

Thiago Rodrigues de Souza Pessanha

**RASTREAMENTO OCULAR NO PROCESSO
DE AVALIAÇÃO DE USABILIDADE DE
INTERFACES DE PÁGINAS WEB**

Campos dos Goytacazes, RJ

03 de julho de 2023

Thiago Rodrigues de Souza Pessanha

RASTREAMENTO OCULAR NO PROCESSO DE AVALIAÇÃO DE USABILIDADE DE INTERFACES DE PÁGINAS WEB

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro como requisito para a obtenção do título de Bacharel em Ciência da Computação, sob orientação de Prof. Dr. Luis Antonio Rivera Escriba.

Universidade Estadual do Norte Fluminense Darcy Ribeyro – UENF

Centro de Ciência e Tecnologia – CCT

Laboratório de Ciências Matemáticas – LCMAT

Curso de Ciência da Computação

Orientador: Prof. Dr. Luis Antonio Rivera Escriba

Campos dos Goytacazes, RJ

03 de julho de 2023

Thiago Rodrigues de Souza Pessanha
RASTREAMENTO OCULAR NO PROCESSO DE AVALIAÇÃO DE USABILIDADE DE INTERFACES DE PÁGINAS WEB/ Thiago Rodrigues de Souza Pessanha. – Campos dos Goytacazes, RJ, 03 de julho de 2023-
119 p. : il. (algumas color.) ; 30 cm.
Orientador: Prof. Dr. Luis Antonio Rivera Escriba
Monografia (Bacharelado) – UENF-CCT-LCMAT-Ciência da Computação, 03 de julho de 2023.
1. Computação Gráfica. 2. Processamento de Imagens 3. Rastreamento Ocular. 4. Usabilidade. 5. Interação Humano-Computador.
CDU 004.41 : 004.4'2 : 004.5 : 004.6 : 004.92 :

Thiago Rodrigues de Souza Pessanha

RASTREAMENTO OCULAR NO PROCESSO DE AVALIAÇÃO DE USABILIDADE DE INTERFACES DE PÁGINAS WEB

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro como requisito para a obtenção do título de Bacharel em Ciência da Computação, sob orientação de Prof. Dr. Luis Antonio Rivera Escriba.

Trabalho aprovado. Campos dos Goytacazes, RJ, 03 de julho de 2023:

Prof. Dr. Luis Antonio Rivera Escriba
Orientador

Prof. Dr. Luis Mariano del Val Cura
Membro da Banca

Prof. Dr. Joao Luiz de Almeida Filho
Membro da Banca

Campos dos Goytacazes, RJ 03 de julho de 2023

Este trabalho é dedicado também aos amigos, professores e todos os outros que me ajudaram a chegar até aqui, mas principalmente à minha família que sempre me apoiou ao longo de toda a vida .

Agradecimentos

Agradeço primeiramente a Deus por tudo o que tive e tenho para conseguir chegar até aqui.

Agradeço à toda minha família que desde sempre me ajudou e deu forças durante toda a vida. Agradeço aos meus pais, Silvano e Adriana, que desde pequeno sempre me incentivaram a estudar e sempre fizeram o possível para me ajudar, desde pagar boas escolas no ensino fundamental que me deram uma boa base à sentar ao meu lado para me ajudar a fazer minhas tarefas e estudar para as provas. À eles, também agradeço por seus exemplos que pude observar e seguir de amor, carinho, caráter, honestidade e gentileza. Ao meu irmão, Rodrigo, agradeço por me permitir aprender a compartilhar. À nossa cachorra, Layka, agradeço por nos últimos seis anos trazer mais vida e alegria para a casa.

Agradeço aos amigos que vieram antes e durante ao meu tempo na graduação, por também terem me apoiado e também terem sido de grande ajuda para deixar todo o caminho mais leve. Aos que vieram antes, obrigado por aprendermos o que é amizade juntos. Aos que vieram durante, agradeço por serem apoio e alívio em diversos momentos e por toda confiança compartilhada. Aos que foram colegas de curso, agradeço também por todos os momentos de dificuldade, crescimento e alegria que dividimos.

Agradeço a todos os professores que tive desde os três anos, em todas as instituições, que me deram a base e me auxiliaram a continuar aumentando meus conhecimentos nos ensinos infantil, fundamental, médio e superior. Em especial, as professoras Tânia, Soraia e Rosângela do Centro Educacional Virgílio Paula; os professores Leandro Gonçalves e José Leandro Alves da Escola São João Batista; aos numerosos professores do Instituto Federal Fluminense; e aos professores da UENF, principalmente os quatro professores de Ciência da Computação que tornam possível a existência do curso: Luis Rivera, Annabell Del Real, Fermin Tang e Ausberto Castro.

*“ [...] até o dia em que Deus dignar-se a desvelar o futuro para o homem, toda a sabedoria humana estará nessas palavras:
Esperar e ter esperança.”
(Alexandre Dumas)*

Resumo

A usabilidade é um fator considerado há muito tempo para definir o quanto simples e fácil é a utilização de um produto ou equipamento é utilizado. E com a constante crescente do uso e dependência de tecnologias digitais por parte da sociedade, esse conceito também se adapta com o tempo, se fazendo presente na interação humano-computador. Seja durante a compra de algum produto online, ou em páginas informacionais ou de apresentação, instituições se expõem com interfaces digitais na tentativa de conseguir dinheiro e reconhecimento através de novos e antigos usuários. Por conta dessa exposição através das interfaces, a usabilidade é extremamente importante, já que se uma interface tem uma usabilidade insatisfatória para o usuário, este além de ficar com uma imagem ruim sobre a instituição, tende a procurar outras opções. Sendo assim, diversas formas de avaliação de usabilidade são usadas para determinar se existem problemas na interface e quais são eles, para que possam ser corrigidos. Uma dessas formas é com o rastreamento ocular que geralmente é feito com o uso de caros equipamentos específicos para essa tarefa. O sistema desenvolvido nesse trabalho tem como objetivo mostrar uma forma de avaliação de usabilidade que usa o rastreamento ocular em computadores simples que possuem uma webcam, e resulta na exibição de possíveis indícios de problemas com a interface. A imagem do usuário captada pela câmera passa por um processo de detecção da face, olhos e pupila, utilizando técnicas de processamento da imagem, como o Histograma de Gradientes Orientados. A partir destas detecções, a posição dos olhos do usuário durante a execução de tarefas propostas são registradas e usadas para identificar fixações e sacadas, e produzir Mapas de Calor e Scanpaths. Esses dados são usados em comparações a partir de métricas estabelecidas em estudos prévios para encontrar os indícios de problemas, que podem ser visualizados por quem estiver responsável pela avaliação sempre que for desejado.

Palavras-chaves: Usabilidade. Rastreamento Ocular. Interação Humano-Computador. Avaliação. Histograma de Gradientes Orientados.

Abstract

Usability is a factor that has long been considered to define how simple and easy it is to use a product or equipment. And with society's ever-increasing use and dependence on digital technologies, this concept also adapts over time, becoming present in human-computer interaction. Whether during the purchase of a product online, or on informational or presentation pages, institutions expose themselves with digital interfaces in an attempt to get money and recognition through new and old users. Due to this exposure through the interfaces, usability is extremely important, since if an interface has unsatisfactory usability for the user, in addition to having a bad image of the institution, he tends to look for other options. Therefore, several forms of usability evaluation are used to determine if there are problems in the interface and what they are, so that they can be corrected. One such way is with eye tracking which is usually done using expensive equipment specifically for this task. The system developed in this work aims to show a form of usability evaluation that uses eye tracking on simple computers that had a webcam, and results in the display of possible signs of problems with the interface. The user's image captured by the camera undergoes a face, eye and pupil detection process, using image processing techniques, such as the Histogram of Oriented Gradients. From these detections, the position of the user's eyes during the execution of proposed tasks are registered and used to identify fixations and saccades, and to produce Heat Maps and Scanpaths. This data is used in comparisons based on metrics established in previous studies to find signs of problems, which can be viewed by whoever is responsible for the evaluation whenever desired.

Key-words: Usability. Eye Tracking. Human-Computer Interaction. Evaluation. Histogram of Oriented Gradients.

Listas de ilustrações

Figura 1 – Exemplo de mapa de calor.	19
Figura 2 – Exemplo de scanpath.	20
Figura 3 – Árvore de dimensões da usabilidade de interfaces.	25
Figura 4 – Análise cognitiva por rastreamento de olhar: (a) Criança participando do estudo a partir do rastreamento. (b) Registro do foco de olhar de uma criança na interface do jogo.	31
Figura 5 – Fixações (círculos vermelhos) e sacadas (setas amarelas) observados em uma imagem.	32
Figura 6 – Mapas de calor: (a) Áreas de foco em cores quentes em interface de jogo utilizado em experimento; (b) Cores quentes nas áreas de maior foco na interface de portal de notícias.	36
Figura 7 – Mapas de calor: (a) Uso de cores mais quentes em áreas com maior foco; (b) Áreas onde não houve muito foco são cobertas por uma espécie de neblina.	37
Figura 8 – Arquitetura do Sistema	39
Figura 9 – Câmeras: (a) Exemplo de webcam externa; (b) Exemplo de câmera de notebook/laptop	40
Figura 10 – Etapas até a detecção da pupila	41
Figura 11 – Marcação na região da pupila	43
Figura 12 – Olhar e posição do cursor do mouse possuem forte relação	43
Figura 13 – Local dos 25 pontos de calibração em uma interface.	45
Figura 14 – Exemplo de usuário sendo submetido a tarefa com rastreamento ocular.	46
Figura 15 – Classificação dos pontos de olhar.	46
Figura 16 – Exemplo da identificação de sacadas.	47
Figura 17 – Mapa de calor feito a partir do rastreamento ocular de uma home page.	55
Figura 18 – Exemplo de transformação de fixações de uma hipotética Tarefa X para exibição como <i>scanpath</i>	56
Figura 19 – Webcam e OpenCV são as ferramentas necessárias para aquisição da imagem	59
Figura 20 – Sequência de passos para a detecção da face	60
Figura 21 – Sequência usada no algoritmo com HOG para detecção em humanos.	62
Figura 22 – Representação de exemplo de pixel e sua vizinhança com intensidades dos gradientes	63
Figura 23 – Representação de exemplo da aplicação de máscara para cálculo da variação horizontal no pixel central	64

Figura 24 – Representação de exemplo da aplicação de máscara para cálculo da variação vertical no pixel central	64
Figura 25 – Representação de como pode ser vista a relação entre magnitude do gradiente e variações de eixo	65
Figura 26 – Representação dos passos desde a divisão em células(a) com mesma quantidade de pixels (b) até o cálculo dos intervalos de gradientes (c e d) e a construção do histograma(e).	66
Figura 27 – Representação de exemplo da obtenção de histogramas em cada célula.	67
Figura 28 – Representação de exemplo do agrupamento das células em blocos descritores.	67
Figura 29 – Representação de exemplo da sobreposição de blocos no processo de normalização.	68
Figura 30 – 68 pontos da face	70
Figura 31 – Pontos da face que fazem referência aos olhos.	71
Figura 32 – Exemplo da imagem produzida logo após o processo de isolamento do olho	73
Figura 33 – Exemplo da imagem produzida após corte da área desnecessária	73
Figura 34 – Sequência para detecção da pupila	74
Figura 35 – Sequência para detecção da íris.	75
Figura 36 – Comparação das fórmulas matemáticas dos filtros Gaussian Blur e Filtro Bilateral	76
Figura 37 – Etapas do processo de calibragem.	78
Figura 38 – Exemplo de arquivo CSV resultante do processo de calibragem.	79
Figura 39 – Etapas para realização da tarefa.	80
Figura 40 – Página WEB construída para demonstração do sistema	81
Figura 41 – Aparência do navegador Opera GX.	83
Figura 42 – Ilustração de como é considerada a numeração e a área dos alvos da página de demonstração.	83
Figura 43 – Exemplo do armazenamento das coordenadas de olhares do usuário durante a tarefa.	84
Figura 44 – Etapas do processo de identificação de fixações e sacadas.	85
Figura 45 – Exemplo de arquivo de armazenamento das métricas de uma tarefa.	93
Figura 46 – Exemplo do arquivo de armazenamento que recebe o conteúdo de resultsList.	95
Figura 47 – Exemplo de representação de mapa de calor feito pelo sistema.	101
Figura 48 – Exemplo da janela que apresenta os resultados problemáticos das métricas em cada tarefa	105
Figura 49 – Exemplo da janela que apresenta os resultados da avaliação de usabilidade e os indícios que os justificam	106

Figura 50 – Página do Boletim de Ciências Geodésicas hospedada pela SciELO . . .	108
Figura 51 – Página do Boletim de Ciências Geodésicas hospedada pela UFPR	109
Figura 52 – Janela de Problemas de Usabilidade da página da SciELO	110
Figura 53 – Janela de Problemas nas Tarefas da página da SciELO	110
Figura 54 – Janela de Problemas de Usabilidade da página da UFPR	111
Figura 55 – Janela de Problemas nas Tarefas da página da UFPR	111

Lista de tabelas

Tabela 1 – Exemplo de como fica a base de dados da calibragem após o primeiro ponto de calibragem.	44
Tabela 2 – Exemplo de continuação da Tabela 1 após o segundo ponto de calibragem	44
Tabela 3 – Métricas das fixações, significados e possíveis relações com a usabilidade	48
Tabela 4 – Métricas das sacadas, significados e possíveis relações com a usabilidade	49
Tabela 5 – Métrica da razão fixações/sacadas, significado e possíveis relações com a usabilidade	50
Tabela 6 – Métricas dos scanpaths, significados e possíveis relações com a usabilidade	50
Tabela 7 – Quantidade de Indícios de Problemas encontrados no Boletim de Ciências Geodésicas por tarefas	112
Tabela 8 – Quantidade de Indícios de Problemas encontrados no Boletim de Ciências Geodésicas por tarefas	112

Lista de abreviaturas e siglas

300-W	300 Faces In-The-Wild Challenge
ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
BGR	Blue/Green/Red
CHT	Circle Hough Transform
CSS	Cascading Style Sheets
CSV	Comma-separated values
HTML	HyperText Markup Language
IBGE	Instituto Brasileiro de Geografia e Estatística
IEC	International Electrotechnical Commission
IHC	Interação Humano-Computador
ISO	International Organization for Standardization
IV	Infravermelho
JSON	JavaScript Object Notation
HOG	Histogram of oriented gradients
Multi-PIE	Multi Pose, Illumination, Expressions
NBR	Normas Técnicas Brasileiras
RGB	Red/Green/Blue
SciELO	Scientific Electronic Library Online
SVM	Support Vector Machine
UFPR	Universidade Federal do Paraná

Sumário

1	INTRODUÇÃO	17
1.1	Problemática	20
1.2	Hipótese	20
1.3	Objetivo	20
1.4	Justificativa	21
1.5	Método seguido	21
1.6	Resultados esperados	22
2	AVALIAÇÃO DE USABILIDADE EM INTERFACES	23
2.1	Usabilidade	23
2.1.1	Usabilidade de Interfaces	24
2.1.1.1	Eficiência	25
2.1.1.2	Eficácia	26
2.1.1.3	Satisfação	26
2.1.1.4	Aprendizagem	26
2.1.2	Usabilidade e Acessibilidade	27
2.2	Medidas de usabilidade	28
2.3	Elementos de interface e o olhar	29
2.4	Olhar no Rastreamento de elementos	30
2.5	Trabalhos relacionados	33
2.5.1	Avaliação de usabilidade	33
2.5.2	Avaliação de usabilidade por olhar	34
2.5.3	Rastreamento do olho	34
2.5.4	Predição de coordenadas	37
3	MODELAGEM CONCEITUAL	38
3.1	Aquisição	39
3.2	Detecção	40
3.3	Calibragem	43
3.4	Tarefa	45
3.5	Identificação	46
3.6	Verificação	47
3.6.1	Fixações	48
3.6.2	Sacadas	48
3.6.3	Razão Fixações/Sacadas	49
3.6.4	Scanpaths	50

3.7	Verificações das Métricas	51
3.7.1	Verificação da Eficiência: Precisão	51
3.7.2	Verificação da Eficácia: Apresentação e Navegabilidade	52
3.7.3	Verificação da Satisfação: Conteúdo e Atratividade	52
3.7.4	Verificação da Aprendizagem: Simplicidade	53
3.8	Representações	54
3.8.1	Mapa de Calor	54
3.8.2	Scanpath	55
3.9	Resultados	56
4 MODELAGEM FÍSICA		57
4.1	Ambiente de Desenvolvimento e Linguagem	58
4.2	Capturando Imagens	58
4.3	Detecção da Face, Olhos e Pupila	59
4.3.1	Detecção da Face	59
4.3.1.1	Detecção da Região da Face	61
4.3.1.2	Marcação dos Pontos da Face	69
4.3.2	Detecção dos Olhos	71
4.3.2.1	Identificação do Olho	71
4.3.2.2	Isolamento	72
4.3.3	Detecção da Pupila	73
4.3.3.1	Detecção da Íris	74
4.3.3.2	Cálculo do Centroide	77
4.4	Calibragem	77
4.5	Tarefa	79
4.5.1	Página Web	80
4.5.2	Mapeamento dos Alvos	81
4.5.3	Execução e Registro	84
4.6	Identificação	84
4.6.1	Determinar Coordenadas Válidas	85
4.6.2	Identificar Locais dos Olhares	85
4.6.2.1	Vetor de Áreas Vazias	86
4.6.2.2	Vetor de Locais	86
4.6.3	Identificar Focos	87
4.6.4	Identificação das Fixações	88
4.6.4.1	Elementos desconsiderados	88
4.6.4.2	Aquisição das Fixações	89
4.6.5	Identificação das Sacadas	90
4.7	Verificação	90
4.7.1	Cálculo das Métricas	90

4.7.1.1	Momento da Primeira Fixação no Alvo	90
4.7.1.2	Número Total de Fixações	91
4.7.1.3	Densidade Espacial das Fixações	91
4.7.1.4	Número Total de Sacadas	91
4.7.1.5	Tamanho Médio das Sacadas	91
4.7.1.6	Duração Total das Sacadas	92
4.7.1.7	Razão da Duração entre Fixações e Sacadas	92
4.7.1.8	Largura Total do Scanpath	92
4.7.1.9	Duração Total do Scanpath	92
4.7.1.10	Registro	93
4.7.2	Comparação com o Resultado Ideal	93
4.7.3	Identificação de Problemas	94
4.7.4	Análise de Usabilidade	95
4.8	Representações	96
4.8.1	Mapa de Calor	97
4.8.1.1	Atribuição de Pesos	97
4.8.1.2	Definição de Limites	98
4.8.1.3	Aplicação de Cores	99
4.8.2	Scanpath	102
4.8.2.1	Desenhar Sacadas	102
4.8.2.2	Desenhar Fixações	102
4.9	Exibição de Resultados ao Avaliador	103
5	RESULTADOS	107
6	CONCLUSÃO	113
6.1	Trabalhos Futuros	114
	REFERÊNCIAS	115

Capítulo 1: Introdução

Atualmente a humanidade, em geral, vive numa sociedade mediada por tecnologias digitais, onde o uso de interfaces digitais se torna cada vez mais comum em todas suas atividades. É possível ver isso, durante a compra de algum produto online em qualquer das diversas lojas virtuais existentes, ou até mesmo como forma de apresentação de informações e/ou notícias por parte de muitas instituições e organizações ao redor do mundo. Com todas essas formas de exposição, aumentou a importância de se manter um padrão aceitável das interfaces no contexto do tempo.

Levando em conta esse crescimento digital, a área de estudo que começou automaticamente a ganhar importância é a Interação Humano-Computador (IHC), que tem como objetivo básico melhorar as interações feitas entre os usuários e os computadores, tornando estes últimos utilizáveis e receptivos às necessidades do usuário. A IHC tem no geral dois fatores principais a serem considerados: a funcionalidade e a usabilidade (SI-NHA; SHAHI; SHANKAR, 2010). A análise da usabilidade é exatamente uma das formas utilizadas de serem avaliados os padrões das interfaces, buscando indicar se o produto é suficientemente aceitável em seu uso natural para atender a demanda do usuário.

A usabilidade, como menciona Ehmke e Wilson (2007), quando é insatisfatória não é tolerada por usuários que simplesmente acabam optando por outras opções. Esse comportamento dos usuários é o que predomina atualmente com os sistemas interativos, como nos sistemas e-commerce nos quais os fatores de usabilidade são os principais responsáveis por afetar no comportamento dos usuários, e consequentemente influenciar no balanceamento da concorrência entre as empresas (WAHYUNINGRUM; KARTIKO; WARDHANA, 2020). Em alguns casos mais críticos, como nos sistemas da área da saúde, a ausência de usabilidade nos sistemas interativos operacionais pode gerar frustrações e aumento de sensações de esgotamentos nos usuários médicos e enfermeiros; e isso, pode influenciar nos erros humanos representando um maior perigo à saúde dos pacientes (CAYRAYON; HOONAKKER, 2019).

As interfaces passaram a estar intimamente ligadas, principalmente, com a representação da imagem de atitude positiva ou negativa, que pode ser passada de instituições para todos de forma a influenciar nas preferências de uso. São vários fatores que determi-

nam uma aceitabilidade do usuário, entre eles está o fator da distribuição e representação dos elementos da interface.

Desde que foi criada a avaliação de usabilidade, diversos métodos foram criados e utilizados na avaliação das interfaces, entre eles é possível destacar a “avaliação heurística” que trabalha em função de dez fatores de funcionalidade da interface: Visibilidade do *status* do sistema, correspondência entre o sistema e o mundo real, controle e liberdade do usuário, consistência e padrões, prevenção de erros, reconhecimento em vez de recordação, flexibilidade e eficiência de uso, estética e design minimalista, ajuda aos usuários para reconhecer, diagnosticar e prover ajuda e documentação. (MOLICH; NIELSEN, 1990; NIELSEN, 1994).

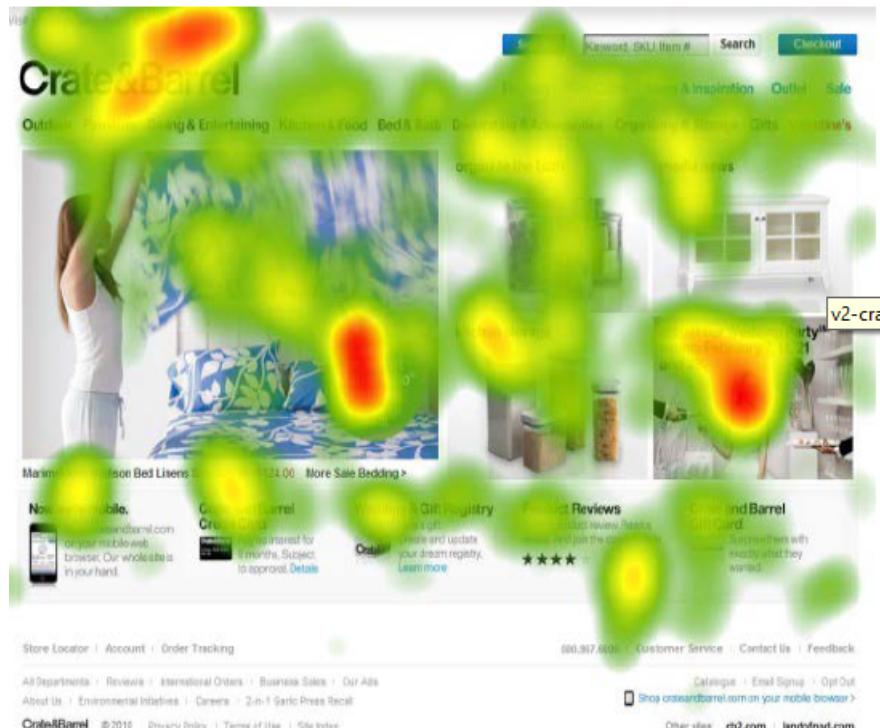
Esse método de avaliação, continua funcionando eficientemente, sendo que com a grande diversidade de interfaces e contextos possíveis, são comumente adicionadas outras regras com propósitos específicos, como observado por Quinones e Rusu (2017) e Zardari et al. (2021). Porém a utilização de grande parte desses métodos, apesar do baixo custo, pode acarretar em problemas graves no teste já que dependem inteiramente de ações humanas, que apesar de serem geralmente conduzidos por especialistas, não estão isentas de muitas falhas, como o esquecimento de algo importante ou a confusão envolvendo informações, e assim, a usabilidade da interface pode continuar ruim e afastando o interesse de usuários.

A forma da distribuição dos elementos da interface, incluindo os atributos visuais e semânticos desses elementos, faz que o usuário tenha uma impressão e conceito de uso com o simples olhar, de forma natural e intuitiva o olhar incidente e repetitiva nos objetos da interface pode passar uma mensagem relacionada com usabilidade. Sobretudo, quando são efetuadas atividades mediante a interface, o foco do olhar do usuário e as transições realizadas definem implicitamente grafos de ações possíveis. Em função desses comportamentos pode se medir alguns parâmetros de usabilidade da interface. Na avaliação heurística podem ser considerados vários fatores confiáveis para uma avaliação por rastreamento ocular. Por exemplo, a visibilidade de instruções para uma clara percepção do usuário e a existência de saídas claras, caso o usuário se veja em alguma situação indesejada. Para isso, um dos fatores utilizados atualmente, e que ganhou bastante popularidade entre os estudiosos, é o de rastreamento ocular. Com isso, a demanda por análises de usabilidade de sites vem crescendo e, também, está se tornando algo mais comum incluir o rastreamento ocular (*eye tracking*) na gama de técnicas utilizadas (EHMK; WILSON, 2007). O rastreamento ocular faz uma análise da movimentação dos olhos em relação a uma interface, podendo usar diversos algoritmos que trazem os resultados esperados (WANG et al., 2019). Sendo possível trazer como informações, a posição dos olhos, tempo em cada posição ou até mesmo velocidade.

Uma das formas de capturar e analisar essas informações é com o uso de mapas de calor e caminhos seguidos (*scanpaths*). Os mapas de calor conseguem transformar a inten-

sidade dos focos de quem está olhando a interface em cores, permitindo que seja possível saber qual área mais chama a atenção dos usuários. Os *scanpaths* também representam graficamente os focos do usuário, marcando os principais pontos e mostrando caminhos sequenciais. Nesse sentido, neste trabalho objetiva-se formular e analisar o rastreamento ocular, com a identificação de mapas de calor (Figura 1) e *scanpaths* (Figura 2) em relação aos objetos da interface para representações mais claras e a avaliação de usabilidade da interface dentro de um contexto usando métricas específicas para ele.

Figura 1 – Exemplo de mapa de calor.



Fonte: Djamasbi (2014)

Figura 2 – Exemplo de scanpath.



Fonte: O autor

1.1 Problemática

As técnicas de avaliação de usabilidade ignoram o comportamento natural da exploração visual pela interface durante a realização das atividades de uso. Parte da informação de agrado da interface para um usuário é capturada pelo olho. Elementos agradáveis são atrativos ao usuário, porém existe apenas um direcionamento rápido do olhar para eles, geralmente esse tipo de olhar passaria despercebido pelo próprio avaliador na hora de expor sua opinião.

1.2 Hipótese

O uso de técnicas de seguimento do olho pode fornecer informações, através do grau de incidência e frequências de direcionamento de olhar nos elementos da interface, do conceito do usuário em relação ao aplicativo que deseja usar.

1.3 Objetivo

Demonstrar o conceito do grau de aceitabilidade natural dos usuários para a avaliação de usabilidade de interfaces de aplicativos interativos. Para tal caso, pretende-se

implementar um modelo com geração de mapas de calor e *scanpaths* e uma lista de indícios de problemas na interface em função de incidências do olhar por técnicas de *eye tracking*.

1.4 Justificativa

A frequência de incidência do olhar pode ter vários significados, um ponto pode ser focado demais por possuir alguma informação importante ao usuário, porém também pode significar que existe algo incomodando naquele lugar. A partir disso, o olhar pode ser usado como informação importante para o rastreamento ocular que permite descobrir a intenção e do contexto da interface qual a situação e os problemas transmitidos por ela.

1.5 Método seguido

Será feito o uso de webcam para aquisição de imagem e assim utilizar a técnica do rastreamento ocular em conjunto com técnicas para a dedução da sequência do olhar, possibilitando a produção de mapas de calor e *scanpaths* a partir do grau de incidência do olhar do avaliador na interface, tendo diversos elementos com diferentes posições, tamanhos, e com os dados obtidos, permitir a sugestão de possíveis problemas de usabilidade presentes na interface.

Para a implementação do sistema será a utilização da linguagem *Python* em conjunto com bibliotecas que permitem trabalhar principalmente com imagens, *machine learning* e armazenamento de dados, sendo as principais: *OpenCV*, *Dlib*, *Pandas* e *NumPy*.

