1, 1) + I(1, 2) + I(1, 3) + I(1, 4) + I(2, 1) + I(2, 2) + I(2, 3) + I(2, 4) \\
&\quad + I(3, 1) + I(3, 2) + I(3, 3) + I(3, 4) + I(4, 1) + I(4, 2) + I(4, 3) + I(4, 4) \\
&= (0 + 0 + 0 + 0) + (0 + 1 + 1 + 0) + (0 + 1 + 1 + 0) + (0 + 0 + 0 + 0) = 4
\end{aligned}$$

$$\begin{aligned}
M_{10} &= \sum_x \sum_y xI(x, y) \\
&= 1 \times (I(1, 1) + I(1, 2) + I(1, 3) + I(1, 4)) \\
&\quad + 2 \times (I(2, 1) + I(2, 2) + I(2, 3) + I(2, 4)) \\
&\quad + 3 \times (I(3, 1) + I(3, 2) + I(3, 3) + I(3, 4)) \\
&\quad + 4 \times (I(4, 1) + I(4, 2) + I(4, 3) + I(4, 4)) \\
&= 1 \times (0 + 0 + 0 + 0) + 2 \times (0 + 1 + 1 + 0) \\
&\quad + 3 \times (0 + 1 + 1 + 0) + 4 \times (0 + 0 + 0 + 0) = 10
\end{aligned}$$

$$\begin{aligned}
M_{01} &= \sum_x \sum_y yI(x, y) \\
&= 1 \times (I(1, 1) + I(2, 1) + I(3, 1) + I(4, 1)) \\
&\quad + 2 \times (I(1, 2) + I(2, 2) + I(3, 2) + I(4, 2)) \\
&\quad + 3 \times (I(1, 3) + I(2, 3) + I(3, 3) + I(4, 3)) \\
&\quad + 4 \times (I(1, 4) + I(2, 4) + I(3, 4) + I(4, 4)) \\
&= 1 \times (0 + 0 + 0 + 0) + 2 \times (0 + 1 + 1 + 0) \\
&\quad + 3 \times (0 + 1 + 1 + 0) + 4 \times (0 + 0 + 0 + 0) = 10
\end{aligned}$$

A partir dos cálculos realizados temos que os momentos  $M_{00}$ ,  $M_{10}$  e  $M_{01}$  é dado respectivamente pelos valores 4, 10 e 10. Substituindo esses valores na equação do centroide, é obtido o seguinte resultado.

$$C_x = \frac{M_{10}}{M_{00}} = \frac{10}{4} = 2.5$$

$$C_y = \frac{M_{01}}{M_{00}} = \frac{10}{4} = 2.5$$

Portanto, temos que para o exemplo da imagem binarizada representada na Figura 3.8, que o centroide do centro da pupila possui as coordenadas  $(x, y)$  de  $(2, 5, 2, 5)$  conforme representado na Figura 3.9

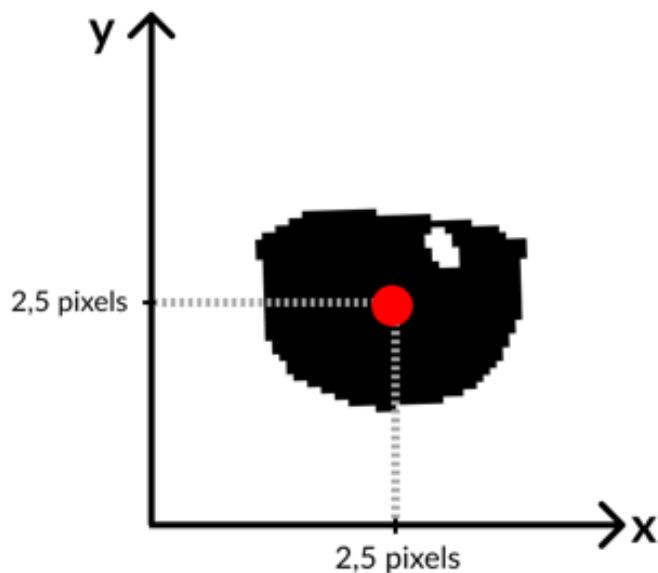


Figura 3.9 – Representação do centro da pupila.

Esse processo é realizado tanto para o olho esquerdo quanto para o olho direito, e ao final é obtido as coordenadas do centro da pupila de ambos os olhos.

### 3.4 CALIBRAÇÃO

O processo de calibração é essencial para realizar a estimativa do olhar do usuário, inspirado nos trabalhos de Brachter e Gerhardt (2020), é feito uma associação entre os pontos de calibração com as coordenadas das pupilas do usuário, onde é definido um modelo de regressão linear, para ser utilizado para prever a posição em que o usuário está olhando na tela em coordenadas (x,y) a partir de uma entrada da coordenada da posição atual da pupila.

A Figura 3.10 representa o processo de calibração, que consiste em apresentar 9 pontos distribuídos uniformemente na tela, assim como feito no trabalho de Brachter e Gerhardt (2020). Onde cada um dos pontos aparece de forma sequencial.

O processo de associação dos pontos de calibração com as coordenadas da pupila do usuário é realizado com o usuário olhando para o ponto exibido na tela e apertando uma tecla específica do teclado, onde ao aperta essa tecla o sistema coleta a coordenada da pupila de ambos os olhos e associa com a coordenada do ponto exibido na tela. Esse processo se repete até finalizar a associação entre todos os pontos de calibração com as coordenadas da pupila do usuário.

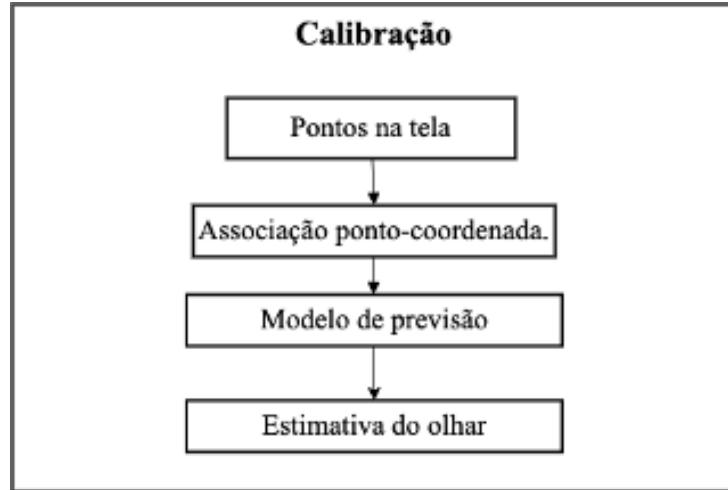


Figura 3.10 – Etapas de calibração.

Por exemplo, considerando uma tela de 1920x1080, na qual é definido 9 pontos de calibração distribuído de forma uniforme, onde cada ponto possui uma coordenada ( $x, y$ ), conforme representado na Figura 3.11.

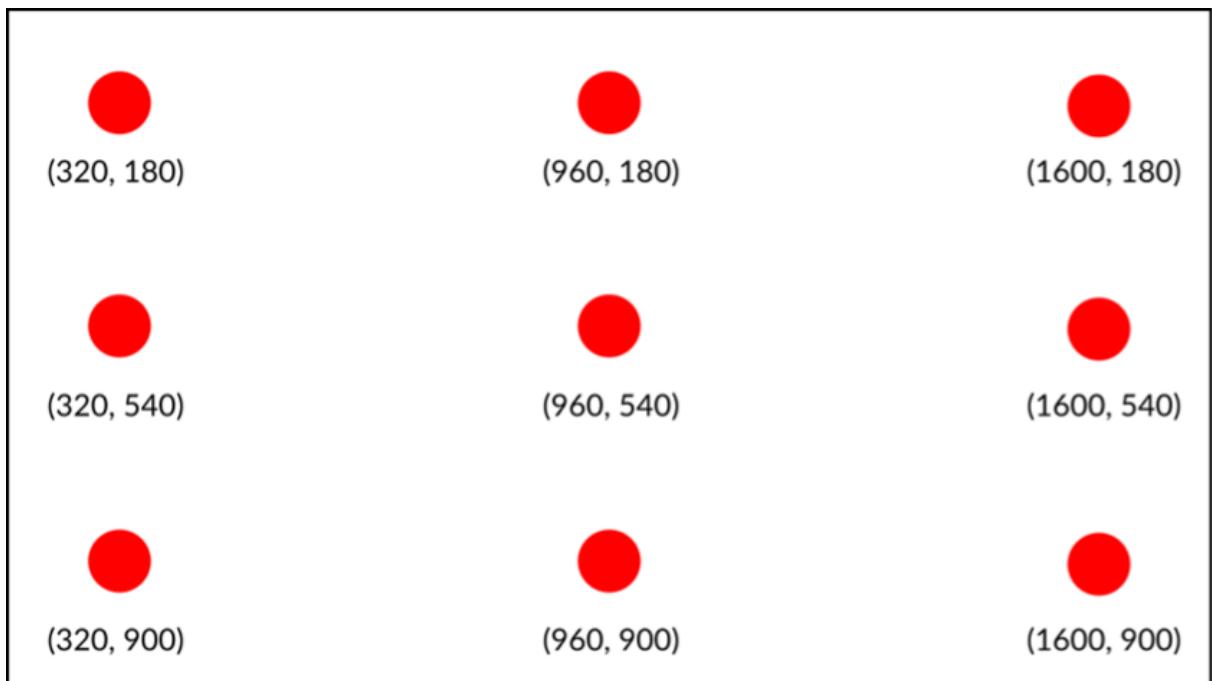


Figura 3.11 – Representação dos pontos de calibração exibido na tela.

Conforme o usuário for olhando para os pontos apresentados e apertando uma tecla no teclado, é feito a associação de pontos de calibração com as coordenadas da pupila do usuário, onde ao final desse processo é obtido como resultado uma tabela com as coordenadas ( $x, y$ ) dos pontos de calibração, e também o ponto médio entre as coordenadas da pupila de ambos os olhos, assim como apresentado na Tabela 3.2 de exemplo:

Tabela 3.2 – Exemplo dos pontos de calibração associado com as coordenadas das pupilas.

Ponto de calibração (x, y)	Pupila direita (x,y)	Pupila esquerda (x,y)	Ponto médio das pupilas (x,y)
(320, 180)	(120, 450)	(150, 480)	(135, 265)
(960, 180)	(340, 670)	(370, 600)	(355, 635)
(1600, 180)	(560, 780)	(590, 710)	(575, 745)
(320, 540)	(890, 230)	(820, 260)	(855, 245)
(960, 540)	(450, 560)	(480, 590)	(465, 575)
(1600, 540)	(670, 120)	(690, 130)	(680, 125)
(320, 900)	(230, 890)	(250, 870)	(240, 880)
(960, 900)	(780, 340)	(790, 350)	(785, 345)
(1600, 900)	(900, 120)	(920, 110)	(910, 115)

De acordo com os dados apresentado na Tabela 3.2, os pontos de calibração que possui as coordenadas de (320, 180) está associado com as coordenadas (120, 450) da pupila do olho direito e com as coordenadas (150, 480) para a pupila do olho esquerdo, e o ponto médio possui as coordenadas (135, 265) para os ambos os olhos. E assim por diante para todos os pontos de calibração.

Após esse processo de associação dos pontos de calibração com as coordenadas da pupila do usuário, é criado dois modelos de regressão linear, um para coordenada  $x$  e outro para coordenada  $y$ . Onde por meio desses modelos é possível estimar a partir de uma coordenada do ponto médio da pupila de entrada  $x$  e  $y$  uma coordenada de saída na qual representa o ponto na qual o usuário está olhando na tela. Onde o modelo de regressão linear segundo Weisberg (2005) é definido pelas seguintes equações:

$$Y = \beta_0 + \beta_1 \chi$$

Onde,  $Y$  é a variável dependente;  $\chi$  é variável independente; e o  $\beta_0$  é a constante,  $\beta_1$  é coeficiente de  $\chi$ .

Para calcular o  $\beta_0$  e  $\beta_1$  é utilizado as seguintes equações:

$$\beta_0 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\beta_1 = \bar{y} - \beta_0 \bar{x}$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

Por exemplo, considerando os pontos de calibração e as coordenadas do ponto médio da pupila do usuário apresentado na Tabela 3.2, é criado duas funções de regressão linear uma para coordenada  $x$  e outra para coordenada  $y$ .

Começando pela coordenada  $x$ , onde  $x_1$  é a coordenada  $x$  do ponto médio da pupila e  $x_2$  é coordenada  $x$  do ponto de calibração, e temos que  $\bar{x}_1$  e  $\bar{x}_2$ , como a média do somatório das coordenadas  $x$  do ponto médio da pupila e do ponto de calibração, respectivamente dividido por 9, que é quantidade de pontos de calibração. Com isso é feita a Tabela 3.3 com os resultados dos cálculos para todos os pontos de calibração e ponto médio da pupila do usuário no eixo  $x$ .

Tabela 3.3 – Resultado dos cálculos para cada ponto de calibração e ponto médio da pupila no eixo x.

$x_1$	$x_2$	$(x_1 - \bar{x}_1)$	$(x_2 - \bar{x}_2)$	$(x_1 - \bar{x}_1) \times (x_2 - \bar{x}_2)$	$(x_1 - \bar{x}_1)^2$
135,0	320,0	-223,3	-640,0	142897,8	49853,0
355,0	960,0	-3,3	0,0	0,0	10,7
575,0	1600,0	216,7	640,0	138702,2	46968,5
139,0	320,0	-219,3	-640,0	140337,8	48082,7
361,0	960,0	2,7	0,0	0,0	7,4
580,0	1600,0	221,7	640,0	141902,2	49160,7
139,5	320,0	-218,8	-640,0	140017,8	47863,7
359,5	960,0	1,2	0,0	0,0	1,5
580,5	1600,0	222,2	640,0	142222,2	49382,7

A partir dos resultados obtidos da Tabela 3.3, é calculado o  $\beta_{0x}$  e  $\beta_{1x}$  para coordenada  $x$ , mostrado nas equações abaixo:

$$\beta_{0x} = \frac{\sum_{i=1}^9 (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{\sum_{i=1}^9 (x_{1i} - \bar{x}_1)^2} = \frac{846080}{291331.1} = 2.9$$

$$\beta_{1x} = \bar{y} - \beta_{0x}\bar{x} = 960 - 2.9 \times 358.3 = -80.5$$

Com isso, temos que o  $\beta_{0x}$  é dado por 2,9 e  $\beta_{1x}$  possui o valor de -80,5. Substituindo os valores de  $\beta_{0x}$  e  $\beta_{1x}$  na função da regressão linear temos que o modelo de previsão para o eixo  $x$  é dado por:

$$Y_x = \beta_{0x} - \beta_{1x}\chi = 2.9 - 80.5\chi$$

Para coordenada  $y$ , onde  $y_1$  e  $y_2$  são respectivamente as coordenadas  $y$  do ponto médio da pupila do usuário e do ponto de calibração. Onde  $\bar{y}_1$  e  $\bar{y}_2$  são a média do somatório das coordenadas  $y$  dividido pela quantidade de pontos de calibração, para os pontos médios da pupila e para o ponto de calibração, respectivamente. Com isso é montado a Tabela 3.4 com os resultados dos cálculos a partir dos pontos de calibração e ponto médio da pupila do usuário no eixo  $y$ .

Tabela 3.4 – Resultado dos cálculos para cada ponto de calibração e ponto médio da pupila no eixo y.

$y_1$	$y_2$	$(y_1 - \bar{y}_1)$	$(y_2 - \bar{y}_2)$	$(y_1 - \bar{y}_1) \times (y_2 - \bar{y}_2)$	$(y_1 - \bar{y}_1)^2$
265,0	180,0	-335,4	-360,0	120740,0	112485,7
269,5	180,0	-330,9	-360,0	119120,0	109487,5
271,5	180,0	-328,9	-360,0	118400,0	108167,9
645,0	540,0	44,6	0,0	0,0	1990,2
648,0	540,0	47,6	0,0	0,0	2266,8
651,5	540,0	51,1	0,0	0,0	2612,3
880,0	900,0	279,6	360,0	100660,0	78182,4
885,5	900,0	285,1	360,0	102640,0	81288,3
887,5	900,0	287,1	360,0	103360,0	82432,8

Com os resultados obtidos da Tabela 3.4, é calculado o  $\beta_{0y}$  e  $\beta_{1y}$  para coordenada  $x$ , mostrado nas equações abaixo:

$$\beta_{0y} = \frac{\sum_{i=1}^9 (y_{1i} - \bar{y}_1)(y_{2i} - \bar{y}_2)}{\sum_{i=1}^9 (y_{1i} - \bar{y}_1)^2} = \frac{664920}{578913.8} = 1.14$$

$$\beta_{1y} = \bar{y}_2 - \beta_{0y}\bar{y}_1 = 540 - 1.14 \times 600.4 = -149.6$$

Com isso, temos que 1,14 e -149,6 é respectivamente o  $\beta_{0y}$  e  $\beta_{1y}$ . Substituindo esses valores na função de regressão linear, é obtido o seguinte modelo de previsão para o eixo  $y$ :

$$Y_y = \beta_{0y} + \beta_{1y}\chi = 1.14 + 149.6\chi$$

Com os modelos de regressão linear definidos, tanto para o eixo  $x$  quanto para o eixo  $y$ , é possível realizar a estimativa do olhar, que é um processo fundamental para o contexto de rastreamento ocular, pois é através dele que podemos identificar para onde que o usuário está olhando e uma determinada tela, e a partir disso, compreender o comportamento do usuário. Esse processo consiste em receber uma coordenada da posição do ponto médio atual da pupila do usuário e retornar uma posição estimada na qual usuário está olhando na tela.

Por exemplo, através dos modelos de regressão linear definidos anteriormente e supondo que as coordenadas  $(x, y)$  da posição atual do ponto médio da pupila do usuário é de (250, 500). Temos que a coordenada estimada do olhar é dada por:

$$Y_x = 2.9 - 80.5 \times 250 = -19937.5$$

$$Y_y = 1.14 + 149.6 \times 500 = 74961.14$$

Com isso, de acordo com os resultados dos cálculos realizados, temos que a coordenada  $(x, y)$  que o usuário está olhando na tela é de (645,5, 435,7).

### 3.5 ANÁLISE DE DADOS

O processo de análise de dados no contexto desse projeto consiste em extrair os dados coletados durante o processo de rastreamento ocular e identificar informações relevantes, na qual possam ser utilizadas para a tomada de decisão. Essa etapa é fundamental para obter a compreensão dos padrões de comportamento do usuário ao navegar em um website.

Inicialmente é realizado identificação da atenção visual através do cálculo da dispersão dos pontos do olhar, e a etapa seguinte é responsável por gerar os relatórios que mostram visualmente as áreas detectadas de atenção do usuário no website.

#### 3.5.1 IDENTIFICAÇÃO DA ATENÇÃO VISUAL

A etapa de identificação de atenção visual consiste em calcular a dispersão dos pontos estimados dos dados obtidos durante o processo de rastreamento ocular. Para isso foi utilizado o algoritmo de *Dispersion-Threshold Identification* (I-DT) do trabalho de Salvucci e Goldberg (2000), que consiste em calcular a dispersão para detectar as fixações. A dispersão é calculada como a distância entre o ponto central de uma coordenada com os demais pontos ao seu redor. Em que é definido um limiar de dispersão, caso a dispersão seja menor que esse limiar, então a coordenada é considerada parte da fixação. A equação pode ser representada da seguinte forma:

$$D = [\max(x) - \min(x)] + [\max(y) - \min(y)]$$

Onde,  $D$  é o valor da dispersão;  $\max(x)$ : é maior valor da coordenada do ponto no eixo  $x$ ;  $\min(x)$  é o menor valor da coordenada do ponto no eixo  $x$ ;  $\max(y)$  é o maior valor da coordenada do ponto no eixo  $y$ ;  $\min(y)$  é o menor valor da coordenada do ponto no eixo  $y$ .

Por exemplo, considerando um limiar de 20 pixels de dispersão e 19 pontos do olhar, a verificação se esses pontos fazem parte de uma fixação, é feita através do cálculo da dispersão entre esses pontos, onde o resultado tem que dar um valor menor ou igual á 20 pixels para ser considerado uma fixação. Desta forma, os pontos são representados na Figura 3.12.

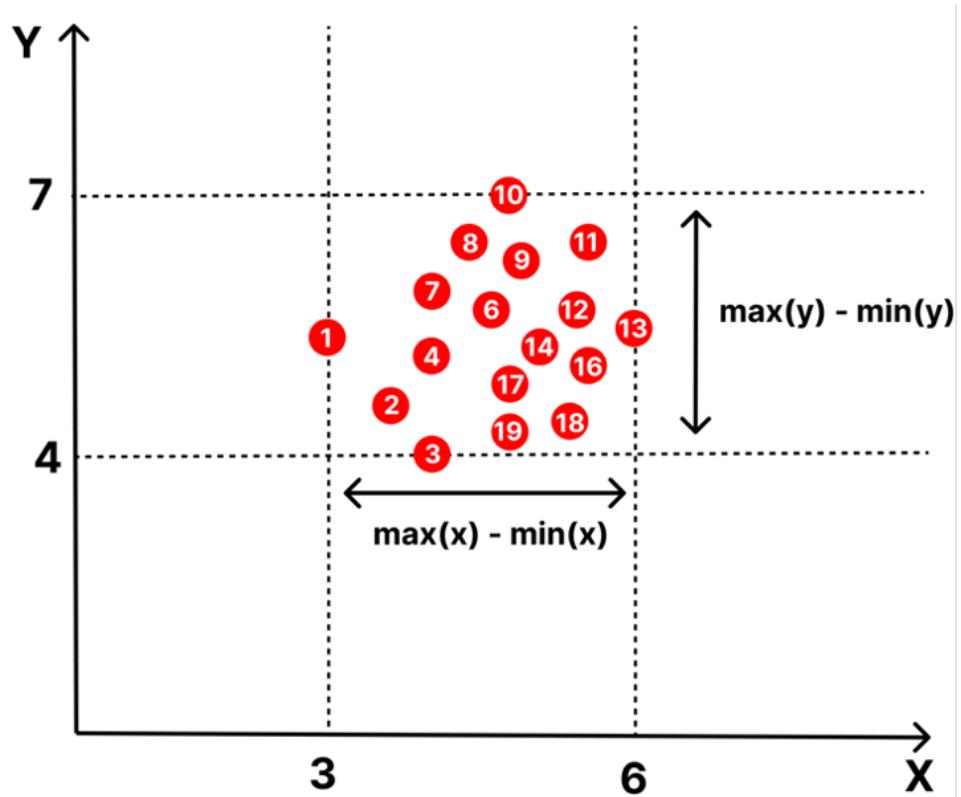


Figura 3.12 – Exemplo de pontos do olhar.

Conforme exibido na Figura 3.12, o ponto 13 possui o valor 6 que é o maior valor para coordenada no eixo  $x$ , enquanto o menor valor é o 3 para o ponto 1. Já para o eixo  $y$ , o maior e menor valor é respectivamente 7 e 4 para os pontos 10 e 3. Adicionando essas informações na equação da dispersão, temos que:

$$D = [6 - 3] + [7 - 4] = 6$$

Com isso, é obtido o valor de 6 pixels de dispersão entre os pontos, como esse valor é inferior ao valor do limiar de 20 pixels, portanto, esses pontos fazem parte de uma fixação.

### 3.5.2 GERAÇÃO DE RELATÓRIOS

O processo de geração de relatórios foi inspirado nos trabalhos de Vera *et al.* (2017) e Menges *et al.* (2020), onde consiste em mostrar as regiões e os elementos do website que obteve a atenção visual. Onde foram criados quatro tipos de relatórios, o mapa de fixação, scanpath, mapa de calor e por fim os elementos web que obteve atenção visual.

#### A. MAPA DE FIXAÇÕES

O Mapa de fixações consiste em mostrar as imagens capturadas durante o experimento com todos os pontos fixações coletadas sobrepostas na imagem (BERGSTROM;

SCHALL, 2014). Para realização desse processo, o sistema irá acessar os dados coletados referente àquele experimento e irá extrair as imagens juntamente com as fixações identificadas para cada imagem.

A partir disso, é utilizado técnicas de processamento de imagem, para cada posição de fixações identificada pelo sistema, é desenhado círculos na imagem de tamanho variável de acordo com a quantidade de pontos presente na fixação, onde quanto maior o círculo maior será a quantidade de pontos pertencente aquela fixação e consequentemente maior será a atenção do usuário naquela região, conforme representado na Figura 3.13.

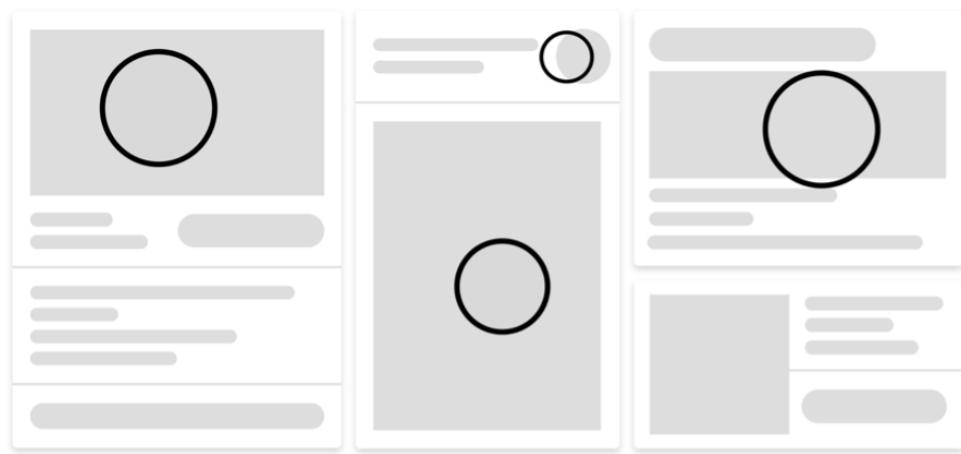


Figura 3.13 – Representação do mapa de fixação.

Por exemplo, considerando uma captura de tela com 400 pixels de largura e altura, e 3 pontos de fixações, onde para cada fixação será utilizada as informações de coordenada central da fixação e a quantidade de pontos que fazem parte dessa fixação. Esse processo consiste desenhar um círculo na imagem na posição referente às coordenadas (x,y) central da fixação, e o tamanho do raio é definido proporcionalmente a quantidade de pontos que pertencem a fixação, onde é multiplicado por um fator de escala de 10 pixels, com o objetivo de ter uma melhor visualização. Com isso, temos os seguintes círculos na captura de tela, conforme pode ser visualizado na Figura 3.14.

