PM

Sparks
Interação Humano-Computador com base num apontador laser

PROJETO DE MESTRADO

Carlos Eduardo Perestrelo Gouveia
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

setembro | 2021

Sparks
Interação Humano-Computador com base num apontador laser

PROJETO DE MESTRADO

Carlos Eduardo Perestrelo Gouveia
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Leonel Domingos Telo Nóbrega

RESUMO

Este projeto de mestrado contextualiza e descreve a criação de uma *framework* de interação humano-computador que utiliza um apontador laser como dispositivo de entrada. Perante o desafio identificado, e considerando as restrições impostas com o surgimento da pandemia COVID-19, a proposta assenta na utilização de um apontador laser para possibilitar uma interação *contactless*. Embora as preocupações com o contágio e propagação tenham estado presentes na elaboração desta abordagem, em verdade, a sua validade e utilidade não se esgota com o fim da pandemia. Pelo contrário, esta abordagem à interatividade humano-computador pode ser um elemento distintivo. Assim, existe a necessidade de estudar um método que possibilite a interação com uma tela projetada, criando uma alternativa inovadora e lúdica que promova maior interatividade com as aplicações web.

Perante o desafio identificado, iremos começar por realizar uma introdução à temática, metodologia e contextualização com o caso de estudo. De seguida, serão apresentados os principais conceitos de processamento de imagens, juntamente com os avanços tecnológicos que existem na área de visão computacional. Posteriormente, será realizado uma análise e levantamento de requisitos para a seleção das melhores estratégias, seguido da proposta de uma solução integrada.

O principal objetivo consiste em fornecer uma base para futuros estudos no campo da visão computacional, apresentando um protótipo que permita validar e demonstrar o potencial de um sistema particular ao qual designamos: Sparks.

Palavras-chave: Visão Computacional, Framework, Interação Humano-Computador, Laser

ABSTRACT

This project contextualizes and describes a human-computer interaction framework that uses a laser pointer as an input device. Given the identified challenge, and considering the restrictions imposed with the emergence of the COVID-19 pandemic, the proposal is based on a laser pointer to enable a contactless interaction. Although concerns about contagion and spread were present in the elaboration of this approach, in fact, its validity and usefulness do not end with the pandemic. On the contrary, this approach to human-computer interactivity can be a distinctive element. Thus, it is necessary to study a method that enables interaction with a projected screen, creating an innovative and playful alternative that promotes greater interactivity with web applications.

Looking at the identified challenge, we will start by performing an introduction, methodology and contextualization with the case study. The main concepts of image processing will be presented, along with the technological advances in the field of computer vision. Subsequently, an analysis and a survey of requirements will be carried out to select the best strategies, followed by the proposal of an integrated solution.

The main objective is to provide a basis for future studies, presenting a prototype that allows us to validate and demonstrate the potential of a particular system that we designate: Sparks.

Keywords: Computer Vision, Framework, Human-Computer Interaction, Laser

AGRADECIMENTOS

Aproveito o tópico para deixar uma palavra de agradecimento a todos os que de uma forma ou de outra estiveram ao meu lado durante toda ou parte desta caminhada.

Quero agradecer ao Doutor Leonel Domingos Telo Nóbrega pela oportunidade de aprendizagem, confiança, esclarecimento de dúvidas, foi incansável e imprescindível à formação exigida de cada uma das etapas e desafios colocados.

Agradeço à Universidade da Madeira e às restantes pessoas envolvidas pelo suporte do material necessário à realização do projeto.

Um agradecimento especial aos meus pais por todo o acompanhamento psicológico e pela ajuda financeira que proporcionou toda esta aventura.

Um muito obrigado à namorada pelo espírito crítico e construtivo ao crescimento pessoal exigido no decorrer desta etapa.

Estarei eternamente grato e agradeço a presença, o humor, a sinceridade, a compreensão e a confiança em mim depositada.

“A fé não me guia para onde eu quero, a fé conduz-me para onde Deus quer que eu chegue.”