Os principais dados gerados ao longo do sistema, com a captura da imagem e implementação de detecções com a técnica do algoritmo de Histograma de Gradientes Orientados, são armazenados em arquivos *CSV* e então, analisados e comparados de acordo com métricas obtidas em estudos prévios para que haja a identificação de indícios de possíveis problemas de usabilidade. Estes podendo ser exibidos a qualquer momento desejado no formato de listas mais detalhadas.

O sistema será usado em duas páginas, ambas do Boletim de Ciências Geodésicas, porém com design diferente e hospedadas, uma pela SciELO e outra pela UFPR. Estas, já tendo passado pelo processo de avaliação de usabilidade em outro estudo, com a ajuda de usuários voluntários, e terá seus resultados comparados com o do estudo em questão para confirmar sua efetividade.

1.6 Resultados esperados

Uma aplicação para seguir as frequências, transições e incidências de olhar do usuário na interface, mapeamento parcial de mapas de calor e *scanpaths* e exibição de possíveis problemas de usabilidade.

Ao final os responsáveis pela avaliação terão listas com indícios de problemas de usabilidade que permitirão a identificação dos locais que devem ter maior atenção para atualizações com objetivo de melhoramento da usabilidade.

Capítulo 2: Avaliação de Usabilidade em Interfaces

Uma interface pode possuir diversas definições, dependendo da fonte a ser utilizada, uma delas para servir como base é a da ISO (Organização Internacional de Normalização) que enfatiza em ISO/IEC (1993) que uma interface é “uma fronteira compartilhada entre duas unidades funcionais, definida por várias características pertencentes às funções, interconexões físicas, trocas de sinal e outras características, conforme apropriado”.

Considerando o contexto presente no trabalho, é possível dizer que a fronteira abordada neste se faz presente entre o usuário e o sistema para a realização das interações, e é o uso dessa fronteira que está em constante estudo e aprimoramento para que não seja criada uma barreira. Também são colocadas características funcionais relacionadas às interconexões, trocas de sinais, entre outras características apropriadas que as interfaces de interação humano-computador devem possuir para uma interação aceitável, de forma a atender as demandas do usuário humano.

Um dos estudos feitos em relação às interfaces é o da usabilidade, que consiste no uso de medidas que tem o objetivo de garantir que a página analisada possua flexibilidade, que seja intuitiva, de fácil uso para todos os usuários, entre outros. Para isso foram desenvolvidas diversas técnicas que envolvem desde contar apenas com a opinião de avaliadores até a obtenção de dados para análise por movimentos da face ou dos olhos, conhecida na literatura como rastreamento ocular.

2.1 Usabilidade

A usabilidade em geral é definida pela ABNT (2002) como: “medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso”.

Ela é uma medida que, em termos gerais, serve para garantir que um produto consiga ser utilizado da melhor forma tanto por usuários já experientes e/ou habilidosos, quanto por aqueles que são inexperientes e nunca o utilizaram. Tudo isso de forma que

possa proporcionar uma interação que seja feita o máximo possível de forma agradável, intuitiva e que diminua a probabilidade de se cometer erros. Pode-se dizer que deve ser agradável no sentido de que o usuário se sinta bem e não tenha incômodo ao usar o produto. Deve ser intuitiva para que até mesmo os usuários que não possuíram contato anterior saibam como operar o produto devido a organização e padrão que este segue. E deve diminuir a probabilidade de erros considerando que as informações devem ser claras para o usuário de forma a facilitar seu uso da forma desejada, e mesmo que este cometa um equívoco seja possível retomar ao caminho desejado sem grandes dificuldades.

2.1.1 Usabilidade de Interfaces

Quando se trata especificamente da usabilidade de interfaces a definição geral da usabilidade e seus conceitos ainda se fazem valer como um todo. Porém em alguns pontos se faz necessária a adição de mais especificidade para o contexto. Em [Mkpojiogu, Hussain e Hassan \(2018\)](#), ao se avaliar a usabilidade de interfaces, são consideradas quatro dimensões de objetivos e dez subdimensões destas (Figura 3).

No geral, eficiência, eficácia e satisfação continuam como pilares assim como nas interfaces em geral mantendo suas definições de base, que estarão descritas a seguir, além delas é adicionada a aprendizagem, que é basicamente o quanto rápido um sistema consegue ser assimilado para que o usuário use sem problemas, e é importante principalmente para usuários novatos ([MKPOJIOGU; HUSSAIN; HASSAN, 2018](#)). Ainda são citadas as subdimensões, sendo cada uma a representação de um objetivo intrínseco as suas dimensões mães.

Figura 3 – Árvore de dimensões da usabilidade de interfaces.



Fonte: [Mkpojiogu, Hussain e Hassan \(2018\)](#)

2.1.1.1 Eficiência

A eficiência está mais relacionada com o tempo, basicamente é possível dizer que é a medida de quanto tempo foi gasto na realização das tarefas, é uma medida bastante relacionada à produtividade do usuário. As unidades de tempo são comumente utilizadas para expressar a ideia de eficiência, podendo serem considerados os tempos individuais de execução de cada tarefa, caso haja mais de uma, podendo-se utilizar uma soma de todos os tempos ou até mesmo considerar uma média de tempo de execução, podendo ser usada como valor de comparação ([GOLDBERG; WICHANSKY, 2003; WANG et al., 2019; MASLOV; NIKOU, 2020; ZARDARI et al., 2021](#)). [Mkpojiogu, Hussain e Hassan \(2018\)](#) consideram que os principais pontos ao se avaliar a eficiência de uma interface são compatibilidade, tempo de resposta e precisão:

- **Compatibilidade:** Ela se refere a capacidade de uma interface ter menos problemas em diferentes tipos de dispositivos.
- **Tempo de Resposta (ou de Carregamento):** É sobre o tempo de carregamento da aplicação e também o tempo de resposta ao usuário.
- **Precisão:** O desempenho da aplicação em determinado tempo realizando uma tarefa com sucesso e precisão. Além do retorno ao usuário de resultados precisos.

2.1.1.2 Eficácia

A eficácia é uma medida que faz referência ao quanto de uma tarefa um usuário foi capaz de completar, considerando qual era o objetivo desejado ao iniciar a tarefa. Geralmente, para expressar a eficácia de um usuário ao realizar uma tarefa é utilizada a porcentagem (%) da relação do concluído com o esperado, podendo haver variâncias conforme a existência de erros e dúvidas (GOLDBERG; WICHANSKY, 2003; WANG et al., 2019; MASLOV; NIKOU, 2020). Os focos ao se avaliar a eficácia, segundo Mkpojiogu, Hussain e Hassan (2018), são:

- **Apresentação:** Como os botões e ícones estão apresentados. Deve ser intuitivo ao usuário qual a função dos elementos.
- **Navegação:** Deve haver uma lógica com sentido por trás de como a interface é estruturada para que o usuário ache o que o deseja.

2.1.1.3 Satisfação

Para entender a satisfação de um usuário é necessário considerar que esta vem a partir de medidas subjetivas, ou seja, não é possível afirmar com exatidão seus resultados, mas estes ainda assim devem ser levados em conta. Normalmente a satisfação é colocada a prova no final de uma, ou todas as tarefas determinadas, por meio questionários com abordagem direta ou utilizando-se de um sistema de escala (GOLDBERG; WICHANSKY, 2003; GOBBI et al., 2017; WANG et al., 2019). Para gerar satisfação, de acordo com Mkpojiogu, Hussain e Hassan (2018) uma interface normalmente precisa ter:

- **Conteúdo:** O conteúdo deve ter algum valor ao usuário e atender aos seus requisitos.
- **Guia de Uso:** interface deve fornecer auxílio ao usuário, não necessariamente na forma de um manual de uso extenso, podendo estar em caixas de mensagem. Com o uso desse auxílio é possível fazer com que o usuário cometa menos erros e caso ocorram, consiga obter formas de desfaze-lo.
- **Atratividade:** A interface deve ser suficientemente atrativa ao usuário.

2.1.1.4 Aprendizagem

Já quando falado da aprendizagem, que foi brevemente explicada acima, os pilares buscados são:

- **Simplicidade:** A interface deve ser simples ao ponto de permitir que o usuário entenda e realize suas tarefas com o menor esforço possível.

- **Familiaridade:** Devem existir elementos que façam com que o usuário se sinta familiarizado e aumente seu entendimento do que há.

2.1.2 Usabilidade e Acessibilidade

Segundo o Censo do IBGE, feito em 2010, 11% da população brasileira era composta por idosos, isso representava cerca de 21 milhões de pessoas, e com a pandemia do vírus COVID-19, iniciada em 2020, muitos desses idosos ingressaram no mundo tecnológico por curiosidade ou mesmo por necessidade, já que alguns serviços passaram a serem feitos de forma online. Nesse momento se fez cada vez mais necessário um fator que permitisse não só aos idosos, mas a todas as outras pessoas com dificuldades cognitivas ou físicas de conseguirem interagir com essas interfaces de forma fácil: a acessibilidade.

O Decreto Federal nº 5.296/2004 no artigo 8º define acessibilidade como “condição para utilização, com segurança e autonomia, total ou assistida, dos espaços, mobiliários e equipamentos urbanos, das edificações, dos serviços de transporte e dos dispositivos, sistemas e meios de comunicação e informação, por pessoa portadora de deficiência ou com mobilidade reduzida”. Já a *ISO 9241-171* define acessibilidade simplesmente como “a usabilidade de um produto, serviço, ambiente ou instalação por pessoas com a mais ampla gama de capacidades”.

Assim como mencionado anteriormente, a acessibilidade é um conceito que por alguns é considerado paralelo a usabilidade, porém na visão de outros é algo que faz parte desta. Em [Sauer, Sonderegger e Schmutz \(2020\)](#) é dito que usabilidade, acessibilidade e experiência do usuário podem ser vistas como uma trindade, onde pode haver interseções em seus conceitos. Usando a própria definição da *ISO* seria possível ver que há uma relação muito próxima entre usabilidade e acessibilidade, sendo que o foco da acessibilidade é garantir que todo o tipo de usuário, com ou sem qualquer tipo de deficiência consiga utilizar um produto com uma boa usabilidade.

Considerando outra visão, [Soler, Farina e Florian \(2021\)](#) diz que a usabilidade inclui a acessibilidade, porém não se foca tanto em garantir que pessoas com dificuldades tenham maior facilidade de acesso e uso, e sim nos três pilares padrões da eficiência, eficácia e satisfação.

Em todo caso, sendo ou não uma parte da usabilidade, quando se é falado de tecnologias digitais, a acessibilidade é requerida pela usabilidade para garantir que mais pessoas usem o produto (*site* ou aplicativo), de forma em que se faz atuante um dos pilares da usabilidade de interfaces citados a cima, que é a simplicidade, onde o usuário consiga realizar suas tarefas com o menor esforço. Isso permite a inclusão de pessoas com diferentes tipos e graus de deficiência que passam a encontrar menos barreiras para fazer o uso dessas interfaces.

2.2 Medidas de usabilidade

Assim como visto nas definições gerais de usabilidade, suas principais medidas a serem consideradas em um teste de usabilidade de interface são eficiência, eficácia e satisfação. A eficiência e a eficácia podem ser medidas de forma mais simples, já que representam quantitativamente, respectivamente, o tempo gasto para realizar uma tarefa e o quanto dela foi concluída. Já a satisfação do usuário, normalmente se dá por meio da confiança do julgamento do usuário através de entrevistas ou questionários.

De acordo com o trabalho de [Falcão e Soares \(2013\)](#) as medidas de usabilidade podem ser classificadas nos três tipos mostrados abaixo:

- **Medidas operacionais de usabilidade de caráter objetivo:** são as medidas que se referem à eficiência e à eficácia, ou seja, o desempenho e o tempo levado na tarefa.
- **Medidas objetivas de usabilidade:** Essas medidas consideram o nível de conhecimento dos usuários. Basicamente, como um usuário experiente performa, a capacidade de como um iniciante aprende e também como se sai um usuário casual tendo que reaprender a usar.
- **Medidas subjetivas:** São as que se baseiam na percepção do usuário e em sua experiência de uso.

A satisfação, claramente se enquadra nas medidas subjetivas já que seus resultados vão depender do que o usuário sentir, perceber e lembrar ao utilizar a interface através de diversos métodos, como os já citados acima, questionários. O principal problema nisso é que esses resultados, já que são subjetivos, acabam precisando ser interpretados possibilitando a ocorrência de erros a partir dessa interpretação ([GOBBI et al., 2017](#)).

É nessa parte onde o rastreamento ocular se faz mais útil, já que mesmo que o usuário não se lembre ou não perceba, o lugar onde seus olhos passaram ou focaram será registrado por diferentes medidas, que possibilitam obter resultados quantitativos que podem ser analisados com maior exatidão. As medidas obtidas pelo rastreamento ocular no geral vêm principalmente de dois movimentos: fixações e sacadas.

As fixações conseguem dar retorno de quais elementos nas interfaces atraírem mais o foco do olhar do usuário e quanto tempo elas demoraram, resultando na resposta de se a interface faz com que o usuário foque no local que deveria ou não. As sacadas, sendo os movimentos rápidos que acontecem entre uma fixação e outra trazem os caminhos feitos pelos olhos e resultam por trazer a hierarquia entre os elementos da interface.

2.3 Elementos de interface e o olhar

O olhar pode proporcionar uma ideia do quanto atrativo e importante os elementos de uma interface podem ser. Assim, existem algumas propriedades que podem determinar um maior, ou menor, estímulo visual presente em determinado objeto ou área (FORSTER, 2017). Assim, o olhar é uma importante fonte de captação de parâmetros de usabilidade. A partir das informações provenientes dele, o corpo interpreta como sensações, e trazem uma ideia do nível de agradabilidade de uma interface.

Além disso, durante o olhar é possível dizer que o corpo passa por processos cognitivos, que são basicamente nesse caso a atenção, o uso da memória, processamento e resolução de possíveis problemas (ABRAHÃO; SILVINO; SARMET, 2005). Haesner et al. (2017) relata que em estudos feitos anteriormente, pessoas com Comprometimento Cognitivo Leve (CCL) apresentavam simultaneamente déficits de memória e também dos processamentos que tinham relação à atenção visual, de modo que ao serem submetidos a um teste de usabilidade com rastreamento ocular, levavam mais tempo e necessitavam de muito mais fixações para realizarem as tarefas.

Existem diversas operações cognitivas executadas por um ser humano, em adição existem muitas formas de se classificar essas operações. Uma dessas formas é feita de forma hierárquica e baseada na complexidade das operações (HARVEY, 2019). Ela é conhecida como *top-down versus bottom-up*. As propriedades *bottom-up*, que são mais concretas e mais ligadas aos processos da cognição mais básicos, sendo elas a cor, o contraste, o brilho, a simetria, entre outras (FORSTER, 2017). As propriedades *top-down*, que também podem gerar estímulo aos olhos, mas diferentemente das *bottom-up* estes já dependem da interpretação do sujeito, do seu passado e do que ele conhece, podendo serem desencadeados até mesmo por uma sensação de desconforto, de que falta algo que era esperado e não se encontra presente. Essa forma de classificação funciona de forma a assumir que as operações mais básicas são menos complexas, enquanto o raciocínio e a resolução de problemas, que são também conhecidos como funcionamento executivo, são mais complexos. As operações de funcionamento executivo sempre envolvem em seu funcionamento as operações menos complexas, enquanto o contrário geralmente não ocorre e as operações básicas raramente usam algum processamento de nível mais alto (HARVEY, 2019).

Em outras palavras as operações cognitivas *bottom-up* são aquelas que geralmente se iniciam a partir dos processos sensoriais mais básicos e futuramente serão entendidos e devolverão alguma resposta. Já as operações *top-down*, são aquelas que decidirão ações a serem tomadas serão decididas a partir de raciocínio e interpretação, onde seu conhecimento será levado em conta.

Em uma interface podem existir diversos elementos, que serão os responsáveis por

desencadear os processos cognitivos. Desde imagens e menus a outros elementos muito menores e que podem parecer que passam despercebidos aos olhos humanos, como um *input* do tipo rádio com seus itens (VANDERDONCKT, 2010). Basicamente qualquer coisa que possa chamar uma mínima atenção do olhar é um elemento a se considerar.

2.4 Olhar no Rastreamento de elementos

De acordo com Forster (2017), o rastreamento ocular no geral pode ser visto como o monitoramento da posição relativa dos olhos durante tarefas que envolvem a visualização de estímulos visuais, sejam eles imagens, objetos ou textos escritos.

Existem diversas técnicas para realização desse tipo de rastreamento, desde as mais invasivas, por exemplo, o implante de uma lente que adere a região da esclera do olho e trabalha de forma extremamente precisa na detecção da mudança de posicionamento dos olhos. Existem, também, as mais comuns que se utilizam da captura da imagem dos olhos por uma câmera e então são calculadas as mudanças de direcionamento do olhar em determinados intervalos. Existem diversos tipos de movimentos dos olhos assim como pode ser observado por Goldberg e Wichansky (2003) e Carter e Luke (2020), sendo alguns destes os movimentos de sacadas, perseguição suave, compensatórios, vergência (convergência ou divergência em relação ao foco), miniatura e nistagmo (movimento involuntário do olho).

Esse tipo de análise está relacionado não só com a área tecnológica em si, como também nos estudos da área psicolinguística, onde inclusive é possível dizer ser esta sua área de origem, onde se torna muito importante a possibilidade de entender os indivíduos a partir de estímulos visuais. Por exemplo, em Sánchez-Morales, Durand-Rivera e Martínez-González (2020) é utilizado o rastreamento ocular, entre outros métodos para encontrar problemas cognitivos em crianças por meio da utilização de um jogo com imersão (Figura 4). Nesse experimento o principal fator percebido para estudo foi o tempo contínuo de atenção dos participantes enquanto interagiam com o jogo. Sendo encontrado, em uma das crianças, um déficit na lateralidade, que é quando se tem maior preferência na utilização de um dos lados do corpo e pode causar transtornos no aprendizado.

Figura 4 – Análise cognitiva por rastreamento de olhar: (a) Criança participando do estudo a partir do rastreamento. (b) Registro do foco de olhar de uma criança na interface do jogo.



(a)



(b)

Fonte: Sánchez-Morales, Durand-Rivera e Martínez-González (2020)

Os diferentes tipos de movimento de olhar considerados por [Goldberg e Wichansky \(2003\)](#) e [Carter e Luke \(2020\)](#) são:

- **Movimento de sacada:** São movimentos dos olhos que possuem uma aceleração muito alta, podendo ser considerados movimentos bruscos. São movimentos que acontecem nos dois olhos ao mesmo tempo, tem uma variação na angulação dos olhos de 2 a 10 graus, acontecem em um tempo inferior a 100ms por movimento e tem uma velocidade de rotação de aproximadamente 500 a 900 graus por segundo. A duração e distância percorrida pelas sacadas variam de acordo com a tarefa.
- **Movimento de perseguição suave:** Esse tipo de movimento tem esse nome devido a sua baixíssima velocidade de rotação, que é de 1 a 30 graus por segundo. Ele ocorre quando há uma tentativa de focar um alvo em movimento, podendo sobrepor o movimento de sacada de forma totalmente involuntária.

- **Movimento compensatório:** É o movimento dos olhos que acontece, como o próprio nome diz, para compensar uma movimentação da cabeça ou do tronco, seja de forma voluntária ou não, com o objetivo de estabilizar a imagem enxergada.
- **Movimento de convergência:** Esse movimento pode ser explicado como sendo basicamente, a movimentação dos olhos em direções opostas devido à dificuldade de se focar em dois objetos que estão se aproximando ou se afastando.
- **Movimento miniatura:** Este tipo de movimento é caracterizado simplesmente por sua baixa mudança angular de posição, sendo esta menor que 1 grau. Esse tipo de movimento pode ser observado mais comumente na forma de pequenos tremores.
- **Movimento do nistagmo:** Essa movimentação involuntária consiste num padrão onde os olhos movem-se rapidamente de um lado para o outro sem que haja uma direção sempre seguida. Pode ser causado por diversos fatores, entre eles movimentos rápidos da cabeça, estímulos opto cinéticos e até mesmo a fadiga.

Além das movimentações dos olhos, também são normalmente consideradas as fixações, que são períodos de tempo onde os olhos estão focados em algum alvo específico. As fixações podem ter durações variadas dependendo de diversos fatores, como o estímulo visual, a atenção do indivíduo e a complexidade da tarefa. Elas geralmente duram de 180 a 330 milissegundos. Por exemplo, na Figura 5 ocorrem várias sequências de fixações, indicadas por círculos vermelhos, e sacadas, indicadas por setas amarelas. Normalmente, fixações e sacadas são os maiores responsáveis pela obtenção de resultados ao se trabalhar com o rastreamento ocular (CARTER; LUKE, 2020).

Figura 5 – Fixações (círculos vermelhos) e sacadas (setas amarelas) observados em uma imagem.



Fonte: Carter e Luke (2020)

2.5 Trabalhos relacionados

Nos últimos 30 anos, vários estudos e pesquisas passaram a ser feitos na área da usabilidade para que esta se tornasse mais simples e entendível. Porém junto com os avanços de teorias e do entendimento do ser humano num todo, além das mudanças de contexto no mundo, se tornou necessária uma forma de julgar se a usabilidade estava sendo aplicada da melhor forma nas mais diversas interfaces.

Para isso pesquisadores começaram a se empenhar a estudar a relação da usabilidade com os mais diversos fatores e áreas, de forma a possibilitar uma avaliação mais completa e precisa com seus trabalhos. Isto inclui desde o entendimento mais básico da usabilidade e do rastreamento ocular, assim como o desenvolvimento e uso dos mais diversos algoritmos.

2.5.1 Avaliação de usabilidade

A avaliação de usabilidade de uma interface pode ser feita de diferentes formas. Podem ser feitas utilizando-se de métodos tanto empíricos, quanto por não empíricos. Tratando-se das avaliações não empíricas é possível dizer que estas apresentam uma forma comum entre elas, que se trata do fato de que são feitas por especialistas, geralmente antes de que o produto seja lançado no mercado, tendo o objetivo de descobrir problemas na interface dele, e assim adequá-lo aos seus critérios (CATECATI et al., 2017).

É possível dizer que o método não empírico mais difundido e utilizado é a Avaliação Heurística de Nielsen e Molich, que consiste na utilização de avaliadores, sendo de 3 a 5 a quantidade recomendada, que farão uma análise das interfaces seguindo principalmente 10 regras pré-determinadas pelos autores deste método, já citadas anteriormente, e então exibirão suas opiniões em um relatório após a avaliação.

Nos métodos empíricos, a avaliação da interface é feita a partir da interação direta do usuário com o produto, de forma a ser possível a análise dos aspectos de usabilidade presentes na interface. Geralmente são planejadas tarefas que serão executadas pelos usuários do experimento, e a partir de suas execuções são avaliadas eficiência, eficácia e satisfação do usuário em relação à interface, a partir de um determinado conjunto de métricas (CATECATI et al., 2017).

Existem diversas formas de se realizar uma avaliação a partir de um método empírico, podendo se utilizar de diversos equipamentos, por exemplo, câmeras para a captação de imagens da face e dos olhos que poderão ser analisadas, microfones para obter a voz e sons emitidos pelo usuário, e até mesmo cronômetros para marcar o tempo de execução das tarefas (CATECATI et al., 2017).

2.5.2 Avaliação de usabilidade por olhar

O rastreamento ocular não existe para encontrar todos os problemas, porém, permite o melhoramento de alguns fatores presentes nas avaliações de usabilidade mais comuns, mais especificamente aqueles que possuem grande dependência da percepção do usuário. Sabendo disso diversos trabalhos foram desenvolvidos na área.

Nesta categoria, encontram-se dois trabalhos, [Goldberg e Wichansky \(2003\)](#) e [Ehmke e Wilson \(2007\)](#), como alguns dos precursores de avaliação de usabilidade por olhar baseados em técnicas de *eye tracking* (rastreamento ocular). O primeiro dá uma visão geral do que é o trabalho de realizar uma avaliação de usabilidade a partir de dados captados na movimentação dos olhos desde sua forma mais básica. Isto é, nele é explicado como são os movimentos dos olhos que podem ser usados num rastreamento ocular, e que geralmente apenas as sacadas e fixações entre elas são consideradas por diversos fatores presentes nos outros possíveis movimentos.

O trabalho de Ehmke e Wilson tratou de sintetizar os dados de diferentes relações entre métricas do rastreamento ocular e problemas de usabilidade que se encontravam presentes em vários outros trabalhos dos temas, que apresentavam suas pesquisas focando em medidas diferentes. Por exemplo, alguns focaram-se na duração das fixações, outros no número destas, e ao final muitas das vezes o problema encontrado era basicamente o mesmo ou muito parecido. Mostrando assim que mesmo que feitos com focos diferentes às avaliações de usabilidade por rastreamento ocular, possuem um padrão na sua relação de resultados, mesmo não sendo totalmente concretizado.

Mais recentemente, e mais especificamente falando sobre rastreamento ocular, é possível citar os trabalhos de [Malan, Eloff e Bruin \(2018\)](#) e também [Joseph e Murugesh \(2020\)](#). Joseph e Murugesh fazem um estudo sobre possíveis métricas e indicadores a serem levados em conta durante uma avaliação na área da interação humano-computador. É comparável ao trabalho de Ehmke e Wilson, feito 13 anos antes, trazendo algumas métricas diferentes e comparações com as que ambos os trabalhos compartilham. Já [Malan, Eloff e Bruin \(2018\)](#) usam uma possível forma de quantificar algumas das métricas encontradas nos trabalhos anteriores, mais especificamente se tratando de fixações e sacadas, além de um caminho mais específico do que fazer de forma conjunta com esses valores que foram obtidos a partir de diferentes características.

2.5.3 Rastreamento do olho

Existem diversas técnicas quando se trata do rastreamento ocular. Quando falamos da sua relação com a avaliação de usabilidade estes basicamente se focam no mapeamento da pupila como ponto de partida. Para isso foram desenvolvidos diferentes algoritmos.

Como exemplos o *CHT* e o *Starburst* mostrados em [Pasarica et al. \(2015\)](#). Além deles, como descrito em [Boyko, Basystiuk e Shakhovska \(2018\)](#) para o rastreamento tanto da face quanto dos olhos, existe também o uso do Histograma de Gradientes Orientados (em inglês, *Histogram of Oriented Gradients*, ou abreviadamente, HOG, em conjunto com um preditor de marcas do rosto (Mais detalhes na seção 4.3.1.1).

O trabalho de [Boyko, Basystiuk e Shakhovska \(2018\)](#) mostra que o algoritmo HOG, que começou a ser usado na detecção de características humanas a partir do trabalho de [Dalal e Triggs \(2005\)](#), continua sendo popular nos dias de hoje e pode ser usado no reconhecimento facial. Esse algoritmo como o nome já sugere, trabalha com o conceito de cálculo de gradientes, onde estes se fazem fundamentais para encontrar onde estão as bordas, no caso, os contornos que formam uma face. A partir da detecção da face se faz necessário encontrar 68 pontos específicos que funcionam como um padrão na face, e podem ser úteis para isolar determinadas partes (Mais detalhes na seção 4.3.1.2).

Muitos dos estudos realizados acabam por não utilizar a implementação de algoritmos e optam pela utilização de equipamentos projetados propriamente para que o rastreamento ocular seja feito, além da exibição de mapas de calor. Sendo necessária apenas uma análise dos resultados exibidos para a tirada de conclusões. Exemplos disso são o estudo de [Zain et al. \(2013\)](#), que trabalhou com a análise de jogos educativos utilizando-se do rastreamento ocular para qualificar as interfaces e o posicionamento de elementos (Figura 6a), e também [Weichbroth, Redlarski e Garnik \(2016\)](#) que tem basicamente o mesmo intuito, porém com portais de notícias (Figura 6b). Entretanto é necessário dizer que este tipo de equipamento citado é extremamente caro, tornando assim inviável sua utilização em muitos possíveis estudos.