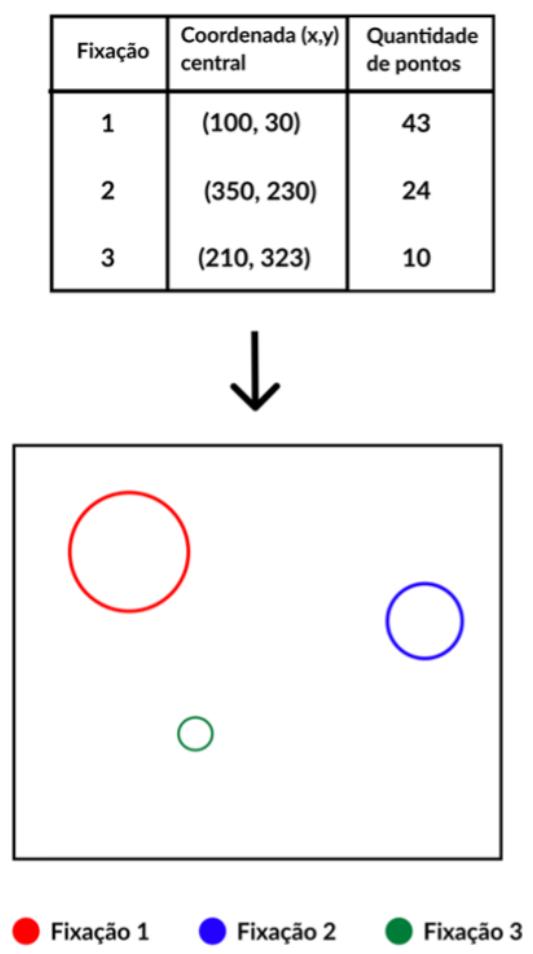


Figura 3.14 – Exemplo do processo para gerar o mapa de fixação.

De acordo com as informações apresentadas na Figura 3.14, para fixação 1, é desenhado um círculo na posição de coordenada (100, 137), onde essa fixação possui 43 pontos do olhar, com isso multiplicando pelo fator de escala 10, temos que o raio do círculo é de 430 pixels. Já para fixação 2 é desenhado um círculo na coordenada (350, 230) com o raio de 240 pixels. Por fim, para a fixação 3 é desenhado um círculo na posição (210, 323) com o raio de comprimento de 100 pixels.

## B. SCANPATH

O Scanpath se trata de um tipo de visualização que mostra em sequência o caminho percorrido pelo olhar do usuário durante a navegação (BERGSTROM; SCHALL, 2014). Para a geração desse tipo de relatório, o sistema irá acessar os dados referentes ao experimento, extraiendo as capturas de tela e as fixações referente a cada captura realizada.

Conforme representado na Figura 3.15, o sistema irá desenhar na imagem para cada fixação um círculo com tamanho variável de acordo com a quantidade de pontos estimados pertencentes aquela fixação, na qual significa que quanto maior o círculo, maior será a o tempo de atenção visual naquele ponto. Além disso, o sistema irá desenhar uma

linha conectando os pontos de fixação, destacando o caminho percorrido pelo olhar do usuário durante a navegação.

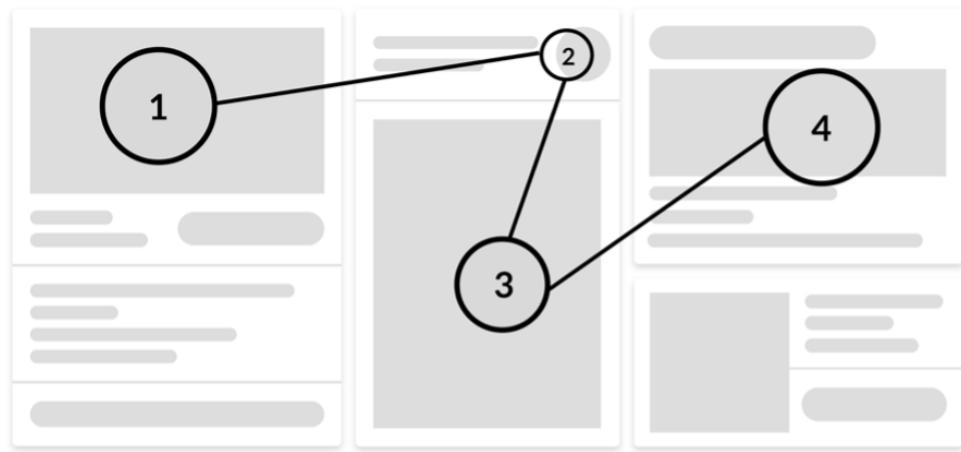


Figura 3.15 – Representação do Scanpath.

Por exemplo, considerando o mesmo exemplo da fixação, onde tem uma captura de tela 400 pixels de largura e altura e 3 pontos de fixações. Com isso, é desenhado um círculo para cada uma das fixações, com o número dentro desse círculo que representa a sequência da fixação, e uma linha conectando as fixações. Onde cada um dos círculos possui um raio de tamanho variável, que é definido multiplicando a quantidade de pontos das fixações por um fator de escala 10. A partir disso, temos o seguinte scanpath representado na Figura 3.16:

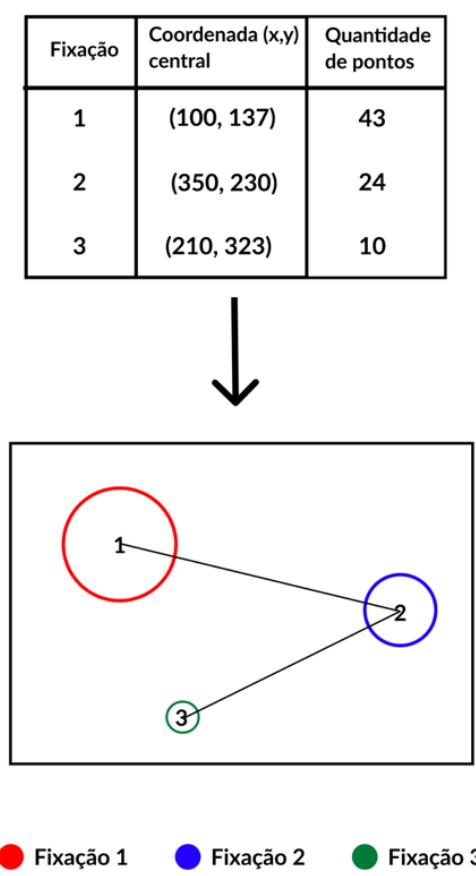


Figura 3.16 – Exemplo do processo para gerar o scanpath.

Com isso, conforme representado na Figura 3.16, temos que a fixação 1 foi visualizada primeiro, na qual possui a posição de (100, 137) de raio de 430 pixels, em seguida a fixação 2 foi visualizada, na posição central de (350, 230) com raio de 240 pixels, e por fim a fixação 3 foi a última a ser visualizada, na posição (210, 323) contendo um raio de 100 pixels.

### C. MAPA DE CALOR

O mapa de calor é um tipo de visualização que mostra nas imagens capturadas as regiões em que foram mais visualizadas durante o experimento para cada captura de tela realizada (BERGSTROM; SCHALL, 2014). Para realização desse tipo de relatório é utilizado a biblioteca *Pygaze* de código aberto do trabalho de Dalmajer *et al.* (2014), onde é disponibilizado uma série de recursos para auxiliar no processo de rastreamento ocular, sendo um deles o módulo de mapa de calor.

Onde inicialmente esse módulo extrair as capturas de tela e as coordenadas do experimento realizado, onde para cada captura realizada é construído uma matriz do mesmo tamanho da imagem em pixels, tanto em largura quanto em altura. Em seguida, essa matriz será preenchida com zeros, e a partir disso, para cada ponto estimado referente aquela imagem capturada, será incrementado 1 na matriz na posição referente a coordenada

estimada, e nas posições vizinhas em um raio de 50 pixels. Ao final do processo de incremento, a matriz final terá os valores referentes a quantidade de frequência das regiões visualizadas.

Após o processo de incrementação das frequências dos pontos de estimativa do olhar, é realizado o agrupamento dessas frequências em 5 grupos de cores, definidas com base na biblioteca *Pygaze*, conforme representado na Tabela 3.5 abaixo:

Tabela 3.5 – Grupo de cores (Baseado na biblioteca *Pygaze*).

Grupo	Quantidade de Frequências	Cor
1	1 a 5	Azul
2	6 a 10	Ciano
3	11 a 15	Verde
4	16 a 25	Amarelo
5	Maior que 25	Vermelho

Em seguida, é substituído na matriz as frequências pela cor do grupo correspondente, onde as regiões mais visualizadas serão representadas por cores mais quentes como vermelho e amarelo, e as regiões menos visualizadas serão representadas por cores mais frias como azul, ciano e verde. Onde desta forma essas cores são sobrepostas na imagem capturada, resultando em uma imagem de mapa de calor, assim como representado na Figura 3.17.

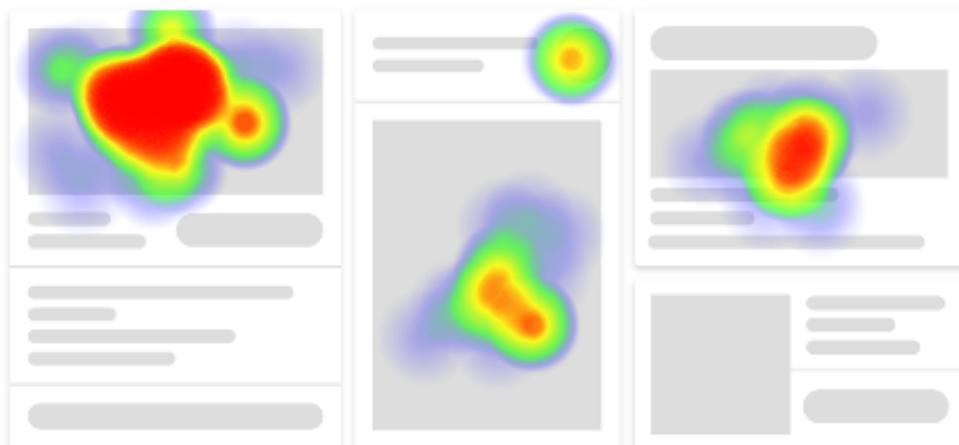


Figura 3.17 – Representação do mapa de calor aplicado.

Por exemplo, considerando uma imagem que possui 10 pixels de largura e altura, com isso é criado uma matriz com 10 colunas e 10 linhas conforme representada na Figura 3.18:

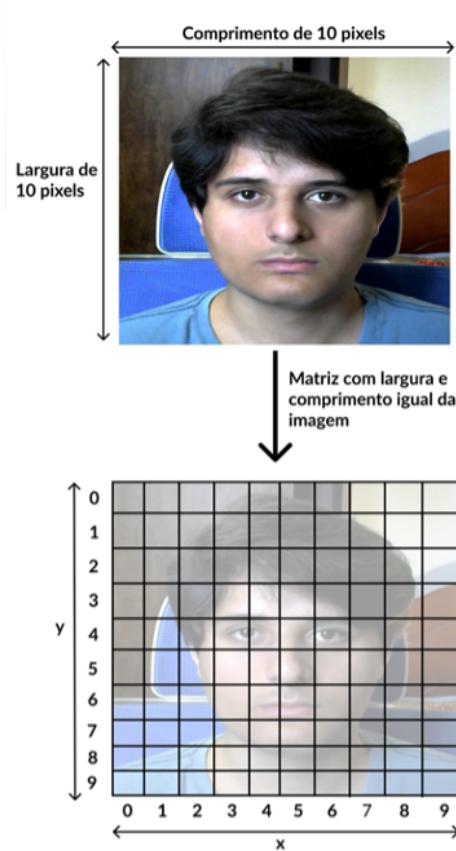


Figura 3.18 – Representação da imagem em uma matriz.

Em seguida, a matriz é preenchida com o número zero, conforme representado na matriz abaixo:

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

Considerando os pontos (3,2), (4,5) (1,4), (4,5),(3,2),(3,2) e (3,3) pontos do olhar e também um raio de 1 pixel, para simplificar o exemplo. Começando pelo ponto (3,2), é incrementado 1 na matriz nas posições (3,2) e nas posições vizinhas de raio de 1 pixel, nas posições vizinhas (2, 1), (3, 1), (4, 1), (2, 2), (4, 2), (2, 3), (3,3) e (4, 3). Com isso a

matriz atualizada é dada por:

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

Em seguida é incrementado 1 na matriz na posição (4,5) e nas posições vizinhas (4, 4), (3, 4), (5, 4), (3,5) (5, 5), (3,6), (4,6) e (5, 6) onde temos que a matriz resultante é dada por:

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

Esse processo se repetir para os pontos restantes, (1,4), (4,5), (3,2), (3,2) e (3,3), onde ao final é obtido a seguinte matriz:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 3 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 5 & 4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 3 & 3 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Com isso, após o processo de preenchimento da matriz com a frequência de cada ponto estimado, é substituído as frequências pelas cores dos grupos correspondente, para

simplificação do exemplo, é considerado que os valores das frequências 1, 2, 3, 4, 5 presenta na matriz resultante, faça parte do grupo 1, 2, 3, 4 e 5 respectivamente. Com isso, temos que o mapa de calor resultante é representado na Figura 3.19.

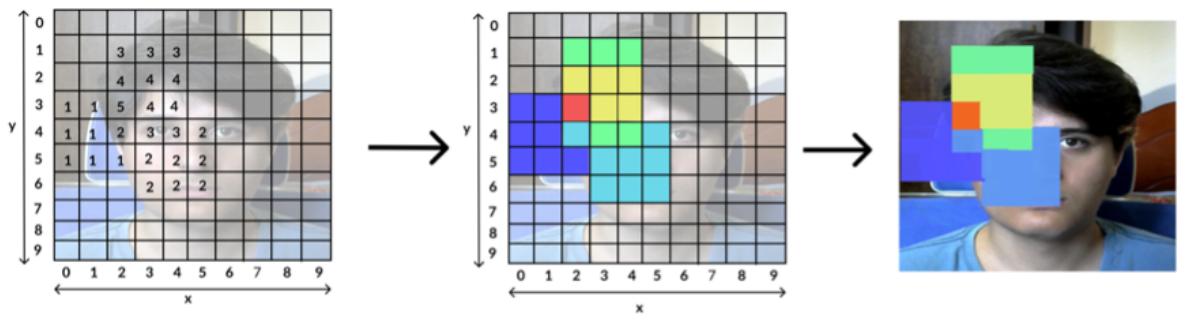


Figura 3.19 – Processo de criação do mapa de calor.

Conforme ilustrado na Figura 3.19, temos que a região que possui a cores vermelha e amarelo foram as regiões que tiveram os maiores valores de frequências, ou seja, foram as regiões mais visualizadas. Já as cores mais frias como azul e ciano foram as regiões menos visualizadas.

### C. ATENÇÃO VISUAL EM ELEMENTOS WEB

Inspirado no trabalho dos autores Barz e Sonntag (2021), o relatório de atenção visual em elementos irá extrair de cada imagem capturada durante o processo de rastreamento ocular as regiões que tiveram fixações. Além disso, irá classificar se essa região se trata de um botão, uma imagem ou um texto. Esse processo é dividido em três etapas: detecção dos elementos do site, classificação de elementos e intersecção dos elementos com as fixações.

#### I. DETECÇÃO DE ELEMENTOS DO SITE

A detecção de elementos do site foi inspirada nos trabalhos dos autores Goyal *et al.* (2021) e Bukhari *et al.* (2011), onde o processo é feito utilizando a técnica de processamento de imagem, que consiste em transformar a imagem para uma escala de cinza, aplicação de um filtro de suavização, limiarização, método canny para a detecção de bordas e por fim a busca por contornos.

Inicialmente a partir de uma captura de tela, é realizado uma conversão da imagem para escala de cinza, pois as cores presentes nas imagens, não será levado em consideração durante o processo. Além disso, esse processo fará com que diminua a demanda computacional, além de facilitar os processos posteriores (KANAN; COTTRELL, 2012). Esse processo é representado na Figura 3.20.



Figura 3.20 – Ilustração do processo de conversão de uma imagem para escala de cinza.

Em seguida, é feito o processo de binarização da imagem, onde é aplicado um limiar em cada pixel presente na imagem, em que irá converter os pixels com valores de intensidade maior que o limiar para branco, e os pixels com valores de intensidade inferior para preto, resultando desta forma uma imagem binarizada com apenas duas cores, preto e branco (BHUYAN, 2019). Destacando assim, os pontos de interesse na imagem, conforme é representado na Figura 3.21.

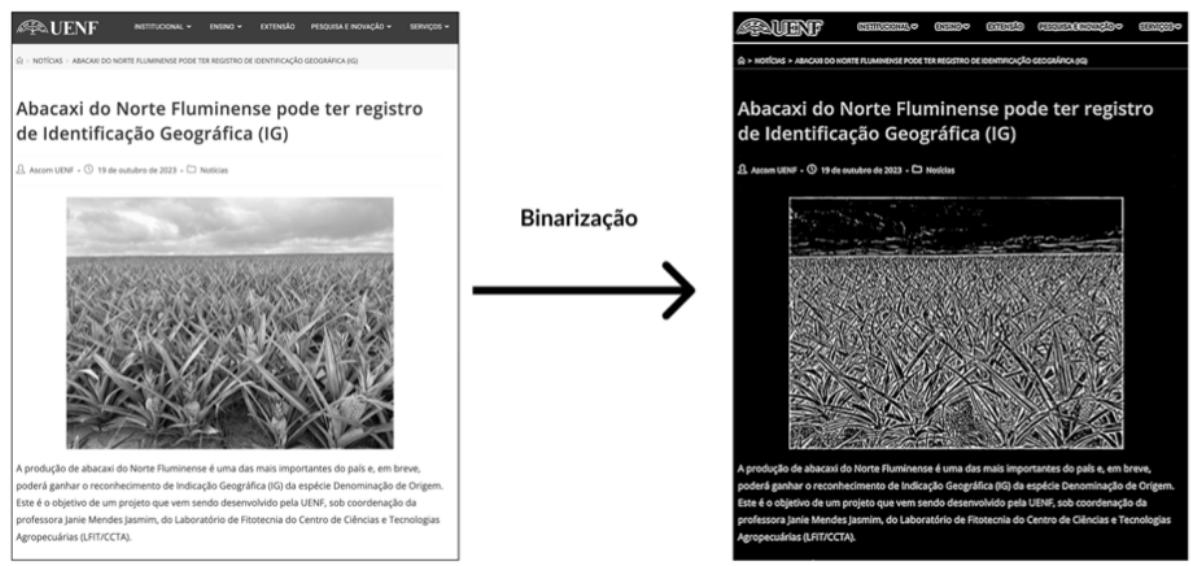


Figura 3.21 – Ilustração do processo de binarização de uma imagem.

Posteriormente, é aplicado o método de Canny com o objetivo de destacar as bordas presentes na imagem binarizada, conforme representado na Figura 3.22:

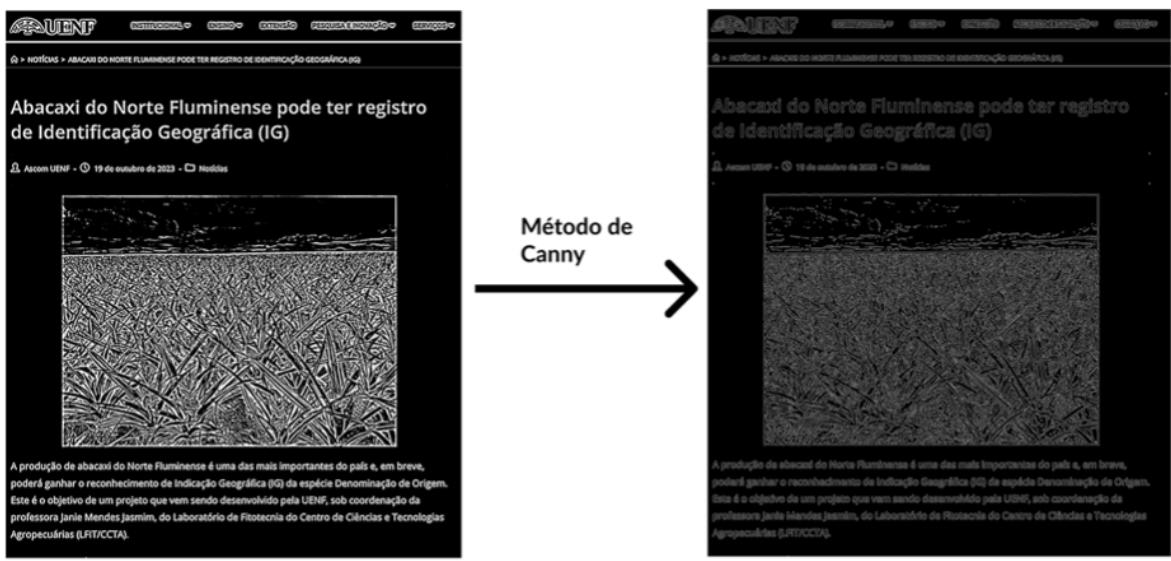


Figura 3.22 – Ilustração do processo do método de Canny aplicado em uma imagem.

Com as bordas detectadas, é aplicado o processo de dilatação, que consiste em expandir as regiões da cor branca presente na imagem, proporcionando assim, a remoção de ruídos e a conexão de objetos próximos ao realizar o preenchimento de pequenos espaçamentos entre objetos (SRISHA; KHAN, 2013). Onde é apresentado na Figura 3.23.

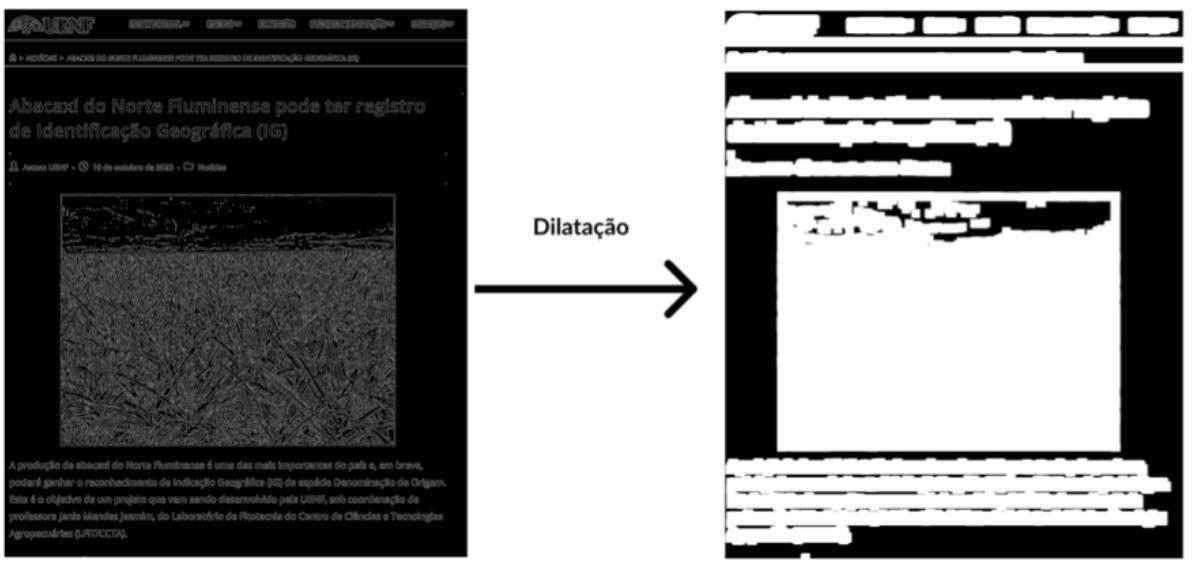


Figura 3.23 – Ilustração do processo de dilatação aplicado em uma imagem.

Posteriormente é realizado o procedimento de erosão, que realizar o processo inverso da dilatação, onde é retraído as regiões da cor branca e a expansão das cores pretas, desta forma, diminuindo os ruídos (SRISHA; KHAN, 2013). o resultado desse processo é representado na Figura 3.24.

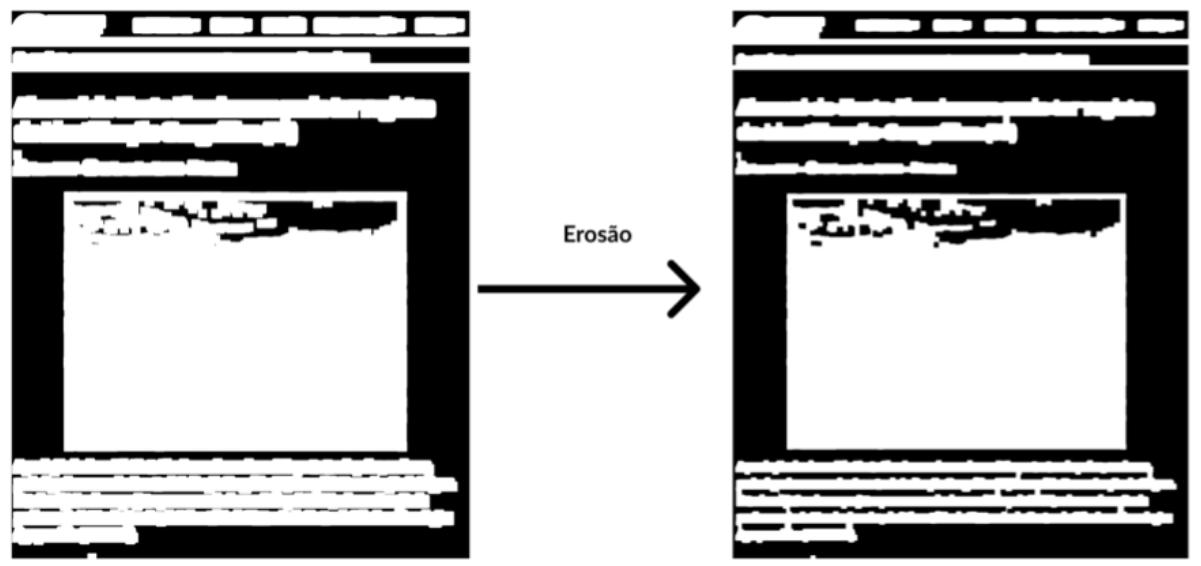


Figura 3.24 – Ilustração do processo de erosão aplicado em uma imagem.

Por fim, após a realização dos procedimentos anteriores de processamento de imagem, é realizado o processo de detecção de contornos. Onde o resultado é apresentado na Figura 3.25, onde os elementos detectados estão delimitados.



Figura 3.25 – Exemplo do resultado de detecção de elementos em uma captura de tela.

## II. CLASSIFICAÇÃO DE ELEMENTOS

Após o processo de recorte dos elementos fixados, a próxima etapa é inspirada nos trabalhos de Goyal *et al.* (2021) e Barz e Sonntag (2021), onde é realizado a classificação desses elementos, para isso é utilizado *Support Vector Machine* (SVM), que se trata de um algoritmo de aprendizagem de máquina para realização de classificações de elementos.