Júnior Trovão

ÍNDICE

1. Introdução 1
1.1. Contexto 1
1.2. Objetivos 2
1.3. Metodologia 2
1.4. Organização 4
2. Estado da arte 5
2.1. Laser 5
2.2. Fundamento das cores 6
2.3. Visão computacional 7
2.4. Imagem 8
2.4.1. Imagem binária 9
2.4.2. Imagem em tons de cinza 10
2.5. Modelo de cor 10
2.5.1. RGB 10
2.5.2. HSV 11
2.6. Processamento de imagem 12
2.7. Suporte 13
2.7.1. Raspberry Pi 13
2.7.2. Câmaras 14
2.7.3. Python 15
2.7.4. Scikit-image 15
2.7.5. OpenCV 16
2.8. Projetos relacionados 16
2.8.1. Técnicas de interação utilizando o apontador laser 16
2.8.2. Interação humano-computador para telas de projeção 18
2.9. Sumário 19
3. Desenho do sistema 20
3.1. Requisitos 20
3.2. Cenários de qualidade 21
3.2.1. Modificabilidade 21
3.2.2. Disponibilidade 21
3.2.3. Desempenho 22
3.3. Táticas arquiteturais 23
3.3.1. Modificabilidade 23
3.3.2. Disponibilidade 23
3.3.3. Desempenho 24
3.4. Arquitetura de software 25
3.5. Processos do Sparks 27
3.5.1. Sparks Core 27
3.5.2. Sparks Web 30
3.6. Broker 32
3.7. Rotinas de manutenção 34
3.8. Sumário 35
4. Desenvolvimento do sistema 37
4.1. Deteção de bordas 37
4.2. Deteção do ponto laser 38

4.3. Rastreio do ponto laser 39
4.4. Gestos circulares 41
4.5. Gestos nas bordas 44
4.6. Processo de Calibração 46
4.6.1 Conectividade 47
4.6.2. Reconhecimento da tela 47
4.6.3. Calibração manual da tela 50
4.6.4. Deteção de colisões 50
4.6.5. Transformação de perspetiva 51
4.7. Local Storage 52
4.8. Interface Gráfica 53
4.9. Sumário 53
5. Experiências e resultados 55
5.1. Testes durante o desenvolvimento 55
5.1.1. Ambiente de testes 56
5.1.2. Efeitos da luminosidade 56
5.1.3. Testes de Luminosidade 60
5.1.4. Medidor de luminosidade 67
5.1.5. Realização de gestos 69
5.2 Avaliações Heurísticas 70
5.3. Testes de usabilidade 71
5.3.1. Tutorial 73
5.3.2. Estabilidade do ponto laser 73
5.3.3. Gesto circular sobre botões aleatórios 74
5.4. Resultados 75
5.5. Sumário 78
6. Conclusão 80
6.1. Trabalho Futuro 80
Referências 82
Anexos 86
Anexo A – Arquitetura de software do Sparks 87
Anexo B – Processamento visual 88
Anexo C – Protótipos de alta fidelidade 89
Anexo E – Depuração do gesto circular 91
Anexo D – Testes de usabilidade 92

LISTA DE FIGURAS

Figura 1 - Cronograma do projeto 3
Figura 2 - Espectro eletromagnético – Fonte [3] 6
Figura 3 - Automóvel da Tesla utilizando a visão computacional - Fonte [7] 7
Figura 4 - Identificação de tumor cerebral através da visão computacional - Fonte [8] 8
Figura 5 - Representação de um pixel – Fonte [10] 9
Figura 6 - Exemplo de imagem binária - Fonte [11] 9
Figura 7 - Representação da escala em tons de cinza - Fonte [12] 10
Figura 8 - Representações do modelo de cores RGB – Fonte [16] 11
Figura 9 – Representações cônicas e cilíndricas do modelo de cores HSV – Fonte [19] ... 11
Figura 10 - Raspberry Pi 3B – Fonte [22] 13
Figura 11 – Pi Camera V2 e Pi NoIR Camera V2 – Fonte [24] 14
Figura 12 - Evolução da biblioteca OpenCV - Fonte [22, p.] 16
Figura 13 - Exemplificação do projeto #1 - Fonte [1] 17
Figura 14 - Técnicas de interação utilizando o apontador laser - Fonte [1] 17
Figura 15 – Exemplificação do projeto #2 – Fonte [26] 18
Figura 16 - Apontador laser utilizado no projeto – Fonte [26] 18
Figura 17 - Processos do sistema 26
Figura 18 - Abstração do processo Sparks Core 27
Figura 19 - Funcionamento do middleware do Sparks Core #1 29
Figura 20 - Funcionamento do middleware do Sparks Core #2 29
Figura 21 - Paradigma Master-Slave entre a comunicação dos processos 30
Figura 22 - Abstração do processo Sparks Web 30
Figura 23 - Middleware (Sparks Web) – Configuração Inicial 31
Figura 24 - Gestão dos dados (Sparks Web Toolkit) 31
Figura 25 - Funcionamento do middleware do Sparks Web 32
Figura 26 - Representação do intermediário de mensagens 33
Figura 27 – Resultado do operador de deteção de bordas 38
Figura 28 - Técnica de subtração de fundo 39
Figura 29 - Matriz desenhada sobre a imagem capturada 40
Figura 30 - Histórico do ponto laser sobre a matriz (Fator de escala - 80) 41
Figura 31 - Histórico do ponto laser sobre a matriz (Fator de escala - 40) 41
Figura 32 - Identificação do ponto médio do gesto circular 43
Figura 33 - Ponto médio do gesto circular 43
Figura 34 - Dados do gesto circular 44
Figura 35 - Representação da imagem capturada num plano cartesiano 45
Figura 36 - Etapa de calibração (Teste de conetividade) 47
Figura 37 - Tentativa de identificação da tela sem sucesso 48
Figura 38 - Deteção da tela projetada (primeira estratégia) 48
Figura 39 - Deteção da tela projetada através da utilização de uma borda 49
Figura 40 - Calibração manual da tela projetada 50
Figura 41 - Área que deve ser ocupada pela tela projetada 51
Figura 42 - Transformação de perspetiva 52
Figura 43 - Armazenamento dos dados de calibração 52
Figura 44 - Cenário de testes inicial 55
Figura 45 - Ambientes de teste utilizados durante o desenvolvimento 56