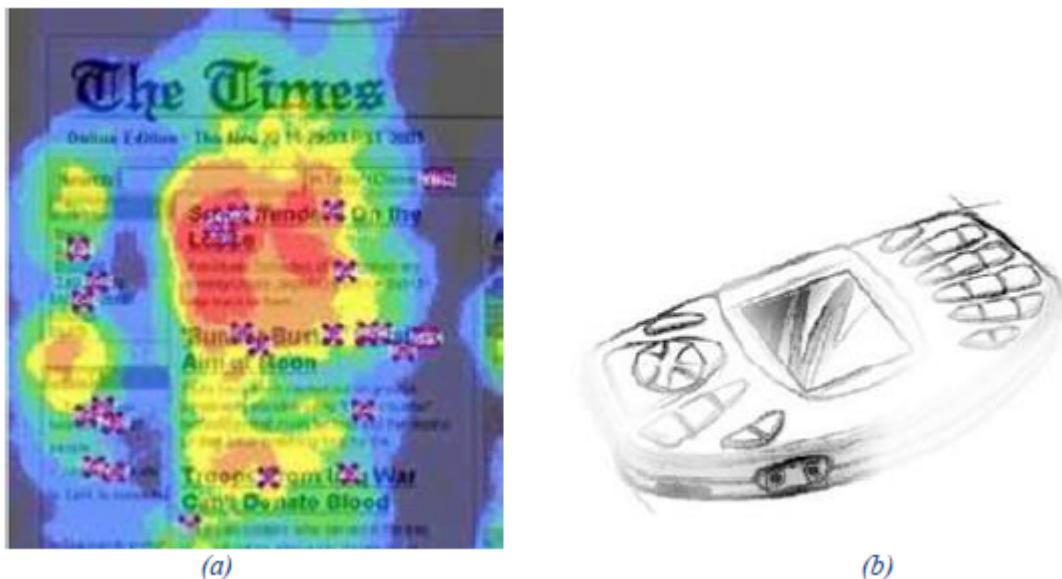
Figura 6 – Mapas de calor: (a) Áreas de foco em cores quentes em interface de jogo utilizado em experimento; (b) Cores quentes nas áreas de maior foco na interface de portal de notícias.



Fonte: (a): [Zain et al. \(2013\)](#) / (b): [Weichbroth, Redlarski e Garnik \(2016\)](#)

Além disso, como mostrado por [Duchowski et al. \(2012\)](#), os dados conseguidos com a avaliação feita por meio do rastreamento ocular pode ter como sua forma de representação os mapas de calor, isto é, a incidência das fixações coletadas durante a avaliação, forma áreas com coloração geralmente de cor mais quente, onde é possível ver quais são os pontos mais focados em determinada interface e assim considerar se os resultados encontrados foram correspondentes com os esperados de forma antecipada, do modo como pode ser visto na Figura 7a. Ou mesmo, como mostrado por [Spakov e Miniotas \(2007\)](#), que introduzem mostram uma versão modificada do mapa de calor, onde áreas em que não é encontrado grande foco dos olhos parecem estar sob sombras ou névoa (Figura 7b). Fato é que, assim como foi dito por [Djamasbi \(2014\)](#), os mapas de calor oferecem uma representação muito mais amigável do rastreamento ocular que outras formas de representação existentes

Figura 7 – Mapas de calor: (a) Uso de cores mais quentes em áreas com maior foco; (b) Áreas onde não houve muito foco são cobertas por uma espécie de neblina.



Fonte:[Spakov e Miniotas \(2007\)](#)

2.5.4 Predição de coordenadas

Após conseguir estabelecer uma relação inicial entre as coordenadas dos olhos e as da tela a partir da calibração, é preciso que haja uma forma de calcular a partir das próximas coordenadas dos olhos, qual será o lugar correspondente na interface e é aí que entra a utilização das regressões, como pode ser visto no trabalho de [Papoutsaki et al. \(2016\)](#). Esse trabalho, que se utiliza das regressões, funciona da seguinte forma: considerando a existência da relação entre o lugar interage com uma interface com o lugar em que ele olha, o usuário é solicitado a clicar em pontos mostrados na interface para calibragem do sistema. A partir desta calibragem é que se faz uso das regressões. Estas funcionam de forma a calcular a partir dos dados obtidos na calibragem, o local na interface referente ao local que o usuário está olhando, mostrando-se um método que mesmo com o uso de câmeras comuns, não equipadas com IV, é possível obter um resultado de bom nível.

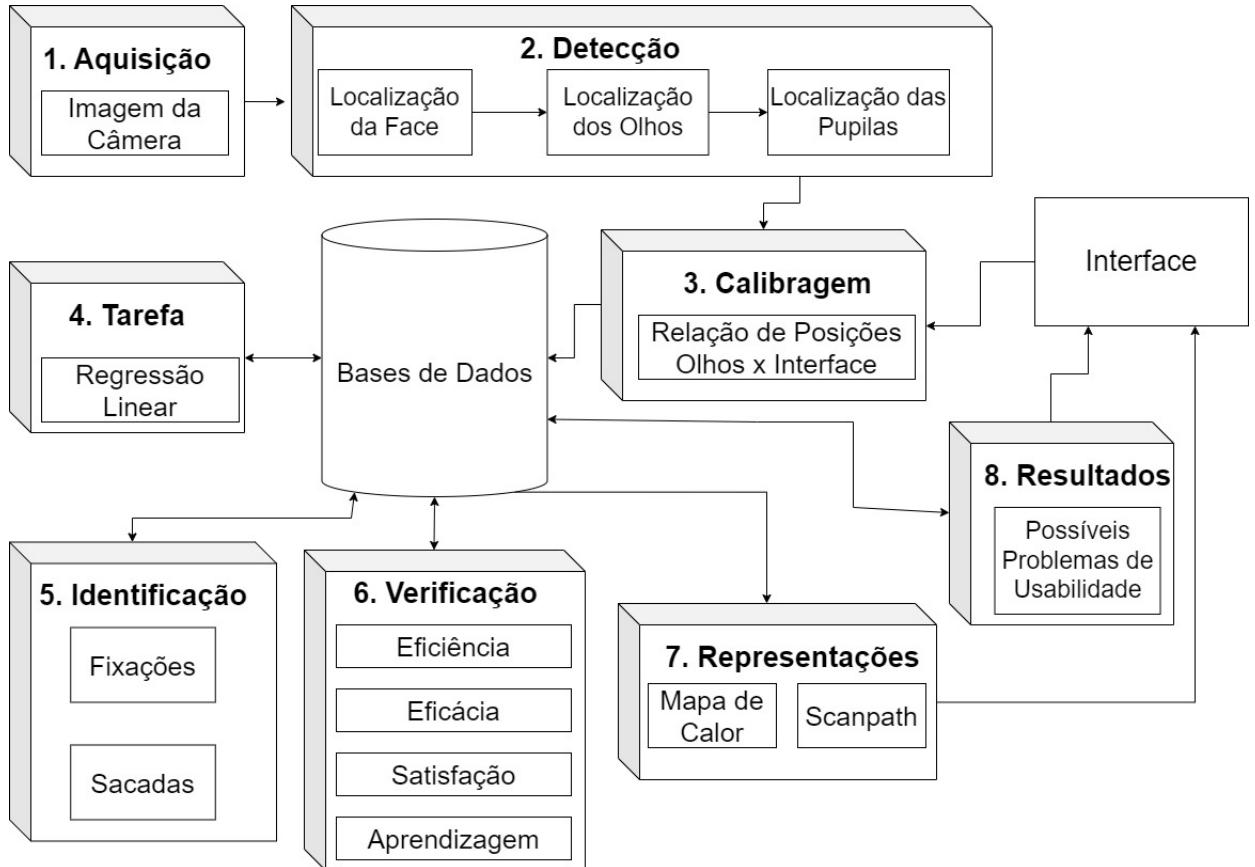
Capítulo 3: Modelagem Conceitual

Neste trabalho pretende-se como objetivo construir uma forma de análise de usabilidade fazendo uso do rastreamento ocular como uma ferramenta para alcançá-lo. A partir do rastreamento ocular feito sobre uma imagem em tempo real obtida de uma câmera se faz possível a percepção de diversos movimentos. Esses movimentos são objetos de pesquisas a várias décadas e observou-se uma possibilidade de relação bem próxima entre estes e fatores que influenciam na usabilidade de interfaces (BALL; RICHARDSON, 2022). Baseando-se nisso, esse trabalho, a partir da tabulação dos movimentos do usuário, fará uso das relações encontradas por todos esses anos para sintetizar possíveis problemas de usabilidade presentes em determinada interface e apresenta-los ao final junto a outras formas gráficas.

O sistema construído para esse fim é composto basicamente de oito blocos funcionais, como pode ser visto na Figura 6: aquisição, detecção, calibragem, tarefa, identificação, representações, verificação e resultados.

Primeiramente, na aquisição é obtida a imagem a partir da câmera utilizada, então com ela pode ser feita a localização de onde estão os olhos do usuário na imagem e em seguida mais especificamente, da pupila. Com esses objetos detectados, se torna possível fazer a calibragem do reconhecimento dos olhos, onde a relação entre o cursor e a posição dos olhos se faz muito importante. Após a calibragem já se torna possível saber para onde o usuário está olhando na tela. Assim, a partir de determinadas tarefas propostas são obtidos dados que permitem a identificação de diferentes tipos de movimentos feitos pelos olhos (mais especificamente, fixações e sacadas) e então, nos dados de cada movimento e da relação entre alguns deles, podem ser verificadas diversas métricas responsáveis por identificar possíveis problemas de usabilidade, seguidamente com os resultados obtidos, se torna possível a produção de mapas de calor, caminhos de seguimento (*scanpaths*) e outros resultados quantitativos importantes para a avaliação da usabilidade.

Figura 8 – Arquitetura do Sistema



Fonte: O autor

3.1 Aquisição

O bloco de aquisição é bem simples e consiste basicamente em obter as imagens vindas de uma câmera conectada ao processador, para que seja possível fazer os reconhecimentos faciais necessários. Dependendo do ambiente pode ser necessária também a utilização de uma fonte de iluminação adequada.

Figura 9 – Câmeras: (a) Exemplo de webcam externa; (b) Exemplo de câmera de notebook/laptop

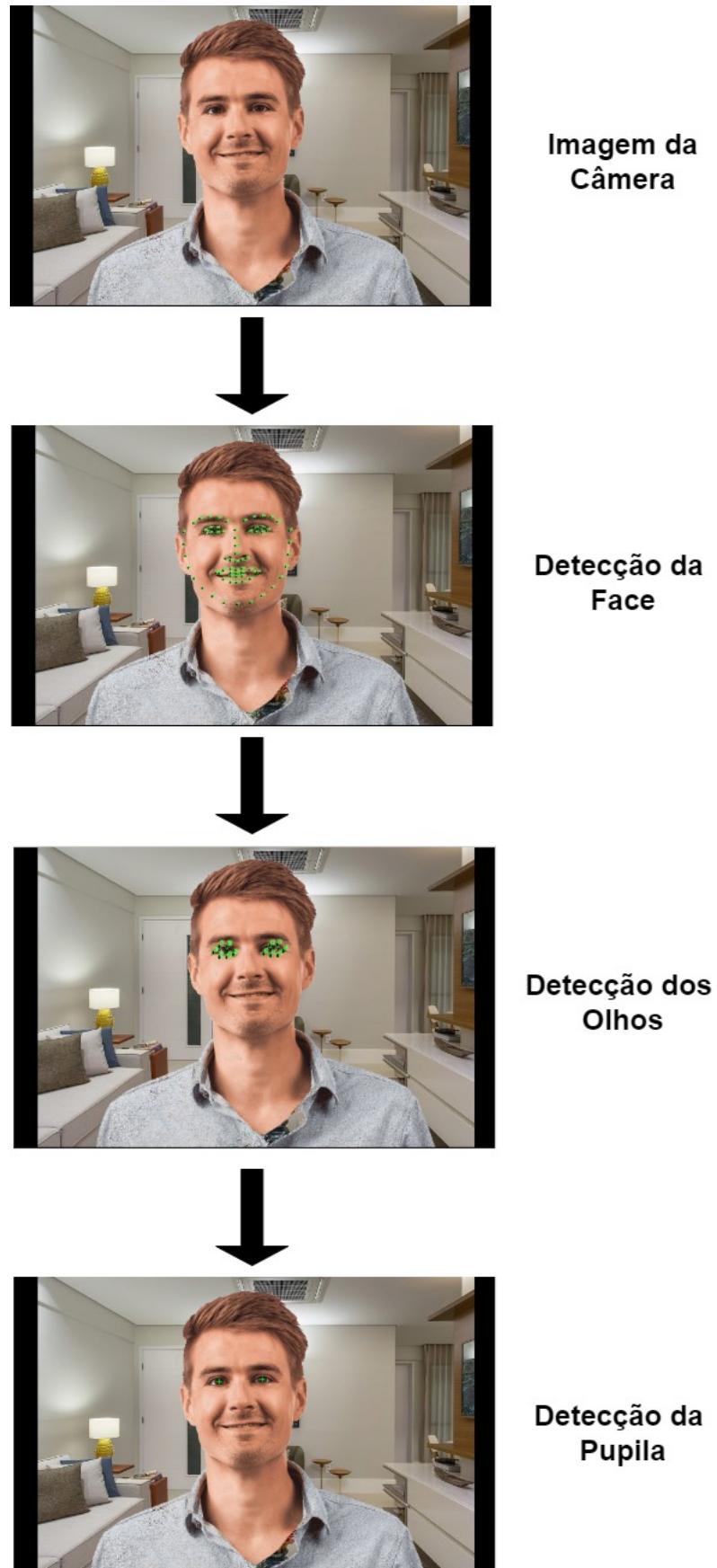


Fonte: O autor

3.2 Detecção

Na detecção é onde ocorre a parte do reconhecimento facial a partir da imagem sendo capturada em tempo real pela câmera, mais especificamente o objetivo desta é o reconhecimento da posição dos olhos na face, e então da pupila nos olhos, como pode ser visto abaixo no diagrama da Figura 10. Para isso, neste trabalho são utilizadas bibliotecas disponíveis que realizam a detecção da face e dos olhos a partir de gradientes e pontos da face (principalmente *OpenCV* e *Dlib*).

Figura 10 – Etapas até a detecção da pupila



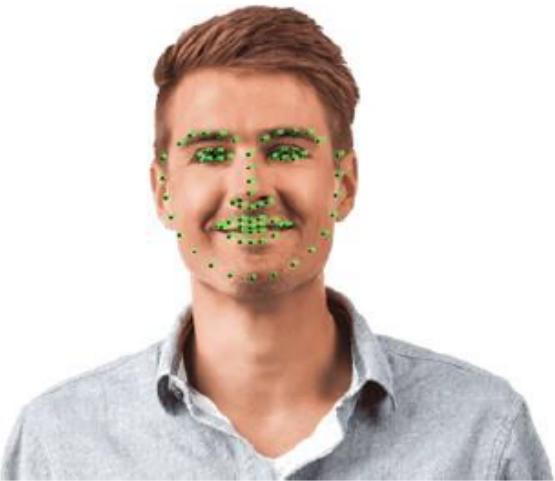
Fonte: O autor

Na detecção da face a imagem que está sendo capturada pela câmera é submetida a uma sequência que utiliza o algoritmo HOG a partir de uma biblioteca já existente (*Dlib*), onde a face do usuário será reconhecida e a partir de seus contornos, são marcados com o auxílio de um modelo treinado, de forma sem que o usuário possa ver acontecendo, pontos na face (Figura 10) que servem como guias para encontrar partes mais específicas.

A partir dos pontos marcados, para encontrar a posição onde estão os olhos do usuário é preciso apenas considerar como foco uma certa sequência de 12 desses pontos, onde cada olho está marcado por 6. Esses pontos possuem numeração padrão já que o modelo é sempre o mesmo. No caso, o olho esquerdo está compreendido entre os pontos 36, 37, 38, 39, 40 e 41 e o olho direito está entre 42, 43, 44, 45, 46 e 47.

Para encontrar a pupila são considerados três fatores: a cor da íris em relação a esclera, a íris é circular, e a pupila geralmente está no centro da íris. A cor da íris na imagem tem grande diferença em relação a esclera dos olhos, assim é feita a binarização da imagem, permitindo que a região da íris seja detectada, a partir dela, a posição da pupila é assumida em ser o centroide da região (Figura 11).

Parte da Figura 10



Parte da Figura 10



Figura 11 – Marcação na região da pupila



Fonte: O autor

3.3 Calibragem

Liebling e Dumais (2014) mostram uma grande relação entre o olhar e a posição do cursor em uma interface, fazendo com que este possa ser de grande ajuda no caso do rastreamento ocular.

Figura 12 – Olhar e posição do cursor do mouse possuem forte relação



Fonte: O autor

Baseando-se em Papoutsaki et al. (2016), pode-se considerar que para fazer a calibragem, um local de interação na tela corresponde à onde está direcionado o olhar. Assim durante esse processo, são exibidos, um de cada vez, um determinado número de pontos na tela espaçados uniformemente. Esses pontos são representados pelo ponteiro

do mouse que se posiciona exatamente acima da coordenada do ponto. O usuário olha e confirma a ação para cada um deles sequencialmente.

Os dados obtidos na calibragem são enviados a uma base de dados para serem utilizados na(s) tarefa(s) pela frente. Essa base de dados é basicamente uma tabela e contém as coordenadas x e y de cada um dos olhos (x_{esquerdo} , x_{direito} , y_{esquerdo} , y_{direito}) e, também, a posição dos pontos de calibragem (x_{mouse} , y_{mouse}). Cada linha da tabela contém a captação dos olhos em cada ponto.

Considerando, por exemplo, que o primeiro ponto de calibragem de um usuário seja o (0,0), e enquanto ele está olhando para esse ponto as coordenadas do olho esquerdo dele sejam (214, 310) e as do olho direito sejam (310, 308). A primeira linha da base de dados seria preenchida e ficaria com a lógica da Tabela 1 a seguir.

Tabela 1 – Exemplo de como fica a base de dados da calibragem após o primeiro ponto de calibragem.

x_{esquerdo}	x_{direito}	y_{esquerdo}	y_{direito}	x_{mouse}	y_{mouse}
214	214	310	310	-	-
-	-	-	-	-	-
-	-	-	-	-	-

É possível determinar a quantidade de pontos de calibragem desejada antes do início desta. Essa quantidade será relacionada com a resolução da tela para determinar onde serão as coordenadas de todos os pontos. Supondo que o usuário do exemplo da tabela acima esteja utilizando uma tela com resolução 1600x900 e foi definido que existirão 25 pontos de calibragem (Figura 13), o próximo ponto estaria na coordenada (400,0). Sendo as coordenadas dos olhos esquerdo e direito nesse ponto, respectivamente, (206,311) e (305,308), a base de dados seria atualizada e ficaria como na Tabela 2.

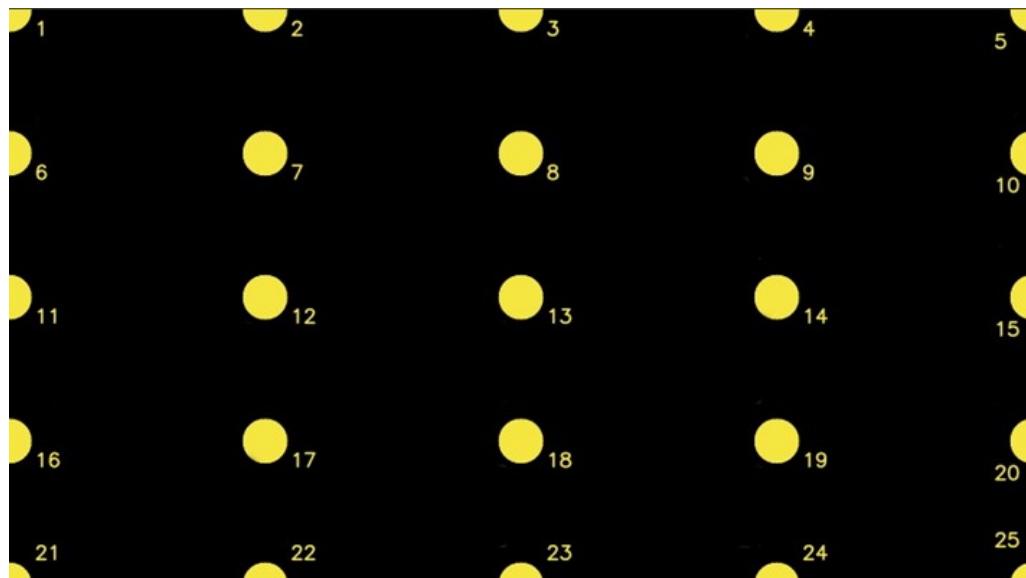
Tabela 2 – Exemplo de continuação da Tabela 1 após o segundo ponto de calibragem

x_{esquerdo}	x_{direito}	y_{esquerdo}	y_{direito}	x_{mouse}	y_{mouse}
214	214	310	310	-	-
206	305	311	308	400	-
-	-	-	-	-	-

E assim continuamente por todos os pontos a base de dados com os dados da calibragem vai sendo preenchida para que possam ser usados na(s) tarefa(s). A Figura 13 abaixo representa como seria a distribuição de 25 pontos de calibragem em uma interface

com a resolução de 1600x900, assim como a do exemplo, por meio de esferas sobre as coordenadas. A numeração próxima das esferas indica a ordem em que os pontos aparecem na tela.

Figura 13 – Local dos 25 pontos de calibração em uma interface.



Fonte: O autor

3.4 Tarefa

A tarefa não possui plano de realização específico e varia entre as avaliações de usabilidade, variando até mesmo dentro de uma mesma avaliação quando existem várias tarefas. Geralmente, existe uma meta a ser alcançada na tarefa e é durante esse percurso onde dados da câmera são coletados ([GOLDBERG; WICHANSKY, 2003](#)) (Figura 12).

Durante a tarefa são feitas correspondências com as posições dos olhos e são utilizados modelos de regressão para supor após a calibragem, com os dados obtidos nesta, os locais correspondentes na tela apenas com a posição do olhar. Cada usuário responsável por participar da avaliação, realiza sua(s) tarefa(s) e então os pontos do olhar obtidos durante esta são guardados bem como as suas coordenadas correspondentes na tela e o tempo destas em cada posição são enviados a base de dados.

Figura 14 – Exemplo de usuário sendo submetido a tarefa com rastreamento ocular.

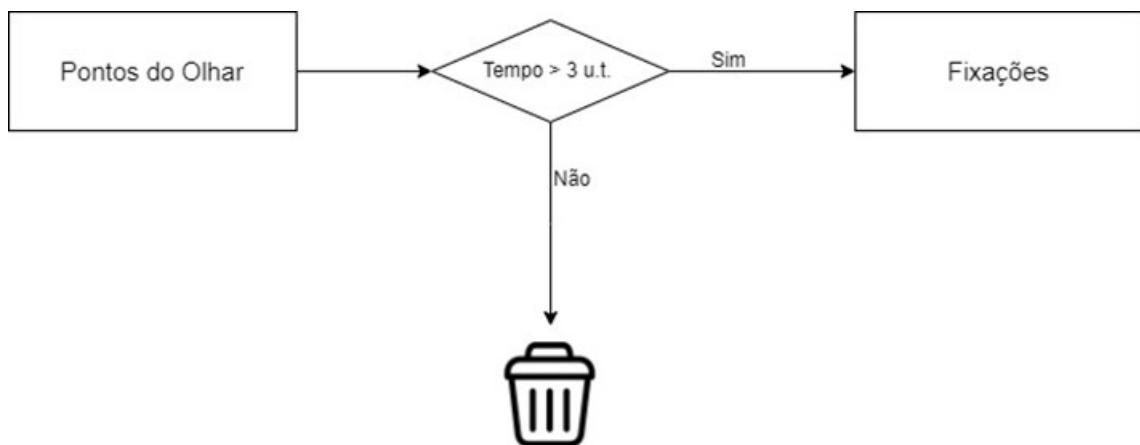


Fonte:[Goldberg e Wichansky \(2003\)](#)

3.5 Identificação

A partir da tabela com os pontos de olhar do usuário fornecida pela tarefa, começa o processo de diferenciação desses pontos (Figura 15). É feita uma varredura na tabela e aqueles pontos que tiverem duração de pelo menos 3 unidades de tempo, podem ser considerados fixações e assim serão úteis para as etapas posteriores, sendo salvos na base de dados. Aqueles que não alcançarem essa duração são basicamente descartados para os processos futuros.

Figura 15 – Classificação dos pontos de olhar.



Fonte: O autor

Depois de salvas as fixações, vem o processo de identificação das sacadas. Esse

processo acontece basicamente como no exemplo mostrado na Figura 16. A partir de cada fixação e a sua próxima reconhecida em sequência será considerada uma sacada. A primeira fixação com a segunda, a segunda com a terceira, a terceira com a quarta e assim por diante. Todas essas sacadas também serão guardadas na base de dados para serem usadas nas próximas etapas.

Figura 16 – Exemplo da identificação de sacadas.

Fixações		Sacadas	
		Início	Fim
1.	(150, 267)		
2.	(303, 294)		
3.	(718, 82)	(150, 267)	(303, 294)
4.	(822, 943)	(303, 294)	(718, 82)
		(718, 82)	(822, 943)

Fonte: O autor

3.6 Verificação

A partir de diversos trabalhos é possível constatar que as principais métricas utilizadas vêm principalmente a partir da análise independente das quantidades e durações dos movimentos oculares de fixações e sacadas, da análise de scanpaths que podem ser construídos com os dois movimentos citados anteriormente, ou até mesmo com a taxa de piscadas de um usuário ao realizar a tarefa.

Depois da identificação dos movimentos e seu armazenamento nas bases de dados no bloco anterior, são feitas verificações nos dados de cada um destes com base em métricas, já existentes e que serão discutidas a seguir, para que sejam fornecidos ao fim da análise os possíveis problemas encontrados, e que serão guardados para serem exibidos ao final.

Nas tabelas abaixo estão descritas algumas das principais métricas advindas dos objetos de análise citados logo acima e que podem ser úteis, além de suas possíveis relações com a usabilidade ([MKPOJIOGU; HUSSAIN; HASSAN, 2018](#)) e onde podem estar os problemas decorrente delas. Para sua construção foram considerados diversos trabalhos (que estão também referenciados na tabela) que ajudaram a entender o significado de cada métrica. Após as tabelas está uma seção destinada ao entendimento do que representa cada uma das relações com usabilidade que foram encontradas.

3.6.1 Fixações

As fixações como já explicado diversas vezes anteriormente são os momentos em que os olhos ficam um período de tempo focado num mesmo lugar. A partir da contagem e duração desses momentos é que se faz possível entender o seu significado e qual a importância deles quando relacionados a usabilidade.

Tabela 3 – Métricas das fixações, significados e possíveis relações com a usabilidade

Métrica	Significado	Referências	Relação com Usabilidade
Tempo até a primeira fixação no alvo desejado	Quanto menor o tempo levado pelo usuário até realizar a primeira fixação no alvo desejado, melhor.	Jacob e Karn (2003); Poole e Ball (2006); Ehmke e Wilson (2007); Liu e Zhu (2012); Joseph e Murugesh (2020); Ball e Richardson (2022);	<ul style="list-style-type: none"> • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade
Número Total de Fixações	Quanto maior o número de fixações, menor a eficiência da interface.	Goldberg e Kotval (1999); Jacob e Karn (2003); Poole e Ball (2006); Ehmke e Wilson (2007); Bhoir et al. (2015); Kumar e Kumar (2019); Joseph e Murugesh (2020); Ball e Richardson (2022);	<ul style="list-style-type: none"> • Eficiência: Precisão • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade • Aprendizagem: Simplicidade
Densidade espacial de fixações (Razão das Fixações no Alvo pelas Fixações totais)	Fixações concentradas em uma pequena área indicam busca focada e eficiente, já quando estão espalhadas indicam busca inefficiente.	Goldberg e Kotval (1999); Jacob e Karn (2003); Poole e Ball (2006); Ehmke e Wilson (2007); Joseph e Murugesh (2020); Ball e Richardson (2022);	<ul style="list-style-type: none"> • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade • Aprendizagem: Simplicidade

3.6.2 Sacadas

As sacadas que são basicamente os caminhos feitos entre as fixações podem ser usadas para entender o quanto o usuário percorreu em sua busca e quanto tempo ele durou

nesses percursos. Esses movimentos são amplamente utilizados para avaliar a qualidade da busca dos usuários na interface.

Tabela 4 – Métricas das sacadas, significados e possíveis relações com a usabilidade

Métrica	Significado	Referências	Relação com Usabilidade
Número de sacadas	Quanto maior o número de sacadas feitas, mais o usuário teve que buscar até chegar onde devia.	Goldberg e Kotval (1999) ; Poole e Ball (2006) ; Ehmke e Wilson (2007) ; Bhoir et al. (2015) ; Kumar e Kumar (2019) ; Nugraha et al. (2019) ; Joseph e Murugesh (2020) ; Ball e Richardson (2022) ;	<ul style="list-style-type: none"> • Eficiência: Precisão • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade • Aprendizagem: Simplicidade
Tamanho médio das sacadas	Se feita uma média do tamanho das sacadas, quanto maior for, melhor tende a ser a interface.	Goldberg e Kotval (1999) ; Poole e Ball (2006) ; Ehmke e Wilson (2007) ; Bhoir et al. (2015) ; Joseph e Murugesh (2020) ; Ball e Richardson (2022) ;	<ul style="list-style-type: none"> • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade
Duração total das sacadas	Quanto maior a duração das sacadas, mais buscas foram realizadas e ocorreu menos atenção as áreas esperadas.	Vuori et al. (2004) ; Ehmke e Wilson (2007) ; Kumar e Kumar (2019) ; Joseph e Murugesh (2020) ;	<ul style="list-style-type: none"> • Eficiência: Precisão • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade • Aprendizagem: Simplicidade

3.6.3 Razão Fixações/Sacadas

Ao obter a duração das fixações e a duração das sacadas, quando é feita a divisão da primeira métrica pela segunda, é possível ter uma ideia de proporcionalmente o quanto usuário demorou mais para procurar os elementos do que focando neles.