Para isso, são criadas três classes de elementos, botão, figuras e textos. Onde para cada classe é criado um conjunto de dados de treinamento, onde possui ao todo 556 imagens, 198 botões, 183 figuras e 175 textos.

Com isso utilizando o SVM, a classificação é realizada a partir de uma imagem de entrada em que é feito uma extração de características, onde o algoritmo SVM verifica qual das classes possui as características mais próximas da imagem da entrada. A Figura 3.26 mostra um exemplo de elementos delimitados e classificados, onde os elementos em vermelho são textos, o de verde são figuras e o de azul são botões.



Figura 3.26 – Exemplo de elementos classificados.

## III. INTERSECÇÃO DOS ELEMENTOS COM AS FIXAÇÕES

A etapa de Intersecção dos elementos com as fixações, irá realizar a associação dos elementos extraídos do website com as fixações identificadas. Assim como no trabalho de Sun *et al.* (2019) para identificação da sobreposição de objetos, será utilizado o algoritmo de *Intersection over Union* (IoU) que consiste em realizar um cálculo que mede a sobreposição de entre dois objetos, onde é definido pela área de interseção entre os dois objetos, dividido

pela área da união. Resultando em um valor que varia entre 0 e 1, onde quanto mais próximo de 1 maior será a sobreposição entre os dois objetos, e quanto mais próximo de 0 menor será a sobreposição (PADILLA *et al.*, 2020). Esse processo é representado na Figura 3.27.

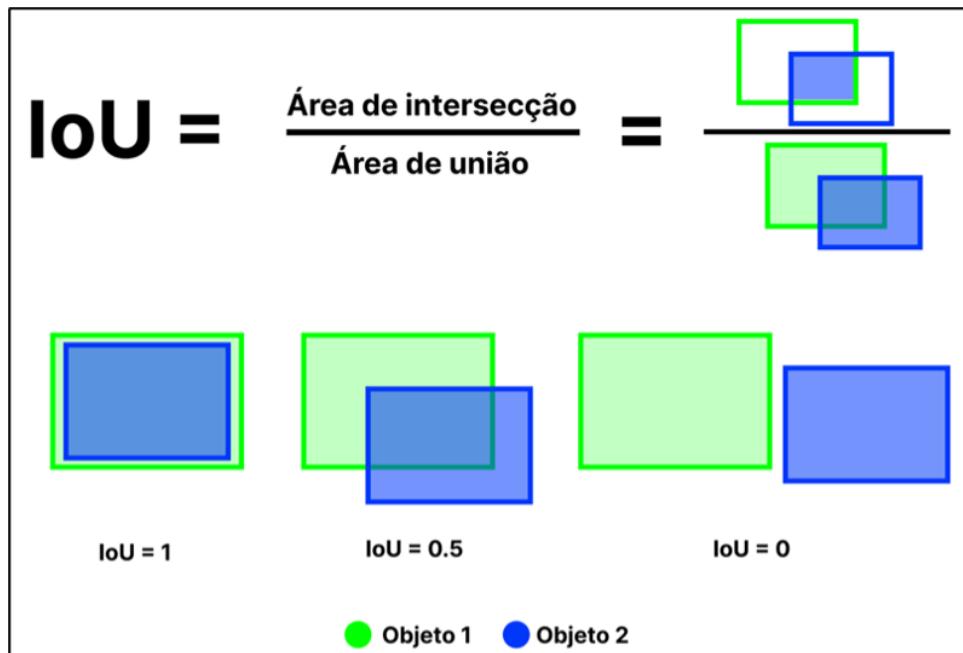


Figura 3.27 – Ilustração do funcionamento do IoU (Modificado de Padilla *et al.* (2020)).

Para realizar o cálculo da interseção entre a fixação e o elemento web, através do algoritmo IoU, é necessário que os ambos sejam delimitados por uma caixa delimitadora. Onde esse processo é realizado apenas para os pontos estimados do olhar presentes na fixação, pois para todos os elementos web esse processo já foi realizado durante o processo de extração. Com isso, é utilizado o algoritmo de *bounding box* na qual é delimitado toda a região dos pontos da fixação, formando uma caixa delimitadora em volta dos pontos, conforme representado na Figura 3.28.

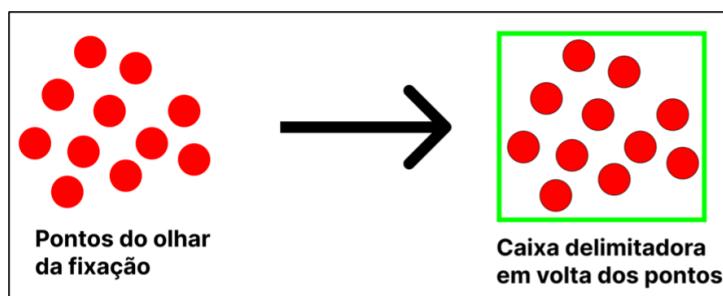


Figura 3.28 – Exemplo da delimitação de uma fixação.

Por exemplo, considerando que a região delimitada pelos pontos da fixação tenha as coordenadas (x,y) para os pontos superior direito e esquerdo, e pontos inferior direito e esquerdo, seja dada respectivamente por (50,25), (150, 25) (50, 95) e (150, 95). E para

caixa delimitadora do elemento web, os pontos superior direito e esquerdo, e os pontos inferiores direito e esquerdo, é dada pelas respectivas coordenadas (x,y), (0, 0), (100, 0) (0, 100) e (100, 100). Onde é representado na Figura 3.29.

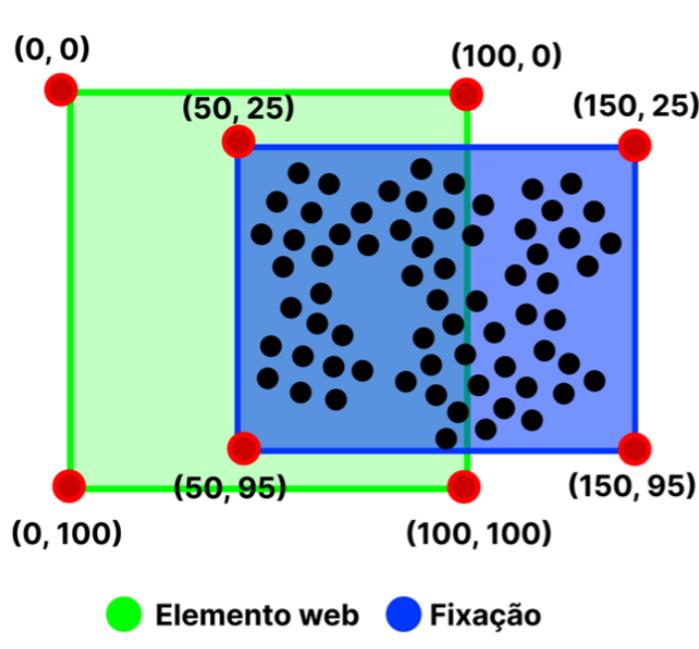


Figura 3.29 – Exemplo de uma sobreposição entre um elemento e uma fixação.

De acordo com os dados apresentados na Figura 3.29, e considerando  $A_E$  como a área do elemento web,  $A_F$  como a área da fixação e  $A_I$  como a área da interseção entre o elemento web e a fixação. Calculando a área para  $A_E$ ,  $A_F$  e  $A_I$  é obtido os seguintes resultados.

$$A_E = 10\,000$$

$$A_F = 7\,000$$

$$A_I = 3\,500$$

A partir disso, é realizado o cálculo do IoU, onde é dividido a área de interseção pela união entre as áreas dos objetos, que é definido pela soma das áreas do elemento web com a fixação subtraindo com a área da interseção, representado na equação abaixo:

$$IoU = \frac{A_I}{A_E + A_F - A_I} = \frac{3\,500}{10\,000 + 7\,000 - 3\,500} \approx 0.2592$$

Com isso, é obtido o valor 0.2592, o que significa que a fixação e o elemento web possuem uma sobreposição de 25,92%. Esse processo é realizado entre todos os elementos web com as fixações, onde caso exista um IoU maior que zero, é feito a associação entre esses dois objetos. Ao final, é armazenado todos os elementos que obtiveram uma intersecção com pelo menos uma fixação.

## 4 DESENVOLVIMENTO

O desenvolvimento deste trabalho consistiu em criar um sistema de rastreamento ocular para identificar atenção visual ao navegar em um website, onde o sistema deverá ser capaz de analisar os dados coletados durante processo e gerar relatórios no qual será possível visualizar as regiões e os elementos do website que obtiveram a atenção visual do usuário. Para isso foi desenvolvido uma interface do usuário para facilitar o processo de interação com o sistema, onde é possível inicializar o processo de rastreamento ocular, visualizar relatórios existentes além de realizar a configuração do sistema.

Onde também foi desenvolvido um navegador web, que possui dois objetivos principais, possibilitar a navegação em um website, e realizar capturar imagens da tela do navegador conforme o conteúdo do website mude. Onde através disso o usuário irá realizar a navegação no website enquanto é realizado o processo de rastreamento ocular.

A implementação do módulo de rastreamento ocular é responsável por receber uma entrada de vídeo, e a partir disso retorna à posição da coordenada da pupila do usuário, na qual será utilizado para o processo de calibração e estimativa do olhar. Para realização do processo de rastreamento ocular é feito uma série de identificações, como a detecção da face, onde foi utilizado aprendizado de máquina para identificar os pontos faciais do usuário. Posteriormente é identificado a região dos olhos a partir dos pontos faciais e o processamento de imagem para realizar o recorte da região dos olhos. Onde esse recorte será utilizado para a detecção da íris e por fim o cálculo da coordenada do centro da pupila.

Onde essa informação é utilizada no processo de calibração, que consiste em realizar a associação da coordenada da pupila com a posição dos pontos apresentados na tela, na qual foi utilizado um modelo regressão linear para estimar a posição do olhar do usuário a partir da coordenada da posição da pupila.

Após o processo de calibração e a coleta de dados durante a navegação do usuário no website, é realizado processo de análise desses dados onde é identificado a atenção visual do usuário, e partir disso é gerado relatórios como mapa de fixação, mapa de calor, scanpath e por fim os elementos web que obtiveram a atenção do usuário.

### 4.1 FERRAMENTAS E TECNOLOGIAS UTILIZADAS

Para o desenvolvimento deste projeto foi utilizado a linguagem *Python*, devido as diversas bibliotecas disponibilizadas relacionadas a área de visão computacional e aprendizado de máquina. Onde para a instalação das bibliotecas foi utilizado o gerenciador de pacotes *pip*. A codificação do sistema foi realizada através do editor de código *Visual Studio Code*, devido a sua facilidade e os diversos recursos disponibilizados pela ferramenta. Foi utilizado a biblioteca de código aberto *OpenCV* devido as diversas funcionalidades para realizar o processamento de imagens, e além disso também foi utilizada a biblioteca

*Scikit-learn* devidos aos recursos de classificação de objetos e aprendizado de máquina.

## 4.2 INTERFACE

O desenvolvimento da interface do sistema foi realizado através do framework *Flet* que permite desenvolver interfaces utilizando a linguagem *Python*, onde o resultado é apresentado na Figura 4.1, onde mostra a tela inicial do sistema, que possui botões que dão acesso as páginas principais do sistema, que é a página de execução, página de visualização de relatórios gerados, página de configurações e por fim a página de informações do projeto.

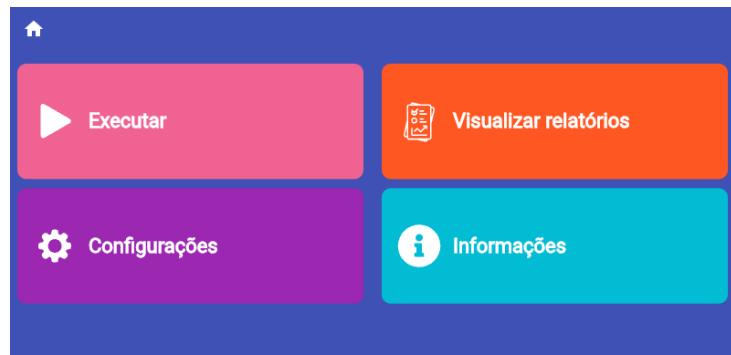


Figura 4.1 – Interface da página inicial.

A Figura 4.2, mostra a página de execução do sistema, onde o usuário deve preencher o nome da execução, que é identificador único, o endereço URL do site, onde o sistema irá realizar o processo de rastreamento ocular, e por fim a resolução da tela.



Figura 4.2 – Interface da página de execução.

## 4.3 NAVEGADOR

O navegador desenvolvido para a ferramenta possui duas funcionalidades principais, mostrar o conteúdo do site escolhido pelo usuário e realizar capturas de tela conforme o conteúdo do site mude.

Para mostrar o conteúdo do site foi inspirado em Fitzpatrick (2021), onde é utilizado a biblioteca *PyQt5*, na qual possibilita a criação de interfaces gráficas utilizando a linguagem *Python*. Essa biblioteca oferece uma série de ferramentas para construção de

interfaces. O Código 4.1 apresenta a implementação da funcionalidade de exibir o conteúdo do site.

Código 4.1 – Implementação da funcionalidade de exibir o conteúdo do site.

```
class Browser(QWebEngineView):
    def __init__(self, url, app, width, height):
        super().__init__()
        self.app = app
        self.resize(width, height)
        self.load(QUrl(url))
        self.show()

    def run(self):
        sys.exit(self.app.exec_())

app = QApplication(sys.argv)
view = Browser(url, app)
view.run()
view.run()
```

Para iniciar a base gráfica da interface do usuário foi usado a função de *QApplication*. E para exibir o conteúdo da internet foi utilizado a combinação entre os módulos *QWebEngineView* e *QUrl*. Onde *QWebEngineView* é responsável pelo componente gráfico do navegador, e o *QUrl* é usado para manipular os endereços dos sites.

O Código 4.2 apresenta a implementação da captura de tela, que são realizadas através da função *grab().toImage()* e antes de salvar a captura de tela é feito uma comparação entre a captura feita com a captura realizada anteriormente, em que será identificado, se essas imagens são iguais ou não. Caso essas imagens sejam iguais, a captura atual será descartada. Já nos casos em que é feita a comparação, e se concluir que as imagens são diferentes, nesse caso a imagem atual será salva.

Código 4.2 – implementação da captura de tela.

```
def capture_screenshot(self):
    screenshot = self.grab().toImage()
    difference = cv2.absdiff(self.previous_screenshot,
                             current_screenshot)
    if np.all(difference) == 0:
        os.remove(filename)
    else:
        screenshot.save(screenshot)
```

Esse processo de comparação foi inspirado no trabalho de Kiruthika *et al.* (2021), que foi realizado utilizando a biblioteca *OpenCV* e *numpy*. Onde é realizado o cálculo da

diferença absoluta das intensidades dos pixels entre a captura atual e a captura anterior, onde isso é feito através da função *absdiff*, que retorna uma matriz com a diferença entre essas duas imagens. E por fim é verificado se a matriz com a diferença é composta apenas por zero. Isso é realizado através do método *all* da biblioteca *numpy*. Essa função retornar um valor numérico que corresponde a diferença entre as duas capturas onde caso o valor retornado seja 0 significa que a captura atual e a captura feita anteriormente são iguais, então nesse caso a captura atual é descartada. Caso o valor retornado seja diferente de 0, significa que as capturas são diferentes e nesse caso a captura atual é salva.

#### 4.4 RASTREAMENTO OCULAR

O processo de rastreamento ocular implementado, consiste em a partir de uma entrada de vídeo, realizar algumas etapas de identificação como o processo de detecção da face, onde foi utilizado aprendizado de máquina para identificação de pontos faciais. Onde o resultado desse processo foi utilizado na etapa seguinte de detecção dos olhos e como resultado, foi obtido o recorte da imagem com a região dos olhos. Onde esse recorte é utilizado na etapa de detecção da pupila, em que foi usado técnicas de processamento de imagens para realizar extração da região de interesse e posteriormente realizar o cálculo do centroide da pupila.

##### A. AQUISIÇÃO DE IMAGEM

Para realizar capturar a imagem do usuário na webcam foi utilizado a função *VideoCapture* do *OpenCV*, onde recebe como parâmetro o índice da câmera conectada. Caso o usuário tenha mais de um dispositivo de câmera conectado ao computador esse índice pode variar de 0, 1, 2, 3 e assim por diante. O Código 4.3 apresenta a implementação da captura de imagem através da webcam.

Código 4.3 – implementação da captura de imagem.

```
cap = cv.VideoCapture(0)
```

Para a realização desse projeto foi utilizado uma única webcam conectada ao dispositivo, neste modo o índice é zero.

##### B. DETECÇÃO DE FACE

Para fazer a detecção da face do usuário, inspirado no trabalho de Hangaragi *et al.* (2023) foi utilizado um algoritmo de aprendizagem de máquina e visão computacional para extrair os pontos faciais do usuário a partir de uma imagem de vídeo como entrada. Esse processo foi realizado através do *MediaPipe* que é uma biblioteca desenvolvida pela Google de aprendizagem de máquina e visão computacional, onde por meio dela é possível detectar os pontos faciais do usuário, isso é realizado através do módulo chamado *Face mesh* (FERNANDEZ, 2023).

O *Face Mesh* se trata de um módulo responsável por identificar a partir de um frame, os pontos faciais em 3D em tempo real de um indivíduo presente na imagem. Esse módulo utiliza aprendizagem de máquina para inicialmente detectar a face do usuário. E posteriormente utilizando redes neurais para extraír os pontos faciais do usuário (FERNANDEZ, 2023).

A detecção de face é realizada utilizando o modelo de aprendizagem de máquina, o *BlazeFace*, na qual possui uma de suas características a sua velocidade de detecção da face (BAZAREVSKY *et al.*, 2019). Esse modelo recebe como entrada, uma imagem de vídeo, onde realiza o processamento e retorna a região de face delimitada recortada para o processo posterior de extração dos pontos faciais.

De acordo com Kartynnik *et al.* (2019), o processo de extração dos pontos faciais utiliza redes neurais que consiste segundo o autor Aggarwal (2018) em uma abordagem de aprendizagem de máquina na qual imita o processo de aprendizagem dos seres humanos, onde são utilizado neurônios artificiais para realizar o processamento de dados. Esse conceito possui aplicação em diversas áreas, como por exemplo, análise de dados para o reconhecimento de padrões, finanças, reconhecimento da voz e entre outros (MIJWIL *et al.*, 2019). Onde também é utilizado para o contexto de detecção de malha facial, através do *Face mesh*, com o intuito de identificar os pontos faciais de um indivíduo, como por exemplo, olhos, nariz, boca, entre outros (KARTYNNIK *et al.*, 2019).

O *Face mesh* foi treinado com o objetivo de disponibilizar os pontos faciais de um indivíduo, onde para isso foi utilizado uma base de dados que conta com 30 mil imagens capturadas com diferentes variações de iluminação, expressões faciais, câmeras diferentes, entre outros (KARTYNNIK *et al.*, 2019). Com isso resultando em um modelo que possui uma boa precisão na detecção dos pontos faciais. O modelo possui a capacidade de detectar 468 pontos faciais, onde cada ponto representa uma coordenada 3D, nos eixos X, Y e Z, que são utilizados para a construção da malha facial do indivíduo (HANGARAGI *et al.*, 2023). Esse processo é ilustrado na Figura 4.3.



Figura 4.3 – Ilustração do processo de extração de pontos faciais.

Conforme mostrado na Figura 4.3, o processo de extração dos pontos faciais, começa a partir de um frame de entrada, onde é realizado o processamento para a detecção das

malhas faciais e ao final é retornado os pontos faciais.

O Código 4.4 apresenta uma classe chamada *FaceMeshDetector*, na qual é responsável através de uma entrada de um frame de vídeo, detectar a malha facial e retornar os pontos faciais. Para isso, inicialmente é declarado no construtor da classe o objeto *mp\_face\_mesh*, na qual é utilizado posteriormente para a detecção da malha facial. Esse objeto é inicializado com quatro parâmetros principais: o número de faces, o valor de refinamento de precisão dos pontos faciais, o valor mínimo para detecção e por fim o valor mínimo de confiança. A partir disso ele retorna as malhas faciais detectadas e com isso será possível extrair a região que possui esses pontos na imagem.

Código 4.4 – Classe que implementa o módulo Face Mesh.

```
class FaceMeshDetector:
    def __init__(self):
        mp_face_mesh = mp.solutions.face_mesh
        self.face_mesh = mp_face_mesh.FaceMesh(
            max_num_faces=1,
            refine_landmarks=True,
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )
```

Através do método *detect\_face\_mesh*, apresentado no Código 4.5, é realizado o processo de detecção da malha facial, onde esse método recebe como parâmetro um frame de vídeo, e através disso, inicialmente é realizado uma conversão da imagem de *BGR* para *RGB* através do módulo do *OpenCV* o *cvtColor*. Pois, o módulo do *Face mesh* utiliza imagens no formato *RGB* (FERNANDEZ, 2023). Após esse processo, é realizada a extração dos valores da altura e largura da imagem, onde são salvas em duas variáveis, *img\_h* e *img\_w*.

Código 4.5 – Função que detecta a face do usuário.

```
def detect_face_mesh(self, frame):
    rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    img_h, img_w = frame.shape[:2]
    results = self.face_mesh.process(rgb_frame)

    if results.multi_face_landmarks:
        mesh_points = np.array([
            [
                np.multiply([p.x, p.y], [img_w, img_h]).astype(int)
                for p in results.multi_face_landmarks[0].landmark
            ]
        ])
```

```
    return mesh_points  
  
return None
```

Em seguida é chamado o método `process` do objeto `face_mesh` que foi inicializado no construtor da classe, onde irá realizar o processo de detecção da malha facial e retornar os pontos faciais. Por fim, é feito uma verificação se existir algum ponto facial detectado, caso não exista é retornado `None`, o que significa que não foi detectado nenhum ponto facial na imagem, caso contrário é retornado um array com os pontos faciais. A Figura 4.4 ilustra os 468 pontos faciais detectados pelo `face mesh`, onde cada ponto representa uma coordenada X, Y e Z (HANGARAGI *et al.*, 2023).

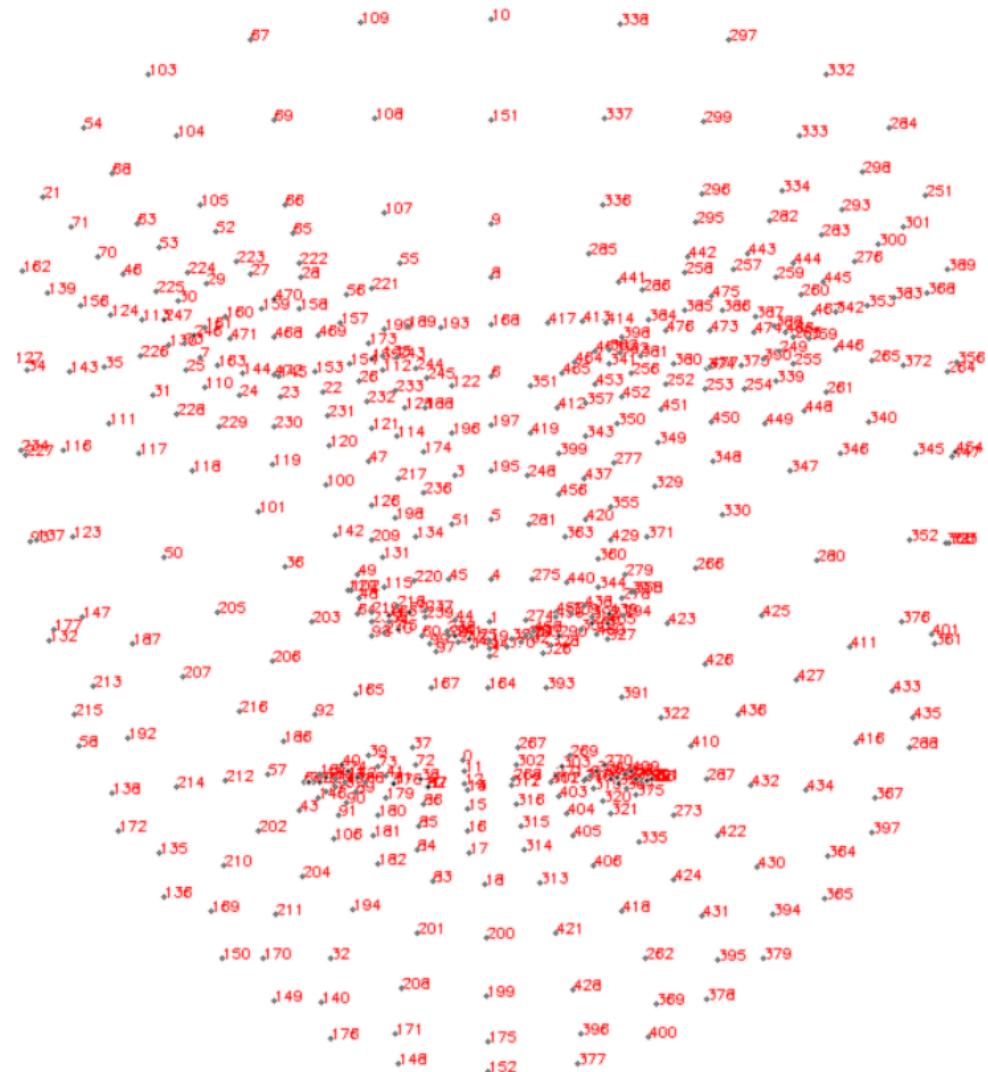


Figura 4.4 – Ilustração dos pontos faciais.

Desta forma, com pontos detectados será possível dar prosseguimento ao processo seguinte de detecção da região dos olhos.

### C. DETECÇÃO DOS OLHOS

Dos 468 pontos detectados na malha facial, a região dos olhos é composta por 32 pontos, sendo 16 para cada olho (WANG, H. *et al.*, 2022). De acordo com o autor Aiphile (2022) a região dos olhos possui os seguintes pontos pré-determinados na malha facial, conforme apresentado no Quadro 4.1 e na Figura 4.5.

Quadro 4.1 – Pontos dos olhos.

<b>Pontos do olho esquerdo</b>	362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385, 384, 398
<b>Pontos do olho direito</b>	33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161, 246



Figura 4.5 – Representação dos pontos dos olhos.

Conforme demonstrado na Figura 4.5, os pontos tanto do olho direito quanto do olho esquerdo delimitam a região dos olhos. Desta forma através desses pontos é possível realizar a extração da região de interesse que no caso são os olhos. A partir disso, o próximo passo é realizar a parte de recorte e processamento da imagem.

Esse processo é realizado através da função *extract\_eye*, apresentado no Código 4.6, que recebe como parâmetro, o frame do vídeo da face do usuário, os pontos faciais detectados e os pontos referente à região dos olhos apresentado anteriormente no Quadro 4.1. Inicialmente é feito um loop para encontrar os pontos faciais referente a região dos olhos no parâmetro *face\_landmarks* que possui todos os 468 pontos facial detectados. Onde esses pontos são armazenados no array *eye\_points*.