Figura 46 - Imagens capturadas sobre diferentes superfícies de projeção 57
Figura 47 - Testes com cores de alta luminosidade 57
Figura 48 - Página de teste com cor de fundo verde 58
Figura 49 - Página de teste com cor de fundo castanho 58
Figura 50 - Resultado da deteção do ponto laser sobre fundo castanho 59
Figura 51 - Resultado da deteção do ponto laser sobre fundo verde 59
Figura 52 - Teste em ambiente diurno 60
Figura 53 - Matriz de cores 61
Figura 54 - Comparação entre os sensores Pi Camera #1 62
Figura 55 - Comparação entre os sensores Pi Camera #2 62
Figura 56 - Comparação entre os sensores Pi Camera #3 63
Figura 57 - Comparação dos níveis de luminosidade (com tela) #1 64
Figura 58 - Comparação dos níveis de luminosidade (com tela) #2 65
Figura 59 - Comparação dos níveis de luminosidade #1 66
Figura 60 - Comparação dos níveis de luminosidade #2 66
Figura 61 - Comparação dos níveis de luminosidade #3 67
Figura 62 - Medidor de luminosidade - Lutron Lx-101 68
Figura 63 – Medição através do luxímetro 68
Figura 64 - Problemas na deteção do gesto circular 70
Figura 65 - Ambiente onde foram realizados os testes de usabilidade 72
Figura 66 – Teste de usabilidade #1 73
Figura 67 - Teste de usabilidade #2 74
Figura 68 - Teste de usabilidade #3 74
Figura 69 - Execução do teste de usabilidade #2 75
Figura 70 - Execução do teste de usabilidade #3 76
Figura 71 - Arquitetura de software 87
Figura 72 - Monitorização do ponto laser (técnica de subtração de fundo) 88
Figura 73 - Segmentação da tela projetada 88
Figura 74 - Página inicial do sistema 89
Figura 75 - Página de configuração #1 89
Figura 76 - Página de configuração #2 90
Figura 77 - Deteção de um gesto circular 91
Figura 78 - Efeitos de alta luminosidade 92
Figura 79 - Deteção da tela projetada antes dos testes de usabilidade #1 92
Figura 80 - Deteção da tela projetada nos testes de usabilidade #2 93
Figura 81 - Testes de usabilidade com os participantes #1 93
Figura 82 - Testes de usabilidade com os participantes #2 94