Tabela 5 – Métrica da razão fixações/sacadas, significado e possíveis relações com a usabilidade

Métrica	Significado	Referências	Relação com Usabilidade
Razão da Duração Fixações/- Sacadas	Dividindo a duração das fixações pela das sacadas, quanto menor for o resultado, mais tempo o usuário passou procurando os alvos do que observando eles.	Goldberg e Kotval (1999) ; Ehmke e Wilson (2007) ; Bhoir et al. (2015) ; Joseph e Murugesh (2020) ; Ball e Richardson (2022) ;	<ul style="list-style-type: none"> • Eficiência: Precisão • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade • Aprendizagem: Simplicidade

3.6.4 Scanpaths

Os scanpaths, que representam os locais das fixações e sacadas, podem ser vistos como um reflexo da busca como um todo já que engloba os dois movimentos na sua construção. Assim como as sacadas, os scanpaths também tem como seu papel na avaliação de usabilidade, refletir como está a qualidade da busca.

Tabela 6 – Métricas dos scanpaths, significados e possíveis relações com a usabilidade

Métrica	Significado	Referências	Relação com Usabilidade
Largura do Scanpath	Maior comprimento do scanpath indica site menos eficiente e navegação ruim.	Goldberg e Kotval (1999) ; Ehmke e Wilson (2007) ; Kumar e Kumar (2019) ; Joseph e Murugesh (2020) ; Ball e Richardson (2022) ;	<ul style="list-style-type: none"> • Eficiência: Precisão • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade • Aprendizagem: Simplicidade
Duração do Scanpath	Quanto maior a duração do scanpath, pior foi a eficiência da busca.	Goldberg e Kotval (1999) ; Ehmke e Wilson (2007) ; Joseph e Murugesh (2020) ; Ball e Richardson (2022) ;	<ul style="list-style-type: none"> • Eficiência: Precisão • Eficácia: Apresentação e Navegação • Satisfação: Conteúdo e Atratividade • Aprendizagem: Simplicidade

3.7 Verificações das Métricas

Considerando as dimensões e subdimensões da usabilidade de interfaces obtidas a partir do trabalho de [Mkpojiogu, Hussain e Hassan \(2018\)](#) é possível fazer uma relação dos significados de cada métrica apresentada nas tabelas acima com determinadas possíveis subdimensões de cada dimensão que podem auxiliar a demonstrar onde estão problemas de usabilidade presentes em interfaces que forem avaliadas fazendo uso do rastreamento ocular.

Abaixo cada uma das possíveis relações apresentadas nas tabelas tem uma pequena explicação e demonstração dos problemas que podem representar.

3.7.1 Verificação da Eficiência: Precisão

A eficiência, que é relacionada com o tempo gasto ao realizar uma tarefa pode ser afetada em uma interface por problemas de compatibilidade com diferentes dispositivos ou mesmo pela questão do tempo de resposta, mas ela se faz por vez em sua relação com o rastreamento ocular bastante refletida na subdimensão da precisão.

O tempo necessário para identificar um alvo pode indicar uma pior exibição de elementos que sejam precisos ao usuário, isto é, que seu significado esteja claro, podendo causar confusão e deixá-lo com dúvidas. Isso pode ser visto quando o usuário faz muito mais buscas do que realmente seria necessário.

Por exemplo, um elemento que não tenha sido construído com um significado claro para o usuário pode fazer com que ele não entenda realmente o que viu, ou entenda algo diferente do que deveria. Podendo passar a olhar para diversos lugares sem relação alguma com o alvo esperado e gastando mais tempo para chegar, ou então passando mais tempo que o necessário num mesmo elemento sem relação alguma com o alvo.

Serão analisados aqui os dados das fixações, sacadas e *scanpaths*. Nessa verificação serão consideradas possíveis problemas de precisão, e em consequência de eficiência, quando:

- **Nas fixações:** O número total de fixações for maior que o esperado.
- **Nas sacadas:** O número de sacadas e a duração total delas forem maiores que o esperado.
- **Na razão fixações/sacadas:** A razão das durações das fixações e das sacadas for menor que o valor esperado.
- **Nos *scanpaths*:** O tamanho e a duração totais tiverem valor maior que o esperado.

3.7.2 Verificação da Eficácia: Apresentação e Navegabilidade

A medida da eficácia que é geralmente vista com o quanto o usuário completou de uma tarefa, e é uma das medidas de usabilidade onde os resultados obtidos a partir de um rastreamento ocular podem ser mais utilizados. As suas duas dimensões, apresentação e navegabilidade, podem afetar em como o usuário usa a interface e isso é refletido através da movimentação do seu olhar.

A apresentação, como falado anteriormente, diz respeito a se os elementos da interface estão apresentados da melhor forma, permitindo que o usuário consiga entendê-los e agir intuitivamente em seguida. Já a navegabilidade, é responsável por garantir que a interface foi construída a partir de uma lógica que permita o usuário se localizar da melhor forma.

Quando um usuário acaba percorrendo muitos lugares até fixar seu olhar na área desejada, pode significar que ele está tendo problemas pra identificar o alvo e conseguir progredir na tarefa, e isso, consequentemente pode ser reflexo de uma apresentação e/ou uma navegação ruim.

Nesta verificação são analisados os dados das fixações, sacadas e *scanpaths*. Aqui serão consideradas possíveis problemas de apresentação e navegação, e em consequência de eficácia, quando:

- **Nas fixações:** O tempo até a primeira fixação no alvo desejado e o número total de fixações forem maiores, e a densidade espacial de fixação no alvo for menor que o esperado.
- **Nas sacadas:** O número total e a duração total das sacadas forem maiores, e quando o tamanho médio das sacadas for menor que o esperado.
- **Na razão fixações/sacadas:** A razão das durações das fixações e das sacadas for menor que o valor esperado.
- **Nos *scanpaths*:** O tamanho e a duração totais tiverem valor maior que o esperado.

3.7.3 Verificação da Satisfação: Conteúdo e Atratividade

A satisfação nos testes de usabilidade normalmente vem através de medidas subjetivas, em sua maioria questionários. Para que uma interface gere satisfação, segundo o trabalho de [Mkpojiogu, Hussain e Hassan \(2018\)](#) é necessário respeitar três subdimensões: conteúdo, guias de uso e atratividade. Principalmente duas delas podem ser analisadas através de dados do rastreamento ocular e refletir possíveis problemas de satisfação.

O conteúdo de uma interface tem que ter valor e ser significativo aos usuários que a utilizarão. Ou seja, quando um usuário vai fazer uso de uma interface, enquanto os elementos que a compõem não parecerem nada de importante, ele acabará continuando percorrendo-a em busca de algo que seja significativo no seu entendimento. Isso pode acabar prejudicando sua experiência com a interface.

A atratividade é possivelmente uma das subdimensões que tem mais impacto numa interface e que é mais refletida nos resultados obtidos com o rastreamento ocular. Quando um usuário foca muito em um mesmo local, uma das possíveis interpretações é que este está com uma boa atratividade e faz com que o usuário foque rapidamente nele. Em contraste, se uma área alvo para a tarefa não teve muito foco do usuário, pode significar que ela está pouco atrativa.

Nesta verificação serão analisados os dados das fixações, sacadas e *scanpaths*. Aqui serão consideradas possíveis problemas de conteúdo e atratividade, e em consequência de satisfação, quando:

- **Nas fixações:** O tempo até a primeira fixação no alvo desejado e o número total de fixações forem maiores, e a densidade espacial de fixação no alvo for menor que o esperado.
- **Nas sacadas:** O número total e a duração total das sacadas forem maiores, e quando o tamanho médio das sacadas for menor que o esperado.
- **Na razão fixações/sacadas:** A razão das durações das fixações e das sacadas for menor que o valor esperado.
- **Nos *scanpaths*:** O tamanho e a duração totais tiverem valor maior que o esperado.

3.7.4 Verificação da Aprendizagem: Simplicidade

A aprendizagem é a medida que se refere a quanto um usuário consegue entender e usar da melhor forma uma interface. Tem como subdimensões a simplicidade e a familiaridade, porém se tratando do rastreamento ocular e suas métricas, a primeira é a que mais se relaciona.

A interface pode não estar simples o suficiente, podendo fazer com que o usuário não entenda para onde ir e se esforce mais para chegar no alvo, o que ocasiona em mais buscas do que o necessário.

Na verificação de simplicidade serão analisados os dados das fixações, sacadas e *scanpaths*. Aqui serão consideradas possíveis problemas de conteúdo e atratividade, e em consequência de satisfação, quando:

- **Nas fixações:** O número total de fixações for maior e a densidade espacial de fixação no alvo for menor que o esperado.
- **Nas sacadas:** O número total e a duração total das sacadas forem maiores que o esperado.
- **Na razão fixações/sacadas:** A razão das durações das fixações e das sacadas for menor que o valor esperado.
- **Nos *scanpaths*:** O tamanho e a duração totais tiverem valor maior que o esperado.

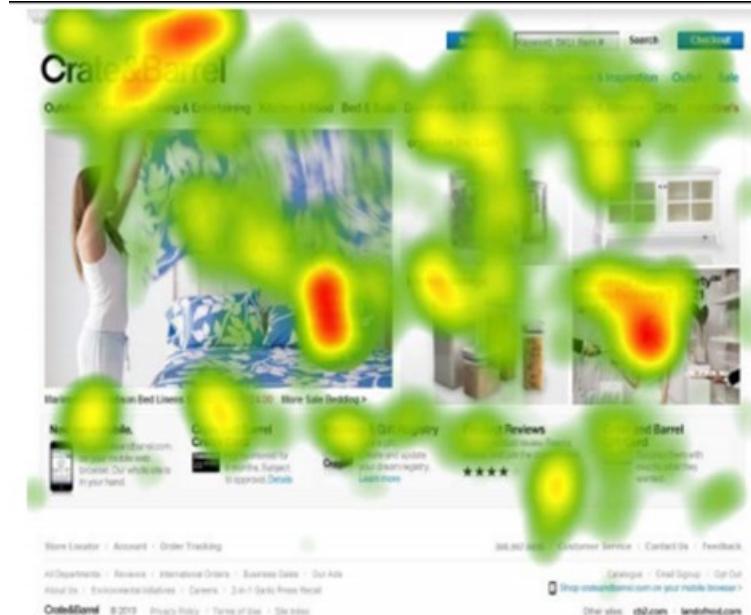
3.8 Representações

Já possuindo as fixações e sacadas se faz possível representar graficamente como foi a tarefa do usuário. Essas representações permitem um entendimento sobre o caminho que o usuário seguiu e os locais mais focados. As formais mais presentes nos trabalhos e que serão utilizadas são o Mapa de Calor e o *Scanpath*.

3.8.1 Mapa de Calor

Para que o mapa de calor seja formado é preciso calcular a intensidade com que cada *pixel* da interface foi focado a partir das fixações do participante, fazendo com que, geralmente, cores mais quentes se formem nas áreas mais focadas seguindo um padrão decrescente destas, como pode ser exemplificado na Figura 17.

Figura 17 – Mapa de calor feito a partir do rastreamento ocular de uma home page.



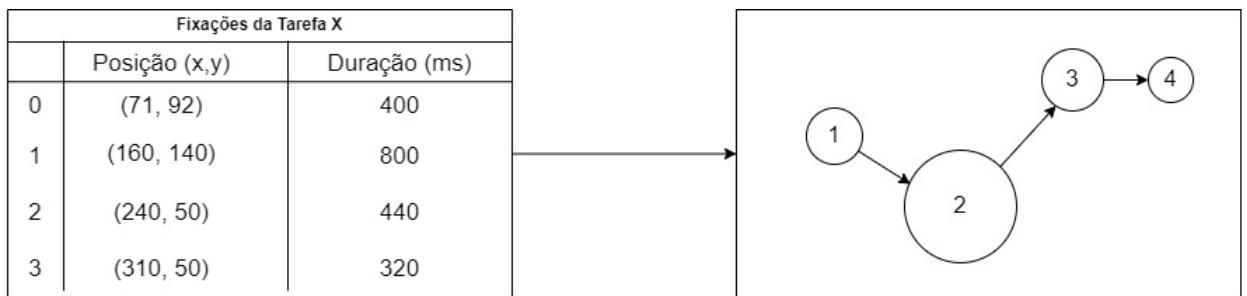
Fonte: Djamasbi (2014)

3.8.2 Scanpath

Na construção do scanpath são demonstrados sequencialmente os dados das fixações e das sacadas feitas pelo usuário durante a tarefa. As fixações são representadas como círculos contendo em seu interior o número que representa a sua ordem ao serem reconhecidas as fixações durante a tarefa, já as sacadas são representadas por setas que também demonstram a sequência das fixações.

Por exemplo, como representado na Figura 18, se em uma determinada Tarefa X, são detectados quatro pontos de fixação, para a construção do *scanpath* esses quatro pontos têm analisadas suas posições e durações. As posições dos pontos são usadas como centro dos círculos e as durações são proporcionalmente representadas no tamanho dos círculos.

Figura 18 – Exemplo de transformação de fixações de uma hipotética Tarefa X para exibição como *scanpath*



Fonte: O autor

3.9 Resultados

Com os resultados obtidos na etapa de associação, ao final da análise é possível a exibição destes de forma mais direta, indicando quais os possíveis problemas (se existentes) que foram encontrados durante a análise da interface em questão, apenas mostrando listas com eles.

Capítulo 4: Modelagem Física

A implementação do modelo proposto no capítulo anterior será feita a partir do desenvolvimento de uma aplicação utilizando *Python* e algumas de suas bibliotecas com o objetivo de fornecer resultados entendíveis ao usuário a partir das operações propostas.

A imagem captada pela webcam do computador usado é constantemente analisada em tempo real, possibilitando a detecção da pupila a partir de processos feitos anteriormente com o auxílio da biblioteca *dlib*, que passam pela detecção da região da face do usuário a partir da utilização do algoritmo de Histograma de Gradientes Orientados (*HOG*), assim como de cada um dos seus olhos, com a aplicação de um modelo treinado existente e de outros processos que permitirão o isolamento da área.

Após a detecção da pupila é feita uma etapa de calibragem, que tem como objetivo a identificação de para onde o usuário está olhando na tela a partir do posicionamento proporcional da sua pupila. Esse processo é feito inicialmente com o auxílio do usuário que visualiza pontos na tela e confirma quando olhar para eles. Esses dados são salvos em um *DataFrame* para serem convertidos em um arquivo *CSV* com a ajuda da biblioteca *Panda*.

Com os dados da calibragem disponíveis, é decidida a tarefa a ser feita pelo usuário. Para isso é usada uma página *Web* previamente construída, onde os possíveis alvos de olhar já têm suas localizações conhecidas, permitindo que possam ser identificados. Na tarefa o usuário terá o objetivo de focar em algum alvo e durante a execução desta, os dados de todos os lugares da interface onde houve a identificação de olhares e fixações serão salvos num arquivo.

A partir do arquivo obtido durante a tarefa, seus dados são analisados para a construção de três objetivos principais: identificação de movimentos do olhar, mapa de calor e *scanpath*. Esses três objetivos são atingidos a partir da observação da sequência de focos do olhar do usuário, sendo o terceiro dependente do primeiro. Sendo a representação dos dois últimos exibida ao fim de cada tarefa.

Após os registros dos movimentos do olhar do usuário (fixações e sacadas) e do caminho que estes formaram, são calculados e salvos os valores de cada uma das métricas previamente apresentadas no Capítulo 3, na Seção 3.6 (Tempo da primeira fixação no alvo,

número total de fixações, densidade espacial das fixações, número de sacadas, tamanho médio das sacadas, duração total das sacadas, razão da duração entre fixações e sacadas, tamanho do *scanpath*, duração do *scanpath*). Cada possível tarefa possui um arquivo próprio onde são armazenados os valores das métricas após cada execução de tarefa.

Para a avaliação das métricas presentes nos arquivos, são definidos previamente valores ideais de cada métrica em cada tarefa que tenha execução prevista. Assim que a(s) pessoa(s) responsável(is) pela avaliação deseja visualizar o resultado encontrado pelo sistema, é feita uma comparação entre os valores ideais estabelecidos para cada tarefa e a média dos seus valores correspondentes em cada um dos seus respectivos arquivos.

4.1 Ambiente de Desenvolvimento e Linguagem

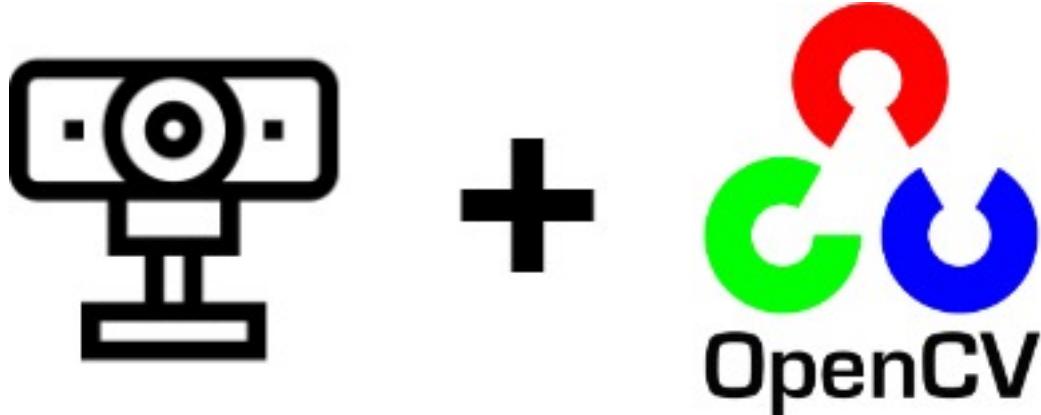
Para a construção do projeto foi escolhido editor de texto o *Visual Code Studio* que permite através de extensões maior facilidade na codificação e, além disso, permite que seja utilizado um terminal, *PowerShell* no caso, que se faz necessário na instalação das bibliotecas necessárias e na execução do código.

A linguagem de programação utilizada é o *Python*, que possui uma boa relação com algoritmos e bibliotecas que utilizam *machine learning*. A partir da instalação do *Python* no computador, é necessário para o funcionamento da aplicação, a instalação de diversas bibliotecas. Foi utilizado o *pip*, que é um instalador de pacotes para *Python*, na instalação dessas bibliotecas.

4.2 Capturando Imagens

O primeiro passo da aplicação como mostrado no modelo apresentado no Capítulo 3 (Figura 8), é conseguir a imagem do usuário em tempo real para que nos módulos futuros possa ser analisada. Essa é a parte mais simples e necessita basicamente de duas coisas: uma webcam conectada ao processador e a instalação da biblioteca *OpenCV* para *Python*.

Figura 19 – Webcam e OpenCV são as ferramentas necessárias para aquisição da imagem



Fonte: O autor

O código para conseguir essa imagem é constituído apenas pela linha abaixo:

```
1     webcam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

É utilizado o método *VideoCapture* do *OpenCV*, que tem nesse caso 2 parâmetros. O primeiro parâmetro 0 é o responsável pela identificação da webcam, onde caso a máquina tenha por exemplo uma segunda webcam conectada e quiser usar ela, basta simplesmente colocar o 1 como parâmetro. O segundo parâmetro faz referência a qual *API* está sendo usada para a captura, e que nesse caso é o *DirectShow* que é uma *API* do *DirectX*.

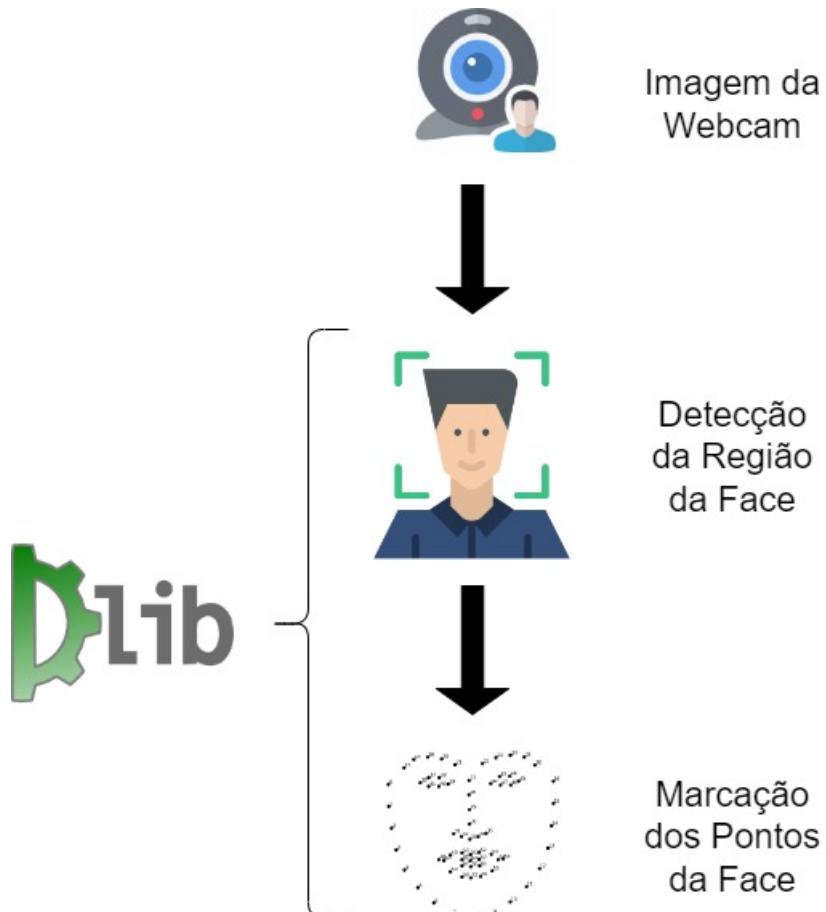
4.3 Detecção da Face, Olhos e Pupila

Após a obtenção da imagem da câmera, os esforços são voltados a sequência de detecção da face, em seguida dos olhos, e então da pupila.

4.3.1 Detecção da Face

Aqui é seguida uma sequência (Figura 20) onde para a detecção da face na aplicação são necessárias duas etapas: a detecção da região onde está a face, e em seguida a marcação dos 68 pontos comuns da face que permitem o reconhecimento de diversas regiões desta. Para essa parte passa a ser utilizada a biblioteca *dlib*, que trabalha com algoritmos de *machine learning* e, também, análise de dados. Os métodos do *dlib* utilizados, como pode ser visto no código abaixo, foram o *get_frontal_face_detector* e o *shape_predictor*.

Figura 20 – Sequência de passos para a detecção da face



Fonte: O autor