Código 4.6 – Função que recorta a região dos olhos.

```
def extract_eye(image, face_landmarks, EYE_POINTS):
    eye_points = [
        (int(landmark.x * image.shape[1]), int(landmark.y * image.
                                                shape[0]))
        for i, landmark in enumerate(face_landmarks.landmark)
        if i in EYE_POINTS
    ]

    eye_mask = np.zeros_like(image)
```

```

cv2.fillPoly(eye_mask, [np.array(eye_mask)], (255, 255, 255))

eye = cv2.bitwise_and(eye_mask, image)

return eye

```

Posteriormente, é feito uma máscara preenchida com zeros que representa a cor preta na imagem do frame do vídeo, onde em seguida através do método *fillPoly* do *OpenCV* é realizado o preenchimento das regiões dos olhos com a cor branca. Por fim, através do método *bitwise\_and* do OpenCV é extraído do frame do vídeo apenas com a região na qual foi preenchida com a cor branca, que representa a região dos olhos. Esse processo é realizado tanto para o olho esquerdo quanto para o olho direito. Onde o resultado desse processo é ilustrado na Figura 4.6.



Figura 4.6 – Ilustração do recorte dos olhos.

#### D. DETECÇÃO DAS PUPILAS

Após a realização do processo de extração da imagem da região dos olhos, baseado no trabalho de Brachter e Gerhardt (2020), é realizado o processamento dessas imagens para detectar a íris e desta forma, extrair as coordenadas da pupila que é um passo fundamental para o processo de rastreamento ocular. Esse processo é dividido em quatro partes: a conversão da imagem para uma escala de cinza, em seguida a realização da binarização da imagem, a detecção de contorno e por fim o cálculo do centroide. A Figura 4.7 mostra a imagem na escala de cinza.



Figura 4.7 – Ilustração da conversão da imagem para escala de cinza.

O processo de conversão para a escala de cinza irá permitir com que se extraiam informações relevantes da imagem e irá facilitar o processamento para os próximos passos (KANAN; COTTRELL, 2012). O Código 4.7 apresenta a conversão para escala de cinza.

Código 4.7 – Conversão para escala de cinza.

```
gray_eye = cv.cvtColor(eye_roi, cv.COLOR_BGR2GRAY)
```

Esse processo é realizado por meio da função `cvtColor` da biblioteca *OpenCV* onde recebe como parâmetro a imagem colorida e uma constante pré-definida do *OpenCV*, o `COLOR_BGR2GRAY`, que irá transformar os três canais de intensidade da imagem para uma escala de cinza. A Figura 4.8 ilustra o processo de remoção de ruído.

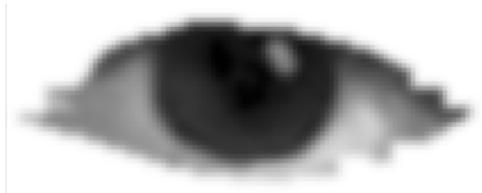


Figura 4.8 – Ilustração da remoção de ruído.

Conforme mostrado na Figura 4.8, esse processo consiste em aplicar um filtro para remover os ruídos que possam atrapalhar a extração de informações da imagem (BHUYAN, 2019). Onde isso é feito através do filtro gaussiano, apresentado no Código 4.8.

Código 4.8 – Implementação do filtro gaussiano.

```
gray_eye = cv2.GaussianBlur(gray_eye, (5,5),0)
```

Para realizar esse processo, foi utilizado o método `GaussianBlur` do *OpenCV* que recebe como parâmetro a imagem, o tamanho da matriz do kernel e por fim o desvio padrão. Em seguida, é realizado o processo de limiarização da imagem, que se trata de um processo de segmentação que irá binarizar a imagem, na qual irá separar a região de interesse e facilitando assim a detecção dos contornos (BHUYAN, 2019).

Conforme ilustrado na Figura 4.9, o processo de binarização consiste em transformar os pixels da imagem que possuem as intensidades com um valor maior do que um limiar para 0 (preto) e a intensidade menor que o limiar para 255 (branco) (BHUYAN, 2019).



Figura 4.9 – Ilustração da binarização da imagem.

Esse processo é realizado através da função `threshold` do *OpenCV*, apresentado no Código 4.9, que recebe como parâmetro a imagem, o valor do limiar que no caso é 107 sugerido pelo (BRACHTER; GERHARDT, 2020), uma constante que atende os critérios de limiarização que representam os pixels na qual foi utilizado 255 e por fim uma constante

pré-definida do *OpenCV THRESH\_BINARY\_INV* que representa o tipo de binarização a ser aplicada.

Código 4.9 – Implementação da binarização.

```
_, threshold_eye = cv.threshold(gray_eye, 107, 255, cv.THRESH_BINARY_INV)
```

O resultado dessa operação será uma imagem binarizada. O próximo passo é através da imagem binarizada detectar as bordas que no caso representam a íris. Esse processo foi realizado através da função *findContours* do *OpenCV*, conforme apresentado no Código 4.10.

Código 4.10 – Implementação da detecção de contornos.

```
contours, _ = cv.findContours(threshold_eye, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
contour = max(contours, key=cv2.contourArea)
```

A função *findContours* recebe como um parâmetro a imagem binarizada e outras duas constantes pré-definidas pelo *OpenCV*. O resultado dessa função irá retornar os contornos detectados na imagem binarizada. Em seguida, é realizada a extração do contorno máximo que representa a região da íris. Após a detecção dos contornos e a extração do maior contorno, o próximo passo é calcular o centroide da pupila, onde para isso foi utilizado, os “momentos da imagem” que se trata de um cálculo que fornece a métrica do centroide (NGUYN, 2019). Para isso foi utilizado a função *moments* do *OpenCV* que recebe como parâmetro o contorno e retorna um conjunto de dados. Esse processo é apresentado no Código 4.11.

Código 4.11 – Implementação do cálculo do centroide.

```
moments = cv2.moments(contour)
if moments[ "m00" ] != 0:
    cx = int(moments[ "m10" ] / moments[ "m00" ])
    cy = int(moments[ "m01" ] / moments[ "m00" ])
else:
    return None
```

Baseado no Nguyn (2019) foi utilizado as constantes  $M_{00}$ ,  $M_{10}$  e  $M_{01}$ , onde foi adicionado na equação do centroide. Ao final desse processo é obtido as coordenadas das pupilas.

## 4.5 CALIBRAÇÃO

Com a disponibilidade da coordenada da pupila, o próximo passo é realizar o processo de calibração. Com base no trabalho de Brachter e Gerhardt (2020), é realizado

uma associação entre as coordenadas da pupila com um ponto específico na tela. Esse processo permite com que seja possível estimar a região na qual o usuário está direcionando o seu olhar na tela. Para isso é realizado algumas etapas, onde a primeira etapa é exibida na tela os pontos em que o usuário terá que olhar, e ao clicar em uma tecla o sistema irá salvar a posição da pupila, juntamente com a posição dos pontos exibidos na tela. Onde ao final terá uma coordenada da pupila associada para cada ponto exibido. Em seguida, é definido o modelo de regressão linear que irá realizar o processo de associação entre as coordenadas da pupila com os pontos capturados da tela. E a partir desse modelo de regressão linear é realizado a estimativa do olhar na tela, através da coordenada de pupila recebida como entrada.

Inicialmente é definido os pontos que serão exibidos na tela, onde a quantidade foi baseada no trabalho de Brachter e Gerhardt (2020), na qual foi definido 9 pontos de calibração que são espalhados de forma uniforme na tela.

Em seguida, conforme apresentado no Código 4.12, é inicializada a variável o *pupil\_points* que se trata de uma array que irá armazenar as coordenadas capturadas da pupila durante o processo de calibração.

Código 4.12 – Variável de armazenamento das coordenadas da pupila.

```
pupil_points = []
```

Conforme apresentado no Código 4.13, é realizado dois loops, onde um feito por um for, que irá inteirar sobre os pontos definidos de calibração e outro é realizado por um while, que irá capturar cada frame do vídeo e irá como parâmetro na classe *eyeTracking*, que é responsável por realizar o processo de rastreamento ocular onde será feito todo o processo relatado no tópico anterior e devolverá as coordenadas x e y da pupila detectada no frame.

Código 4.13 – Captura das coordenadas da pupila para cada ponto de calibração.

```
for point in self.screen_points:
    while True:
        ret, frame = cap.read()
        if not ret:
            break
    coords = eyeTracking(frame)
```

Ainda no loop, apresentado no Código 4.14, é criado uma imagem de fundo da cor branca que irá apresentar os pontos na tela. Para isso foi utilizado a função *one* da biblioteca *numpy* que cria uma matriz de tamanho 1920 x 1080, onde cada elemento da matriz foi preenchido com o número 1, e posteriormente foi multiplicado para 255 para que todos os elementos que antes da matriz que antes estavam com o valor 1 passa a ser 255 que é o código da cor branca em RGB. Onde o resultado dessa operação resulta em uma imagem totalmente branca.

Código 4.14 – Imagem de fundo branco.

```
white_image = np.ones((1920, 1080, 3), dtype=np.uint8) * 255
```

Posteriormente, apresentado no Código 4.15, utilizando a função *circle* da biblioteca *OpenCV* é desenhado círculo preto na imagem de fundo de raio 10, que irá representar visualmente o ponto de calibração que o usuário terá que olhar. ]

Código 4.15 – Desenho de círculos na imagem.

```
cv.circle(white_image, point, 10, (0, 0, 0), -1)
```

Posteriormente, utilizando a função *imshow* do *Opencv* é exibido a tela de calibração. Conforme apresentado no Código 4.16.

Código 4.16 – Apresentação da tela de calibração.

```
cv.imshow('img', white_image)
```

Por fim, apresentado no Código 4.17, é utilizado a função *waitKey*, que é responsável por capturar as teclas digitadas no teclado pelo usuário, em seguida é feito um condicional onde caso o usuário clique a tecla "C"do teclado, irá significar para o sistema que o usuário está olhando para o ponto especificado na tela e portanto poderá salvar a coordenada da pupila.

Código 4.17 – Funcionalidade para detecção de tecla digitada para realizar a calibração.

```
key = cv.waitKey(1)
if key == ord('c'):
    pupil_points.append(coords) break
```

Esse processo irá se repetir para todos os pontos definidos de calibração e ao final desse processo para cada ponto de calibração definido haverá uma coordenada de pupila associada.

O próximo passo, é criar modelo de regressão linear na qual irá possibilitar estimar a coordenada que o usuário está olhando na tela a partir de uma coordenada atual da pupila. Para a realização desse processo foi utilizado a biblioteca *scikit-learn* onde foi definidos dois modelos. Um para coordenada x e outro para coordenada y. Conforme apresentado no Código 4.18.

Código 4.18 – Implementação das funções de regressão linear.

```
model_x = LinearRegression()
model_y = LinearRegression()
```

Com isso, utilizando a função *LinearRegression* do *scikit-learn* é definido o *model\_x* e *model\_y* que são respectivamente o modelo de regressão linear para coordenada x e a coordenada y. Em seguida, apresentado no Código 4.19, é associado os modelos de regressão linear criados aos dados as coordenadas das pupilas com os pontos de calibração.

Código 4.19 – Associação entre pontos de calibração com as coordenadas das pupilas.

```
model_x.fit(pupil_points[:, 0].reshape(-1, 1), screen_points[:, 0].reshape(-1, 1))
model_y.fit(pupil_points[:, 1].reshape(-1, 1), screen_points[:, 1].reshape(-1, 1))
```

Deste modo, utilizando o método *fit* do *scikit-learn*, que recebe dois parâmetros, as coordenadas da pupila e os pontos de calibração, onde esse processo é realizado tanto para as coordenadas em X quanto para coordenadas em Y. E ao final desse processo é retornado o modelo regressão linear associado com as coordenadas da pupila e os pontos de calibração.

Com isso, o sistema já possui a capacidade de estimar a posição do olhar através de uma coordenada da pupila. A realização desse processo, apresentado no Código 4.20, é realizada utilizando o método *predict* do *scikit-learn*, que recebe como parâmetro as coordenadas da posição da pupila e retorna a estimativa do olhar, respectivamente para os eixos X e Y.

Código 4.20 – Implementação da funcionalidade de previsão das coordenadas do olhar na tela.

```
gaze_coordinates_x = self.model_x.predict(x_coordinate)
gaze_coordinates_y = self.model_y.predict(y_coordinate)
```

Ao final desse processo de calibração, o sistema será capaz de receber uma lista coordenada da posição da pupila e a partir disso estimar a posição do olhar do usuário em relação a tela.

## 4.6 ANÁLISE DE DADOS

Após a realização do processo de rastreamento ocular onde os dados capturados já estão salvos, é possível utilizar esses dados para extrair informações relevantes no contexto da detecção de atenção visual do usuário em websites. Onde inicialmente é feito o processo de identificação dos pontos de fixação e em seguida geração de relatórios com as informações extraídas.

### 4.6.1 IDENTIFICAÇÃO DE FIXAÇÕES

A etapa de identificação de fixações consiste em calcular a dispersão dos pontos coletados durante o processo de rastreamento ocular, e assim identificar as fixações. Para isso foi utilizado o algoritmo de Dispersion-Threshold Identification (I-DT) dos autores Salvucci e Goldberg (2000). O cálculo da dispersão é feito através da distância entre o ponto central de uma coordenada com os demais pontos ao seu redor. Na qual possui um limiar de dispersão, onde caso a dispersão seja menor que esse limiar, então a coordenada

é considerada parte da fixação, além disso possui um outro parâmetro que é a quantidade de pontos mínimos para que seja considerado uma fixação.

Segundo Gobbi *et al.* (2017) uma fixação possui um tempo de duração de 200 á 300 milissegundos, desta forma foi utilizado o tempo de 300 milissegundos para o cálculo da dispersão. A partir de testes realizados, o tempo médio para registrar uma coordenada do olhar pela ferramenta de rastreamento ocular desenvolvida foi de 30 milissegundos, ou seja, é necessário de no mínimo 10 pontos para alcançar o tempo de 300 milissegundos.

Com isso, foi definido a quantidade mínima de 10 pontos para que seja considerada uma fixação. Além disso, baseado no trabalho Sun *et al.* (2019) foi definido o valor de 20 pixels como o limiar de dispersão. O Código 4.21 apresenta a implementação da funcionalidade de identificação de fixações.

Código 4.21 – Implementação da funcionalidade de identificação de fixações.

```
def fixation_detection(points, threshold, min_points):
    fixations = []
    i = 0
    while i < len(points) - min_points + 1:
        window = points[i:i+min_points]
        min_x = min(window, key=lambda t: t[0])[0]
        max_x = max(window, key=lambda t: t[0])[0]
        min_y = min(window, key=lambda t: t[1])[1]
        max_y = max(window, key=lambda t: t[1])[1]

        dispersion = max(max_x - min_x, max_y - min_y)

        if dispersion <= threshold:
            while i + len(window) < len(points) and dispersion <=
                threshold:
                window.append(points[i + len(window)])
                min_x = min(window, key=lambda t: t[0])[0]
                max_x = max(window, key=lambda t: t[0])[0]
                min_y = min(window, key=lambda t: t[1])[1]
                max_y = max(window, key=lambda t: t[1])[1]
                dispersion = max(max_x - min_x, max_y - min_y)
            avg_x = sum([t[0] for t in window]) / len(window)
            avg_y = sum([t[1] for t in window]) / len(window)
            fixations.append((avg_x, avg_y, len(window)))

            i += len(window)
        else:
            i += 1
```

```
return fixations
```

A função *fixation\_detection* recebe três parâmetros, o primeiro é uma lista de tuplas contendo as coordenadas do olhar, o segundo é o limiar de dispersão e por fim o terceiro é a quantidade mínima de pontos. Inicialmente é declarado as variáveis, *fixation* responsável por armazenar as fixações, e o *i* é o índice que percorre a lista de pontos. Posteriormente é realizada um loop que irá percorrer a lista de pontos, e dentro do loop é criado uma janela de pontos, onde o tamanho dessa janela é o valor do parâmetro *min\_points* que é quantidade mínima de pontos para ser considerado uma fixação. Em seguida é calculado a dispersão através da diferença entre o maior e o menor valor de x e y da janela, caso o valor da dispersão seja menor que o limiar definido, então é realizado um novo loop que irá adicionar novos pontos na janela e será feito após esse loop a média dos valores de x e y da janela, e por fim será adicionado na lista de fixações. Esse processo, irá se repetir até que o valor da dispersão seja maior que o limiar e atinja o tamanho mínimo de pontos. Onde ao final do processo, a função irá retornar a lista de fixações.

#### 4.6.2 GERAÇÃO DE RELATÓRIOS

Inspirado nos trabalho Vera *et al.* (2017) e Menges *et al.* (2020), foram desenvolvidos quatro tipos de relatórios sendo um deles o mapa de fixações que mostram todas as fixações realizadas em cada captura de tela, o scanpath que mostra os pontos de fixações em sequência onde esse tipo de relatório demonstra a navegação entre um ponto de fixação e outro durante a navegação, o mapa de calor que para cada imagem capturada irá mostrar as regiões na qual o usuário obteve o maior concentração de atenção visual durante a navegação no website e por fim o relatório com as informações dos elementos web que obtiveram a atenção visual do usuário.

#### A. MAPA DE FIXAÇÕES

O desenvolvimento do mapa de fixação foi feito através da biblioteca *OpenCV* que irá realizar o processamento das imagens através dos módulos *imread*, *cicle* e *imgwrite*. O Código 4.22 apresenta a implementação do mapa de fixações.

Código 4.22 – Implementação do mapa de fixações.

```
frame = cv2.imread(image_path)
for fixation in fixations:
    x = int(fixation[0])
    y = int(fixation[1])
    center_point = (x, y)
    length_points = int(fixation[2])

    base_radius = 10
```

```

radius = base_radius * length_points
cv2.circle(frame, center_point, radius, (0, 0, 0), 2)

cv2.imwrite(output_image_path, frame)

```

Inicialmente, é carregada a imagem de captura de tela utilizando o método `imread` que ler a imagem, e posteriormente é realizado uma interação sobre as fixações referente a imagem lida, é carregado então o ponto central da fixação na variável `center_point` e quantidade de pontos pertencente a fixação, através do método cicle desenhar o círculo da cor preta para a coordenada central da fixação da imagem, juntamente com o tamanho variável de acordo com a quantidade pontos da fixação.

Em seguida, é utilizado o método `imgwrite`, responsável por salvar a imagem esse processo se repete para cada imagem capturada do experimento. O resultado final é representado na Figura 4.10



O Coral da UENF, com apoio da Universidade Estadual do Norte Fluminense Darcy Ribeiro (UENF),  
Pró-Reitoria de Extensão da UENF, Assessoria de Cultura da UENF e Assessoria de Comunicação

Figura 4.10 – Mapa de fixação gerado pelo sistema.

## B. SCANPATH

O Scanpath funciona de forma semelhante ao mapa de fixações, porém com uma diferença que neste tipo de mapa é mostrado dentro do círculo o número referente a ordem da fixação realizada e as linhas conectando a trajetória percorrida pelo olhar. O Código 4.23 apresenta a implementação do scanpath.

Código 4.23 – Implementação do scanpath.

```

for idx, fixation in enumerate( fixations ):
    x = int(fixation [0])
    y = int(fixation [1])
    center_point = (x, y)
    length_points = int(fixation [2])

    base_radius = 10
    radius = base_radius * length_points

    cv2.circle(frame, center_point, radius, (0, 0, 0), 2

    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(
        frame, str(idx + 1), (x - 5, y + 5), font, 0.5, (0, 0, 0), 2,
        cv2.LINE_AA
    )

    if prev_point is not None:
        cv2.line(frame, prev_point, center_point, (0, 0, 0), 2)

    prev_point = center_point

cv2.imwrite(output_image_path, frame)

```

Para realização desse processo é utilizado o círculo para desenhar os círculos na imagem, e o módulo *putText* que receber como parâmetro o número da sequência na qual aquela fixação representa, e adiciona o texto dentro do círculo. O método *line* é utilizado para desenhar as linhas que conectam os pontos de fixação, onde é passado como parâmetro o *frame*, o ponto anterior, o ponto atual, a cor da linha e a espessura da linha. Por fim o *imwrite* para salvar a imagem. O resultado final é representado na Figura 4.11.

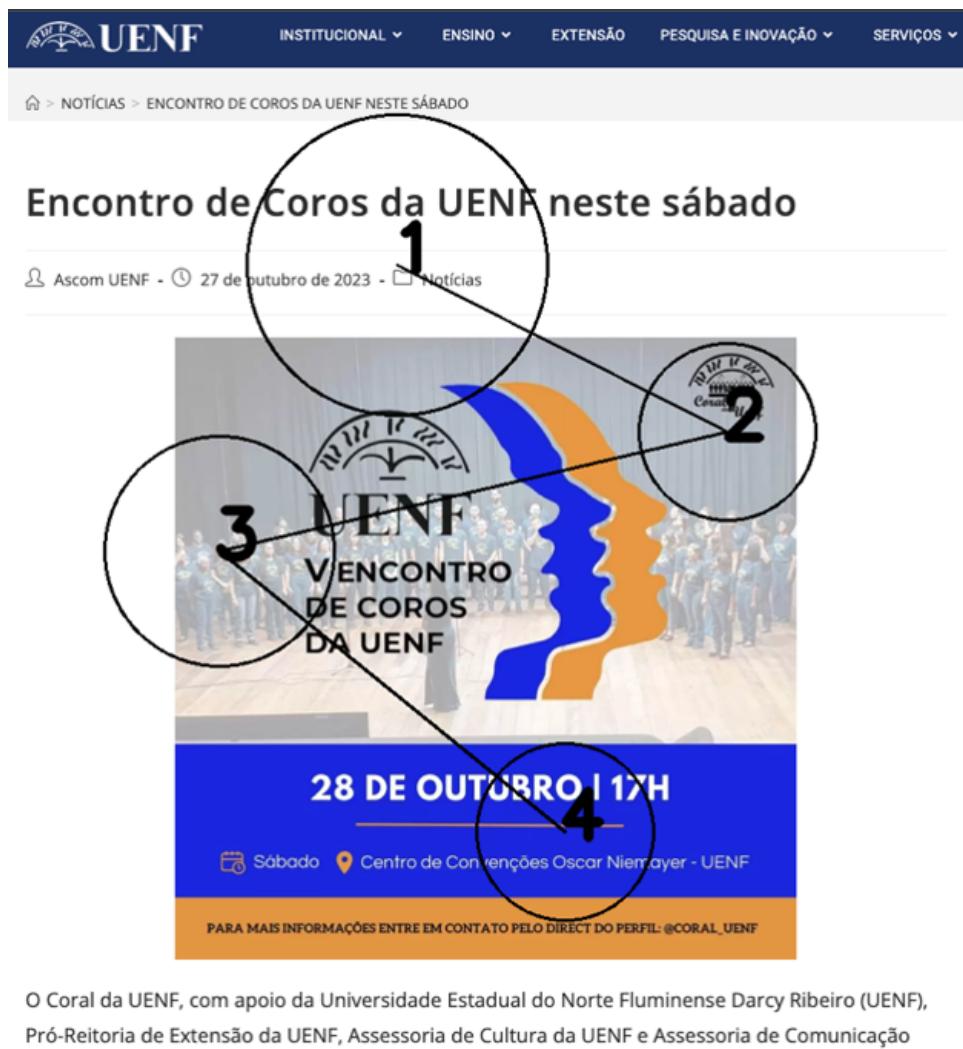


Figura 4.11 – Scanpath gerado pelo sistema.

### C. MAPA DE CALOR

O mapa de calor irá mostrar as regiões visualizadas através dos dados coletados durante o processo de rastreamento ocular. O que possibilitará ter uma visualização clara das regiões que obteve maior atenção do usuário. Para isso é utilizado a biblioteca *Pygaze* de código aberto do trabalho de Dalmaijer *et al.* (2014), que possibilita a criação de mapas de calor. A implementação do mapa de calor é apresentada no Código 4.24.

Código 4.24 – Implementação do mapa de calor.

```
def draw_heatmap(gazepoints, size, imagefile, path_save_file):
    fig, ax = draw_display(size, imagefile=imagefile)
    radius = 50
    heatmap = numpy.zeros((size[1], size[0]), dtype=float)

    for (x, y, _) in gazepoints:
        x += radius
        y += radius
```

```

for i in range(-radius, radius + 1):
    for j in range(-radius, radius + 1):
        if i**2 + j**2 <= radius**2:
            x_new, y_new = x + i, y + j
            if 0 <= x_new < heatmap.shape[1] and 0 <= y_new <
                heatmap.shape[0]:
                heatmap[y_new, x_new] += 1

heatmap = heatmap[radius:-radius, radius:-radius]

heatmap[(heatmap > 1) & (heatmap <= 5)] = 1
heatmap[(heatmap > 5) & (heatmap <= 10)] = 2
heatmap[(heatmap > 10) & (heatmap <= 15)] = 3
heatmap[(heatmap > 15) & (heatmap <= 20)] = 4
heatmap[heatmap > 25] = 5

colors = [
    (128 / 255, 128 / 255, 220 / 255),
    (128 / 255, 214 / 255, 255 / 255),
    (188 / 255, 255 / 255, 190 / 255),
    (239 / 255, 255 / 255, 138 / 255),
    (238 / 255, 128 / 255, 128 / 255),
]
cmap = ListedColormap(colors)

ax.imshow(heatmap, cmap=cmap, alpha=alpha, interpolation="nearest")
fig.savefig(path_save_file)
return fig

```

Para realizar o procedimento da criação de mapa de calor, foi criado uma função chamada *draw\_heatmap* que recebe como parâmetros a lista de coordenadas da estimativa do olhar, captura de tela, o tamanho em pixel da resolução da imagem e por fim o caminho para salvar a imagem processada. Inicialmente é chamado uma função chamada *draw\_display* que é responsável por criar a figura dos eixos do gráfico que serão utilizados ao final para salvar a imagem processada. Depois disso é criado uma matriz de zero com o mesmo tamanho da imagem, utilizando o parâmetro de size recebido como parâmetro.

Para cada ponto da lista de coordenadas do olhar é incrementado o valor de 1 na matriz e no raio de 50 pixels ao redor da posição referente a coordenada do ponto. Após esse processo, é adicionado as cores ao mapa de calor, onde é feito um agrupamento de pontos de acordo com a frequência em cada região da matriz. Onde cada grupo possui

uma cor diferente, representando a frequência do olhar na região, conforme apresentado na Tabela 4.1. As regiões mais visualizadas são representadas pelas cores mais quentes como o vermelho e amarelo, e as regiões menos visualizadas pelas cores mais frias como o azul e o ciano.

Tabela 4.1 – Grupo de cores do mapa de calor (Baseado na biblioteca *Pygaze*).