```

1 self.faceDetector = dlib.get_frontal_face_detector()
2
3 cwd = os.path.abspath(os.path.dirname(__file__))
4 modelPath = os.path.abspath(os.path.join(cwd, "modelo_treinado/
    shape_predictor_68_face_landmarks.dat"))
5 self.predictor = dlib.shape_predictor(modelPath)

```

Nele, inicialmente a variável *faceDetector* recebe o método de detecção da face do *dlib*, *get_frontal_face_detector*. A variável *cwd* recebe o caminho do diretório do arquivo e em seguida *modelPath* recebe a concatenação do caminho presente em *cwd* com o caminho interno que leva ao arquivo que contém o modelo treinado para detecção da face. E finalmente, a variável *predictor* recebe o retorno do método *shape_predictor* usando o modelo indicado por *modelPath*.

4.3.1.1 Detecção da Região da Face

A dlib, quando se trata especificamente de detecção facial, utiliza-se internamente o Histograma de Gradientes Orientados (HOG). O primeiro método dessa biblioteca que é aqui utilizado que faz a detecção da região onde está a face do usuário a partir da imagem obtida é o `get_frontal_face_detector`, que no código, é chamado quando o usuário faz uso da variável `faceDetector`. O código abaixo mostra o momento em que o método é utilizado na imagem obtida, após esta ter suas cores convertida para escala de cinza.

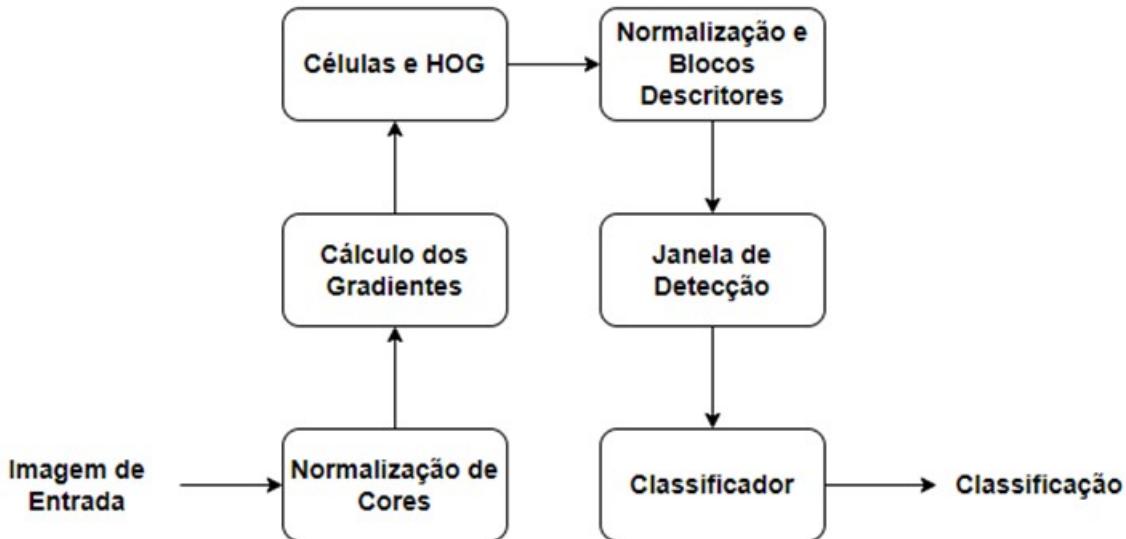
```
1 def analize(self):
2     grayImage = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
3     faces = self.faceDetector(grayImage)
```

O trabalho de [Boyko, Basystiuk e Shakhovska \(2018\)](#) mostra que o algoritmo HOG, que começou a ser usado na detecção de características humanas a partir do trabalho de [Dalal e Triggs \(2005\)](#), continua sendo popular nos dias de hoje e pode ser usado no reconhecimento facial. Esse algoritmo como o nome já sugere, trabalha com o conceito de cálculo de gradientes, onde estes se fazem fundamentais para encontrar onde estão as bordas, no caso, os contornos que formam uma face. A partir da detecção da região da face se faz necessário encontrar 68 pontos específicos que funcionam como um padrão na face, e podem ser úteis para isolar determinadas partes.

O uso do algoritmo HOG para detecções em humanos foi planejado por [Dalal e Triggs \(2005\)](#) para funcionar seguindo uma sequência de processos, que pode ser vista na Figura 19, desde a leitura da imagem até a classificação obtida a partir dela.

Para isso a partir da imagem de entrada, suas cores são normalizadas, cada *pixel* da imagem tem seu gradiente calculado, são criadas células formadas por grupos de *pixel* onde, seus gradientes são analisados para a formação de histogramas. Com o intuito de eliminar efeitos da diferença de luminosidade, as células são agrupadas em blocos e estes são normalizados dando formatos finais aos histogramas que representam a imagem e que são analisados para obter características que permitem a um classificador detectar os componentes que formam a face.

Figura 21 – Sequência usada no algoritmo com HOG para detecção em humanos.



Adaptado de Dalal e Triggs (2005)

A Normalização das Cores

As imagens que passam pelo processo, quando coloridas, utilizam o sistema de cores *RGB* ou o *BGR*, com três canais, ou então estão em escala de cinza ou algum outro canal único. Em ambos os sistemas os seus canais de cores assumem valores que vão do 0 ao 255. Com esses valores o risco de cálculos futuros estourarem a capacidade de armazenado de uma variável inteira é bem grande.

Por isso, se faz necessário um processo de normalização dos valores de cada canal. Nesse processo, os valores variarão de 0 a 1, impondo um limite claro no valor que pode ser alcançado. O exemplo a seguir, permite entender a normalização de cores.

Exemplo: Supondo que seja preciso em um processo A, elevar ao cubo o valor V de um canal de cor:

- Se o valor usado não tiver sido normalizado, o resultado da operação poderá variar entre 0 e 16581375.
- Já se houver sido feita a normalização, o resultado variará de 0 a 1.

O exemplo acima demonstra o quanto grande podem se tornar os valores envolvidos em cálculos simples caso não normalizados, e é preciso considerar que ainda poderiam haver mais cálculos em seguida que aumentariam ainda mais suas dimensões. A normalização permite que independentemente do número de cálculos feitos, seus resultados sempre irão variar entre 0 e 1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

A normalização de um valor de forma geral ocorre pela fórmula acima, onde X é um valor presente em um intervalo que varia entre X máximo e X mínimo.

Mas considerando que o objetivo em questão é a normalização de valores que sempre podem variar entre 0 e 255, o X mínimo se torna desnecessário e a fórmula assume como padrão:

$$X_{norm} = \frac{X}{255}$$

B Cálculo dos Gradientes

Com o objetivo de obter informações de borda, são calculados os gradientes de cada pixel. O gradiente de um *pixel* indica a variação da intensidade de cor entre ele e seus vizinhos laterais, sendo composto por dois componentes que precisam ser calculados: orientação e magnitude.

Em imagens com mais de um canal de cores, é calculado o gradiente de cada canal separadamente, o pixel então, assume o gradiente do canal que apresentar a maior magnitude.

Para obter a orientação e o módulo de um gradiente (vetor) de um *pixel*, primeiro deve ser encontrada a variação entre ele e seus vizinhos dos eixos cartesianos X e Y.

Em ambos os eixos, é feita a aplicação de uma máscara unidimensional com o formato [-1,0,1] centralizada no pixel alvo, respeitando a orientação de seus respectivos eixos.

Figura 22 – Representação de exemplo de pixel e sua vizinhança com intensidades dos gradientes

0.35	0.60	0.70
0.30	0.40	0.50
0.20	0.30	0.45

Fonte: O autor

A Figura 22 acima, representa um *pixel* (marcado em azul) e seus pixels vizinhos. Todos eles tendo nos centros valores que representam a intensidade de um dos canais de cores (possivelmente único) da sua imagem, já normalizados. Para calcular a variação lateral é aplicada a máscara da forma apresentada na Figura 23.

Figura 23 – Representação de exemplo da aplicação de máscara para cálculo da variação horizontal no pixel central

0.35	0.60	0.70
0.30	0.40	0.50
0.20	0.30	0.45

+

-1	0	1
----	---	---

=

-0.30	0	0.50
-------	---	------

Fonte: O autor

Somando os valores resultantes da aplicação da máscara, é obtido o valor da variação no eixo x. No caso: $-0.30 + 0 + 0.50$, então a variação em X é 0.20

No caso do eixo Y, o padrão é seguido (Figura 24):

Figura 24 – Representação de exemplo da aplicação de máscara para cálculo da variação vertical no pixel central

0.35	0.60	0.70
0.30	0.40	0.50
0.20	0.30	0.45

+

1
0
-1

=

0.60
0
-0.30

Fonte: O autor

Somando, a variação em Y é de 0.30.

- Orientação

A orientação do gradiente de um pixel é dada pelo arco tangente da razão da variação do eixo y pela variação do eixo x, como na fórmula abaixo.

$$\Theta = \arctan\left(\frac{\Delta y}{\Delta x}\right)$$

No caso do exemplo acima a orientação do gradiente se daria por

$$\Theta = \arctan\left(\frac{0.30}{0.20}\right)$$

$$\Theta = 56.31^\circ \text{ ou } \Theta \cong 0.9828 \text{ rad}$$

- **Magnitude**

A magnitude de um gradiente tem como uma das formas de se entender na Figura 25, onde as variações no eixo x e y são os catetos e a magnitude (M) é a hipotenusa.

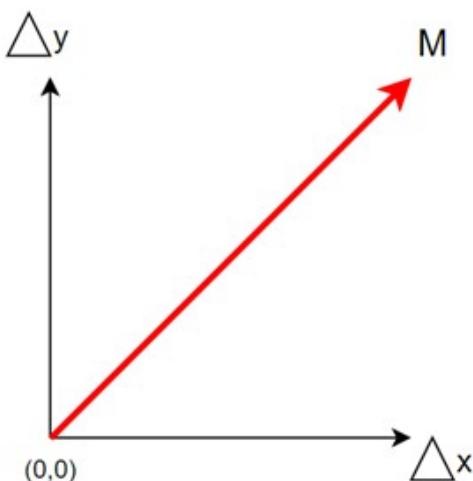
Portanto para o cálculo do valor da magnitude do gradiente é preciso apenas a aplicação do Teorema de Pitágoras usando as variações de eixo como na fórmula abaixo

$$M = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$

Caso aplicada no exemplo de demonstração:

$$M = \sqrt{(0.20)^2 + (0.30)^2} \longrightarrow M \cong 0.36$$

Figura 25 – Representação de como pode ser vista a relação entre magnitude do gradiente e variações de eixo

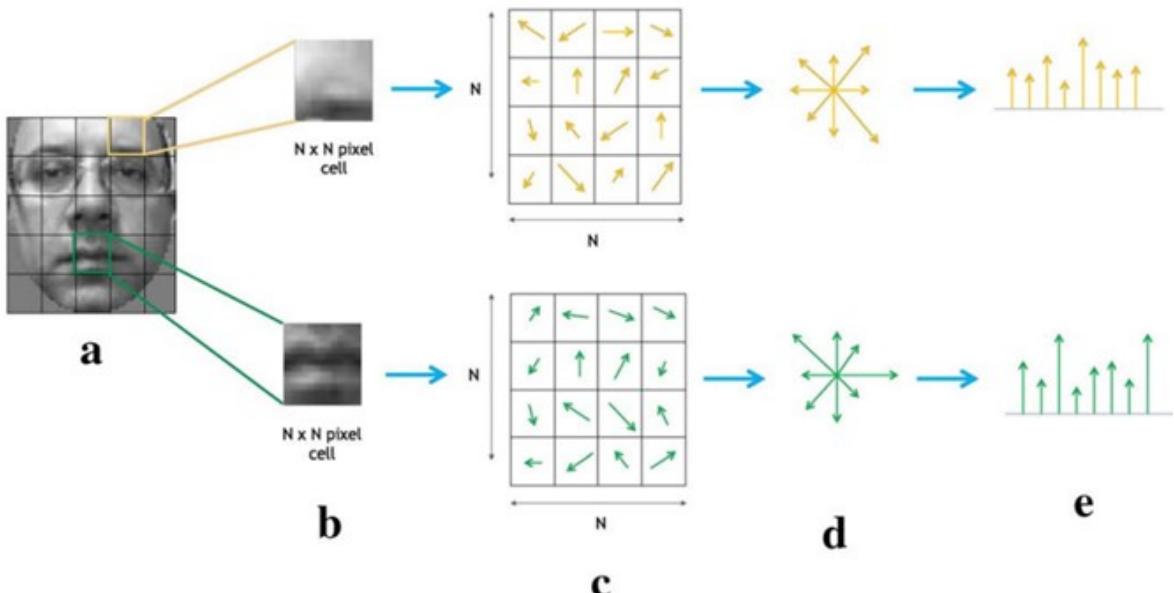


Fonte: O autor

C Células e HOG

O objetivo desta etapa é a separação da imagem em células e a obtenção de um histograma de gradientes orientados para cada célula, como representado na Figura 26.

Figura 26 – Representação dos passos desde a divisão em células(a) com mesma quantidade de pixels (b) até o cálculo dos intervalos de gradientes (c e d) e a construção do histograma(e).



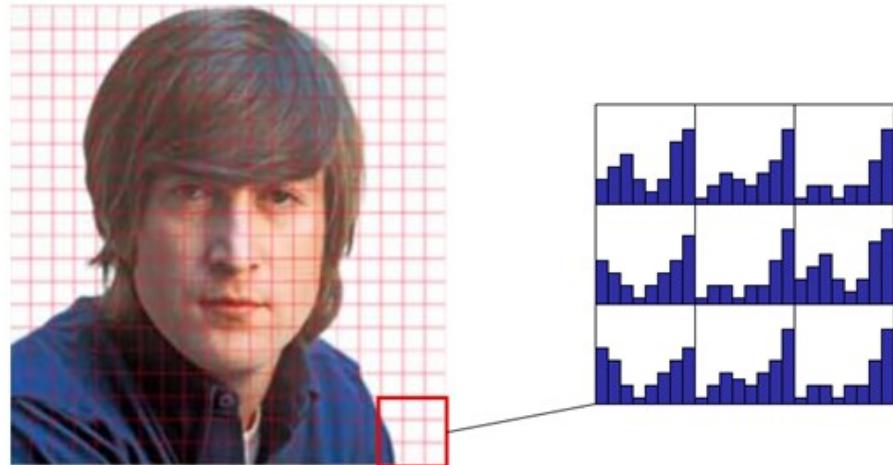
Modificado de Carcagni et al. (2015)

Após cada *pixel* ter seu gradiente calculado, o algoritmo faz a divisão da imagem em pequenos blocos retangulares chamados de “células”, o tamanho das células é ajustado de acordo com cada imagem. Todas as células, portanto, contém um mesmo número limitado de *pixels*, mantendo os dados de seus gradientes. O intuito do processo nessa etapa é fazer com que cada uma das células tenha um histograma próprio, formado a partir da orientação e magnitude de cada pixel que as compõem. Para isso, é justamente aplicado o HOG.

No HOG, cada barra representa intervalo de valores das orientações de cada *pixel* da célula. Nesse algoritmo, a altura de cada barra se dá por um sistema de votos com pesos de cada *pixel*. O peso do voto de cada *pixel* é decidido em função da magnitude do seu gradiente, podendo ser o próprio valor da magnitude ou sua raiz quadrada, em alguns casos.

Esse processo obtém como resultado algo como o mostrado na Figura 27 abaixo, que exemplifica um pequeno conjunto de células de uma imagem, cada uma com seu histograma.

Figura 27 – Representação de exemplo da obtenção de histogramas em cada célula.



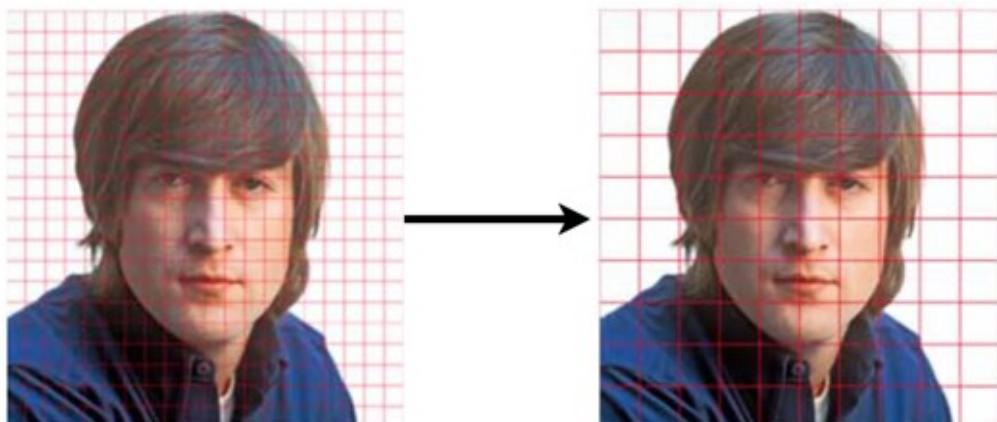
Modificado de Boyko, Basystiuk e Shakhovska (2018)

D Normalização e Blocos Descritores

As variações de iluminação de uma imagem acabam tendo grande influência sobre os valores assumidos pelos gradientes em determinados locais, de modo que a normalização por setores maiores da imagem se faz necessária para neutralizar esses efeitos no resultado final da detecção.

Para isso, nessa etapa do processo um conjunto de células são agrupadas em blocos maiores, nos chamados blocos descritores (Figura 28). Todos os blocos da imagem possuem as mesmas dimensões, consequentemente quantidade de células.

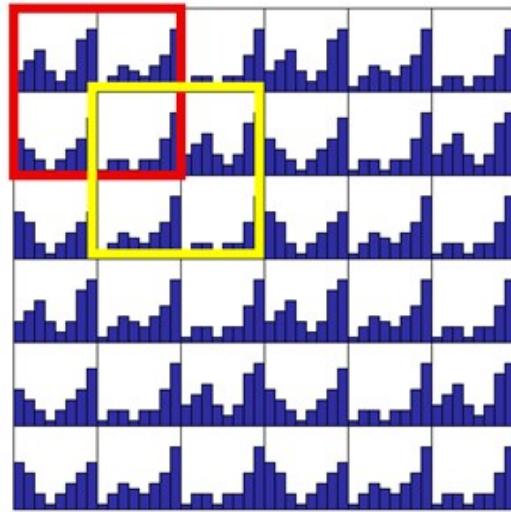
Figura 28 – Representação de exemplo do agrupamento das células em blocos descritores.



Modificado de Boyko, Basystiuk e Shakhovska (2018)

No processo de normalização dos blocos descritores é feita a prática de sobreposição entre blocos (Figura 29), assim, normalmente uma célula contribui na normalização de mais de um bloco.

Figura 29 – Representação de exemplo da sobreposição de blocos no processo de normalização.



Fonte: O autor

Para normalizar os valores dos blocos é utilizada a norma euclidiana com a fórmula:

$$V_N = \frac{V}{\|V\|_2}$$

Onde V é o vetor que contém os valores que denotam a altura de todas as barras de todas as células do bloco em questão e $\|V\|_2$ é a norma euclidiana do vetor V que pode ser obtida fazendo.

$$\|V\|_2 = \varepsilon + \sqrt{\sum_{n=0}^N V(n)^2}$$

Sendo n, a posição do vetor que vai de 0 a N, que representa a última posição possível, e ε sendo uma constante de valor muito pequeno para impedir que no cálculo do valor normalizado, ocorra uma divisão por 0.

Após a normalização de todos os blocos, os histogramas resultantes já podem ser usados para obter características da imagem analisada.

E Janela de Detecção

Nessa parte é incluída uma margem de alguns *pixels* em todos os lados (em seu trabalho, Dalal e Triggs incluem 16 *pixels* em cada lado). Segundo os autores: “essa borda fornece uma quantidade significativa de contexto que ajuda na detecção”.

F Classificador

Após a normalização dos valores de todos os histogramas o algoritmo HOG está completo e a face pode ser detectada. Para isso é usado uma *SVM* (Máquina de Vetores de Suporte, do inglês *Support Vector Machine*), que é um algoritmo de *machine learning* que, após treinado, é usado como classificador.

A vantagem de usar uma SVM é que ela pode lidar com dados simples e complexos de forma eficaz e rápida, o que é extremamente necessário já que a detecção é feita em tempo real enquanto o usuário é captado pela câmera.

Nesse caso, a partir de características extraídas dos histogramas, a SVM analisa cada componente da imagem e diz se ele faz parte ou não da face na imagem, se existir uma.

4.3.1.2 Marcação dos Pontos da Face

Após a detecção da face, como já mencionado antes, se faz necessária a marcação dos 68 pontos comuns em uma face (Figura 30). Esses pontos são extremamente importantes na próxima etapa do módulo de detecção.

Figura 30 – 68 pontos da face



Fonte: Boyko, Basystiuk e Shakhovska (2018)

Para essa tarefa a biblioteca dlib, do *Python*, continua sendo útil e através do método *shape_predictor* e o uso de um modelo já treinado existente entre os arquivos da aplicação, todos os pontos são facilmente identificados á partir da região detectada no processo anterior.

Esse método é o resultado da implementação baseada no trabalho de Sagonas et al. (2016) que apresenta os resultados do *300 Faces In-The-Wild Challenge (300-W)*, que foi uma competição que serviu de comparação entre as diferentes bases de dados de marcação facial existentes e tentar uni-las sob um mesmo protocolo superando os problemas individuais de cada.

Os 68 pontos da face obtidos vêm originalmente das bases de dados *Multi-PIE* e *IBUG*, presentes no *300-W*, sendo a segunda implementada utilizando o mesmo esquema de pontos da primeira. Os pontos foram estabelecidos manualmente no desenvolvimento da base *Multi-PIE* e foram utilizadas centenas de imagens de mais de 300 indivíduos em diferentes condições de iluminação, expressões e poses (GROSS et al., 2010), no caso da *IBUG* foram utilizadas centenas de imagens obtidas pela internet, também com diferentes condições (SAGONAS et al., 2016).

A diversidade das imagens utilizadas no desenvolvimento dessas bases permite

que imagens obtidas em vários tipos de ambientes consigam ser analisadas, e seus pontos podem ser encontrados sem grandes problemas.

```
1 landmarks = self.predictor(grayImage, faces[0])
```

A variável *landmarks* é feita então como a responsável por receber a localização de todos os pontos identificados e usados nos processos seguintes.

4.3.2 Detecção dos Olhos

A detecção da região dos olhos após a marcação dos 68 pontos na face do usuário é um processo contínuo ao anterior. A partir do momento em que o modelo de marcação proposto é utilizado, cada um dos olhos vai ser referenciado por seis números específicos (Figura 29). Essa parte pode ser dividida em duas etapas: identificação do olho e isolamento.

Figura 31 – Pontos da face que fazem referência aos olhos.



Modificado de Boyko, Basystiuk e Shakhovska (2018)

4.3.2.1 Identificação do Olho

Após o momento em que a face é marcada, começa a ser feita a identificação separada de cada um dos olhos onde a face é dividida entre dois lados (esquerda e direita).

```
1 self.leftEye = Eyes(grayImage, landmarks, 0, self.binarizacao)
2 self.rightEye = Eyes(grayImage, landmarks, 1, self.binarizacao)
```

São criadas duas variáveis que chamam as operações da classe *Eyes* que faz parte do arquivo *olho.py*. A variável *leftEye* recebe os dados do olho esquerdo e a *rightEye*, do olho direito. Como pode ser visto no código acima, são passados como terceiros parâmetros, 0 e 1. Esses valores ao serem recebidos em *Eyes* (abaixo), são armazenados em *side* e desempenham o papel de indicar para a classe, as marcas de qual olho estão sendo consideradas no momento. Assim, a etapa de isolamento consegue saber em qual olho deve focar no momento do processamento.

```
1 class Eyes(object):
2     LeftEyeMarks = [36, 37, 38, 39, 40, 41]
3     RightEyeMarks = [42, 43, 44, 45, 46, 47]
4
```

```

5     def __init__(self, image, pontos, side, binarizacao):
6         self.frame = None
7         self.origin = None
8         self.center = None
9         self.pupil = None
10        self._analyze(image, pontos, side, binarizacao)

```

```

1 def _analyze(self, original_frame, landmarks, side, binarizacao):
2     if side == 0:
3         points = self.LeftEyeMarks
4     elif side == 1:
5         points = self.RightEyeMarks
6     else:
7         return

```

4.3.2.2 Isolamento

Com a identificação de quais são os pontos do olho, é iniciada a etapa de isolamento dos olhos na imagem, o processo funciona com o uso de máscaras e outras funções gráficas das bibliotecas *Numpy* e *OpenCV*. O processo é necessário para que desse modo a imagem obtida não tenha as outras partes do rosto que não são mais necessárias.

```
1 self._isolate(original_frame, landmarks, points)
```

No código acima é chamada a função *isolate*, passando como parâmetros, respectivamente, a imagem, todos os pontos da face, e os pontos específicos do olho a ser isolado. Essa função (abaixo), salva em um vetor as coordenadas dos pontos do olho, cria uma imagem totalmente preta e também uma totalmente branca, que é usada como máscara.

```

1 def _isolate(self, frame, landmarks, points):
2     region = np.array([(landmarks.part(point).x, landmarks.part(point).y) for point
3                         in points])
4     region = region.astype(np.int32)
5     height, width = frame.shape[:2]
6     black_frame = np.zeros((height, width), np.uint8)
7     mask = np.full((height, width), 255, np.uint8)
8     cv2.fillPoly(mask, [region], (0, 0, 0))
9     eye = cv2.bitwise_not(black_frame, frame.copy(), mask=mask)
10    margin = 5
11    min_x = np.min(region[:, 0]) - margin
12    max_x = np.max(region[:, 0]) + margin
13    min_y = np.min(region[:, 1]) - margin
14    max_y = np.max(region[:, 1]) + margin
15    self.frame = eye[min_y:max_y, min_x:max_x]

```

Com o uso de funções do *OpenCV*, é criada uma imagem onde o único elemento presente é justamente o olho isolado (Figura 32). Após isso, são adicionadas algumas margens e a imagem é reduzida a apenas a área de interesse desse processo (Figura 33). Dessa forma os processos seguintes não precisam descobrir novamente qual a área relevante na imagem da câmera (os olhos).

Figura 32 – Exemplo da imagem produzida logo após o processo de isolamento do olho



Fonte: O autor

Figura 33 – Exemplo da imagem produzida após corte da área desnecessária

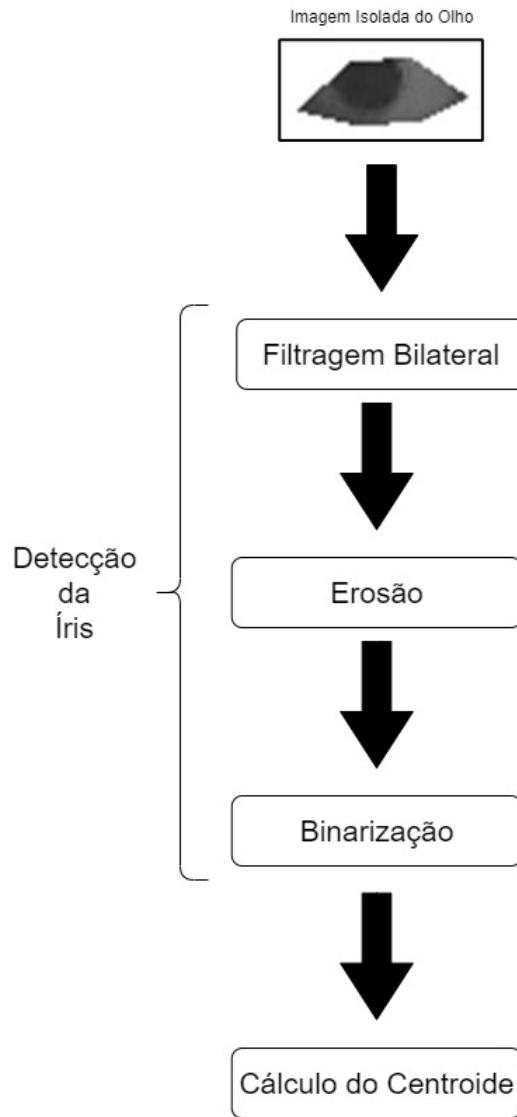


Fonte: O autor

4.3.3 Detecção da Pupila

Com a região do olho isolada, o passo seguinte é encontrar onde está localizada a pupila do usuário na imagem. Para isso, uma das técnicas existentes, considera a assunção de que a pupila está situada no centro da íris. Ou seja, para encontrar a pupila, é preciso que a região da íris seja identificada para que então sejam feitos os cálculos. Esse processo é dividido em duas etapas (Figura 34): detecção da íris e cálculo do centroide.

Figura 34 – Sequência para detecção da pupila

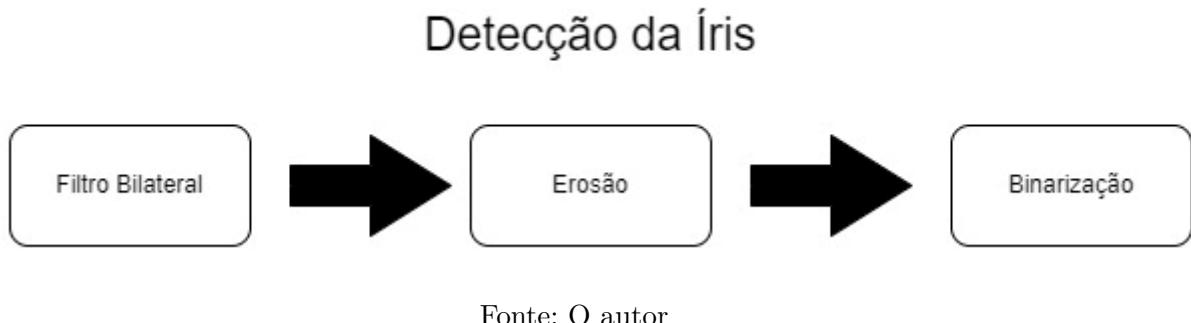


Fonte: O autor

4.3.3.1 Detecção da Íris

Logo após a detecção da região do olho, é necessário tratar a imagem para permitir a detecção da íris. Para isso são feitos três processos sequenciais (Figura 35): a aplicação de um filtro bilateral, o uso de um método de erosão e a binarização da imagem obtida.

Figura 35 – Sequência para detecção da íris.



A Filtragem Bilateral

A filtragem bilateral é o primeiro passo no processo de detecção da íris do usuário e seu uso tem como objetivo a remoção de ruídos. Esse método também usa um tipo de filtro gaussiano, porém ao contrário dos filtros gaussianos normalmente utilizados, que trabalham com a média dos pixels ao redor, na filtragem bilateral as bordas permanecem nítidas.

Isso acontece porque como pode ser visto na Figura 36 logo abaixo, além de manter a função espacial do *Gaussian Blur* como um dos seus componentes, é também utilizada uma função gaussiana de diferença de intensidade que funciona de forma que considera para o cálculo somente os pixels que tiverem uma intensidade próxima ao *pixel central* (último componente da Função Bilateral da figura abaixo). Assim, bordas que, geralmente, tem *pixels* com intensidades bem diferentes do centro da imagem não são afetadas.

Figura 36 – Comparação das fórmulas matemáticas dos filtros Gaussian Blur e Filtro Bilateral

$$GB[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|) I_q$$

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(I_p - I_q) I_q$$

Fator de Normalização Gaussian Blur (Espacial) Gaussian de Intervalo (Intensidade)

Fonte: O autor

B Erosão

Após a aplicação da filtragem bilateral é aplicado um método de erosão na imagem obtida. Esse processo acontece pela aplicação na imagem de um *kernel*, que é uma matriz quadrada de ordem ímpar. Esse kernel trabalha, como o próprio título desta seção sugere, erodindo os componentes da imagem em áreas próximas às bordas.

Esse processo é importante pois é utilizado para simplificar as características da imagem diminuindo a espessura de seus componentes e possíveis ruídos restantes. Isso possibilita que a binarização a ser feita da imagem tenha um resultado mais claro.

C Binarização

Logo após o processo de erosão, é necessário realizar a binarização da última imagem resultante para que assim seja possível determinar com maior precisão as duas principais regiões aparentes do olho, esclera (região branca) e íris, e focar somente na mais importante, a íris.

Para isso, primeiramente, é encontrado o melhor *threshold* (limiar) possível, que é o valor que melhor consegue servir para dividir os *pixels* da imagem entre os que ficarão pretos e os que ficarão brancos no processo. Mais especificamente, os pixels que contém valor abaixo do *threshold* são transformados em *pixels* branco, e todos os outros se tornam *pixels* pretos. Dessa forma, é possível separar claramente a íris da esclera.

4.3.3.2 Cálculo do Centroide

Com a obtenção da imagem binarizada é feita a obtenção dos contornos, e a partir destes são obtidos os momentos da imagem que são dados necessários para o cálculo do centroide. Esses dois processos são feitos utilizando-se de métodos da biblioteca *OpenCV*.

Os momentos são as médias de intensidade dos pixels presentes na imagem. Com eles pode-se achar diferentes propriedades da imagem, como áreas, raio, orientações entre outras informações. Incluindo nestas, as coordenadas de um centroide.

Para o cálculo do centroides são utilizadas duas fórmulas pré-estabelecidas, exibidas abaixo, que servem para definir a coordenada (Cx, Cy) do centroide da imagem.

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

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

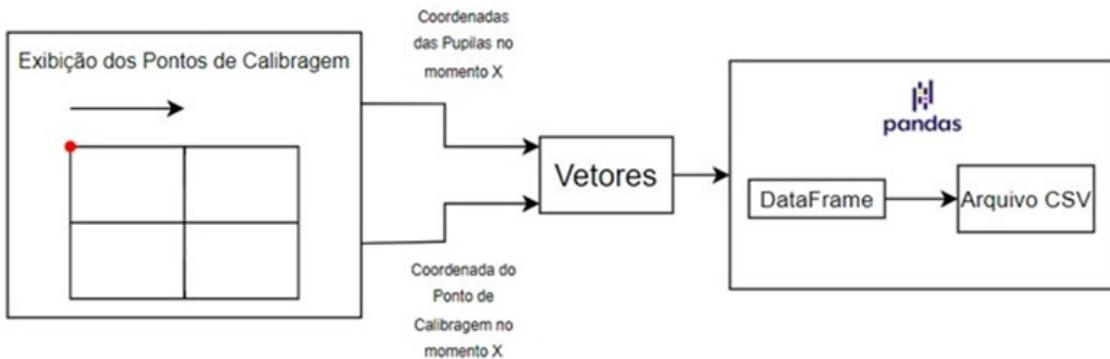
Como pode ser visto no código utilizado abaixo, é necessário que os momentos sejam chamados utilizando-se exatamente os mesmos identificadores vistos nas fórmulas acima ($m00$, $m01$ e $m10$).

```
1 moments = cv2.moments(contours[-2])
2 self.x = int(moments['m10'] / moments['m00'])
3 self.y = int(moments['m01'] / moments['m00'])
```

4.4 Calibragem

A partir da identificação da localização das pupilas do usuário, é necessário que exista uma forma de relacionar seus posicionamentos com respectivos pontos na interface que são, de fato, o local da interface para onde o usuário está olhando. Para isso, uma das formas de obter essa relação é por meio do processo de calibragem, salvando registros de algumas localizações. O processo é feito basicamente a partir da aquisição de coordenadas das pupilas e da interface em determinados momentos e esses dados são manipulados até serem salvos em um arquivo para uso de processos futuros (Figura 37).

Figura 37 – Etapas do processo de calibragem.



Fonte: O autor

No processo de calibragem, primeiramente são recebidas informações sobre as medidas da interface utilizada, então ela é virtualmente dividida como uma matriz, e é definida no código a quantidade específica de linhas e colunas que são consideradas a partir do fracionamento das dimensões da interface. No código abaixo, por exemplo, o código define a divisão virtual da interface em 4 linhas e 4 colunas. As variáveis *x_next* e *y_next* recebem as coordenadas, x e y respectivamente, dos pontos de calibragem.

```

1 x_next += (screenSizeX / 4)
2 y_next += (screenSizeY / 4)
  
```

Os pontos de calibragem são mostrados sequencialmente usando como marcação o ponteiro do mouse em todos os pontos onde as bordas das linhas e colunas virtuais na interface se cruzam. Em cada um desses pontos o usuário, olha na direção dele e através de um comando (pressionar uma tecla específica no teclado), a posição das pupilas e a posição dos pontos são salvas em vetores criados.

Para conseguir armazenar e trabalhar com os valores desses vetores, estes são convertidos em um DataFrame e em seguida em um arquivo de formato *CSV (comma-separated-values)*, que é traduzido como “valores separados por vírgula”). Para fazer essas conversões é usada a biblioteca *Pandas*, bastante utilizada para manipulação de dados. O arquivo csv resultante assume a forma mostrada na Figura 38.

Figura 38 – Exemplo de arquivo CSV resultante do processo de calibragem.

```
1 x_left,x_right,y_left,y_right,x_mouse,y_mouse
2 305,416,269,265,0,0
3 300,410,268,265,400,0
4 288,399,271,267,800,0
5 274,383,268,265,1200,0
6 257,364,268,265,1600,0
7 301,411,271,267,0,225
8 291,399,271,268,400,225
9 278,386,272,269,800,225
10 266,373,272,269,1200,225
11 255,361,272,270,1600,225
12 308,417,274,272,0,450
13 300,409,274,273,400,450
14 283,392,276,275,800,450
15 268,376,278,278,1200,450
16 255,361,280,281,1600,450
17 312,421,284,283,0,675
18 295,405,281,281,400,675
19 277,387,283,281,800,675
20 264,370,281,281,1200,675
21 250,354,279,281,1600,675
22 311,421,287,286,0,900
23 295,405,290,289,400,900
24 276,384,286,288,800,900
25 268,375,285,289,1200,900
26 252,354,282,284,1600,900
27
```

Fonte: O autor

4.5 Tarefa

No processo da tarefa três passos são necessários (Figura 39), a página WEB que será avaliada e pode ser construída tendo como objetivo a própria avaliação, o mapeamento de quais são os possíveis alvos para onde o usuário olhará, além da própria execução da tarefa pelo usuário onde os movimentos de seus olhos são convertidos em dados registrados em arquivos.

Figura 39 – Etapas para realização da tarefa.



Fonte: O autor

4.5.1 Página Web

A página Web é o meio a ser avaliado pelo sistema, portanto a sua existência prévia a avaliação é indispensável, podendo ser construída usando qualquer ferramenta disponível. Durante a construção do sistema foi construída uma página simples (Figura 40) para demonstração utilizando-se apenas de *HTML*, *CSS* e *JavaScript*, onde algumas condições se mostraram necessárias para o funcionamento correto. Sendo estas:

- Os elementos são apresentados no navegador de modo que o usuário não precise usar a barra de rolagem.
- Elementos de animações como o *dropdown* não são identificadas pelo sistema como novo alvo, portanto a área ocupada por ele após a animação e a área “atrás” dele são consideradas o mesmo local. Portanto apesar de bastante utilizados pela *internet*, não são utilizados na página de demonstração.

Figura 40 – Página WEB construída para demonstração do sistema



Fonte: O autor

Na construção da página foram utilizados como componentes duas imagens encontradas na Web, três blocos de notícia da página *globo.com* e uma notícia do portal *Brasil Escola*. Eles foram arrumados em ordem aleatória de modo que formassem uma grade.

4.5.2 Mapeamento dos Alvos

Após a página ser construída, é necessário fazer a identificação das áreas consideradas possíveis alvos para as tarefas. Para esse fim, é preciso o conhecimento das coordenadas na página em que aparecem os alvos para o usuário. Esses dados são salvos em um arquivo *json*, nomeado *data.json*, com a estrutura vista no código abaixo onde *posicaoInicial* contém as coordenadas do canto superior esquerdo da área do alvo e *posicaoFinal* as do canto inferior direito.

```

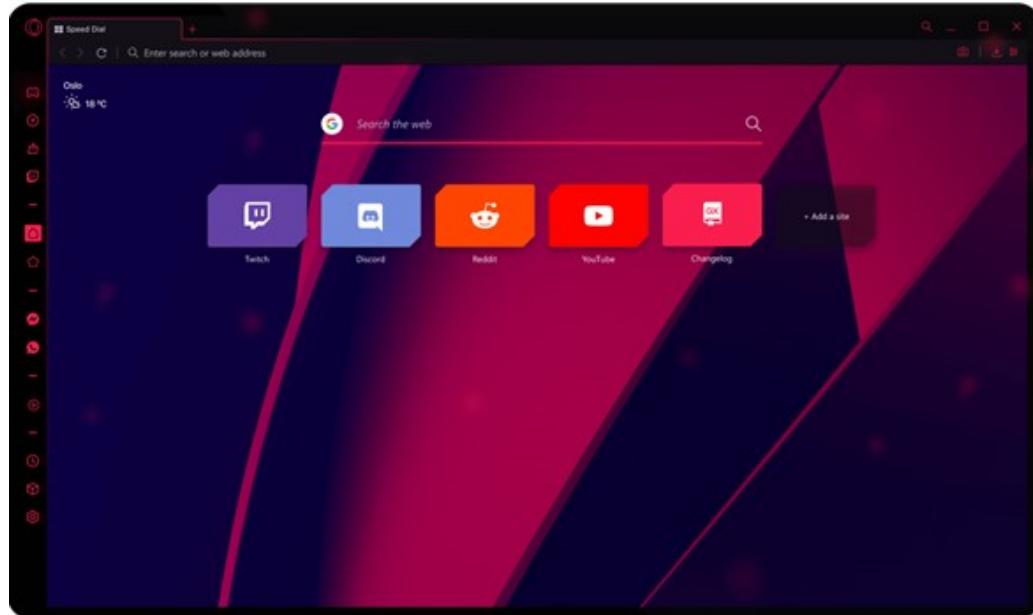
1 {
2     "objetos": [
3         {
4             "posicaoInicial": [50,50],
5             "posicaoFinal": [750,350],
6             "centro": [400,200],
7             "descricao": "Noticia da suspensao do Telegram"
8         },
9         {
10            "posicaoInicial": [770,50],
11            "posicaoFinal": [1400,200],
12            "centro": [1085,125],
13        }
14    ]
15 }
```

```

13     "descricao": "Noticia do jacare na praia"
14 },
15 {
16     "posicaoInicial": [770,200],
17     "posicaoFinal": [1400,350],
18     "centro": [1085,275],
19     "descricao": "Noticia do gol de Igor Paixao"
20 },
21 {
22     "posicaoInicial": [90,400],
23     "posicaoFinal": [540,750],
24     "centro": [315,575],
25     "descricao": "Imagen do cachorro e gato"
26 },
27 {
28     "posicaoInicial": [600,400],
29     "posicaoFinal": [1100,750],
30     "centro": [850,575],
31     "descricao": "Noticia imagens reais e virtuais"
32 },
33 {
34     "posicaoInicial": [1200,500],
35     "posicaoFinal": [1400,700],
36     "centro": [1300,600],
37     "descricao": "Imagen dos pinguins"
38 }
39 ]
40 }
```

Durante a construção do sistema foi utilizado o navegador *Opera GX* com suas configurações de aparência padrão (Figura 41), portanto o tamanho de suas barras superior e lateral são considerados para os cálculos de coordenadas na interface como um todo, sendo necessário fazer acréscimos aos valores das coordenadas informadas, mais especificamente, de 49 no eixo X e de 78 no eixo Y. Sendo assim o uso desse mesmo navegador sempre que o sistema é executado é totalmente indispensável.

Figura 41 – Aparência do navegador Opera GX.



Fonte: O autor

Na página de demonstração (Figura 40) são considerados 6 possíveis alvos, identificados por numeração do 0 ao 5, assim como demonstrado na Figura 42. Os retângulos vermelhos demarcam a área dos alvos a partir das coordenadas informadas no arquivo *data.json*, visto acima.

Figura 42 – Ilustração de como é considerada a numeração e a área dos alvos da página de demonstração.



Fonte: O autor

4.5.3 Execução e Registro

Já estando feita a página e os possíveis alvos com as coordenadas mapeadas, podem ser realizadas as tarefas para a avaliação. Para a execução de uma tarefa o usuário é informado anteriormente do que deve ser buscado na página.

Na prática, sempre que o usuário realiza uma tarefa proposta é feita a calibração e logo após, a tarefa é iniciada. Durante a execução da tarefa todos os olhares do usuário estarão sendo identificados e armazenados temporariamente em tempo real em vetores e em seguida, permanentemente em arquivos CSV, seguindo a estrutura mostrada na Figura 43. Em paralelo, estará sendo salvo em um vetor os tempos (considerando que cada olhar foi feito em uma unidade de tempo após o anterior) de cada olhar. Assim que o usuário entende que atingiu o objetivo da tarefa, ele pressiona um comando no teclado informado previamente.

Figura 43 – Exemplo do armazenamento das coordenadas de olhares do usuário durante a tarefa.

1	eye_x,eye_y
2	1558,251
3	1352,316
4	1117,342
5	1036,354
6	1036,333
7	1057,312
8	1078,312
9	1105,287
10	1150,279
11	1165,284
12	1125,266
13	1086,249

Fonte: O autor

4.6 Identificação

Após o registro das coordenadas de olhar do usuário que foram captadas durante a tarefa, é preciso filtrar e trabalhar esses dados. A partir deles são encontrados a sequência de lugares que tiveram atenção do usuário e com isso quais momentos podem ser considerados fixações. A identificação das fixações, e em consequência das sacadas, acontece em quatro etapas, exibidas na Figura 44.

Figura 44 – Etapas do processo de identificação de fixações e sacadas.



Fonte: O autor

4.6.1 Determinar Coordenadas Válidas

Antes de conseguir identificar as fixações, o primeiro passo é determinar quais das coordenadas do olhar salvas durante a tarefa podem ser consideradas válidas. Isso é necessário porque pode ter sido captado algum momento em que o olhar do usuário não estava na área da página criada, podendo estar em alguma das barras do próprio navegador ou em outros elementos do sistema operacional.

```
1 for i in range(len(posR_df)):  
2     if (((xEye[i] > 41) and (xEye[i] < 1585)) and ((yEye[i] > 71) and (yEye[i] <  
845))):  
3         validXEye.append(xEye[i])  
4         validYEye.append(yEye[i])  
5         validEyesTime.append(timeEye[i])
```

O código acima é o responsável por determinar quais olhares são válidos. Usando como base a interface e o navegador onde o sistema foi desenvolvido, foram determinadas coordenadas limites para a área válida da interface, sendo essas o canto superior esquerdo na coordenada [41, 71] e o canto inferior direito em [1585, 845].

Assim que a coordenada de um olhar é considerada válida, seus valores de x e y são salvos em vetores separados específicos (*validXEye* e *validYEye*, respectivamente), assim como a unidade de tempo definida anteriormente em outro (*validEyesTime*).

4.6.2 Identificar Locais dos Olhares

Com as coordenadas válidas identificadas, o passo seguinte é determinar onde estão localizadas cada uma delas na página. Para isso, dois passos são necessários: criacão

de um vetor de “áreas vazias” e a modificação desse vetor com os locais onde olhares correspondem a alvos.

4.6.2.1 Vetor de Áreas Vazias

Inicialmente, a partir da quantidade de olhares válidos que foram identificados, é criado, com o auxílio da biblioteca *NumPy*, um vetor, nomeado “local”, com esse tamanho, tendo todos seus elementos iguais a “Área Vazia” (linha de código abaixo). Nesse contexto, o termo “Área Vazia” é a representação de qualquer lugar da página onde não esteja um dos alvos definidos. Então mesmo que haja algum objeto num local, se este não for identificado como um alvo, ele será considerado aqui uma “Área Vazia”.

Esse vetor é criado com o objetivo proposital de ser modificado posteriormente, para facilitar a identificação de correspondência dos locais com os alvos, tendo seus elementos “Área Vazia” nesses locais.

```
1 local = np.repeat("Área Vazia", len(validXEye))
```

4.6.2.2 Vetor de Locais

Após a criação do vetor de Áreas Vazias, é iniciada a etapa de modificação deste, fazendo com que alguns elementos sejam substituídos pelos identificadores dos alvos estabelecidos, que podem ser vistos na Figura 42.

```
1 for i in range(len(validYEye)):
2     for j in range(len(centros)):
3         eyePoint = [validXEye[i], validYEye[i]]
4         if ((eyePoint[0] > posicoesIniciais[j][0]) and (eyePoint[0] <
5             posicoesFinais[j][0])):
6             if ((eyePoint[1] > posicoesIniciais[j][1]) and (eyePoint[1] <
7                 posicoesFinais[j][1])):
8                 local[i] = (j)
```

O código acima mostra como é feito o processo. O vetor de olhares válidos terá seus elementos sequencialmente verificados, onde em cada um desses elementos é feita a comparação das suas coordenadas com o posicionamento dos alvos, que foram previamente informados a partir do arquivo data.json.

Caso seja identificado que um olhar esteja dentro da área de um alvo (as bordas são desconsideradas por motivos de possíveis conflitos entre objetos que compartilham bordas), sua posição correspondente no vetor de Áreas Vazias é substituída pelo identificador desse objeto. Tendo sido percorridos todos os olhares, o vetor “local” já foi totalmente

modificado, possuindo agora a informação de todos os locais onde os olhares válidos estão incidindo.

4.6.3 Identificar Focos

Antes de identificar as fixações é preciso filtrar os olhares já obtidos, de forma que olhares isolados que não se repetem ou não se mantêm em áreas próximas sejam desconsiderados futuramente. Para isso são verificadas e registradas temporariamente as existências de focos, seguindo a lógica abaixo:

1. Criação de 4 novos vetores para serem armazenados Posição dos Focos, Local dos Focos, Tempo de Entrada dos Focos e Contagem
2. Percorrer cada elemento em paralelo dos vetores de olhares válidos, locais e tempos
3. Caso seja o primeiro elemento dos olhares válidos, suas coordenadas, local e tempo serão adicionados nos primeiros elementos dos novos vetores de Posição, Local e Tempo de Entrada, respectivamente, e o vetor de Contagem receberá como primeiro elemento, 1.
4. Se não for o primeiro elemento, verificar cinco possíveis situações:
 - a) Se o último olhar foi em um dos alvos e o atual continua nele;
 - b) Se o último olhar foi em um dos objetos e o atual é outro objeto;
 - c) Se o último olhar foi em um dos objetos e o atual é uma área vazia;
 - d) Se o último olhar foi em uma área vazia e o atual é um objeto;
 - e) Se o último foco foi em uma área vazia e o atual continua sendo:
 - Na situação “a”, já que o olhar do usuário continua no alvo, é considerado que o foco ainda é o mesmo, portanto o único valor modificado é o do último valor adicionado ao vetor de Contagem, sendo adicionado 1 ao valor prévio.
 - Nas situações “b”, “c” e “d”, como os locais entre o último elemento verificado e o atual não são mais os mesmos, é necessário adicionar novos elementos em todos os quatro novos vetores, seguindo a mesma lógica de adição dos primeiros elementos.
 - Na situação “e”, já que a Área Vazia da interface pode ser extremamente ampla, são consideradas duas possibilidades:
 - Se a distância entre as coordenadas do último elemento verificado e do atual for menor que um limite estabelecido (no caso 80), então é entendido que o foco continua o mesmo e assim o único valor modificado é

o do último valor adicionado ao vetor de Contagem, sendo adicionado 1 ao valor prévio.

- Se a distância for maior que o limite, então é considerado um novo foco e assim é necessário adicionar novos elementos em todos os quatro novos vetores, seguindo a mesma lógica de adição dos primeiros elementos.

Após esse processo, existem 4 novos vetores a serem considerados:

- Vetor de Posição dos Focos: possui as coordenadas de todos os focos encontrados.
- Vetor de Local dos Focos: possui o local (indicador de objeto ou Área Vazia) de todos os focos encontrados
- Vetor de Tempo de Entrada dos Focos: possui o momento de início de cada um dos focos encontrados.
- Vetor de Contagem: possui a quantidade de tempo que o usuário permaneceu em cada um dos focos.

Para facilitar os processos posteriores, é criado um vetor que passa a englobar todos esses quatro vetores, permitindo que cada elemento possua as quatro informações de cada foco (código abaixo).

```
1 focos = []
2 for i in range(len(focoPos)):
3     focos.append([focoPos[i], focoLocal[i], focoCount[i], focoTime[i]])
```

4.6.4 Identificação das Fixações

A partir da produção do vetor de focos, faltam poucos passos para que as fixações sejam identificadas. A verificação de focos que devem ser desconsiderados para o cálculo e a análise dos focos restantes.

4.6.4.1 Elementos desconsiderados

Alguns dos elementos presentes no vetor de focos precisam ser desconsiderados. Isso ocorre porque eles não existem por tempo suficiente para realmente serem considerados pontos de foco do usuário. Essa situação acontece normalmente porque o sistema captou o olhar no momento em que o usuário estava percorrendo o caminho entre os verdadeiros focos nos alvos, ou foi captado algum tipo de desvio ínfimo, possivelmente involuntário, enquanto o usuário estava focando um alvo.

Como mostrado no código abaixo, para verificar quais são os elementos que precisam ser eliminados, o vetor de focos é percorrido e a contagem de permanência de cada elemento (`focos[i][2]`) é verificada, sendo apenas os elementos que a possuem com valor maior que 3 armazenados em um novo vetor que continuará sendo trabalhado no próximo processo.

```

1 usefulFocus = []
2     for i in range(len(focos)):
3         if (focos[i][2] > 3):
4             usefulFocus.append(focos[i])

```

4.6.4.2 Aquisição das Fixações

Com os focos úteis definidos, o próximo passo é a aquisição da lista de fixações. Para obter essa lista é preciso analisar todos os elementos do novo vetor de focos. O código abaixo mostra como é feito.

```

1 fixations = []
2     for i in range (len(usefulFocus)):
3         if (i == 0):
4             fixations.append(usefulFocus[i])
5         else:
6             if ((usefulFocus[i-1] [1] == usefulFocus[i] [1]) and (usefulFocus[i] [1]
7 != 'Área Vazia')):
8                 fixations[-1] [2] = usefulFocus[i-1] [2] + usefulFocus[i] [2]
9             else:
10                fixations.append(usefulFocus[i])

```

Primeiramente é criado um vetor responsável por armazenar a lista de fixações obtidas. A partir disso o vetor de focos úteis (`usefulFocus`) tem seus elementos analisados.

- a Caso seja o primeiro elemento do vetor de focos úteis, ele é automaticamente armazenado no vetor de fixações.
- b Caso não seja o primeiro elemento é feita uma análise:
 - Se o último elemento percorrido e o atual estão no mesmo local e esse local não é uma Área Vazia, ambos focos são considerados parte de uma mesma fixação e a única modificação feita é que a contagem de permanência do último elemento tem adicionada ao seu valor a contagem do elemento atual.
 - Caso as condições do item acima não se cumpram, o elemento atual é adicionado como uma nova entrada no vetor de fixações.

4.6.5 Identificação das Sacadas

Com as fixações já obtidas, a tarefa de identificar as sacadas é extremamente simples, já que elas são os caminhos entre as fixações, ou seja, para trabalhar com as sacadas é necessário trabalhar com a sequência de fixações e considerar que o início e o fim de uma sacada são as coordenadas de duas fixações em sequência.

4.7 Verificação

Na etapa da verificação é onde serão calculados quais são os possíveis problemas de usabilidade da interface. Para isso são necessários três processos: o cálculo das métricas de avaliação propostas, a comparação com o resultado ideal e a identificação dos possíveis problemas da interface.

4.7.1 Cálculo das Métricas

São propostas nove métricas para a avaliação da usabilidade da interface com base em resultados de rastreamento ocular: Momento da primeira fixação no alvo, número total de fixações, densidade espacial das fixações, número total de sacadas, tamanho médio das sacadas, duração total das sacadas, razão da duração entre fixações e sacadas, largura total do scanpath e duração total do scanpath. Essas métricas são obtidas através da análise das fixações e sacadas identificadas.

4.7.1.1 Momento da Primeira Fixação no Alvo

Como falado anteriormente, antes que a tarefa seja executada é definido um objetivo para ela. Esse objetivo consiste na identificação de algum alvo específico, onde cada tarefa tem seu alvo. É a primeira fixação nesse alvo em questão que precisa ser encontrada.

Para fazer isso, como ocorre no código abaixo, é preciso simplesmente percorrer o vetor de fixações, de modo que na primeira vez que seja encontrada uma fixação que ocorreu no local que é o alvo da tarefa, seu tempo de entrada será salvo em uma variável para posteriormente ser incluído em um arquivo, e o loop no vetor será interrompido.

```
1 for i in range(len(fixations)):
2     if (fixations[i][1] == target):
3         ft = fixations[i][3]
4         break
```

4.7.1.2 Número Total de Fixações

Para encontrar o número total de fixações identificadas é preciso somente calcular o comprimento do vetor de fixações obtido anteriormente. No código, isso é feito com o uso de apenas a linha abaixo:

```
1 totalFixationsCount = len(fixations)
```

4.7.1.3 Densidade Espacial das Fixações

A densidade espacial das fixações é um valor que representa a proporção de fixações dentro do alvo da tarefa para o total identificadas em toda página. O número total de fixações para fazer o cálculo já foi obtido como valor de uma métrica anteriormente, já o número de fixações dentro do alvo pode ser encontrado usando um loop parecido com o usado para encontrar a primeira fixação no alvo (código abaixo).

```
1 for i in range(len(fixations)):
2     if (target == fixations[i][1]):
3         fixationsIn += 1
4
5 fixationsDensity = fixationsIn/totalFixationsCount
```

As diferenças são que o loop não é interrompido quando encontra uma fixação dentro do alvo e cada vez que uma dessas fixações é encontrada uma variável que faz o papel de contador tem seu valor acrescido em 1.

4.7.1.4 Número Total de Sacadas

Considerando que uma sacada ocorre sempre entre duas fixações sequenciais, o número total de sacadas é o valor do número total de fixações menos um. Isso porque a última fixação não pode ser ponto de partida de uma nova sacada, já que não haveria lugar para ela terminar.

```
1 saccadesCount = (len(fixations)-1)
```

4.7.1.5 Tamanho Médio das Sacadas

O tamanho médio das sacadas é obtido a partir da divisão do valor da soma dos tamanhos das sacadas pelo número total de sacadas, já obtido anteriormente.

```
1 for i in range(saccadesCount):
2     sizeSaccades += math.dist(fixations[i][0], fixations[i+1][0])
```

```

3
4 saccadesMedia = sizeSaccades/saccadesCount

```

A soma dos tamanhos das sacadas é obtida a partir da soma das distâncias entre as posições das fixações que ocorrem em sequência.

4.7.1.6 Duração Total das Sacadas

A duração total das sacadas é obtida a partir da soma das diferenças dos tempos identificados como momento de entrada de uma fixação e o momento de saída da anterior.

```

1 for i in range(saccadesCount):
2     saccadesDuration += (fixations[i+1][3] - (fixations[i][2] + fixations[i]
] [3]))

```

O momento de entrada de uma fixação já está representado como o quarto elemento no registro de cada fixação no vetor. Já o tempo de saída necessita ser calculado, sendo igual ao momento de entrada na fixação adicionado ao valor da contagem de permanência desta, que é encontrado como o terceiro elemento de cada registro.

4.7.1.7 Razão da Duração entre Fixações e Sacadas

Como o próprio nome diz, o valor dessa razão é encontrado dividindo a duração das fixações pela duração das sacadas, esse último já tendo sido obtido.

Para obter a duração total das fixações é necessário somente somar a contagem de permanência de todas as fixações.

4.7.1.8 Largura Total do Scanpath

O *scanpath* sendo o caminho de fixações percorrido pelo usuário tem seu tamanho totalmente concentrado nas sacadas já que as fixações são pontos, ou seja, o tamanho do scanpath é o resultado da soma dos tamanhos das sacadas, o que já foi obtido no caminho para obter o tamanho médio das sacadas e ficou armazenado na variável *sizeSaccades*.

4.7.1.9 Duração Total do Scanpath

A duração total do *scanpath* pode logicamente ser obtida pela diferença do momento de saída da última fixação pelo momento de entrada da primeira.

```

1 startMoment = fixations[0][3]
2 endMoment = fixations[-1][3] + fixations[-1][2]

```

```

3
4 scanpathDuration = endMoment - startMoment

```

4.7.1.10 Registro

Após o cálculo de todas as métricas, estas serão salvas em um arquivo específico para a tarefa onde foram obtidas. Ou seja, toda vez que uma tarefa é executada, as métricas obtidas são salvas em um arquivo exclusivo da tarefa e todas as tarefas tem seu próprio arquivo. A Figura 45 abaixo mostra o exemplo de um arquivo que possui quatro linhas de métricas obtidas, portanto, a tarefa correspondente a esse arquivo foi executada quatro vezes.

Figura 45 – Exemplo de arquivo de armazenamento das métricas de uma tarefa.

```

1 First Fixation on Target,Total Number of Fixations,Density of Fixations,Number of Saccades,Average Scanpath Duration
2 73,6,0.1666666666666666,5,345.8273286964899,21,2.6666666666666665,1729.1366434824495,77
3 41,3,0.3333333333333333,2,589.8438928922682,6,6,0,1179.6877857845363,42
4 23,2,0.5,1,416.39404414568656,1,29,0,416.39404414568656,30
5 28,3,0.3333333333333333,2,404.79591546678444,1,34,0,809.5918309335689,35
6

```

Fonte: O autor

4.7.2 Comparação com o Resultado Ideal

Buscando entender como estão as métricas obtidas é preciso compará-las com os valores ideais esperados pela avaliação para cada tarefa. Esses valores assim como outros dados são informados através de um arquivo *JSON*.

Para fazer essa comparação, a tabela de cada tarefa é relacionada apenas com a parte do arquivo de valores ideais que corresponde àquela mesma tarefa. O cálculo é feito entre os valores ideais das métricas e a média dos valores daquela mesma métrica na tabela correspondente, consequentemente obtendo valores de proporções entre eles.

Como ocorre no código abaixo, as proporções resultantes da comparação da Primeira fixação no alvo, número de fixações, densidade de fixações, número de sacadas, tamanho médio das sacadas, duração total das sacadas, razão da duração de fixações e sacadas, tamanho do *scanpath* e duração do *scanpath* são armazenadas nas variáveis *ftProp*, *fixNumProp*, *denProp*, *sacProp*, *sacSizeProp*, *sacDurProp*, *ratioProp*, *scanProp* e *scanDurProp*, respectivamente, e então salvas em um novo vetor de vetores (*comp*).