Grupo	Quantidade de Frequências	Cor
1	1 a 5	Azul
2	6 a 10	Ciano
3	11 a 15	Verde
4	16 a 25	Amarelo
5	Maior que 25	Vermelho

Por fim, através do método *savefig*, o mapa de calor gerado é salvo. O resultado é apresentado na Figura 4.12.



Figura 4.12 – Mapa de calor gerado pelo sistema.

## D. ATENÇÃO VISUAL EM ELEMENTOS WEB

Inspirado no trabalho de Barz e Sonntag (2021), o relatório atenção visual em elementos web é realizado em três etapas, onde inicialmente é feito processamento das capturas de telas coletas, com o objetivo de extrair os elementos web, onde em seguida esses elementos são classificados em texto, botão ou figura e por fim é feito a interseção dos elementos web com as fixações detectadas.

### I. DETECÇÃO DE ELEMENTOS DO WEBSITE

O processo de detecção de elementos foi baseado nos trabalhos Goyal *et al.* (2021) e Bukhari *et al.* (2011), onde foi utilizado processamento de imagem para realização da delimitação dos elementos de um website, utilizando conceitos de visão computacional.

Inicialmente para realização do processo de identificação de elementos de um website, é lido a captura de tela do site, através do módulo *imread* da biblioteca *OpenCV*. Conforme é apresentado no Código 4.25.

Código 4.25 – Implementação da funcionalidade de identificação de elementos.

```
image = cv2.imread('screenshot')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
binarized = cv2.threshold(gray_image, 40, 255, cv2.THRESH_BINARY_INV)
edges = cv2.Canny(binarized, 50, 100)
kernel = np.ones((10,10), np.uint8)
edges = cv2.dilate(edges, kernel, iterations=1)
edges = cv2.erode(edges, kernel, iterations=1)
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.
    CHAIN_APPROX_SIMPLE)
```

A Figura 4.13 ilustra o resultado do processo de conversão da imagem original para escala de cinza por meio do método *cvtColor*, utilizando a constante *COLOR\_BGR2GRAY*, que é responsável por converter a imagem para escala de cinza.

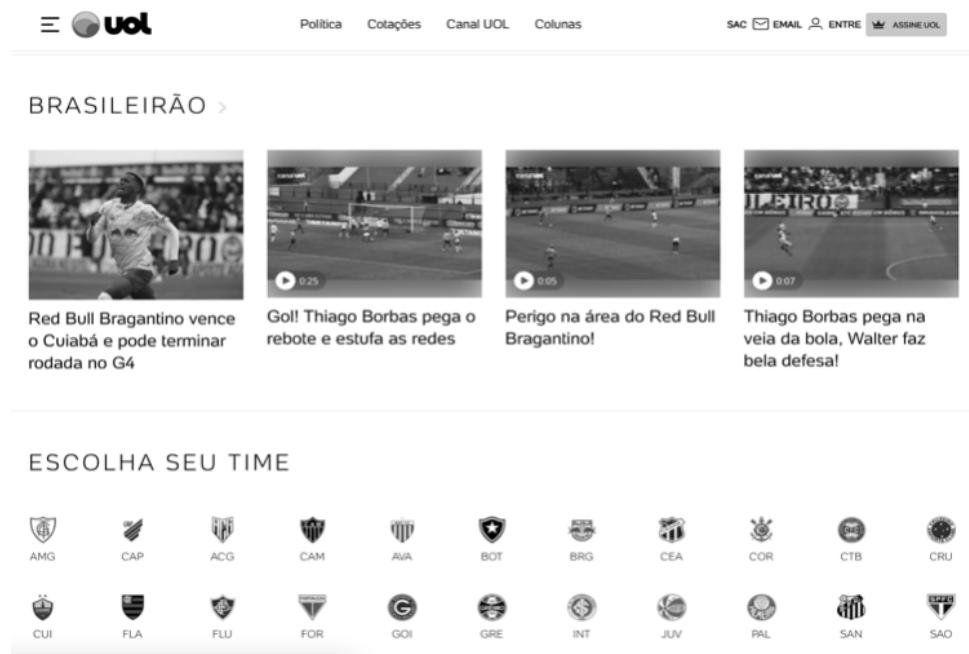


Figura 4.13 – Imagem convertida para cor cinza.

Em seguida, é realizado o processo de limiarização, na qual é feito a binarização da imagem, que consiste em transformar a imagem para apenas dois valores intensidade, 0 ou 255, onde o 0 representa o preto e o 255 o branco, onde desta forma irá destacar os pontos de interesse da imagem (BHUYAN, 2019). Para realização desse processo é utilizado o método *threshold* do *OpenCV*. A Figura 4.14 ilustra o resultado do processo de binarização da imagem.



Figura 4.14 – Imagem após o processo de binarização.

Posteriormente, o próximo passo realizado é aplicar o algoritmo de *CANNY*, para identificar as bordas da imagem, que é realizado através do método *canny* da biblioteca *OpenCV*, em que recebe 3 parâmetros, a imagem binarizada, o valor mínimo do limiar, e por fim o valor máximo do limiar. A Figura 4.15 ilustra o resultado desse processo.



Figura 4.15 – Imagem após o processo do Método de Canny.

Em seguida é realizado o processo de aplicação das operações morfológicas, onde é realizado três etapas, que são a criação do kernel dilatação e erosão. A criação do kernel consiste em criar uma matriz com valores preenchidos com o número 1 na qual será utilizada para realização de operações morfológicas, pelas etapas de dilatação e erosão. Onde esse processo é realizado através do método *ones* da biblioteca *NUMPY*.

Após a criação do kernel, é realizada a dilação da imagem, que se trata de um processo que irá expandir as regiões da cor branca da imagem, possibilitando realizar o preenchimento de um espaçamento entre os pixels (SRISHA; KHAN, 2013). Esse processo é realizado através do método *dilates* do *OpenCV* que recebe como parâmetros a imagem após o método de *Canny*, o kernel criado, e por fim o número de vezes que será aplicado a operação. A Figura 4.16 ilustra o resultado do processo de dilatação da imagem.



Figura 4.16 – Imagem após o processo de dilatação.

Em seguida é realizado o processo de erosão, que faz o trabalho inverso da dilatação, onde consiste em encolher as regiões brancas da imagem, onde também possui o objetivo de corrigir imperfeições na imagem (SRISHA; KHAN, 2013). Onde esse processo é realizado através do método *erode* do *OpenCV* que recebe como parâmetros a imagem, o kernel criado, e por fim o número de vezes que será aplicado a operação. A Figura 4.17 ilustra o resultado do processo de erosão da imagem.



Figura 4.17 – Imagem após o processo de erosão.

A Figura 4.18 mostra os contornos identificados apóas a operação do método *findContours* do *OpenCV*, na qual recebe como parâmetro a imagem apóas o processo de aplicacão das operações morfológicas. Como resultado dessa função é retornada uma lista dos elementos identificados na imagem.

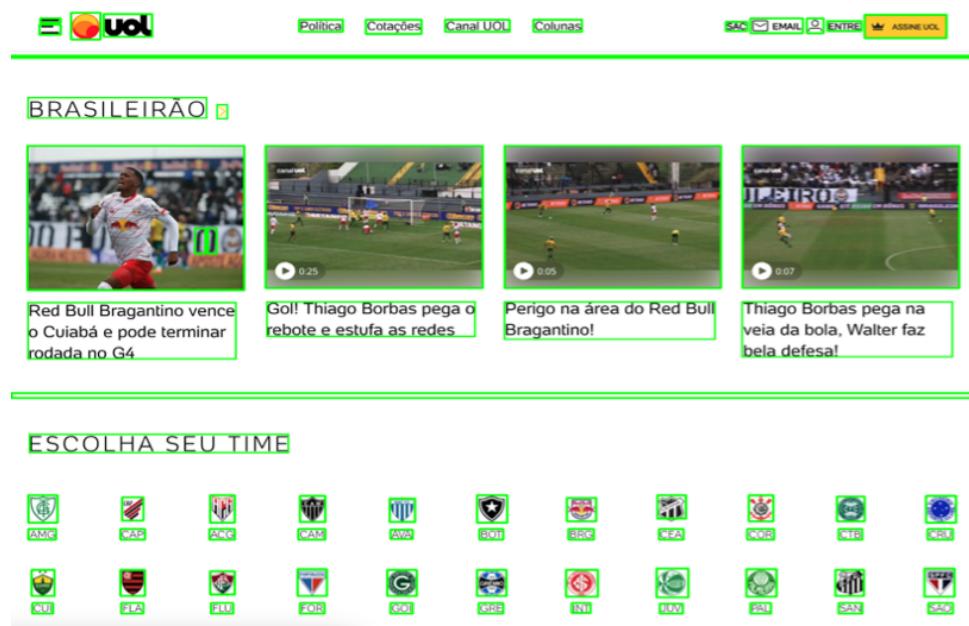


Figura 4.18 – Ilustraçao dos elementos identificados.

## II. CLASSIFICAÇÃO

Inspirado nos trabalhos Goyal *et al.* (2021) e Barz e Sonntag (2021), é realizada a classificaçao dos elementos, onde foi feito através do *Support Vector Machine* (SVM). O treinamento do modelo foi realizado a partir de uma base de dados de treinamento, na qual foram usadas 556 imagens, onde cada imagem foi extraída manualmente de sites aleatórios na internet, onde no total essas imagens são 198 botões, 183 figuras e 175 textos. O Código 4.26 apresenta o treinamento do modelo de classificaçao.

Código 4.26 – Treinamento do modelo de classificaçao.

```
directories = [ "./data_images/botao" , "./data_images/texto" , "./
               data_images/imagem" ]
image_features = []
image_labels = []

for i , dir in enumerate(directories):
    for filename in os.listdir(dir):
        if filename.endswith( ".png" ):
            img = cv2.imread(os.path.join(dir , filename))
            gray = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)
            resized = cv2.resize(gray , (64 , 64))
            flat = resized.flatten()
```

```

    image_features.append( flat )
    image_labels.append( i )

image_features = np.array(image_features)
image_labels = np.array(image_labels)
clf = SVC()
clf.fit(image_features, image_labels)

```

Para realizar o treinamento do modelo, foi utilizado a biblioteca *scikit-learn*. Inicialmente é criado uma série de variáveis, a primeira é a de *directories*, onde possui o caminho das pastas que contém as imagens de treinamento, a segunda variável *image\_features* que é uma lista que irá armazenar as características das imagens, e a terceira variável *image\_labels*, que é uma lista que irá armazenar os rótulos das imagens.

Posteriormente é realizado um loop na qual irá percorrer a variável *directories*, e para cada pasta será percorrido as imagens que estão dentro dela. Para cada imagem é feito processo de leitura através do método *imread*, posteriormente é realizada conversão dessa imagem para escala de cinza através do método *cvtColor*, o redimensionamento da imagem para um tamanho fixo de 64x64, e por fim é passado essa imagem para um vetor.

Ao final de cada iteração são adicionados nas variáveis *image\_features* e *image\_labels*, respectivamente as características e os rótulos das imagens. Onde o rótulo é atribuído por meio de um índice, onde por exemplo o botão possui o índice 0, o texto o índice 1 e a imagem o índice 2. Ao final do loop é feito a conversão das variáveis *image\_features* e *image\_labels* para um array. Por fim é chamado o método *SVC* da biblioteca *scikit-learn*, que irá criar o classificador utilizando o método *fit*, onde é realizado o treinamento do classificador, em que é passado como parâmetro as características e os rótulos das imagens.

Após o processo do treinamento do modelo, o sistema irá pegar as capturas de telas realizadas que foram segmentadas, pela parte de extração de elementos, e irá realizar a classificação desses elementos. O Código 4.27 apresenta a implementação da classificação de elementos.

Código 4.27 – Implementação da classificação de elementos.

```

predicts = []
names = []

for filename in os.listdir(dir):
    if filename.endswith('.png'):
        img_path = os.path.join(dir, filename)

        # Leitura e processamento da imagem
        img = cv2.imread(img_path)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

```

```

resized = cv2.resize(gray, (64, 64))
flat = resized.flatten()
X_new = flat.reshape(1, -1)
y_pred = clf.predict(X_new)
predicts.append(directories[y_pred[0]])
names.append(filename)

```

Inicialmente é declarado as variáveis, *predicts* e *names*, na qual irá armazenar respectivamente as previsões e os nomes dos arquivos que foram classificados.

Posteriormente é realizado a previsão da imagem, por meio do método *predict* da biblioteca *scikit-learn*, na qual recebe como parâmetro as características da imagem e retorna o índice do rótulo da imagem. Ao final é adicionado nas variáveis *predicts* o índice do rótulo da imagem e na variável *names* o nome do arquivo.

A Figura 4.19 mostra um exemplo dos elementos web classificados pelo sistema, onde os elementos delimitados pela cor azul são tipo botão, da cor vermelho do tipo texto e por fim da cor verde do tipo imagem.



Figura 4.19 – Elementos classificados pelo sistema.

### III. INTERSECÇÃO DOS ELEMENTOS COM AS FIXAÇÕES

O processo de intersecção dos elementos com as fixações, consiste em utilizar o algoritmo *Intersection over Union* (IoU), para verificar a sobreposição entre o elemento e a fixação assim como feito no trabalho de (SUN *et al.*, 2019). O algoritmo IoU é definido pela da área de intersecção entre o elemento e a fixação dividido pela soma das áreas do elemento e da fixação subtraído pela área de intersecção. Onde o resultado é valor que varia de 0 a 1, onde um valor maior que 0 indica que existe uma sobreposição entre os elementos, e um valor igual a 0 indica que não possui sobreposição entre os elementos (PADILLA *et al.*, 2020). O Código 4.28 apresenta a função para identificar a intersecção entre as fixações e os elementos.

Código 4.28 – Função para identificar a intersecção entre as fixações e os elementos.

```
def check_intersection(element, fixation):
    xA = max(element[0], fixation[0])
    yA = max(element[1], fixation[1])
    xB = min(element[2], fixation[2])
    yB = min(element[3], fixation[3])

    intersection_area = max(0, xB - xA + 1) * max(0, yB - yA + 1)

    element_area = (element[2] - element[0] + 1) * (element[3] -
        element[1] + 1)
    fixation_area = (fixation[2] - fixation[0] + 1) * (fixation[3] -
        fixation[1] + 1)

    iou = intersection_area / float(element_area + fixation_area -
        intersection_area)

    return iou > 0
```

A realização desse processo é feita através da função *check\_intersection*, que recebe dois parâmetros, o elemento e a fixação. Onde inicialmente é definido as coordenadas de intersecção e em seguida é calculado a área de intersecção, onde posteriormente é calculado a área do elemento e área da fixação. E por fim é realizado o cálculo do IoU, onde caso o resultado seja igual a 0, é retornado *false* que significar que a fixação não está associada ao elemento. Caso o resultado seja maior que 0 é retornado *true*, onde nesse caso é feita a associação entre o elemento e a fixação. Esse processo é repetido para todas as fixações identificadas.

## 5 TESTES E RESULTADOS

Neste capítulo serão discutidos sobre os testes realizados e os resultados que foram obtidos. Foram realizados três tipos de testes para avaliar as funcionalidades do sistema de rastreamento ocular desenvolvido neste trabalho, que são eles, o teste de precisão da estimativa do olhar, teste de eficácia de identificação dos elementos web, e por fim foi feito um teste geral do sistema para análise dos relatórios obtidos.

Para a realização dos testes, foi utilizado a webcam Logitech c920 na resolução de 1920x1080 pixels, a uma taxa de 30 frames por segundo, e o computador utilizado possui um processador Intel Core i9-11900k de 3.5Ghz, com 32GB de memória ram, juntamente com uma placa de vídeo RX6600 XT de 8GB.

### 5.1 TESTE DE PRECISÃO DA ESTIMATIVA DO OLHAR

A realização do teste de precisão da estimativa do olhar foi realizado da seguinte forma, foi exibido uma imagem com pontos fixos em diferentes posições, e desta forma, enquanto o usuário olhava para um ponto fixo, o sistema coletava as coordenadas do olhar do usuário na tela. Conforme apresentado na Figura 5.1.

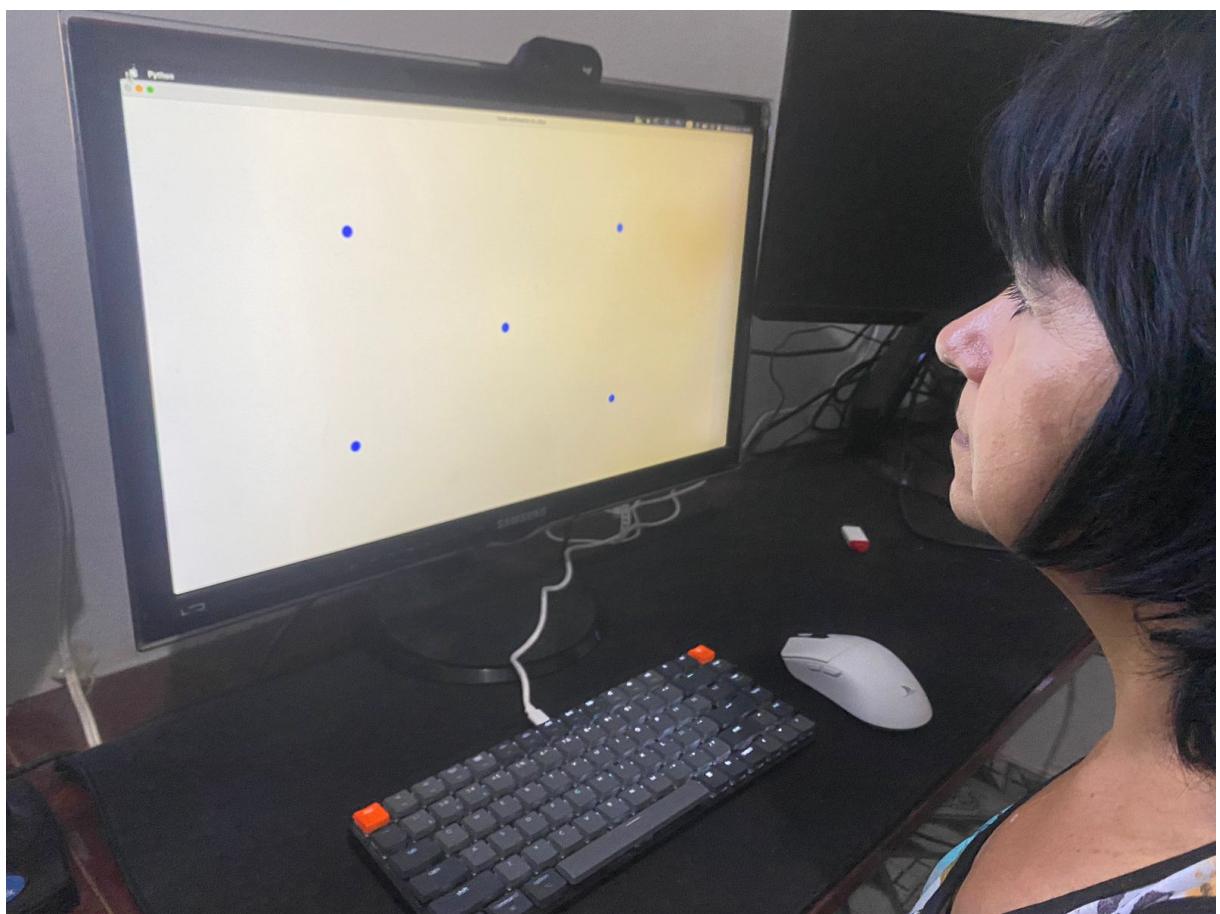


Figura 5.1 – Usuário olhando para um ponto fixo na tela.

A coleta das coordenadas do olhar do usuário foi realizada em cinco pontos fixos, em diferentes posições na tela, conforme apresentado na Figura 5.2 que mostra os 5 pontos em diferentes posições na tela, e a Tabela 5.1 apresenta as coordenadas dos pontos.

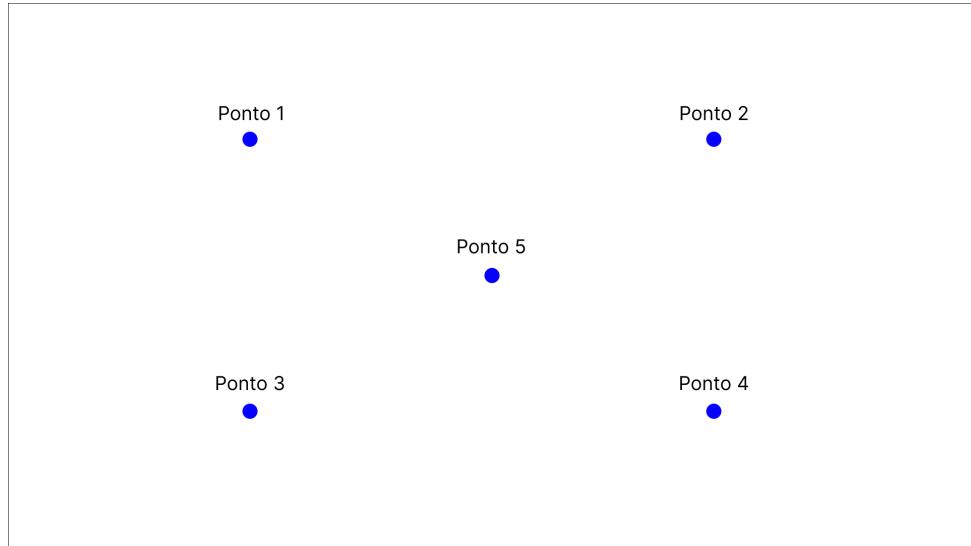


Figura 5.2 – Os pontos que o usuário olhou durante o teste.

Tabela 5.1 – Pontos apresentado na tela.

Ponto	Coordenada (x,y)
1	(480, 270)
2	(1400, 270)
3	(480, 810)
4	(1400, 810)
5	(960, 540)

Após o processo do usuário olhar para os pontos apresentados na tela e consequentemente a coleta das coordenadas do olhar do usuário para cada ponto, foi calculado a precisão da estimativa do olhar em cinco diferentes níveis de tamanho de raio em relação ao ponto fixado na tela. Na Tabela 5.2 é apresentado para cada nível o tamanho do raio em pixels (px).

Tabela 5.2 – Tamanho do raio por nível.

Nível	Tamanho do raio (Pixels)
5	200
4	150
3	100
2	50
1	25

O cálculo da precisão é feito verificando-se a porcentagem de pontos do olhar que está dentro do raio definido.

Donuk *et al.* (2022) e Papoutsaki *et al.* (2016) mostraram que a iluminação pode influenciar os resultados do teste de precisão da estimativa do olhar, com isso, inspirado nesses autores, os testes foram realizados em dois cenários diferentes, um ambiente com boa iluminação, e outro ambiente com baixa iluminação. A Figura 5.3 apresenta o usuário realizando o teste de precisão da estimativa do olhar em um ambiente com boa iluminação e baixa iluminação.



Figura 5.3 – Usuário realizando o teste de precisão da estimativa do olhar em um ambiente com boa iluminação e baixa iluminação.

Após o processo de coleta das coordenadas do olhar do usuário para cada ponto fixado na tela no ambiente com boa iluminação, o sistema criou um arquivo CSV com as coordenadas do olhar do usuário para cada ponto fixado na tela, onde a primeira coluna representa a coordenada X e a segunda coluna representa a coordenada Y, e cada linha representa uma coordenada do olhar do usuário. A Figura 5.4 apresenta os arquivos CSV gerados pelo sistema com as coordenadas do olhar do usuário para cada ponto.

ponto1.csv	ponto2.csv	ponto3.csv
X, Y 481.25, 313.81 522.79, 324.14 514.71, 272.48 474.22, 280.91 466.27, 289.97 476.51, 298.69 533.97, 262.01 520.97, 201.61 537.49, 219.43 509.35, 229.48 484.98, 249.26 458.63, 278.25 490.84, 303.19 516.20, 290.03 528.28, 282.56 507.84, 259.00 501.99, 241.27 494.13, 237.65 504.03, 228.85 501.42, 222.84	X, Y 1067.40, 197.63 1153.19, 184.19 1198.13, 193.80 1228.49, 202.95 1259.60, 199.19 1295.09, 203.68 1267.16, 227.57 1247.38, 264.26 1220.55, 280.40 1198.60, 289.34 1196.99, 291.15 1209.36, 293.00 1269.63, 256.62 1249.70, 255.72 1228.22, 203.55 1234.22, 197.33 1208.22, 203.00 1231.03, 189.59 1233.32, 190.74 1246.55, 161.67	X, Y 538.86, 690.68 396.86, 760.38 297.68, 784.91 149.24, 757.69 168.01, 735.44 305.44, 780.15 289.35, 784.21 411.22, 799.79 577.11, 800.55 681.37, 781.27 745.10, 770.47 772.84, 823.76 655.68, 806.05 551.39, 786.67 466.37, 785.75 401.83, 753.35 382.04, 712.96 361.46, 711.68 362.29, 675.68 420.80, 725.62
ponto4.csv	ponto5.csv	
X, Y 1181.02, 817.97 1245.40, 771.62 1288.24, 810.89 1278.25, 800.53 1282.47, 875.65 1300.92, 878.58 1225.61, 874.10 1262.79, 874.23 1205.83, 901.00 1233.06, 931.14 1249.74, 866.44 1319.11, 871.52 1341.50, 868.96 1369.28, 834.25 1323.80, 837.19 1307.71, 841.71 1359.47, 840.03 1321.20, 882.86 1290.18, 918.65 1773.63, 970.54	X, Y 910.78, 608.23 934.75, 600.74 877.78, 586.41 842.85, 580.55 827.03, 575.01 827.02, 614.51 826.59, 608.56 814.07, 580.38 839.37, 627.85 844.15, 600.72 836.06, 580.97 831.80, 564.50 841.61, 580.74 857.84, 585.85 880.23, 587.26 885.19, 597.45 901.06, 597.42 901.74, 543.71 922.88, 551.47 923.76, 551.52	

Figura 5.4 – Arquivos CSV com as coordenadas do olhar do usuário para cada ponto fixado na tela no ambiente com boa iluminação.

Com os dados das coordenadas do olhar do usuário para cada ponto fixado na tela, no ambiente com boa iluminação, apresentado na Figura 5.4, foi calculado para cada nível de raio da Tabela 5.2, a porcentagem de pontos do olhar do usuário que estavam dentro do raio definido. Desta forma, a Tabela 5.3 mostra os resultados que foram obtidos no ambiente com boa iluminação para diferentes níveis de raio.

Tabela 5.3 – Resultados da precisão do olhar para o ambiente de boa iluminação.