```

1 comp = []
2     for i in ideal['tarefas']:
3         for j in range(len(measureMeans)):
4             if (i["id"] == measureMeans[j][0]):

```

```

5         idProp = i["id"]
6         ftProp = measureMeans[j][1]/ i["First Fixation on Target"]
7         fixNumProp = measureMeans[j][2]/ i["Total Number of Fixations"]
8         denProp = measureMeans[j][3]/ i["Density of Fixations"]
9         sacProp = measureMeans[j][4]/ i["Number of Saccades"]
10        sacSizeProp = measureMeans[j][5]/ i["Average Saccades Size"]
11        sacDurProp = measureMeans[j][6]/ i["Total Duration of Saccades"]
12        ratioProp = measureMeans[j][7]/ i["Ratio of Fixations/Saccades Duration"
13
14        scanProp = measureMeans[j][8]/ i["Scanpath Size"]
15        scanDurProp = measureMeans[j][9]/ i["Scanpath Duration"]
16        comp.append([idProp,ftProp, fixNumProp, denProp, sacProp, sacSizeProp,
17                      sacDurProp, ratioProp, scanProp, scanDurProp])

```

4.7.3 Identificação de Problemas

O vetor de proporções gerado é usado como base de análise dos problemas de usabilidade. É utilizada uma função que avalia cada métrica de cada tarefa em relação a uma proporção máxima ou mínima, definida em código, para elas.

O código abaixo mostra parte da função responsável pelo processo (não é necessário a apresentação dela como um todo por se tratar apenas da repetição da mesma lógica ao seu decorrer), onde são criados três vetores: *resultsList*, *problemsList* e *problemsListTemp*, dentro do *for*.

A estrutura de repetição principal faz com que cada linha do vetor de proporções seja percorrida. Em cada linha, seus elementos serão analisados. O primeiro elemento de cada linha é a identificação da tarefa que ela representa. Essa identificação é salva também como primeiro elemento dos vetores *resultsList* e *problemsListTemp*.

O segundo elemento é o valor da proporção entre o valor ideal e a média obtida nas tarefas do Tempo da Primeira Fixação no Alvo. Esse valor tem a característica de quanto maior for, pior. Portanto, é definido um limite máximo aceitável para essa proporção, no caso 1. Então, toda vez que essa proporção é maior que 1, o sistema entende como um possível problema e adiciona uma mensagem indicando problema nessa métrica na tarefa analisada no vetor *resultsList* e o elemento de valor 1 ao *problemsListTemp*. Caso o valor não ultrapasse o limite, o vetor *problemsListTemp* adiciona o elemento 0.

```

1 def results(comp):
2
3     resultsList = []
4     problemsList = []
5
6     for i in range(len(comp)):

```

```

7     problemsListTemp = []
8     resultsList.append("Tarefa " + str(comp[i][0]))
9     problemsListTemp.append("Tarefa " + str(comp[i][0]))
10
11
12     if (comp[i][1] > 1):
13         resultsList.append("O tempo da primeira fixacao no alvo foi em media "
14         + str(round(comp[i][1],5)) + " vezes maior que o esperado")
15         problemsListTemp.append(1)
16     else:
17         problemsListTemp.append(0)

```

Ao final das métricas de cada tarefa, o vetor *problemsList*, recebe o conteúdo do *problemsListTemp*, se tornando ao final da função um vetor de vetores.

Esse processo é repetido para cada métrica em todas as tarefas, e ao final da análise de todas o conteúdo do vetor *resultsList* é salvo em um arquivo CSV que assume o formato mostrado na Figura 46 abaixo.

Figura 46 – Exemplo do arquivo de armazenamento que recebe o conteúdo de *resultsList*.

```

Tarefa 1
O tempo da primeira fixação no alvo foi em média 2.8 vezes maior que o esperado
O tamanho do scanpath no alvo foi em média 1.80691 vezes maior que o esperado
A duração do scanpath no alvo foi em média 2.1 vezes maior que o esperado
"
"
Tarefa 2
O tempo da primeira fixação no alvo foi em média 1.7 vezes maior que o esperado
O tamanho médio das sacadas no alvo foi em média 0.95159 vezes menor que o esperado
A duração total das sacadas no alvo foi em média 1.33333 vezes maior que o esperado
A razão da duração entre fixações e sacadas no alvo foi em média 0.8125 vezes menor que o esperado
"
"
Tarefa 3
A densidade de fixações no alvo foi em média 0.5 vezes menor que o esperado
O tamanho do scanpath no alvo foi em média 1.03725 vezes maior que o esperado
A duração do scanpath no alvo foi em média 1.4 vezes maior que o esperado
"
"

```

Fonte: O autor

4.7.4 Análise de Usabilidade

Nessa etapa o alvo de análise é o vetor *problemsList*, construído previamente. Ele é basicamente construído de forma que cada linha representa uma tarefa e seus elementos são, com exceção do primeiro que é o identificador da tarefa da linha, zeros ou uns. Todo elemento que é 0 é o indicador de um possível problema.

O primeiro passo aqui é a criação de quatro vetores, um para cada dimensão

da usabilidade onde podem ser encontrados problemas: Eficiência, Eficácia, Satisfação e Aprendizagem.

Ao percorrer o problemsList, toda vez que a métrica de uma tarefa é identificada como tendo um problema, os respectivos vetores de dimensões da usabilidade relacionados a ela têm adicionados um indicativo de problema.

O código abaixo, como exemplo, é uma parte da função que faz referência a métrica do Número Total de Fixações. Caso o valor de seu elemento seja maior que 0, ou seja, 1, é adicionada uma frase aos vetores de Eficiência, Satisfação e Aprendizagem (*effectivenessProblems*, *satisfactionProblems* e *learningProblems*, respectivamente, não tendo mudança no vetor de eficiência, *efficiencyProblems*).

```

1 if problemsList[i][2] > 0:
2     p = "Número Total de Fixações na " + str(problemsList[i][0])
3     effectivenessProblems.append(p)
4     satisfactionProblems.append(p)
5     learningProblems.append(p)

```

Após as repetições, os quatro vetores de problemas são englobados em um só vetor, o *usabilityProblemsList* (código abaixo).

```

1 usabilityProblemsList = []
2     usabilityProblemsList.append(efficiencyProblems)
3     usabilityProblemsList.append(effectivenessProblems)
4     usabilityProblemsList.append(satisfactionProblems)
5     usabilityProblemsList.append(learningProblems)

```

Esse novo vetor é então salvo como um arquivo, mais especificamente o *usability_problems.csv*.

```

1 problemsFile = pd.DataFrame(usabilityProblemsList, index=None)
2 problemsFile.to_csv("usability_problems.csv", index=False)

```

4.8 Representações

Além da produção de arquivos com dados ao longo de todos os processos, o sistema após cada execução de tarefa exibe duas formas de representação dos movimentos do olhar do usuário que foram capturados: mapa de calor e scanpath.

4.8.1 Mapa de Calor

Para a produção do mapa de calor após a tarefa, são considerados os pixels válidos identificados na seção 4.6.1. A partir deles são calculados pesos para eles próprios e aqueles próximos, fazendo assim, com que toda área ao seu redor tenha importância. Para fazer o mapa três etapas sequenciais são necessárias: Atribuição de pesos, definição de limites e aplicação de cores.

4.8.1.1 Atribuição de Pesos

Sabendo quais são as posições dos pixels considerados, é obtida também a quantidade de pixels total da interface a partir da multiplicação das dimensões da sua resolução (Largura e Altura).

Com esses dois dados é criada uma estrutura de repetição que para toda coordenada válida, todos os pixels da interface com uma distância de até 30 *pixels* terão adicionados ao seu peso um valor decrescente proporcional à distância. O primeiro passo é a criação de um vetor de zeros, de tamanho igual a quantidade total de *pixels* da interface. Esse vetor é onde os pesos são armazenados.

O código abaixo, é o responsável pela distribuição desses pesos, ele faz basicamente o que foi falado logo acima. Em mais detalhes:

- 1) O vetor *mapArray* é percorrido, para cada posição sua, o vetor de posições válidas (*arr*) é percorrido.
- 2) Caso os valores (posições) sejam os mesmos, a coordenada do pixel dessa posição é salva em uma variável (*ptCoord*), que passa a servir como um pivô para iteração.
- 3) O *mapArray* é novamente percorrido, desta vez calculando a distância entre as coordenadas de cada uma de suas posições com as coordenadas de *ptCoord*. Sempre que for identificada uma distância menor que 30. O valor da posição em *mapArray* tem adicionado ao seu valor, a diferença entre 50 e a distância encontrada.

```

1 # mapArray = Vetor de zeros com tamanho da quantidade de pixels da tela
2 # arr = Vetor com posies de pixels vlidos
3     for arrayPos in range(mapArray.size):
4         for dfPos in range(len(arr)):
5             if (arrayPos == arr[dfPos]) :
6                 ptCoord = coordPos(arrayPos)
7                 for arrayPosComp in range(mapArray.size):
8                     coordComp = coordPos(arrayPosComp)
9                     distComp = math.dist(ptCoord,coordComp)

```

```

10         if (distComp < 30):
11             mapArray[arrayPosComp] += (50 - distComp)

```

Ou seja, a posição do próprio *ptCoord* da vez que tem distância 0 para ele mesmo tem adicionado 50 ao seu valor e o pixel que tiver a 29,999 pixels de distância terá adicionado em sua posição o valor de 0,001.

4.8.1.2 Definição de Limites

Após o vetor *mapArray* ter sido preenchido pelos pesos das coordenadas válidas e suas proximidades, é preciso fazer uma análise bem simples dos seus valores com a ajuda da biblioteca *NumPy* e o objetivo simples de encontrar qual o maior peso que uma posição do vetor possui (*mapArrayMAX*), e qual é o menor (*mapArrayMIN*), que normalmente é 0.

```

1 mapArrayMAX = npamax(mapArray)
2 mapArrayMIN = npamin(mapArray)

```

Com esses dois, é obtida diferença máxima de pesos que pode existir dentro do vetor a partir da diferença do máximo pelo mínimo. Esse valor é armazenado na variável *weightRange*.

```
1 weightRange = mapArrayMAX - mapArrayMIN
```

É a partir da *weightRange* que são criados quatro limites: *range10*, *range40*, *range70* e *range90*.

```

1 range10 = mapArrayMAX - (weightRange*0.10)
2 range40 = mapArrayMAX - (weightRange*0.40)
3 range70 = mapArrayMAX - (weightRange*0.70)
4 range90 = mapArrayMAX - (weightRange*0.90)

```

Explicando cada um, e consequentemente o que ocorre no código acima:

- *range10*: É o valor que representa o limite que separa os 10% maiores pesos possíveis de serem alcançados.
- *range40*: É o valor que representa o limite que separa os 40% maiores pesos possíveis de serem alcançados.
- *range70*: É o valor que representa o limite que separa os 70% maiores pesos possíveis de serem alcançados.
- *range90*: É o valor que representa o limite que separa os 90% maiores pesos possíveis de serem alcançados.

4.8.1.3 Aplicação de Cores

Com os limites definidos, é iniciada a etapa de coloração. Primeiramente é feita uma captura de tela, que ocorre logo após ao fim da tarefa. Essa captura é feita e salva com o auxílio da biblioteca *pyautogui*.

Para ser trabalhada, a imagem precisa ter seu sistema de cores convertido pela *OpenCV* (código abaixo), para um que possa ser usado por ela, no caso BGR.

```
1 image = cv2.cvtColor(np.array(imagePrint),cv2.COLOR_RGB2BGR)
```

Para a aplicação das cores são feitas quatro repetições que percorrem todos os pixels da interface em paralelo com *mapArray*, cada uma responsável por um grupo de valores e por uma cor.

1) Primeiro Grupo

O primeiro grupo é responsável por identificar as posições que tem peso respectivo em *mapArray* maior que *range90* e menor ou igual que *range70*. Sempre que essa condição é atendida, é desenhado, com o uso da *OpenCV*, um círculo com raio de 35 pixels centrado no pixel identificado e preenchido pela cor *rgb(246,250,130)*, que pode ser visualizada abaixo do código.

```
1 for p in range (screenSizeX):
2     for q in range (screenSizeY):
3         if ((mapArray[screenSizeX*q + p] > range90) and (mapArray[screenSizeX*q
+ p] <= range70)):
4             image = cv2.circle(image, (p,q), 35, (130,250,246), -1)
```



2) Segundo Grupo

O segundo grupo é responsável por identificar as posições que tem peso respectivo em *mapArray* maior que *range70* e menor ou igual que *range40*. Sempre que essa condição é atendida, é desenhado, com o uso da *OpenCV*, um círculo com raio de 35 pixels centrado no pixel identificado e preenchido pela cor *rgb(255,255,0)*, que pode ser visualizada abaixo do código.

```

1 for p in range (screenSizeX):
2     for q in range (screenSizeY):
3         if ((mapArray[screenSizeX*q + p] > range70) and (mapArray[screenSizeX*q
+ p] <= range40)):
4             image = cv2.circle(image, (p,q), 35, (0,255,255), -1)

```



3) Terceiro Grupo

O terceiro grupo é responsável por identificar as posições que tem peso respectivo em *mapArray* maior que *range40* e menor ou igual que *range10*. Sempre que essa condição é atendida, é desenhado, com o uso da *OpenCV*, um círculo com raio de 25 pixels centrado no pixel identificado e preenchido pela cor *rgb(255,140,0)*, que pode ser visualizada abaixo do código.

```

1 for p in range (screenSizeX):
2     for q in range (screenSizeY):
3         if ((mapArray[screenSizeX*q + p] > range40) and (mapArray[screenSizeX*q
+ p] <= range10)):
4             image = cv2.circle(image, (p,q), 25, (0,140,255), -1)

```



4) Quarto Grupo

O quarto e último grupo é responsável por identificar as posições que tem peso respectivo em *mapArray* maior que *range10*. Sempre que essa condição é atendida, é desenhado, com o uso da *OpenCV*, um círculo com raio de 15 pixels centrado no pixel identificado e preenchido pela cor *rgb(255,0,0)*, que pode ser visualizada abaixo do código.

```

1 for p in range (screenSizeX):
2     for q in range (screenSizeY):

```

```

3         if ((mapArray[screenSizeX*q + p]) > range10):
4             image = cv2.circle(image, (p,q), 15, (0,0,255), -1)

```



Após a aplicação das cores na imagem, são feitos dois processos: o primeiro para deixar as cores translúcidas e assim permitir que conteúdo atrás delas ainda continue visível, e o segundo é um processo de redimensionamento para que ela não ocupe toda a interface. Esses processos foram aplicados usando as respectivas linhas do código abaixo.

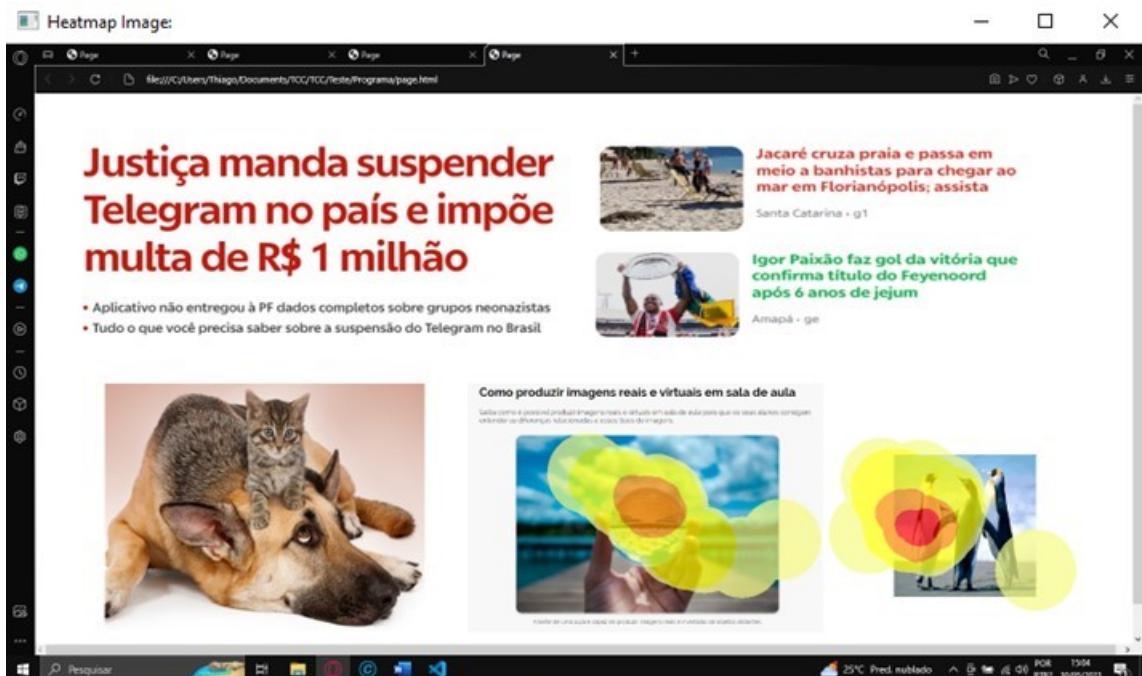
```

1 image_new = cv2.addWeighted(overlay, 0.3, image, 1 - 0.3, 0)
2 image_map = cv2.resize(image_new, (int(screenSizeX/2), int(screenSizeY/2)))

```

Após todos os processos pós-tarefa serem realizados é exibida a imagem da interface com o mapa de calor, sendo algo como a Figura 47 abaixo:

Figura 47 – Exemplo de representação de mapa de calor feito pelo sistema.



Fonte: O autor

4.8.2 Scanpath

O processo de produção do scanpath também ocorre após cada tarefa, porém diferentemente do que foi feito no Mapa de Calor, são consideradas como coordenadas apenas as presentes no vetor de fixações criado anteriormente.

Assim como o mapa de calor, o *scanpath* também utiliza a imagem de captura de tela convertida para BGR pelo *OpenCV*. Nela serão produzidas respectivamente as representações das sacadas e das fixações do usuário.

4.8.2.1 Desenhar Sacadas

A primeira parte a ser desenhada são as sacadas, para que elas não apareçam por cima das fixações. As sacadas são os caminhos entre as fixações em sequência, portanto, podem ser representadas por linhas.

A função abaixo é a responsável por esse processo. Nela o vetor de fixações é percorrido, sendo as coordenadas da primeira fixação salva na variável *startPoint* e as da fixação que vem na sequência sendo salvas em *endPoint*.

Toda sacada precisa de um ponto de partida (*startPoint*) e de um de chegada (*endPoint*), por isso esse processo é repetido até que a penúltima e a última fixações sejam respectivamente *startPoint* e *endPoint*.

Em cada repetição é usado um método da *OpenCV* para desenhar linhas (*cv2.line*) em imagens, sendo necessário informar como parâmetro: a imagem onde será desenhada, o ponto de partida, o ponto de chegada, a cor e a espessura da linha. No caso das cores, foram definidos anteriormente a função valores aleatórios para os canais de azul, verde e vermelho (respectivamente, *blueScale*, *greenScale* e *redScale*)

```

1 def drawLines(i, blueScale, greenScale, redScale):
2     for i in range(len(fixation)-1):
3         startPoint = [fixation[i][0][0], fixation[i][0][1]]
4         endPoint = [fixation[i+1][0][0], fixation[i+1][0][1]]
5         cv2.line(image,startPoint,endPoint,(blueScale,greenScale,redScale),2)

```

4.8.2.2 Desenhar Fixações

Para representar as fixações é feito um processo mais simples que o das sacadas, sendo também necessária uma estrutura de repetição que percorre o vetor de fixações para conseguir suas coordenadas.

Em cada repetição, ou seja, em cada fixação, as três estruturas do código abaixo são executadas.

- A primeira desenha uma circunferência preenchida de cor cinza muito clara, centralizado na fixação e de raio com tamanho igual a 5 vezes o valor do contador de permanência da fixação.
- A segunda desenha uma circunferência sem preenchimento de cor aleatória (do mesmo modo que as sacadas), centralizado na circunferência e de raio com tamanho igual a 5 vezes o valor do contador de permanência da fixação. Funciona na prática como uma borda da primeira circunferência.
- A terceira estrutura escreve o identificador de posição da fixação na sequência em que foram feitas. O identificador é iniciado em 0, para a primeira fixação e sempre é escrito na cor vermelha.

```

1 image = cv2.circle(image, (fixation[i][0][0], fixation[i][0][1]), (5*fixation[i]
    ][2]), (200,255,255), -1)
2 image = cv2.circle(image, (fixation[i][0][0], fixation[i][0][1]), (5*fixation[i]
    ][2]), (blueScale,greenScale,redScale), 5)
3 image = cv2.putText(image, str(i), (fixation[i][0][0], fixation[i][0][1]), cv2.
    FONT_HERSHEY_SIMPLEX, 3, (0,0,255), 5 )

```

4.9 Exibição de Resultados ao Avaliador

Sempre que o avaliador desejar, podem ser visualizados os resultados obtidos com a análise dos dados. Para isso é executado o arquivo *seeResults.py* que está dentro da pasta Resultados.

Esse arquivo, trabalha com o conteúdo apresentado em duas janelas distintas: uma com os resultados dos problemas em cada tarefa (Figura 48) e a outra que é a responsável por dizer quais os possíveis problemas de usabilidade da interface como um todo e exibe os indícios de cada problema (Figura 49).

Essas informações são obtidas a partir dos arquivos criados nas seções 4.7.3 e 4.7.4, *results.csv* e *usability_problems.csv*, respectivamente.

```

1 results = pd.read_csv("../results.csv")
2 usabilityProblems = pd.read_csv("../usability_problems.csv")

```

Na primeira janela são simplesmente exibidos os dados do arquivo recebido por *results*. Nas figuras abaixo, *listbox* é a responsável pelo conteúdo da janela da Figura 48 e adiciona todas as linhas para exibição.

```

1 for i in range(len(results)):
2     listbox.insert(END, results.iloc[i, 0])

```

Para a segunda janela, primeiramente os dados em *usabilityProblems* são separados em quatro vetores, um para cada linha, que representa cada dimensão da usabilidade a ser considerada. Respectivamente, eficiência, eficácia, satisfação e aprendizagem (código abaixo).

```

1 for i in range (len(usabilityProblems.axes[1])):
2     efficiencyProblems.append(usabilityProblems.iloc[0,i])
3     effectivenessProblems.append(usabilityProblems.iloc[1,i])
4     satisfactionProblems.append(usabilityProblems.iloc[2,i])
5     learningProblems.append(usabilityProblems.iloc[3,i])

```

É então verificado se os vetores estão vazios (sem problemas) ou não. Caso tenham conteúdo, a dimensão que representa será adicionada em uma lista, o que representa que esta dimensão tem indícios de estar com problemas.

```

1 usabilityP = [0,0,0,0]
2 if (len(efficiencyProblems) > 0):
3     usabilityP[0] = "Eficiencia: Preciso"
4 if (len(effectivenessProblems) > 0):
5     usabilityP[1] = 'Eficacia: Apresentao e Navegao '
6 if (len(satisfactionProblems) > 0):
7     usabilityP[2] = "Satisfao: Contedo e Atratividade"
8 if (len(learningProblems) > 0):
9     usabilityP[3] = "Aprendizagem: Simplicidade"

```

O conteúdo dessa lista é então adicionado na nova janela, logo no começo.

```

1 listbox2.insert(END, "A pgina apresentou indcios de possveis problemas de
    usabilidade em:")
2 for i in range (len(usabilityP)):
3     if usabilityP != 0:
4         listbox2.insert(END, usabilityP[i])

```

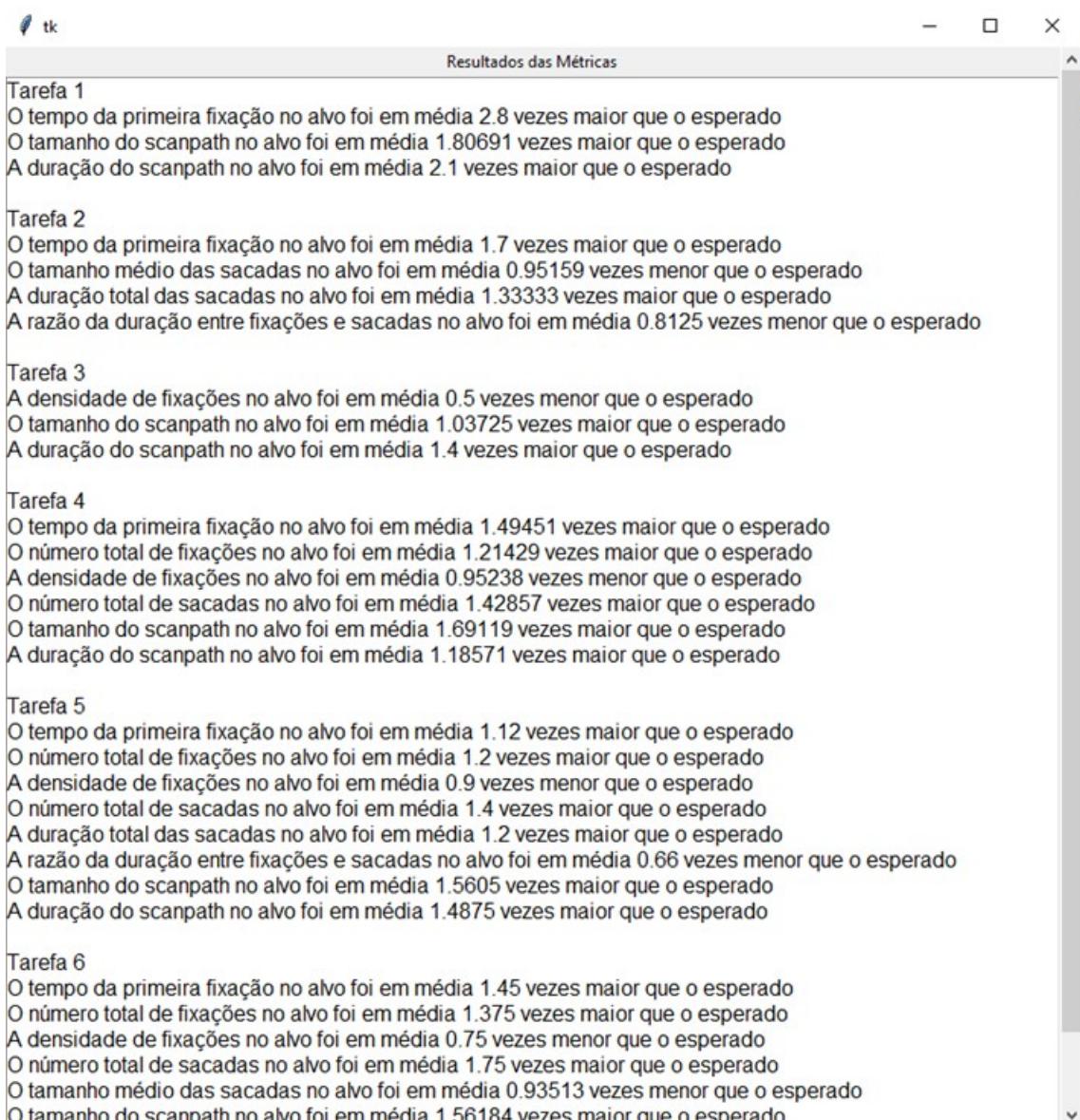
Por último os indícios de problemas de cada vetor de dimensão são adicionados usando a lógica do código abaixo, responsável pelo vetor de eficiência.