Nível	Ponto 1	Ponto 2	Ponto 3	Ponto 4	Ponto 5
5	99,69%	98,81%	94,29%	89,18%	98,76%
4	98,24%	91,28%	83,17%	71,67%	91,90%
3	83,23%	70,60%	51,93%	45,26%	70,29%
2	40,77%	26,07%	14,41%	14,08%	29,96%
1	13,00%	8,80%	4,56%	4,42%	9,33%

Conforme mostrado na Tabela 5.3, a precisão do olhar foi diminuindo conforme o

tamanho do raio foi reduzindo, o que era esperado. Foi obtido em média para o raio de 200 pixels uma precisão de 96,15%, para o raio de 150 pixels uma precisão de 87,25%, para o raio de 100 pixels uma precisão de 64,26%, para o raio de 50 pixels uma precisão de 33%, por fim o raio de 25 pixels uma precisão de 8,02%. Mostrando assim, que 150 pixels é o raio de tamanho mínimo para que o seja capaz de estimar a posição olhar do usuário com uma boa precisão.

Para o ambiente com baixa iluminação, A Figura 5.5, mostra os arquivos CSV gerados pelo sistema com as coordenadas coletadas do olhar do usuário para cada ponto fixado na tela.

**ponto1.csv**

X, Y
921.41, 784.57
599.23, 563.97
540.80, 545.35
450.29, 493.96
363.27, 464.00
454.12, 362.86
302.15, 369.65
215.99, 403.65
284.49, 445.76
354.09, 356.89
533.18, 380.87
456.34, 363.36
529.64, 391.33
519.79, 434.91
480.10, 490.45
426.80, 460.93
468.54, 417.15
481.75, 423.66
593.25, 373.47
559.94, 379.02
585.84, 369.72
582.02, 372.54
644.48, 425.27

**ponto2.csv**

X, Y
1247.95, 271.07
1371.17, 357.19
1310.96, 392.63
1385.20, 386.39
1231.92, 294.37
1151.12, 317.77
1199.42, 402.70
1330.98, 404.15
1484.04, 434.00
1419.70, 430.91
1272.68, 426.92
1310.07, 411.35
1295.21, 387.54
1333.51, 351.95
1416.16, 343.30
1386.41, 368.28
1350.57, 425.05
1213.47, 432.25
1154.22, 372.04
1039.95, 310.39
1102.93, 310.82
1267.69, 375.80
1208.41, 357.47

**ponto3.csv**

X, Y
577.55, 570.48
739.34, 484.99
819.61, 478.46
746.15, 506.29
729.99, 510.94
757.70, 480.34
727.91, 439.93
743.96, 463.60
750.48, 466.53
757.35, 425.26
669.27, 443.28
726.30, 489.79
824.21, 503.51
832.99, 484.51
912.17, 490.64
847.39, 493.36
710.60, 481.83
810.24, 480.84
848.31, 449.25
700.06, 516.00
741.86, 536.92
659.66, 582.49
722.86, 546.54

**ponto4.csv**

X, Y
1208.72, 481.34
1211.02, 486.02
1194.56, 526.33
1206.38, 555.32
1145.67, 616.38
1093.72, 700.34
1103.53, 701.35
1057.80, 690.61
1155.94, 690.71
1209.08, 647.50
1180.27, 647.83
1155.18, 665.89
998.12, 561.65
1006.63, 506.95
1180.22, 543.75
1118.09, 582.70
1108.35, 600.30
1183.62, 554.57
1123.96, 563.54
1099.73, 585.58
1098.55, 544.76
1060.00, 510.64
1156.71, 476.61

**ponto5.csv**

X, Y
950.65, 759.98
1086.87, 776.58
1100.74, 804.25
1073.05, 798.57
1029.79, 775.25
1098.44, 803.75
914.87, 684.51
896.73, 641.00
860.35, 638.77
966.26, 677.45
1068.95, 693.08
1022.68, 767.39
1015.13, 800.63
887.11, 702.26
901.92, 668.24
859.40, 668.39
977.03, 721.20
1059.43, 739.21
1094.41, 698.37
1043.74, 754.56
1004.36, 742.79
1035.42, 684.94
1094.12, 669.95

Figura 5.5 – Arquivos CSV com as coordenadas do olhar do usuário para cada ponto fixado na tela no ambiente com baixa iluminação.

Através das coordenadas do olhar do usuário no ambiente com baixa iluminação, apresentado na Figura 5.5, foi feito o mesmo processo realizado no ambiente com boa

iluminação, onde foi calculado para cada nível de raio da Tabela 5.2, a porcentagem de pontos do olhar do usuário que estavam dentro do raio definido. Com isso, foram obtidos os resultados apresentados na Tabela 5.4.

Tabela 5.4 – Resultados da precisão do olhar para o ambiente de baixa iluminação.

Nível	Ponto 1	Ponto 2	Ponto 3	Ponto 4	Ponto 5
5	46,89%	22,29%	55,09%	29,76%	56,65%
4	27,22%	9,21%	33,12%	15,55%	30,21%
3	11,51%	2,85%	13,68%	5,90%	10,18%
2	2,88%	0,42%	3,08%	1,41%	1,20%
1	0,60%	0,06%	0,65%	0,27%	0,08%

De acordo com os resultados mostrados na Tabela 5.4, assim como o teste realizado no ambiente de boa iluminação, conforme o tamanho do raio foi reduzindo, a precisão do olhar também foi diminuindo. Porém, os resultados da precisão do olhar foram muito inferiores ao ambiente com boa iluminação, onde obteve-se média para o raio de 200px uma precisão de 42,14%, para o raio de 150px uma precisão de 23,06%, para o raio de 100px uma precisão de 8,82%, para o raio de 50px uma precisão de 1,80%, e por fim para o raio 25px de 0,33%. De acordo com os resultados obtidos, o sistema não possui uma boa precisão para estimar o olhar do usuário em situações na qual o ambiente possua uma baixa iluminação.

O Figura 5.6 mostra a comparação da precisão média da estimativa do olhar, da Tabela 5.3 e da Tabela 5.4 para o ambiente com boa iluminação e baixa iluminação, respectivamente.

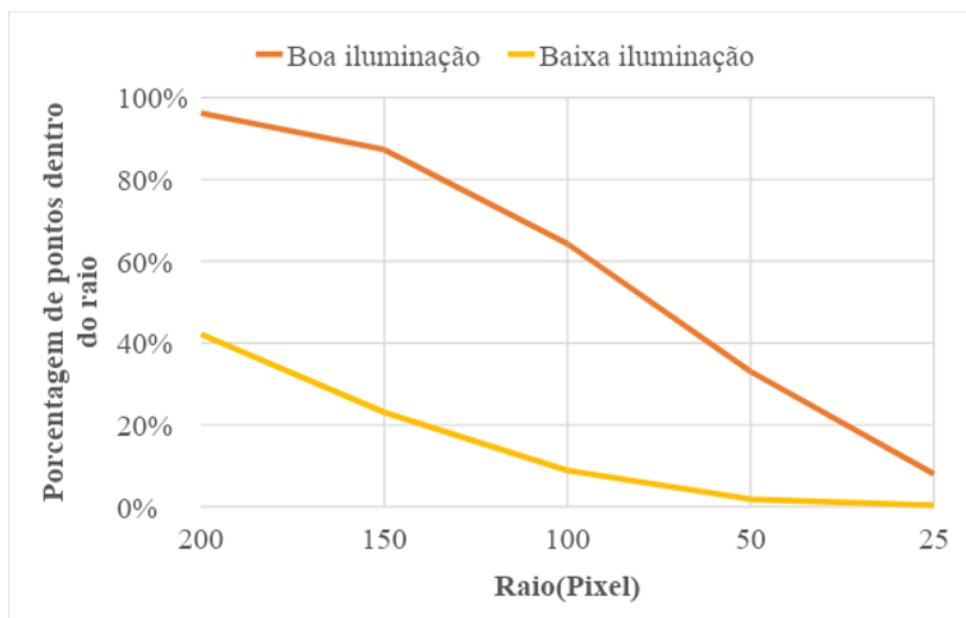


Figura 5.6 – Comparação da média dos resultados obtidos no ambiente com boa iluminação e baixa iluminação.

Conforme mostrado no Figura 5.6, os resultados obtidos no ambiente com boa iluminação foram muito superiores em comparação com o ambiente com baixa iluminação, tendo uma diferença de média de 42,51%.

## 5.2 TESTE DE PRECISÃO DE IDENTIFICAÇÃO DE ELEMENTOS WEB

Os testes de precisão da identificação de elementos web, foram realizados através de capturas de telas de sites, para que o sistema pudesse identificar e retornar os elementos textuais, figuras e botões presentes em cada imagem. Onde o cálculo da porcentagem de precisão foi feito pela divisão da quantidade de cada elemento detectado pelo sistema pela quantidade real de cada elemento que realmente existe na imagem multiplicado por 100.

Os testes foram realizados através de quatro capturas de telas de diferentes sites, na qual cada captura de tela possui uma quantidade de elementos diferentes entre si. A primeira captura de tela foi realizada no site institucional da UENF, a segunda captura de tela foi realizada no site Google images, em uma pesquisa de imagens de pessoas, a terceira captura de tela foi realizada na página inicial do UOL esporte, e por fim a quarta captura de tela foi realizada no site da UENF em uma página de notícia. A Figura 5.7, apresenta as capturas de telas dos sites utilizados para os testes de precisão da identificação de elementos web.

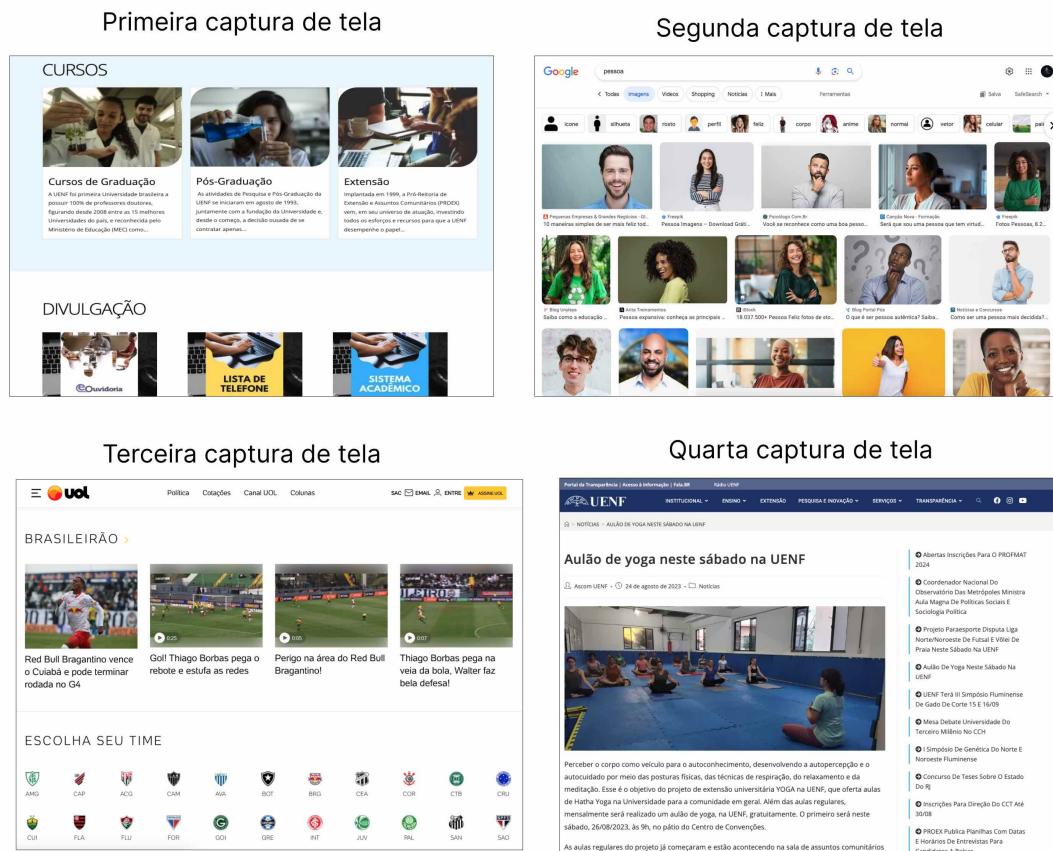


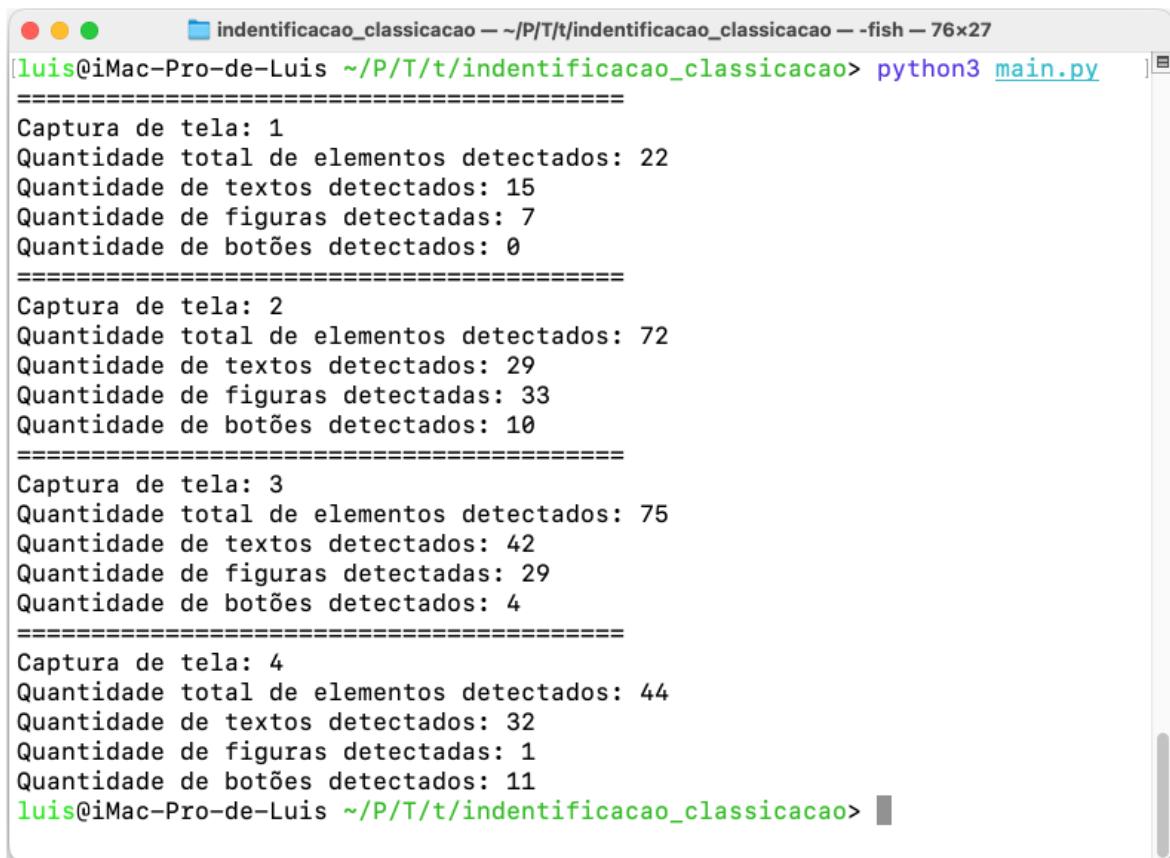
Figura 5.7 – Capturas de telas dos sites utilizados para os testes de precisão da identificação de elementos web.

A partir das capturas de telas da Figura 5.7, foram inicialmente identificados manualmente os elementos textuais, figuras e botões presentes em cada imagem, para que fosse possível realizar a comparação com os elementos identificados pelo sistema. A Tabela 5.5 apresenta a quantidade de elementos textuais, figuras e botões identificados manualmente em cada captura de tela.

Tabela 5.5 – Quantidade de elementos textuais, figuras e botões identificados manualmente em cada captura de tela.

Captura de tela	Texto	Figura	Botão
1	8	6	0
2	21	28	12
3	39	26	4
4	17	1	14

Com os elementos identificados manualmente em cada captura de tela, foi realizada a identificação dos elementos pelo sistema. A Figura 5.8 apresenta a quantidade de cada tipo de elementos identificados pelo sistema para todas as capturas de telas realizadas.



```

indentificacao_classicacao -- ~/P/T/t/indentificacao_classicacao -- -fish -- 76x27
luis@iMac-Pro-de-Luis ~/P/T/t/indentificacao_classicacao> python3 main.py
=====
Captura de tela: 1
Quantidade total de elementos detectados: 22
Quantidade de textos detectados: 15
Quantidade de figuras detectadas: 7
Quantidade de botões detectados: 0
=====
Captura de tela: 2
Quantidade total de elementos detectados: 72
Quantidade de textos detectados: 29
Quantidade de figuras detectadas: 33
Quantidade de botões detectados: 10
=====
Captura de tela: 3
Quantidade total de elementos detectados: 75
Quantidade de textos detectados: 42
Quantidade de figuras detectadas: 29
Quantidade de botões detectados: 4
=====
Captura de tela: 4
Quantidade total de elementos detectados: 44
Quantidade de textos detectados: 32
Quantidade de figuras detectadas: 1
Quantidade de botões detectados: 11
luis@iMac-Pro-de-Luis ~/P/T/t/indentificacao_classicacao>

```

Figura 5.8 – Elementos web identificados pelo sistema.

Através da identificação dos elementos web realizado pelo sistema, apresentado na Figura 5.8, foi calculado a porcentagem de precisão da identificação de elementos web para cada captura de tela, apresentado na Tabela 5.6.

Tabela 5.6 – Resultados da precisão de identificação de elementos WEB.

Captura de tela	Texto	Figura	Botão
1	53,33%	85,71%	100%
2	72,41%	84,85%	83,33%
3	92,86%	89,66%	100%
4	53,13%	100%	78,57%

Conforme mostrado na Tabela 5.6, o sistema foi capaz de identificar elementos de texto com uma precisão média de 67,18%, enquanto que para elementos de figura e botão, a precisão média foi de 90,05% e 90,48%, respectivamente, onde esses resultados é mostrado na Figura 5.9 a seguir.

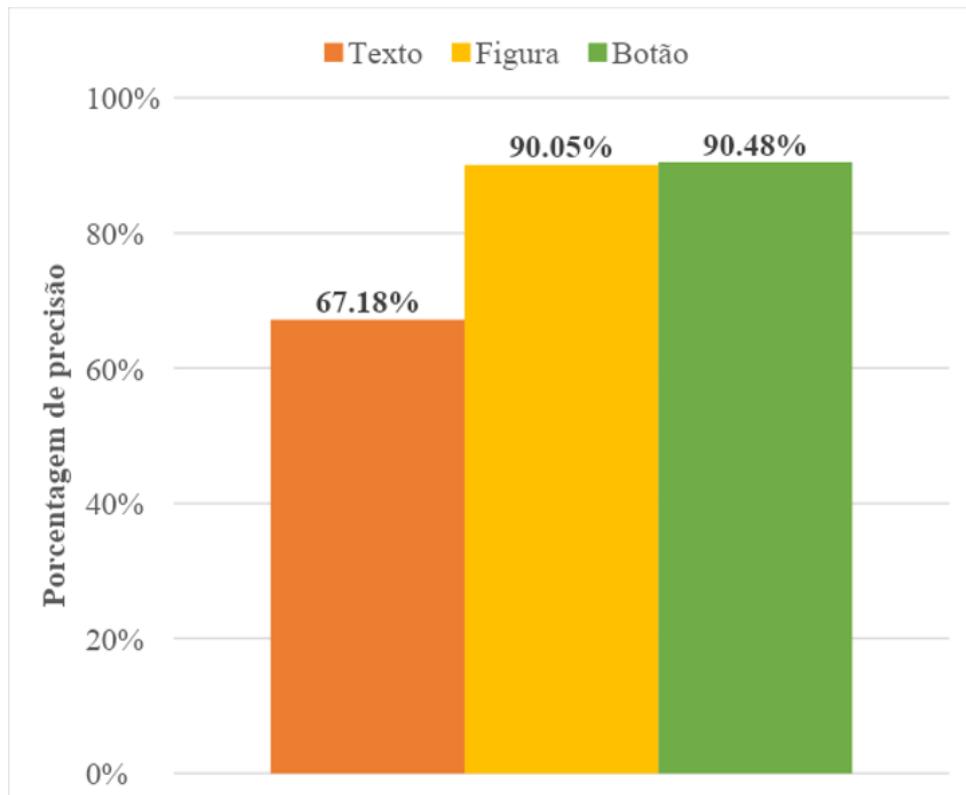


Figura 5.9 – Comparaçāo da média dos resultados por tipo de elemento WEB.

### 5.3 TESTE GERAL DO SISTEMA

O teste geral do sistema consiste, em realizar um experimento real da aplicação executando todas as funcionalidades do sistema, na qual foram analisados os resultados obtidos. Onde os testes foram realizados no site do curso de Ciência da computação da Universidade Estadual do Norte Fluminense (UENF), por meio dessa url <https://cc.uenf.br/>.

Inicialmente, através da interface do sistema, foi selecionada a opção de "executar", onde foi aberto uma nova tela para que o usuário prenchesse as informações do experimento. A Figura 5.10 apresenta o usuário selecionando a opção de "executar" o experimento na interface do sistema.

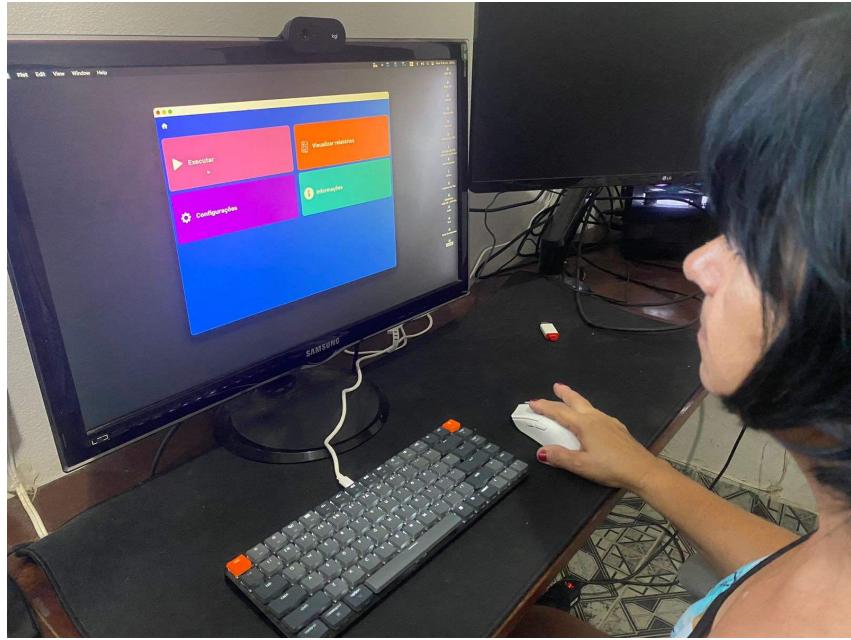


Figura 5.10 – Usuário selecionando a opção de executar o experimento na interface do sistema.

A Figura 5.11, mostra o usuário preenchendo as informações do experimento, onde foi preenchido o nome do experimento, o endereço URL do site e por fim a resolução da tela.

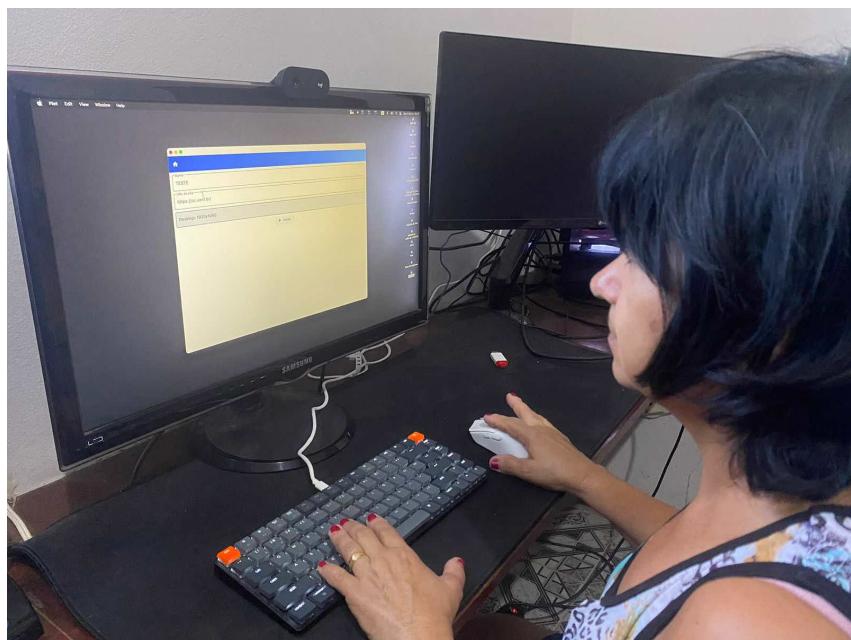


Figura 5.11 – Usuário preenchendo as informações do experimento.

Em seguida, foi feito o processo de calibração do sistema, onde o usuário olhou para os pontos de calibração apresentados na tela, conforme apresentado na Figura 5.12.

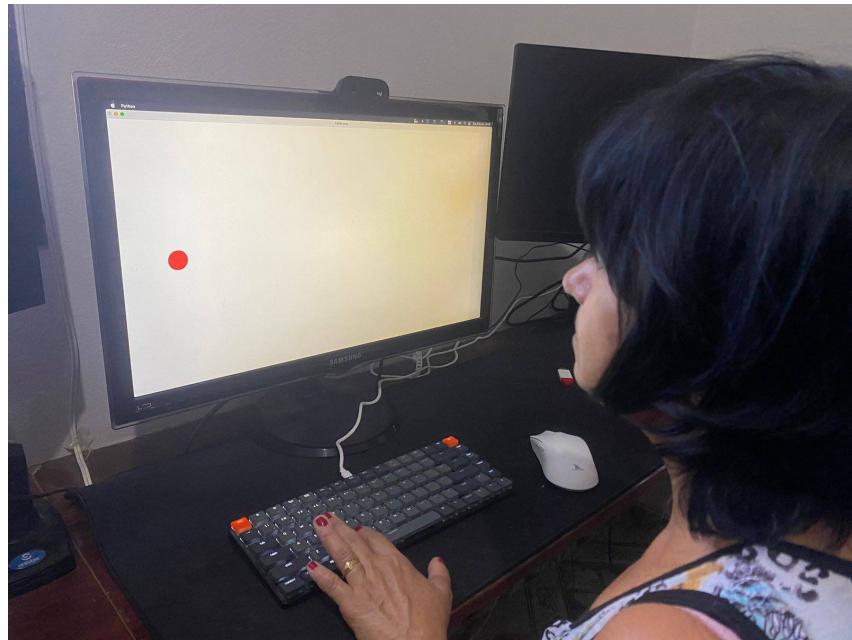


Figura 5.12 – Usuário olhando para os pontos de calibração apresentados na tela.

Após realizar o processo de calibração, iniciou-se o processo de navegação no site onde o sistema coletou tanto as capturas de telas do website quanto os pontos do olhar do usuário na tela. A Figura 5.13 apresenta o usuário navegando no site.

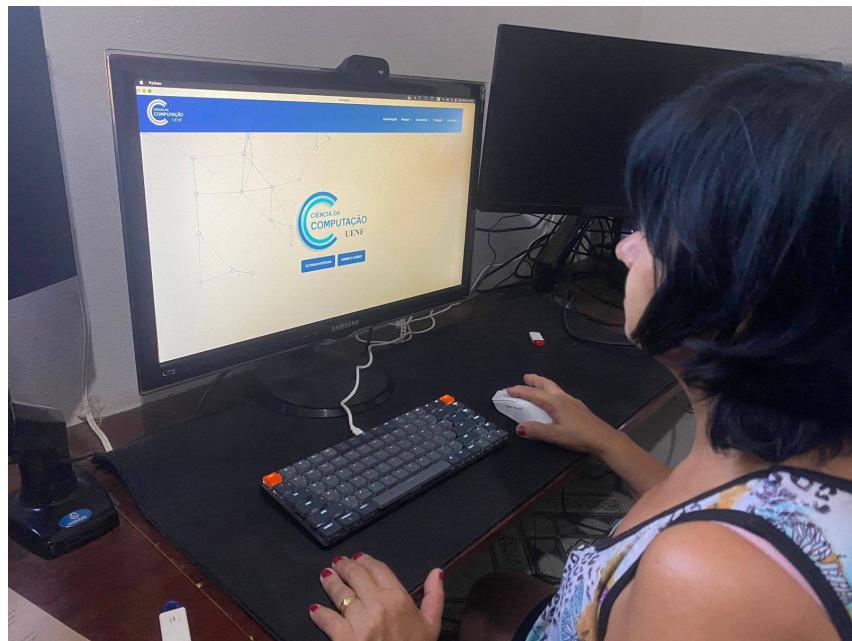


Figura 5.13 – Usuário navegando no site.

O sistema registrou três capturas de telas, na qual a primeira se trata da região inicial do site onde possui a logo do curso e alguns botões de navegação, já segunda captura de tela possui as notícias do site, e por fim a terceira captura de tela possui informações sobre o curso e o rodapé do site, conforme apresentado na Figura 5.14.



Primeira captura de tela



Segunda captura de tela



Terceira captura de tela

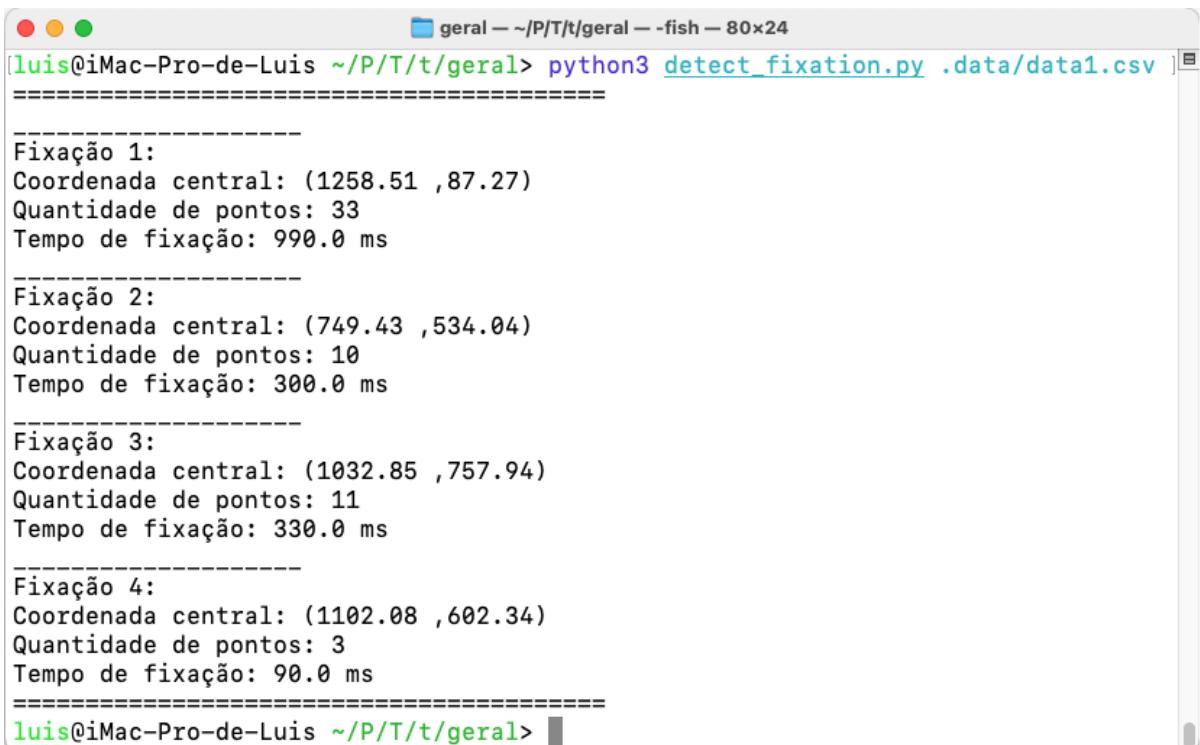
Figura 5.14 – Capturas de telas feitas pelo sistema durante a navegação no site.

Além disso, o sistema coletou as coordenadas do olhar do usuário durante a navegação no site. A Figura 5.15 apresenta as coordenadas do olhar do usuário para cada captura de tela realizada.

X, Y	X, Y	X, Y
1134.98, 88.92	900.63, 349.27	681.30, 306.77
1125.07, 88.45	898.68, 335.99	655.12, 318.01
1177.73, 95.37	887.42, 347.52	709.48, 294.19
1162.17, 108.84	864.61, 347.66	792.76, 337.38
1240.18, 78.39	874.30, 345.14	772.53, 294.03
1208.31, 123.38	872.96, 354.75	807.32, 303.80
1153.03, 113.45	918.29, 367.65	705.48, 292.45
1151.42, 141.88	917.12, 377.09	684.22, 295.38
1182.48, 217.32	895.14, 406.31	637.28, 291.65
1191.82, 150.77	896.14, 380.63	657.74, 284.81
1172.10, 62.98	875.93, 409.18	777.10, 259.36
1169.84, 72.04	898.43, 431.80	870.50, 239.20
1157.95, 73.96	874.84, 432.20	1069.17, 257.32
1181.92, 146.94	863.38, 445.26	1185.00, 297.79
1185.80, 167.75	863.88, 464.39	1268.03, 305.11
1192.51, 138.90	890.31, 446.46	1235.48, 313.24
1215.40, 135.04	849.97, 448.74	1110.10, 204.53
1164.60, 144.56	788.56, 456.44	1007.27, 160.44
1172.00, 147.61	723.71, 474.56	1004.35, 147.25
1155.95, 128.67	776.20, 478.23	969.25, 155.57
1200.17, 128.88	846.79, 457.11	957.64, 169.28
1209.52, 221.94	795.01, 443.10	950.52, 167.25

Figura 5.15 – Coordenadas do olhar do usuário para cada captura de tela realizada.

Com as coordenadas do olhar, foi indentificado pelo sistema, as fixações realizadas pelo usuário em cada captura de tela. A Figura 5.16 mostra que para a primeira captura de tela, foram detectadas 4 fixações, onde a primeira fixação tem sua coordenada central no ponto (1258.51, 87.27), possui 33 pontos e um tempo de duração de 990 milissegundos. A segunda fixação possui a coordenada central no ponto (749.43, 534.04), com 10 pontos e 300 milissegundos de duração. A terceira e a quarta fixação possuem as coordenadas centrais nos pontos (1032.85, 757.94) e (1102.08, 602.34), com 11 e 3 pontos e 330 e 90 milissegundos de duração, respectivamente.



```
gerald -- ~/P/T/t/gerald -- -fish -- 80x24
luis@iMac-Pro-de-Luis ~/P/T/t/gerald> python3 detect_fixation.py .data/data1.csv
=====
-----
Fixação 1:
Coordenada central: (1258.51 ,87.27)
Quantidade de pontos: 33
Tempo de fixação: 990.0 ms

-----
Fixação 2:
Coordenada central: (749.43 ,534.04)
Quantidade de pontos: 10
Tempo de fixação: 300.0 ms

-----
Fixação 3:
Coordenada central: (1032.85 ,757.94)
Quantidade de pontos: 11
Tempo de fixação: 330.0 ms

-----
Fixação 4:
Coordenada central: (1102.08 ,602.34)
Quantidade de pontos: 3
Tempo de fixação: 90.0 ms
=====

luis@iMac-Pro-de-Luis ~/P/T/t/gerald>
```

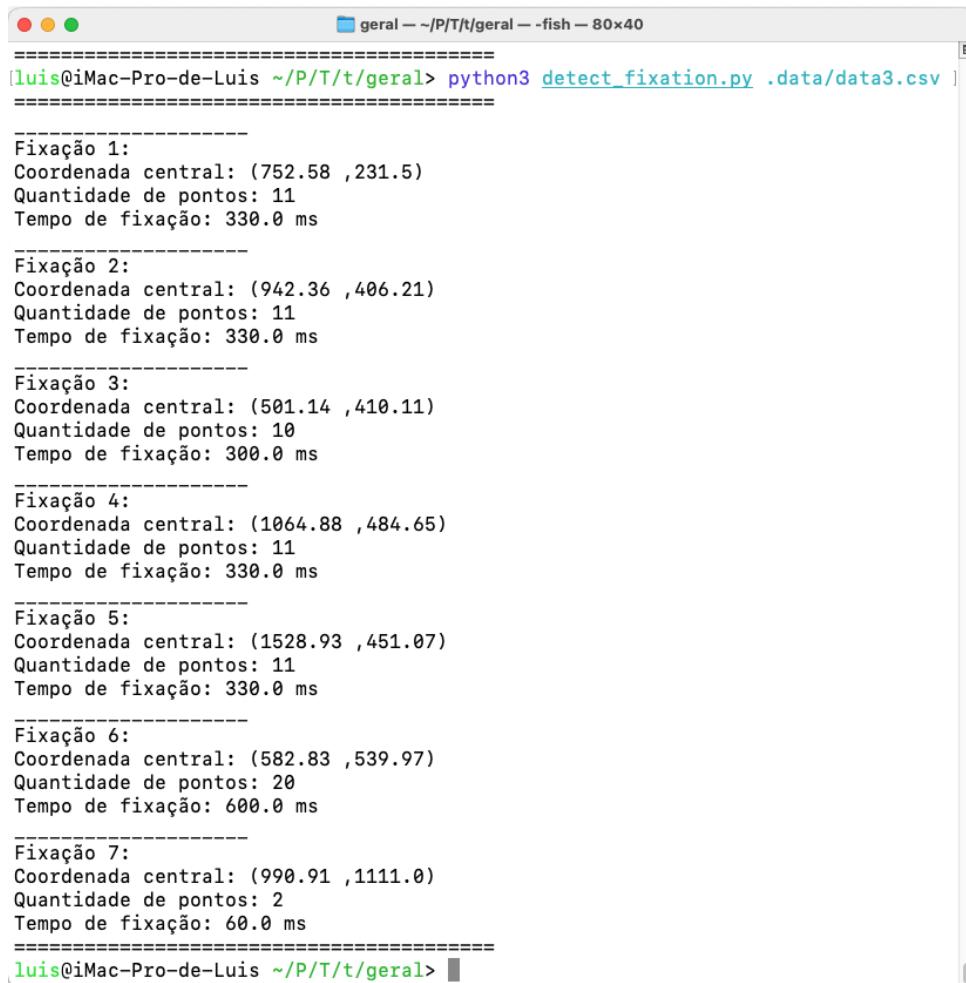
Figura 5.16 – Fixações indentificadas pelo sistema para a primeira captura de tela.

A Figura 5.17, mostra as fixações identificadas para segunda de captura de tela, onde foram detectadas 8 fixações, sendo que a fixação 3, possui a maior quantidade de pontos, com 192 e o maior tempo de duração, com 5760 milissegundos.

```
gerald — ~/P/T/t/geral — fish — 80x44
luis@iMac-Pro-de-Luis ~/P/T/t/geral> python3 detect_fixation.py .data/data2.csv
=====
-----  
Fixação 1:  
Coordenada central: (937.99 ,389.02)  
Quantidade de pontos: 53  
Tempo de fixação: 1590.0 ms  
-----  
Fixação 2:  
Coordenada central: (1391.15 ,368.37)  
Quantidade de pontos: 21  
Tempo de fixação: 630.0 ms  
-----  
Fixação 3:  
Coordenada central: (575.88 ,804.74)  
Quantidade de pontos: 192  
Tempo de fixação: 5760.0 ms  
-----  
Fixação 4:  
Coordenada central: (1048.53 ,589.98)  
Quantidade de pontos: 33  
Tempo de fixação: 990.0 ms  
-----  
Fixação 5:  
Coordenada central: (956.49 ,836.56)  
Quantidade de pontos: 63  
Tempo de fixação: 1890.0 ms  
-----  
Fixação 6:  
Coordenada central: (867.31 ,938.46)  
Quantidade de pontos: 22  
Tempo de fixação: 660.0 ms  
-----  
Fixação 7:  
Coordenada central: (1380.4 ,781.85)  
Quantidade de pontos: 64  
Tempo de fixação: 1920.0 ms  
-----  
Fixação 8:  
Coordenada central: (1425.53 ,895.76)  
Quantidade de pontos: 5  
Tempo de fixação: 150.0 ms  
=====
```

Figura 5.17 – Fixações indentificadas pelo sistema para a segunda captura de tela.

A figura 5.18, apresenta as 7 fixações indentificadas para a terceira captura de tela, onde a fixação 6 teve o maior tempo de duração com 600ms e a maior quantidade de pontos com 20.



```

luis@iMac-Pro-de-Luis ~/P/T/t/geral> python3 detect_fixation.py .data/data3.csv
=====
Fixação 1:
Coordenada central: (752.58 ,231.5)
Quantidade de pontos: 11
Tempo de fixação: 330.0 ms

-----
Fixação 2:
Coordenada central: (942.36 ,406.21)
Quantidade de pontos: 11
Tempo de fixação: 330.0 ms

-----
Fixação 3:
Coordenada central: (501.14 ,410.11)
Quantidade de pontos: 10
Tempo de fixação: 300.0 ms

-----
Fixação 4:
Coordenada central: (1064.88 ,484.65)
Quantidade de pontos: 11
Tempo de fixação: 330.0 ms

-----
Fixação 5:
Coordenada central: (1528.93 ,451.07)
Quantidade de pontos: 11
Tempo de fixação: 330.0 ms

-----
Fixação 6:
Coordenada central: (582.83 ,539.97)
Quantidade de pontos: 20
Tempo de fixação: 600.0 ms

-----
Fixação 7:
Coordenada central: (990.91 ,1111.0)
Quantidade de pontos: 2
Tempo de fixação: 60.0 ms
=====

luis@iMac-Pro-de-Luis ~/P/T/t/geral>

```

Figura 5.18 – Fixações indentificadas pelo sistema para a terceira captura de tela.

A partir das informações das Figuras 5.16, 5.17 e 5.18, foi gerado os relatórios de resultados do sistema, como mapa de fixação, mapa de calor, scanpath e informações dos elementos web detectados para cada captura de tela realizada. A seguir serão analisados os resultados que foram obtidos para cada relatório gerado.

### 5.3.1 MAPA DE FIXAÇÕES

O mapa de fixação mostra as regiões de atenção visual do usuário, onde são representadas por círculos, que ficam posicionados na região que o usuário fixou o seu olhar, onde o tamanho do círculo é variável de acordo com a duração de tempo que o usuário fixou o seu olhar naquela região.

O mapa de fixações gerado para a primeira captura de tela, é apresentado na Figura 5.19, onde é foi detectado quatro região de fixações, uma na barra de navegação focado no botão chamado "Apresentação", onde foi a região na qual levou o maior tempo, como pode ser visualizado pelo tamanho do círculo. A segunda fixação foi feita na logo do curso, a terceira foi numa região que faz parte da logo, porém no texto "Computação", com uma curta duração. A quarta e a última fixação foi feita no botão "Sobre o curso".

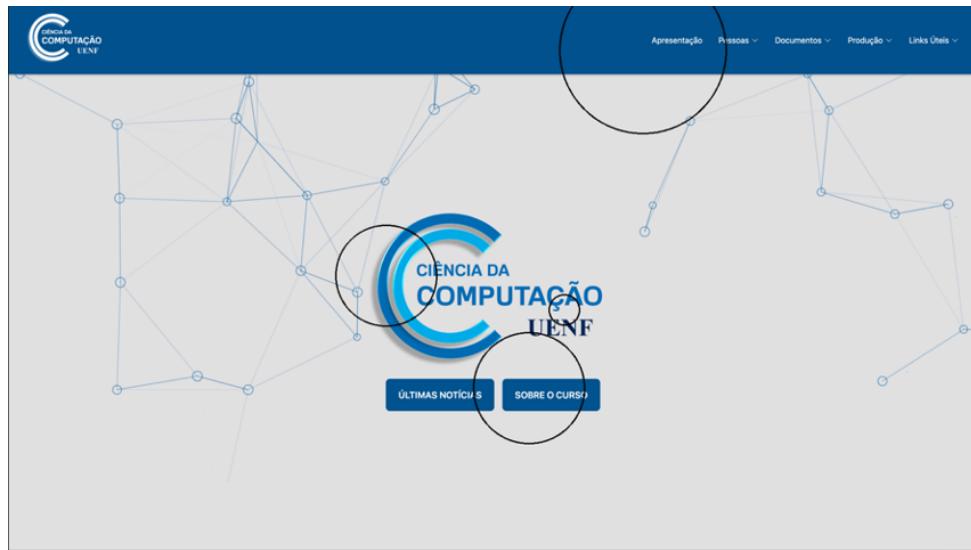


Figura 5.19 – Fixações detectadas para a primeira captura de tela.

O próximo mapa de fixação é para a segunda captura de tela, apresentado na Figura 5.20, onde teve um total de 7 fixações, sendo a primeira fixação feita numa região próxima ao título do tópico "Notícias", a segunda fixação foi feita numa região sem conteúdo com uma curta duração. A terceira fixação foi feita no título da notícia "Divulgado Calendário de Graduação 2022.1", onde teve o maior tempo de fixações nessa captura de tela. A quarta fixação e quinta foram feitas na notícia "Inicia-se XIV Congresso Fluminense de Iniciação Científica e Tecnológica - CONFICT", uma região de fixação no título e uma outra de curta duração na descrição da notícia, respectivamente. A sexta e sétima região de fixação foram feitas na próxima notícia "Eleições de Centro Acadêmico - UENF", onde a sexta fixação foi feita na região do título e por fim a última fixação foi feita na região da descrição da notícia com uma duração maior.

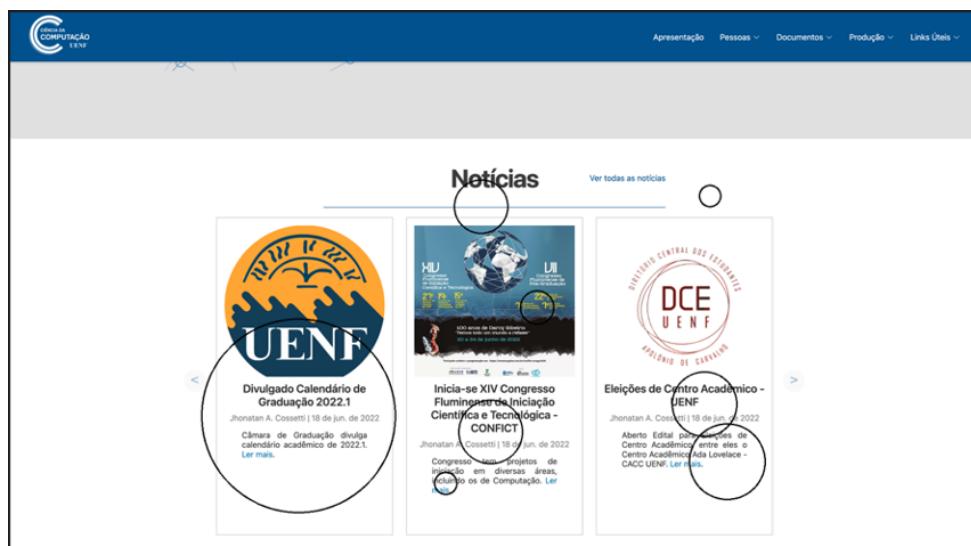


Figura 5.20 – Fixações detectadas para a segunda captura de tela.

O último mapa de fixação foi realizado para terceira captura de tela, apresentado na Figura 5.21, onde teve no total 6 fixações, onde a maior parte deles concentrados na região do texto de apresentação do curso.

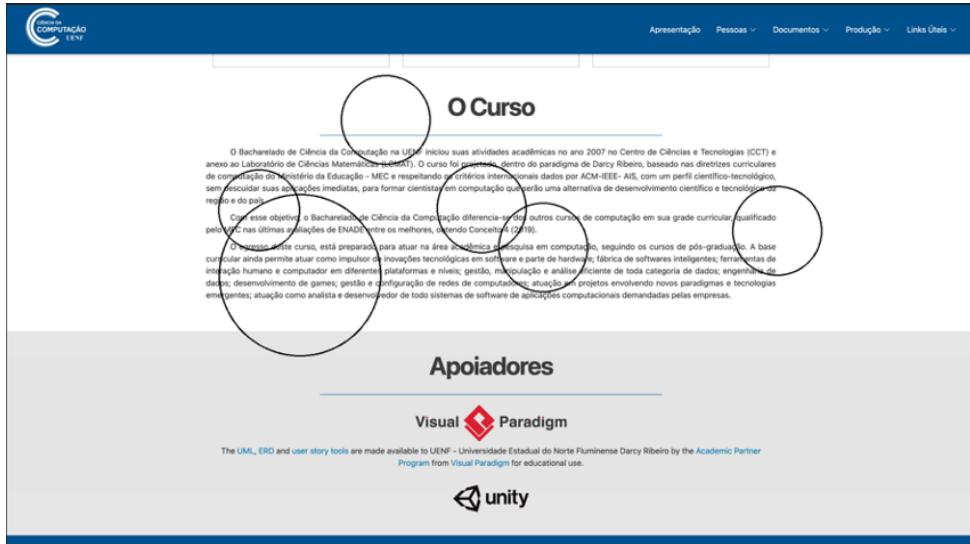


Figura 5.21 – Fixações detectadas para a terceira captura de tela.

Após apresentar os mapas de fixações gerados pelo sistema a partir do experimento realizado no site do curso Ciências da computação da UENF, é apresentado a seguir os resultados do scanpath para cada um desses mapas.

### 5.3.2 SCANPATH

O scanpath mostra a trajetória percorrida pelo olhar do usuário durante a navegação no site, através de círculos conectados por linhas que ficam posicionados nas regiões que o usuário fixou o seu olhar, onde quanto maior o tamanho do círculo, mais tempo o usuário fixou o seu olhar naquela região e os números dentro círculo representam a ordem em que a fixação foi realizada.

O scanpath realizado na primeira captura de tela, apresentado na Figura 5.22, onde mostra a sequência de fixações, onde inicialmente o usuário olhar para a barra de navegação, em seguida para a logo do curso, depois para o botão "Sobre o curso", e por fim olha com uma curta duração para texto pertencente a logo do curso.



Figura 5.22 – Scanpath detectados para a primeira captura de tela.

O próximo scanpath realizado foi para a segunda captura de tela, apresentada na Figura 5.23, onde inicialmente o usuário olhar para o título "Notícias", em seguida, é olhado para uma região sem conteúdo, posteriormente para a primeira notícia onde leva um tempo maior, depois para a segunda notícia, na qual o usuário olhar primeiramente para a imagem, depois para o título e por fim para a descrição da notícia, logo após o usuário olha para a terceira notícia, onde inicialmente ele olha para o título e por fim a descrição da notícia.



Figura 5.23 – Scanpath detectados para a segunda captura de tela.

O último scanpath foi realizada para a terceira captura de tela, apresentado na Figura 5.24, onde mostra que o usuário inicialmente olha para uma região próxima ao título do curso, em seguida os próximos trajetos do olhar são diretamente para o texto de apresentação do curso, na qual posteriormente o olhar do usuário é direcionado para o

rodapé do site, na qual não foi possível visualizar a região pois foi cortado pela captura de tela.

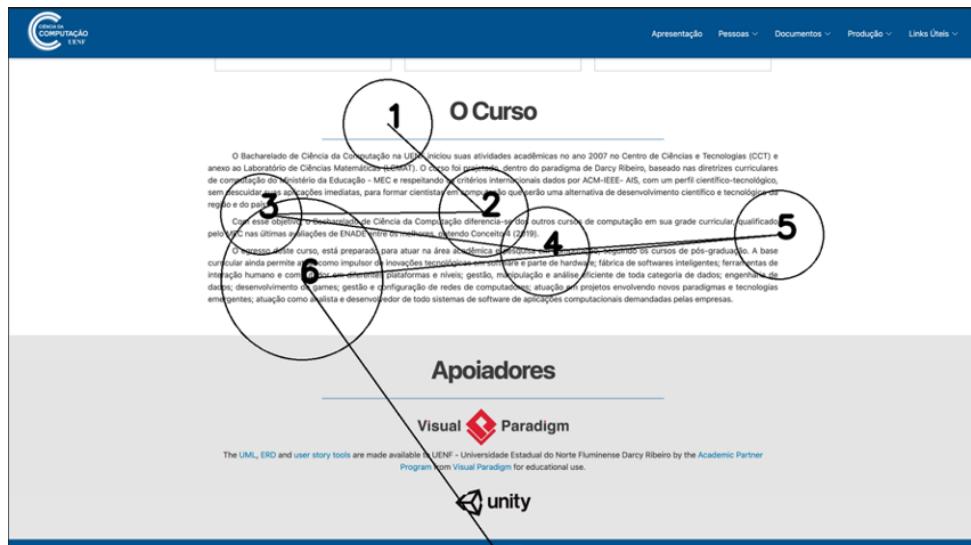


Figura 5.24 – Scanpath detectados para a terceira captura de tela.

Após mostrar a sequência percorrida pelo olhar do usuário através do scanpath, é apresentado o mapa de calor para cada captura de tela realizada.

### 5.3.3 MAPA DE CALOR

O mapa de calor, mostra as regiões que o usuário visualizou, para cada captura de tela realizada, onde as regiões que possuem as cores mais quentes como vermelho e amarelo, representam as regiões mais visualizadas e as cores mais frias como azul, ciano e verde são as regiões menos visualizadas.

No mapa de calor para a primeira captura de tela, apresentado na Figura 5.25, mostra que a região mais visualizada foi a barra de navegação, que teve a cor mais quente, indicando que teve uma duração maior atenção do olhar neste local. A região central da tela que mostra a logo do curso juntamente com os botões também obteve atenção do olhar do usuário considerável.