```

1 if (len(efficiencyProblems) > 0):
2     listbox2.insert(END, "\n")
3     listbox2.insert(END, "Eficiencia: Preciso :")
4     listbox2.insert(END, "\n")
5     for i in range(len(efficiencyProblems)):
6         if (pd.notnull(efficiencyProblems[i])):
7             listbox2.insert(END, efficiencyProblems[i])

```

Figura 48 – Exemplo da janela que apresenta os resultados problemáticos das métricas em cada tarefa



Fonte: O autor

Figura 49 – Exemplo da janela que apresenta os resultados da avaliação de usabilidade e os indícios que os justificam



Fonte: O autor

Capítulo 5: Resultados

Após o desenvolvimento do sistema, foi proposto o seu uso a 5 pessoas voluntárias de diferentes idades em mais de uma máquina como forma de validar suas etapas e consequentemente funcionamento deste. Para isso, foram escolhidas como objeto de análise duas páginas que já foram alvo de um estudo prévio, assim os resultados do estudo e o obtido pelo sistema podem ser comparados.

As páginas usadas, são ambas do *Boletim de Ciências Geodésicas*, porém uma está hospedada pela *SciELO* (Figura 50) e a outra pela *UFPR* (Figura 51). As duas possuem o mesmo objetivo, mas são construídas e tem seus elementos organizados de formas totalmente diferentes. A página na *UFPR* por exemplo, tem um foco maior em textos do que imagens.

Figura 50 – Página do Boletim de Ciências Geodésicas hospedada pela SciELO

The screenshot shows the Scielo website for the **Boletim de Ciências Geodésicas**. At the top, there is the Scielo 25 logo, the journal's name in Portuguese and English, and publication details: **Publicação de: Universidade Federal do Paraná**, **Área: Ciências Exatas E Da Terra**, **Versão impressa ISSN: 1413-4853 Versão on-line ISSN: 1982-2170**. The top right features language options (ESPAÑOL, ENGLISH), manuscript submission links, and editorial board information. Below the header is a navigation bar with links for **home do periódico**, **todos os números**, **« número anterior**, **número atual**, **nächste »**, **buscar**, and **métricas**. A social sharing section follows. The main content area is titled **Nossa Missão** with the text: "Publicar trabalhos inéditos na área de Ciências Geodésicas e afins. Número mais recente". It displays the volume and number: **Boletim de Ciências Geodésicas, Volume: 29, Número: 2, Publicado: 2023**. Below this is a section titled **Notícias** containing seven news items with thumbnails, dates, titles, and brief descriptions.

Thumbnail	Date	Title	Description
	2023-06-02 19:55:34	Reprodução e replicação na pesquisa científica – parte 2	Nesta segunda nota sobre o assunto, abordaremos as diretrizes propostas em 2019 pela NASEM. Analisaremos como a replicabilidade é
	2023-05-24 14:00:07	O movimento é demonstrado caminhando: comunicação e avaliação aberta em uma conferência de ciência aberta	O primeiro Congresso Ibero-Americano de Ciência Aberta aconteceu nos dias 23 e 24 de novembro de 2022 como um fórum de diálogo ibero-
	2023-05-19 18:00:05	Reprodução e replicação na pesquisa científica – parte 1	A replicabilidade é uma questão central quando se discute a confiabilidade da pesquisa científica que se renova na promoção da ciência
	2023-05-05 20:00:12	Mapeamento da produção em análise de conteúdo no Scielo Brasil indica uma técnica que parou no tempo	Uma revisão citemétrica sobre o uso da análise de conteúdo no Scielo-Brasil demonstrou uma notável concentração de
	2023-04-20 18:50:11	BVS 25 Anos: Conquistas, Desafios e Oportunidades	Em 1998, a Rede Latino-Americana e do Caribe de Informação em Saúde aprovou a Declaração de São José "Hacia la Biblioteca Virtual en
	2023-04-10 19:20:18	É preciso um corpo para entender o mundo – por que o ChatGPT e outras IAs de linguagem não sabem o que dizem [Publicado originalmente no The Conversation em...]	É preciso um corpo para entender o mundo – por que o ChatGPT e outras IAs de linguagem não sabem o que dizem [Publicado originalmente no The Conversation em...]
	2023-03-29 12:00:1	Reformular a avaliação por pares para torná-la sustentável	Um artigo publicado recentemente discute a necessidade de uma profunda reforma da avaliação por pares, pois o ato

Fonte: O autor

Figura 51 – Página do Boletim de Ciências Geodésicas hospedada pela UFPR

The screenshot shows the homepage of the UFPR Digital Library of Periodicals. At the top left is the logo of the UFPR Digital Library of Periodicals. The top navigation bar includes links for Home, LAR, SOBRE, ACESSO+, PROCURAR, PROBLEMAS+, ANÚNCIOS, and ESTATÍSTICAS. Below the navigation is a breadcrumb trail: Home > Vol 28, No 2 (2022). The main content area features the BCG logo (a blue globe with a grid pattern) and the text "Departamento de Geomática". To the right, there is information about the journal's accreditation: "Qualis B1" and "ISSN 1982-2170". On the far right, there is a sidebar titled "Estás dentro:" with a dropdown menu set to "Biblioteca Digital de Periódicos (BDP)". Below this is a "CONECTE-SE" form with fields for "Nome de usuário" and "Senha", a "Lembre de mim" checkbox, and a "Conecte-se" button.

Fonte: O autor

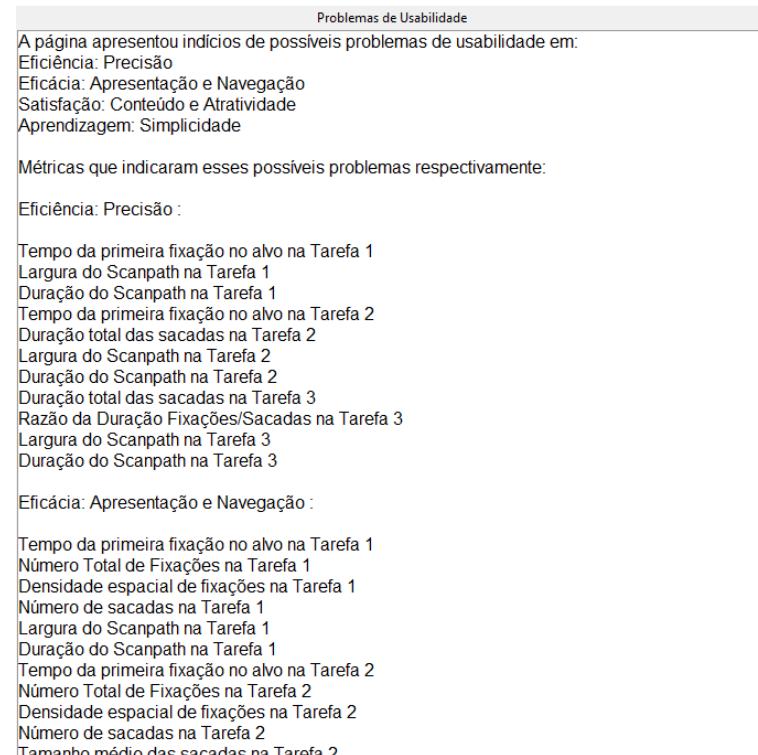
Essas duas páginas tiveram sua usabilidade avaliada previamente no trabalho de [Martins e Schmidt \(2022\)](#) usando como método o uso de questionários remotos por voluntários, onde as tarefas a serem avaliadas pediam elementos para serem encontrados nas páginas. Após a comparação entre as tarefas, foi constatado que apesar de ambas apresentarem problemas, a página da UFPR apresentou melhor desempenho de usabilidade.

Ao utilizar o sistema para avaliar as páginas, foram feitas três tarefas. Foi necessário fazer as tarefas com os voluntários em uma página de cada vez. As tarefas assim como o estudo mencionado tinham como objetivo encontrar determinados elementos, tentando deixar mais próximos o possível dos objetivos do estudo original. Os olhares dos usuários foram captados sem problemas em todas as condições e ao longo dos processos foram re-colhidas informações que permitiram estabelecer as relações esperadas com a usabilidade a partir de métricas de trabalhos anteriores.

Ao final das tarefas, e com os resultados já definidos, as comparações entre as páginas são feitas observando a quantidade de indícios de problemas nas tarefas e também nas dimensões de usabilidade.

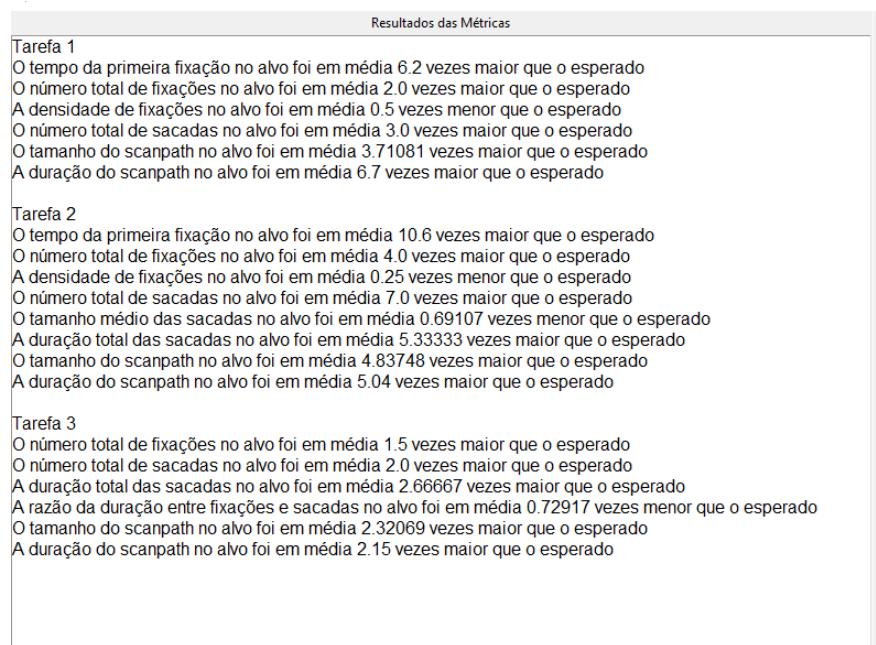
A página da *SciELO* teve como resultados os dados das Figuras 52 (parcialmente) e 53:

Figura 52 – Janela de Problemas de Usabilidade da página da SciELO



Fonte: O autor

Figura 53 – Janela de Problemas nas Tarefas da página da SciELO



Fonte: O autor

Já os da página da UFPR estão nas figuras 54 (parcialmente) e 55:

Figura 54 – Janela de Problemas de Usabilidade da página da UFPR

Problemas de Usabilidade

A página apresentou indícios de possíveis problemas de usabilidade em:

- Eficiência: Precisão
- Eficácia: Apresentação e Navegação
- Satisfação: Conteúdo e Atratividade
- Aprendizagem: Simplicidade

Métricas que indicaram esses possíveis problemas respectivamente:

Eficiência: Precisão :

- Duração do Scanpath na Tarefa 1
- Tempo da primeira fixação no alvo na Tarefa 3
- Duração total das sacadas na Tarefa 3
- Razão da Duração Fixações/Sacadas na Tarefa 3
- Largura do Scanpath na Tarefa 3
- Duração do Scanpath na Tarefa 3

Eficácia: Apresentação e Navegação :

- Densidade espacial de fixações na Tarefa 1
- Tamanho médio das sacadas na Tarefa 1
- Duração do Scanpath na Tarefa 1
- Densidade espacial de fixações na Tarefa 2
- Tamanho médio das sacadas na Tarefa 2
- Tempo da primeira fixação no alvo na Tarefa 3
- Número Total de Fixações na Tarefa 3
- Densidade espacial de fixações na Tarefa 3
- Número de sacadas na Tarefa 3
- Duração total das sacadas na Tarefa 3
- Razão da Duração Fixações/Sacadas na Tarefa 3
- Largura do Scanpath na Tarefa 3
- Duração do Scanpath na Tarefa 3

Satisfação: Conteúdo e Atratividade :

- Densidade espacial de fixações na Tarefa 1
- Tamanho médio das sacadas na Tarefa 1
- Duração do Scanpath na Tarefa 1
- Densidade espacial de fixações na Tarefa 2
- Tamanho médio das sacadas na Tarefa 2
- Tempo da primeira fixação no alvo na Tarefa 3

Fonte: O autor

Figura 55 – Janela de Problemas nas Tarefas da página da UFPR

Resultados das Métricas

Tarefa 1

- A densidade de fixações no alvo foi em média 0.0 vezes menor que o esperado
- O tamanho médio das sacadas no alvo foi em média 0.67545 vezes menor que o esperado
- A duração do scanpath no alvo foi em média 1.5 vezes maior que o esperado

Tarefa 2

- A densidade de fixações no alvo foi em média 0.0 vezes menor que o esperado
- O tamanho médio das sacadas no alvo foi em média 0.72575 vezes menor que o esperado

Tarefa 3

- O tempo da primeira fixação no alvo foi em média 1.06667 vezes maior que o esperado
- O número total de fixações no alvo foi em média 1.5 vezes maior que o esperado
- A densidade de fixações no alvo foi em média 0.66667 vezes menor que o esperado
- O número total de sacadas no alvo foi em média 2.0 vezes maior que o esperado
- A duração total das sacadas no alvo foi em média 7.66667 vezes maior que o esperado
- A razão da duração entre fixações e sacadas no alvo foi em média 0.21014 vezes menor que o esperado
- O tamanho do scanpath no alvo foi em média 3.89218 vezes maior que o esperado
- A duração do scanpath no alvo foi em média 2.6 vezes maior que o esperado

Fonte: O autor

Comparando os resultados das duas páginas pelo sistema, o resultado de [Martins e Schmidt \(2022\)](#) foi mantido, tendo a página hospedada pela *UFPR* com indícios de melhor usabilidade.

Tabela 7 – Quantidade de Indícios de Problemas encontrados no Boletim de Ciências Geodésicas por tarefas

	Quantidade de Indícios de Problemas encontrados no Boletim de Ciências Geodésicas por tarefas	
	SciELO	UFPR
Tarefa 1	6	3
Tarefa 2	8	2
Tarefa 3	6	8

A Tabela 7 acima representa quantos indícios de problemas foram encontrados e exibidos para cada tarefa nas duas páginas. Na página da *SciELO* foram encontrados 20 indícios, já na da *UFPR* foram apenas 13.

Tabela 8 – Quantidade de Indícios de Problemas encontrados no Boletim de Ciências Geodésicas por tarefas

	Quantidade de Indícios de Problemas encontrados no Boletim de Ciências Geodésicas por dimensão de usabilidade	
	SciELO	UFPR
Eficiência	11	6
Eficácia	20	13
Satisfação	20	13
Aprendizagem	20	13

Como observado na Tabela 8, e mais uma forma de confirmação, todas as quatro dimensões de usabilidade tem resultado favorável para a página hospedada pela *UFPR*.

A avaliação fornecida pelo sistema e a feita pelo estudo obtiveram resoluções muito próximas, mantendo o resultado favorável para a página na *UFPR* e mostrando que ainda assim ambas tem possíveis problemas de usabilidade que podem ser consertados.

Capítulo 6: Conclusão

A avaliação de usabilidade vem sendo tema de estudos há várias décadas, e diversas formas para realiza-la foram desenvolvidas. As mais comuns, como a popular Avaliação Heurística de Nielsen e Molich, tem opiniões de avaliadores levadas em conta, o que pode deixar detalhes inconscientes de fora da avaliação.

O desenvolvimento do sistema apresentado nesse trabalho tem como proposta a utilização da captação dos movimentos oculares, identificando a partir da incidência e trajetórias de olhares, fixações e sacadas, para permitir que seja feita uma avaliação com base em dados que não dependam de opiniões, que podem ser tendenciosas.

O sistema considera os dados coletados de qualquer usuário que execute suas tarefas, independente de idade ou conhecimento prévio, desde que seja executado em um computador conectado a uma câmera para captura de vídeo. O usuário, ao fazer a tarefa proposta, segue orientações de quem estiver responsável pela avaliação e ao final não precisa responder nenhum tipo de formulário, pois todos os dados necessários já estarão salvos. Para transformar esses dados em uma fonte de avaliação de usabilidade, são identificadas a partir deles métricas definidas em diversos trabalhos usados como base para este. Comparações entre essas métricas e valores esperados resultam em indícios que apontam ao responsável, possíveis problemas de usabilidade.

O resultado obtido com o desenvolvimento deste trabalho foi satisfatório, cumprindo seus objetivos estipulados, e mostrando uma forma de realizar uma avaliação de usabilidade de uma página *web*, sem a necessidade da utilização de aparelhos específicos para a tarefa que costumam ter um valor de compra ou aluguel muito alto, requisitando para uso apenas um computador e uma *webcam* conectada.

Os problemas encontrados durante o desenvolvimento, basicamente se limitaram a impossibilidade de sua utilização durante os testes mantendo os óculos na face. Isso exigiu que pequenos ajustes fossem feitos para amenizar a situação, como a mudança de tamanho e cor do cursor do *mouse* utilizado.

6.1 Trabalhos Futuros

O sistema tem algumas possibilidades de melhorias que podem ser exploradas futuramente com o objetivo de melhorar a aquisição de dados e a avaliação de usabilidade.

- a **Detecção das Pupilas em Pessoas com Óculos:** Atualmente o sistema de detecção das pupilas a partir dos pontos da face não consegue fazer a identificação em pessoas que usam óculos, confundindo o contorno dos olhos com a armação do equipamento
- b **Identificação Automática da Posição de Elementos na Página Web:** Os objetos distribuídos na página construída para a avaliação têm atualmente a necessidade de ter seu posicionamento informado manualmente em um arquivo JSON.
- c **Uso de Novas Métricas e Revisão das Existentes:** Ao longo dos anos foram descobertas novas métricas do rastreamento ocular que possuem relações com a usabilidade. Portanto, é esperado que novas descobertas sejam feitas futuramente que acrescentem novas métricas e atualizem as já conhecidas.

Referências

- ABRAHÃO, J.; SILVINO, A.; SARMET, M. Ergonomia, cognição e trabalho informatizado. *Psicologia: Teoria e Pesquisa*, v. 21, 08 2005. Citado na página 29.
- ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 9241-11: Requisitos ergonômicos para trabalho de escritórios com computadores, parte 11 – orientações sobre usabilidade*. Rio de Janeiro, 2002. 21 p. Citado na página 23.
- BALL, L. J.; RICHARDSON, B. H. Eye movement in user experience and human-computer interaction research. In: _____. *Eye Tracking: Background, Methods, and Applications*. New York, NY: Springer US, 2022. p. 165–183. ISBN 978-1-0716-2391-6. Citado 4 vezes nas páginas 38, 48, 49 e 50.
- BHOIR, S. et al. Measuring construction workers' attention using eye-tracking technology. In: *5th International/11th Construction Specialty Conference*. Vancouver, British Columbia: [s.n.], 2015. Citado 3 vezes nas páginas 48, 49 e 50.
- BOYKO, N.; BASYSTIUK, O.; SHAKHOVSKA, N. Performance evaluation and comparison of software for face recognition, based on dlib and opencv library. In: *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*. [S.l.: s.n.], 2018. p. 478–482. Citado 5 vezes nas páginas 35, 61, 67, 70 e 71.
- CARAYON, P.; HOONAKKER, P. Human factors and usability for health information technology: Old and new challenges. *Yearbook of Medical Informatics*, v. 28, p. 071–077, 08 2019. Citado na página 17.
- CARCAGNI, P. et al. Facial expression recognition and histograms of oriented gradients: a comprehensive study. *SpringerPlus*, v. 4, p. 645, 11 2015. Citado na página 66.
- CARTER, B.; LUKE, S. Best practices in eye tracking research. *International Journal of Psychophysiology*, v. 155, 06 2020. Citado 3 vezes nas páginas 30, 31 e 32.
- CATECATI, T. et al. Medindo a experiência do usuário por meio de sinais psicofisiológicos. *Ergodesign & HCI*, v. 5, n. Especial, p. 142–152, 2017. ISSN 2317-8876. Citado na página 33.
- DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2005. v. 1, p. 886–893. Citado 3 vezes nas páginas 35, 61 e 62.
- DJAMASBI, S. Eye tracking and web experience. *AIS Transactions on Human-Computer Interaction*, v. 6, p. 37–54, 06 2014. Citado 3 vezes nas páginas 19, 36 e 55.

- DUCHOWSKI, A. et al. Aggregate gaze visualization with real-time heatmaps. *Eye Tracking Research and Applications Symposium (ETRA)*, 03 2012. Citado na página 36.
- EHMKE, C.; WILSON, S. Identifying web usability problems from eye-tracking data. In: *21st British HCI Group Annual Conference on HCI 2007*. [S.l.: s.n.], 2007. v. 1, p. 119–128. Citado 6 vezes nas páginas 17, 18, 34, 48, 49 e 50.
- FALCÃO, C.; SOARES, M. Usabilidade de produtos de consumo: uma análise dos conceitos, métodos e aplicações. *usability of consumer products: an analyzes of concepts, methods and applications*. *Estudos em Design*, v. 21, p. 01–26, 01 2013. Citado na página 28.
- FORSTER, R. Aspectos da utilização do rastreamento ocular na pesquisa psicolinguística. *DELTA: Documentação de Estudos em Lingüística Teórica e Aplicada*, v. 33, p. 609–644, 08 2017. Citado 2 vezes nas páginas 29 e 30.
- GOBBI, A. et al. Uso do eye tracking para obtenção de medidas quantitativas em testes de usabilidade: Um estudo focado na medida da satisfação. *Human Factors in Design*, v. 6, p. 106–125, 12 2017. Citado 2 vezes nas páginas 26 e 28.
- GOLDBERG, J.; KOTVAL, X. Computer interface evaluation using eye movements: Methods and constructs. *International Journal of Industrial Ergonomics*, v. 24, p. 631–645, 10 1999. Citado 3 vezes nas páginas 48, 49 e 50.
- GOLDBERG, J.; WICHANSKY, A. Eye tracking in usability evaluation: A practitioner's guide. In: _____. *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*. [S.l.: s.n.], 2003. p. 493–516. ISBN 0444510206. Citado 7 vezes nas páginas 25, 26, 30, 31, 34, 45 e 46.
- GROSS, R. et al. Multi-pie. *International Conference on Automatic Face and Gesture Recognition*, v. 28, p. 807–813, 05 2010. Citado na página 70.
- HAESNER, M. et al. An eye movement analysis of web usability: Differences between older adults with and without mild cognitive impairment. *Assistive Technology*, v. 30, p. 1–8, 02 2017. Citado na página 29.
- HARVEY, P. D. Domains of cognition and their assessment. *Dialogues in Clinical Neuroscience*, Taylor & Francis, v. 21, n. 3, p. 227–237, 2019. PMID: 31749647. Citado na página 29.
- ISO/IEC. *ISO/IEC 2382-1:1993 - Information technology — Vocabulary — Part 1: Fundamental terms*. [S.l.], 1993. Citado na página 23.
- JACOB, R.; KARN, K. Commentary on section 4 - eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In: _____. *The Mind's Eye*. Amsterdam: North-Holland, 2003. p. 573–605. ISBN 978-0-444-51020-4. Citado na página 48.
- JOSEPH, A. W.; MURUGESH, R. Potential eye tracking metrics and indicators to measure cognitive load in human-computer interaction research. *Journal of Scientific Research of the Banaras Hindu University (JSR-BHU)*, v. 64, p. 168–175, 01 2020. Citado 4 vezes nas páginas 34, 48, 49 e 50.

- KUMAR, J.; KUMAR, J. Investigation of usability issues through physiological tools: An experimental study with tourism websites. In: AHRAM, T.; KARWOWSKI, W.; TAIAR, R. (Ed.). *Human Systems Engineering and Design*. Cham, CHE: Springer International Publishing, 2019. p. 365–371. ISBN 978-3-030-02053-8. Citado 3 vezes nas páginas 48, 49 e 50.
- LIEBLING, D.; DUMAIS, S. Gaze and mouse coordination in everyday work. *UbiComp 2014 - Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, p. 1141–1150, 09 2014. Citado na página 43.
- LIU, M.; ZHU, Z. A case study of using eye tracking techniques to evaluate the usability of e-learning courses. *International Journal of Learning Technology*, Inderscience Publishers, Geneva 15, CHE, v. 7, n. 2, p. 154–171, jul 2012. ISSN 1477-8386. Citado na página 48.
- MALAN, K.; ELOFF, J.; BRUIN, J. de. Semi-automated usability analysis through eye tracking. *South African Computer Journal*, v. 30, 07 2018. Citado na página 34.
- MARTINS, V.; SCHMIDT, M. Avaliação de usabilidade do site scielo e ufpr do periódico boletim de ciências geodésicas (bcg). *Ciência da Informação em Revista*, v. 8, p. 95–108, 02 2022. Citado 2 vezes nas páginas 109 e 112.
- MASLOV, I.; NIKOU, S. Usability and ux of learning management systems: An eye-tracking approach. In: *2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. [S.l.: s.n.], 2020. p. 1–9. Citado 2 vezes nas páginas 25 e 26.
- MKPOJIOGU, E.; HUSSAIN, A.; HASSAN, F. A systematic review of usability quality attributes for the evaluation of mobile learning applications for children. In: *Proceedings of the 3rd International Conference on Applied Science and Technology (ICAST'18)*. Penang, Malaysia: AIP Publishing, 2018. v. 2016, p. 020092. Citado 6 vezes nas páginas 24, 25, 26, 47, 51 e 52.
- MOLICH, R.; NIELSEN, J. Improving a human-computer dialogue. *Communications of ACM*, Association for Computing Machinery, New York, NY, USA, v. 33, n. 3, p. 338–348, mar 1990. ISSN 0001-0782. Citado na página 18.
- NIELSEN, J. Enhancing the explanatory power of usability heuristics. In: *Conference Companion on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 1994. (CHI '94), p. 210. ISBN 0897916514. Citado na página 18.
- NUGRAHA, A. et al. Usability evaluation for user interface redesign of financial technology application. *IOP Conference Series: Materials Science and Engineering*, v. 505, 07 2019. Citado na página 49.
- PAPOUTSAKI, A. et al. Webgazer: Scalable webcam eye tracking using user interactions. In: *25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*. New York City, NY, USA: [s.n.], 2016. Citado 2 vezes nas páginas 37 e 43.
- PASARICA, A. et al. Pupil detection algorithms for eye tracking applications. In: *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)*. Brasov, Romania: IEEE, 2015. p. 161–164. Citado na página 35.

- POOLE, A.; BALL, L. Eye tracking in hci and usability research. In: _____. *Encyclopaedia of human-computer interaction*. [S.l.]: Idea Group Inc., 2006. p. 211–219. Citado 2 vezes nas páginas 48 e 49.
- QUINONES, D.; RUSU, C. How to develop usability heuristics: A systematic literature review. *Computer Standards & Interfaces*, v. 53, p. 89–122, 03 2017. ISSN 0920-5489. Citado na página 18.
- SAGONAS, C. et al. 300 faces in-the-wild challenge: database and results. *Image and Vision Computing*, v. 47, p. 3–18, 01 2016. ISSN 0262-8856. Citado na página 70.
- SAUER, J.; SONDEREGGER, A.; SCHMUTZ, S. Usability, user experience and accessibility: towards an integrative model. *Ergonomics*, v. 63, p. 1–23, 05 2020. Citado na página 27.
- SINHA, G.; SHAHI, R.; SHANKAR, M. Human computer interaction. In: *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. [S.l.: s.n.], 2010. p. 1–4. Citado na página 17.
- SOLER, J. V. A.; FARINA, R. M.; FLORIAN, F. Relevância da acessibilidade intuitiva: como o foco na usabilidade no design de interfaces impacta o usuário. *Revista Interface Tecnológica*, v. 18, p. 194–207, 12 2021. Citado na página 27.
- SPAKOV, O.; MINIOTAS, D. Visualization of eye gaze data using heat maps. *ELEKTRONIKA IR ELEKTROTECHNIKA MEDICINE TECHNOLOGY T*, v. 115, p. 55–58, 01 2007. Citado 2 vezes nas páginas 36 e 37.
- SÁNCHEZ-MORALES, A.; DURAND-RIVERA, A.; MARTÍNEZ-GONZÁLEZ, C. L. Usability evaluation of a tangible user interface and serious game for identification of cognitive deficiencies in preschool children. *International Journal of Advanced Computer Science and Applications*, v. 11, 07 2020. Citado 2 vezes nas páginas 30 e 31.
- VANDERDONCKT, J. Distributed user interfaces: How to distribute user interface elements across users, platforms, and environments. In: *XIth Congreso Internacional de Interacción Persona-Ordenador Interacción'2010*. Valencia, España: AIPO, 2010. p. 3–14. Citado na página 30.
- VUORI, T. et al. Can eye movements be quantitatively applied to image quality studies? In: *Proceedings of the Third Nordic Conference on Human-Computer Interaction 2004*. Tampere, Finland: [s.n.], 2004. v. 82, p. 335–338. Citado na página 49.
- WAHYUNINGRUM, T.; KARTIKO, C.; WARDHANA, A. Exploring e-commerce usability by heuristic evaluation as a complement of system usability scale. In: *2020 International Conference on Advancement in Data Science, E-learning and Information Systems (ICADEIS)*. Lombok, Indonesia: IEEE, 2020. Citado na página 17.
- WANG, J. et al. *International Journal of Human-Computer Interaction*, Taylor Francis, v. 35, n. 6, p. 483–494, 2019. Citado 3 vezes nas páginas 18, 25 e 26.
- WEICHBROTH, P.; REDLARSKI, K.; GARNIK, I. Eye-tracking web usability research. In: *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*. Gdansk, Poland: [s.n.], 2016. v. 8, p. 1681–1684. ISSN 2300-5963. Citado 2 vezes nas páginas 35 e 36.

ZAIN, N. H. M. et al. Eye tracking in educational games environment: Evaluating user interface design through eye tracking patterns. In: *Visual Informatics: Sustaining Research and Innovations*. [S.l.: s.n.], 2013. v. 7067. ISBN 978-3-642-25199-3. Citado 2 vezes nas páginas [35](#) e [36](#).

ZARDARI, B. et al. Quest e-learning portal: applying heuristic evaluation, usability testing and eye tracking. *Universal Access in the Information Society*, v. 20, p. 1–13, 08 2021. Citado 2 vezes nas páginas [18](#) e [25](#).