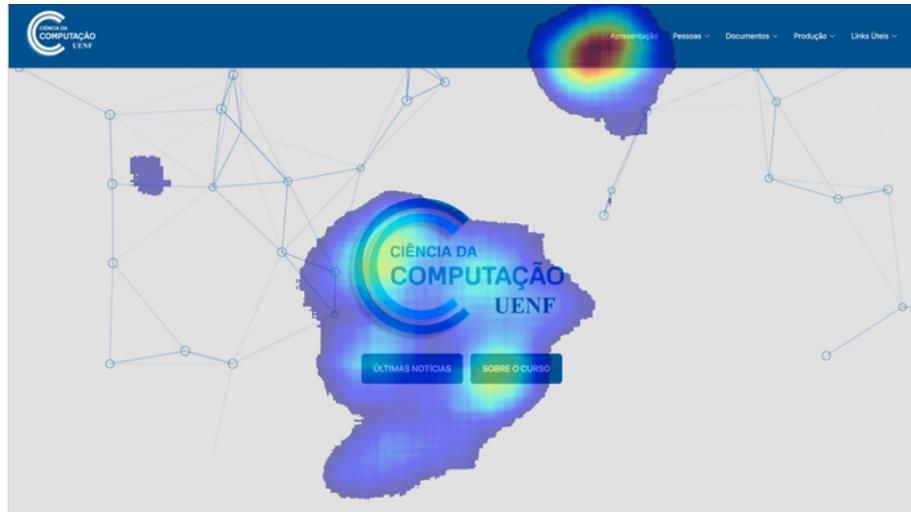


Figura 5.25 – Mapa de calor para a primeira captura de tela.

O próximo mapa calor foi realizado para segunda captura de tela , apresentado na Figura 5.26, onde é mostrado que a região mais visada foi a região das notícias, onde teve a maior concentração na divulgação do calendário acadêmico e a eleição do centro acadêmico.



Figura 5.26 – Mapa de calor para a segunda captura de tela.

O último mapa de calor foi realizado para a terceira captura de tela, apresentado na Figura 5.27, onde a região mais visada pelo usuário foi a área textual da seção de apresentação do curso com uma concentração maior nas laterais do texto. A região do rodapé que mostra seção dos apoiadores também recebeu atenção considerável.

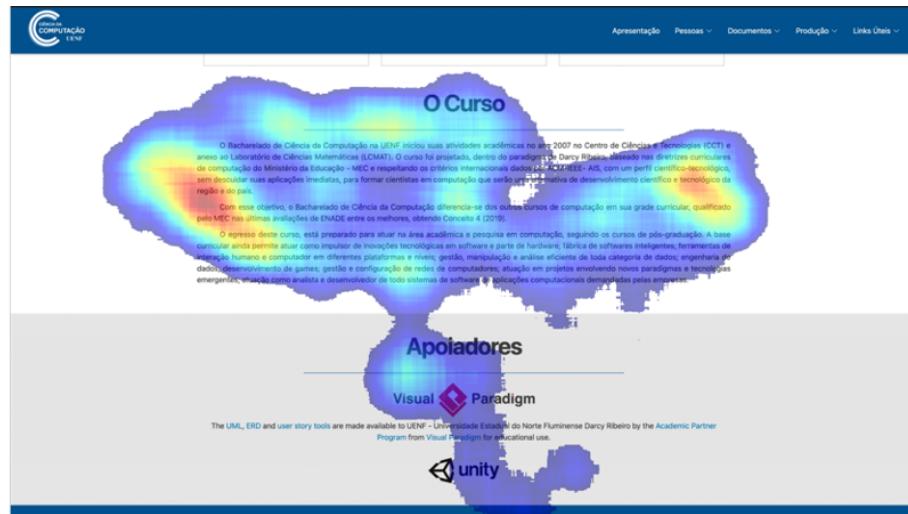


Figura 5.27 – Mapa de calor para a terceira captura de tela.

Após mostrar o mapa de calor das regiões visualizadas pelo usuário durante o experimento para cada captura de tela, é apresentado a seguir as informações de cada elemento web que obteve a atenção visual do usuário.

#### 5.3.4 INFORMAÇÕES DOS ELEMENTOS IDENTIFICADOS

A Figura 5.28 apresenta os três elementos web detectados para a primeira captura de tela, ou seja, os três elementos na qual receberam a fixação do olhar do usuário durante o experimento. Além disso, a Tabela 5.7 mostra as informações sobre cada um dos elementos detectados, onde mostra o tipo de elemento, o tempo de duração das fixações, e as quantidades de fixações e quantidade de pontos do olhar.



Figura 5.28 – Elementos identificados da primeira captura de tela.

Tabela 5.7 – Informações sobre os elementos detectados da primeira captura de tela.

Elemento	Quantidade de Fixações	Quantidade de pontos	Tempo de duração (Milissegundos)	Tipo
A	1	33	990	Botão
B	1	10	300	Figura
C	1	11	330	Botão

Através das informações extraídas da Tabela 5.6, onde mostra que o elemento A e C foi identificado como botões, onde teve apenas 1 fixação em ambos, porém o elemento A teve um tempo de duração de fixação e quantidade de pontos maiores tanto em relação ao elemento C como também para o elemento B, na qual foi identificado, como figura onde também teve uma única fixação e uma quantidade de 10 pontos para essa fixação.

Para a segunda captura de tela foram novamente detectados apenas 3 elementos com fixações, apresentado na Figura 5.29, e na Tabela 5.8 mostra as informações sobre cada um dos elementos.

Onde para a segunda captura de tela os elementos identificados, foram delimitados de uma forma diferente, onde cada elemento foi composto pela imagem, o título e a descrição da notícia. Isso se deve ao fato de que existe uma borda que envolve esses elementos, o que fez com que detecção de contorno considerasse como um único elemento, ao invés de dividir em 3 elementos para cada notícia, um de imagem, título e descrição.

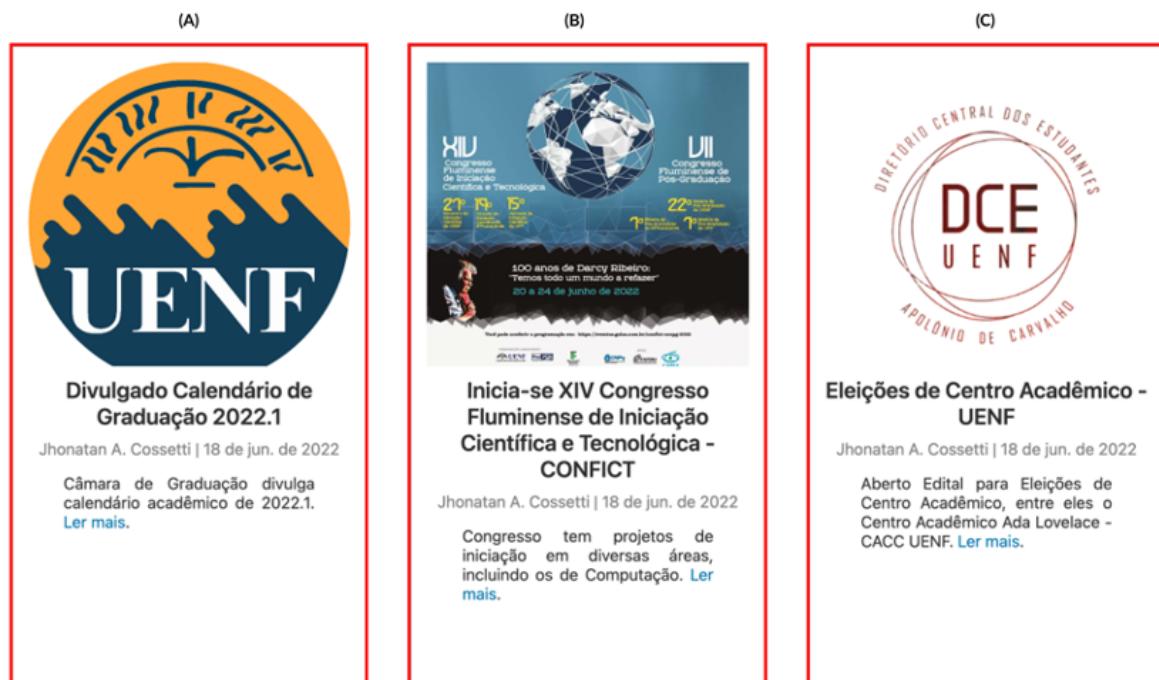


Figura 5.29 – Elementos identificados para a segunda captura de tela.

Tabela 5.8 – Informações sobre os elementos detectados da segunda captura de tela.

Elemento	Quantidade de Fixações	Quantidade de pontos	Tempo de duração (Milissegundos)	Tipo
A	1	192	5760	texto
B	2	118	3540	texto
C	2	69	2070	texto

Conforme mostrado na Tabela 5.8, para todos os elementos identificados, foram classificados como texto, o elemento A teve a maior quantidade de pontos e o maior tempo de duração e obteve apenas uma fixação, enquanto os elementos B e C, tiveram duas fixações, na qual o elemento B teve uma duração e uma quantidade de pontos superior em relação ao elemento C. A Figura 5.30 mostra que para a terceira captura de tela, foram identificados três elementos que teve a intersecção de fixações. E na Tabela 5.9 mostra as informações sobre cada um dos elementos.

- (A) O Bacharelado de Ciência da Computação na UENF iniciou suas atividades acadêmicas no ano 2007 no Centro de Ciências e Tecnologias (CCT) e anexo ao Laboratório de Ciências Matemáticas (LCMAT). O curso foi projetado, dentro do paradigma de Darcy Ribeiro, baseado nas diretrizes curriculares de computação do Ministério da Educação - MEC e respeitando os critérios internacionais dados por ACM-IEEE- AIS, com um perfil científico-tecnológico, sem descuidar suas aplicações imediatas, para formar cientistas em computação que serão uma alternativa de desenvolvimento científico e tecnológico da
- (B) Com esse objetivo, o Bacharelado de Ciencia da Computação diferencia-se dos outros cursos de computação em sua grade curricular, qualificado pelo MEC nas últimas avaliações de ENADE entre os melhores, obtendo Conceito 4 (2019).
- (C) O egresso deste curso, está preparado para atuar na área acadêmica e pesquisa em computação, seguindo os cursos de pós-graduação. A base curricular ainda permite atuar como impulsor de inovações tecnológicas em software e parte de hardware; fábrica de softwares inteligentes; ferramentas de interação humano e computador em diferentes plataformas e níveis; gestão, manipulação e análise eficiente de toda categoria de dados; engenharia de dados; desenvolvimento de games; gestão e configuração de redes de computadores; atuação em projetos envolvendo novos paradigmas e tecnologias emergentes; atuação como analista e desenvolvedor de todo sistemas de software de aplicações computacionais demandadas pelas empresas.

Figura 5.30 – Elementos identificados para a terceira captura de tela.

Tabela 5.9 – Informações sobre os elementos detectados da terceira captura de tela.

Elemento	Quantidade de Fixações	Quantidade de pontos	Tempo de duração (Milissegundos)	Tipo
A	2	21	630	texto
B	1	11	330	texto
C	2	31	930	texto

De acordo com as informações da Tabela 5.9, mostra que todos os elementos foram identificados como o tipo texto, e também mostra que tanto o elemento A quanto o elemento C contém duas fixações, sendo que o elemento C teve a maior quantidade de pontos e também o maior tempo de duração de fixações. Já o elemento B teve apenas uma fixação, e uma quantidade de pontos e o tempo de duração inferior em relação aos outros dois elementos.

## 5.4 DISCUSSÃO DOS RESULTADOS

Os resultados dos testes realizados mostrou uma diferença significativa na precisão da estimativa do olhar do usuário entre o ambiente com boa iluminação e o ambiente com baixa iluminação, onde esse resultado já era esperado e condiz com os resultados já previstos pelos os autores Donuk *et al.* (2022) e Papoutsaki *et al.* (2016), onde eles atribuíram essa diferença ao fato de esta utilizando uma webcam comum para realizar o processo de rastreamento ocular, na qual essa webcam não possui uma boa qualidade para capturar imagens em um ambiente com baixa iluminação, fazendo com que os resultados sejam inferiores em relação ao ambiente com boa iluminação. Porém, para ambientes com uma boa iluminação o sistema consegue obter-se resultados satisfatórios, possibilitando dessa forma estimar as coordenadas do olhar do usuário com uma boa precisão para um raio maior que 150px.

Além disso, o sistema foi capaz de identificar com uma boa precisão os elementos da web, principalmente elementos de figuras e botões da captura de telas. Porém a precisão para identificar elementos textuais foi inferior em relação aos outros elementos, onde muitas vezes, o sistema identificou a região de contorno do elemento de forma errada, ao invés de contornar um parágrafo por exemplo, o sistema contornou vários trechos dentro desse parágrafo, fazendo com que tivesse mais elementos do que deveria.

Através do teste geral do sistema, foi possível observar que o sistema foi capaz de realizar as funcionalidades propostas. Onde foi realizado o processo de exibir a interface do sistema, calibração e navegação no site. E a partir desses processos, foram coletadas informações referente a capturas de telas do site e as coordenadas do olhar do usuário para cada captura de tela realizada. E com essas informações, foi identificado pelo sistema, as fixações realizadas pelo usuário em cada captura de tela. E com isso, foi gerado os relatórios de resultados do sistema, onde o mapa de fixação mostrou as regiões que obteve atenção visual do usuário, o scanpath mostrou a trajetória realizada pelo usuário durante a navegação no site, o mapa de calor mostrou as regiões de maior concentração de atenção visual do usuário, e por fim, as informações dos elementos web que obtiveram atenção visual do usuário.

## 6 CONCLUSÃO

No presente trabalho, foi desenvolvido um sistema de rastreamento ocular na qual realizar a detecção da coordenada da pupila do usuário que navega em uma página web e por meio de um processo de calibração, consegue estimar as regiões que receberam o olhar do usuário em websites.

Os dados coletados pelo sistema, durante o processo de navegação são analisados e gerados quatro tipos de relatórios, os mapas de fixações, scanpath, o mapa de calor e a identificação dos elementos da interface como textos, figuras e botões, que obtiveram a atenção visual do usuário.

Através dos testes realizados, pode-se concluir que o sistema desenvolvido teve um resultado satisfatório em ambientes com boa iluminação, na qual a precisão da estimativa do olhar foi considerada boa para medições feitas em pontos fixos com raio de no mínimo 150 pixels.

Além disso, o sistema obteve uma boa média de precisão para identificar elementos da interface, como figuras, botões e textos. Porém, a precisão da identificação dos textos (67,18%) foi inferior em relação às figuras (90,05%) e botões (90,48%).

Através dos testes realizados e o teste geral do sistema, foi possível comprovar que é possível realizar o processo de rastreamento ocular identificando as coordenadas da pupila do usuário através de uma webcam, e com isso, realizar a estimativa do olhar do usuário em websites, e por meio da análise de dados e a identificação dos elementos da interface, fornecer informações relevantes por meio de relatórios sobre a atenção visual do usuário, e desta forma contribuindo para o entendimento do comportamento do usuário ao navegar em websites.

Em trabalhos futuros poderá ser desenvolvido um sistema que utilize webcam que tenha o recurso de visão noturna ou infravermelho, para que variações de iluminação não interfira na precisão do rastreamento ocular. Além disso, poderá ser implementado a identificação de elementos do tipo texto, através de algoritmos especializados na identificação de textos, como o *Optical Character Recognition* (OCR), para que a precisão na identificação de textos seja maior.

## REFERÊNCIAS

AGGARWAL, Charu. **Neural Networks and Deep Learning: A Textbook.** [S.l.: s.n.], 2018. ISBN 978-3-319-94462-3.

AIPHILE. **IRIS Segmentation Mediapipe Python.** [S.l.: s.n.], jan. 2022. <https://medium.com/mlearning-ai/iris-segmentation-mediapipe-python-a4deb711aae3>. Accessed: 05 Aug 2023.

AWAD, Mariette; KHANNA, Rahul. **Support Vector Machines for Classification.** [S.l.: s.n.], 2015. P. 39–66.

BARZ, M.; SONNTAG, D. Automatic Visual Attention Detection for Mobile Eye Tracking Using Pre-Trained Computer Vision Models and Human Gaze. **Sensors**, v. 21, n. 12, p. 4143, 2021.

BAZAREVSKY, Valentin *et al.* Blazeface: Sub-millisecond neural face detection on mobile GPUs. **arXiv preprint arXiv:1907.05047**, 2019.

BERGSTROM, Jennifer Romano; SCHALL, Andrew. **Eye Tracking in User Experience Design.** [S.l.]: Elsevier, 2014.

BHUYAN, Manas Kamal. **Computer Vision and Image Processing: Fundamentals and Applications.** [S.l.]: CRC Press, 2019.

BLASCHECK, T. *et al.* State-of-the-Art of Visualization for Eye Tracking Data. In: EUROVIS - STARs. [S.l.]: The Eurographics Association, 2014.

BOARDMAN, Rosy; MCCORMICK, Helen; HENNINGER, Claudia E. Exploring attention on a retailer's homepage: an eye-tracking & qualitative research study. **Behaviour & Information Technology**, v. 42, p. 1–17, 2022.

BRACHTER, Tobias; GERHARDT, Dietmar. Camera Image Based Method of Real Time Gaze Detection and Interaction. **International Journal of Scientific and Research Publications (IJSRP)**, v. 10, p. 788–794, 2020.

BRIDGEMAN, Bruce. Eye Movements. In: ENCYCLOPEDIA of Human Behavior. 2. ed. Amsterdam: Elsevier, 2012. v. 2. P. 160–166.

BUKHARI, Syed; SHAFAIT, Faisal; BREUEL, Thomas. Improved Document Image Segmentation Algorithm using Multiresolution Morphology. In: SPIE. PROCEEDINGS of SPIE - The International Society for Optical Engineering. [S.l.: s.n.], 2011. v. 7874, p. 1–10.

BURCH, Michael. **Eye Tracking and Visual Analytics**. [S.l.]: CRC Press, 2021.

BURGER, Carlos Alberto Coletto; KNOLL, Graziela Frainer. Eye tracking: possibilidades de uso da ferramenta de rastreamento ocular na publicidade. **Fronteiras - estudos midiáticos**, UNISINOS - Universidade do Vale do Rio Dos Sinos, v. 20, n. 3, dez. 2018.

CHEN, Hsuan-Chu; WANG, Chun-Chia; HUNG, Jason C.; HSUEH, Cheng-Yu. Employing Eye Tracking to Study Visual Attention to Live Streaming: A Case Study of Facebook Live. **Sustainability**, MDPI, v. 14, n. 12, p. 7494, 2022.

DALMAIJER, E. S.; MATHÔT, S.; VAN DER STIGCHEL, S. PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. **Behavior Research Methods**, Springer, v. 46, p. 913–921, 2014.

DEVINBARRY. **GazeTracker**. [S.l.: s.n.], 2014.

<https://github.com/devinbarry/GazeTracker>. Last modified: 18 Jun 2014; Accessed: 05 Aug 2023.

DONUK, Kenan; ARI, Ali; HANBAY, Davut. A CNN based real-time eye tracker for web mining applications. **Multimedia Tools and Applications**, v. 81, n. 27, p. 39103–39120, 2022.

DUCHOWSKI, T. Andrew. **Eye Tracking: Methodology Theory and Practice**. [S.l.]: Springer, 2017.

FERNANDEZ, Joe. **Face Mesh: Real-time dense face mesh with MediaPipe**. [S.l.: s.n.], 2023.  
[https://github.com/google/mediapipe/blob/master/docs/solutions/face\\_mesh.md](https://github.com/google/mediapipe/blob/master/docs/solutions/face_mesh.md). Accessed: 05 Aug 2023.

FITZPATRICK, Martin. **The first steps building the browser with PYQt5**. [S.l.: s.n.], 2021. <https://www.pythonguis.com/examples/python-web-browser/>. Accessed: 29 Aug 2023.

GAZEPOINTER. **GazePointer**. [S.l.: s.n.], 2023.

<https://gazerecorder.com/gazepointer/>. Accessed: 05 Aug 2023.

GOBBI, Aline Girardi *et al.* Uso do eye tracking para obtenção de medidas quantitativas em testes de usabilidade: um estudo focado na medida da satisfação. Portuguese.

**Human Factors in Design**, v. 6, n. 11, p. 106–125, 2017. Disponível em:

<https://revistas.udesc.br/index.php/hfd/article/view/10252>.

GOYAL, Manoj *et al.* **ScreenSeg: On-Device Screenshot Layout Analysis**.

abs/2104.08052. [S.l.: s.n.], 2021. <https://arxiv.org/abs/2104.08052>.

HAITH, Marshall M.; BENSON, Janette B. **Encyclopedia of Infant and Early Childhood Development**. [S.l.]: Elsevier, 2008.

HANGARAGI, Shivalila; SINGH, Tripty; N, Neelima. Face Detection and Recognition Using Face Mesh and Deep Neural Network. **Procedia Computer Science**, v. 218, p. 741–749, 2023.

HOLMQVIST, Kenneth; ANDERSSON, Richard. **Eye-tracking: A Comprehensive Guide to Methods, Paradigms and Measures**. [S.l.]: [s.n.], 2017. ISBN 978-1979484893.

HUNG, Jason C.; WANG, Chun-Chia. Exploring the website object layout of responsive web design: results of eye tracking evaluations. **The Journal of Supercomputing**, v. 77, n. 1, p. 343–365, 2021.

KANAN, Christopher; COTTRELL, Garrison. Color-to-Grayscale: Does the Method Matter in Image Recognition? **PloS one**, v. 7, e29740, 2012.

KARTYNNIK, Yury *et al.* Real-time facial surface geometry from monocular video on mobile GPUs. **arXiv preprint arXiv:1907.06724**, 2019.

KIRUTHIKA, S. V.; NANDHINI, R.; KRITHIKA, D.; LAVANYA, R. PCB Defect Detection Using Embedded Image Processing. **International Research Journal of Modernization in Engineering Technology and Science**, v. 03, n. 03, 2021.

LOCKHOFEN, Denise Elfriede Liesa; MULERT, Christoph. Neurochemistry of Visual Attention. **Frontiers in Neuroscience**, v. 15, p. 643597, 2021.

- MENGES, R. *et al.* A Visualization Tool for Eye Tracking Data Analysis in the Web. [S.l.: s.n.], 2020. P. 1–5.
- MIJWIL, Maad *et al.* Overview of Neural Networks. v. 1. [S.l.: s.n.], abr. 2019. P. 2.
- NGUYN, Quyn. Eye Gaze Movement Detection for Controlling On-Screen Cursor in Real Time. [S.l.: s.n.], 2019.  
<https://github.com/Controlling-Cursor-With-Eye-Movement>.
- OPENCV. Color Conversions. [S.l.: s.n.], 2023. [https://docs.opencv.org/3.4/de/d25/imgproc\\_color\\_conversions.html#color\\_convert\\_rgb\\_gray](https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray).
- PADILLA, Rafael; NETTO, Sergio; DA SILVA, Eduardo. A Survey on Performance Metrics for Object-Detection Algorithms. [S.l.: s.n.], 2020.
- PANETTA, Karen *et al.* ISeeColor: Method for Advanced Visual Analytics of Eye Tracking Data. **IEEE Access**, v. 8, p. 52278–52287, 2020.
- PAPOUTSAKI, Alexandra *et al.* WebGazer: Scalable Webcam Eye Tracking Using User Interactions. In: AAAI. PROCEEDINGS of the 25th International Joint Conference on Artificial Intelligence (IJCAI). [S.l.: s.n.], 2016. P. 3839–3845.
- RUBENSTEIN, John; RAKIC, Pasko. Neural Circuit Development and Function in the Healthy and Diseased Brain: Comprehensive Developmental Neuroscience. [S.l.]: Academic Press, 2013. v. 3.
- SALVUCCI, Dario D.; GOLDBERG, Joseph H. Identifying Fixations and Saccades in Eye-Tracking Protocols. In: PROCEEDINGS of the 2000 Symposium on Eye Tracking Research & Applications. Palm Beach Gardens, Florida, USA: Association for Computing Machinery, 2000. (ETRA '00), p. 71–78.
- SCHRÖTER, Iris *et al.* Webcam Eye Tracking for Monitoring Visual Attention in Hypothetical Online Shopping Tasks. **Applied Sciences**, v. 11, n. 19, p. 9281, 2021.
- SERENO, A. B.; BABIN, S. L.; HOOD, A. J.; JETER, C. B. Executive Functions: Eye Movements and Neuropsychiatric Disorders. In: SQUIRE, L. R. (Ed.). **Encyclopedia of Neuroscience**. Oxford: Academic Press, 2009. P. 117–122.

- SPILIOTIS, I. M.; MERTZIOS, B. G. Real-time computation of two-dimensional moments on binary images using image block representation. **IEEE Transactions on Image Processing**, IEEE, v. 7, n. 11, p. 1609–1615, 1998.
- SRISHA, R.; KHAN, A. **Morphological Operations for Image Processing: Understanding and its Applications**. [S.l.: s.n.], 2013.
- STOCKMAN, George; SHAPIRO, Linda G. **Computer vision**. [S.l.]: Prentice Hall PTR, 2001.
- STUART, Samuel. **Eye Tracking: Background, Methods, and Applications**. [S.l.]: Springer Nature, 2022. v. 183.
- SUN, S. *et al.* **Fixation Based Object Recognition in Autism Clinic Setting**. [S.l.: s.n.], 2019.
- SZELISKI, Richard. **Concise Computer Vision: An Introduction into Theory and Algorithms**. [S.l.]: Springer-Verlag, 2014.
- TERRA. **Estudo revela que 252 mil sites são criados todos os dias**. [S.l.: s.n.], ago. 2023. <https://www.terra.com.br/noticias/estudo-revela-que-252-mil-sites-sao-criados-todos-os-dias,af310195a36ec138501642ba92e92eee3xdl3zhr.html>. Acesso em: 23 de setembro de 2023.
- VERA, F. *et al.* Akori: A Tool Based in Eye-Tracking Techniques for Analyzing Web User Behaviour on a Web Site. In: IEEE International Conference on Data Mining Workshops (ICDMW). [S.l.: s.n.], 2017. P. 635–640.
- WANG, Hange *et al.* **ILOVEEYE: EYELINER MAKEUP GUIDANCE SYSTEM WITH EYE SHAPE FEATURES**. [S.l.: s.n.], 2022. <https://arxiv.org/abs/2203.13592>.
- WEBGAZER. **Democratizing Webcam Eye Tracking on the Browser**. [S.l.: s.n.], 2023. <https://webgazer.cs.brown.edu/>. Accessed: 05 Aug 2023.
- WEISBERG, Sanford. **Applied Linear Regression**. 3. ed. New York: John Wiley Sons, 2005. P. 528.