LISTA DE TABELAS

Tabela 1 - Cores no espaço HSV – Fonte [3] 12
Tabela 2 - Especificações técnicas do Raspberry Pi 3B 14
Tabela 3 - Comparação entre a câmara do Raspberry Pi e do telemóvel 15
Tabela 4 - Cenário de modificabilidade 21
Tabela 5 - Cenário de disponibilidade 22
Tabela 6 - Cenário de desempenho 22
Tabela 7 - Identificação do gesto circular 42
Tabela 8 - Eventos gerados no cruzamento de uma borda da tela 44
Tabela 9 - Exemplo de combinações possíveis utilizando as bordas 45
Tabela 10 - Deteção de gestos na borda direita da tela 46
Tabela 11 - Valores de luminosidade #1 60
Tabela 12 - Valores de luminosidade #2 60
Tabela 13 - Visibilidade do ponto laser sobre a matriz de cores (Pi Camera V2) 63
Tabela 14 - Visibilidade do ponto laser sobre a matriz de cores (Pi NoIR Camera V2) 63
Tabela 15 - Média dos valores de luminosidade #1 69
Tabela 16 - Média dos valores de luminosidade #2 69
Tabela 17 - Padrões de deteção de um gesto circular 70
Tabela 18 - Informações dos participantes dos testes de usabilidade 72
Tabela 19 - Tempos de execução dos testes de usabilidade 77

LISTA DE ABREVIATURAS

AI	*Artificial Intelligence (Inteligência artificial)*
API	*Application Programming Interface (Interface de programação de aplicações)*
CSI	*Camera Serial Interface (Interface serial da câmara)*
EDA	*Event Driven Architecture (Arquitetura orientada a eventos)*
EM	*Electromagnetic Spectrum (Espectro eletromagnético)*
FIFO	*First in, First out (Primeiro a entrar, primeiro a sair)*
FPS	*Frames per Second (Quadros por segundo)*
HSV	*Hue Saturation and Value (Matiz, Saturação e Valor)*
HTTP	*HyperText Transfer Protocol*
IoT	*Internet of Things (Internet das coisas)*
IR	*Infrared (Infravermelhos)*
JSON	*JavaScript Object Notation (Notação de objeto em javascript)*
Nm	*Nanómetro*
OS	*Operating System (Sistema operativo)*
QoS	*Quality of Service (Qualidade de serviço)*
RGB	*Red, Green e Blue (Vermelho, verde e azul)*
ROI	*Region of Interest (Região de interesse)*
UI	*User Interface (Interface do utilizador)*

1. Introdução

Este capítulo descreve o propósito, o âmbito do projeto e enumera os objetivos delineados para médio e longo prazo.

O apontador laser é um dispositivo portátil e frequentemente utilizado durante apresentações. Ele permite apontar para qualquer lugar independentemente do tamanho do alvo ou distância. Apesar do uso comum, se pensarmos do ponto de vista de uma apresentação, pode ser conveniente o orador não apenas direcionar o apontador, mas também utilizá-lo como dispositivo de entrada.

Desta forma, pretende-se desenvolver uma solução capaz de identificar o ponto laser sobre uma tela, e através do percurso realizado identificar gestos como uma ação a executar no navegador web. A ação a executar depende do gesto identificado, e pode variar: mudar de página, clicar em um botão, entre outros, sem ter de deslocar-se até ao computador. Assim, os dados recolhidos devem ser classificados corretamente, para permitir o mapeamento das coordenadas do ponto laser para as coordenadas do rato no computador.

1.1. Contexto

Este projeto assenta na área da visão computacional e são vários os documentos que contribuem para o estado da arte e se dedicam à exploração da inteligência artificial sobre a visão computacional.

O projeto foi desenvolvido na casa do autor deste documento, sediado em Machico. E todo o material utilizado para as diversas experiências foi obtido através da Universidade da Madeira e de outras pessoas próximas.

A ideia de desenvolver o Sparks derivou de uma visita realizada ao Museu Casa da Luz (MCL), localizado no Funchal. Na visita guiada, a Dra. Carla Freitas e a Dra. Luísa Garrido, mencionaram a necessidade de dotar o museu de elementos tecnológicos que promovessem maior interatividade. Posteriormente, em reunião com o Prof. Leonel Nóbrega, foi colocado a hipótese do desenvolvimento de um conceito de interação baseado na luz. Dado o caráter da proposta e o facto de a visão computacional estar em constante crescimento, estes foram os motivos que suscitaram interesse para o desenvolvimento do corrente projeto de mestrado.

Para a implementação do projeto será utilizado um Raspberry Pi interligado a uma câmara e direcionada para a tela projetada. De seguida, os quadros de imagem capturados serão processados a fim de extrair informações úteis para fins de tomada de decisão.

Assim, pretende-se reunir informação relevante para contextualizar o leitor a compreender a utilidade deste sistema e os desafios mais relevantes.

1.2. Objetivos

O objetivo preliminar comporta uma revisão da literatura e compreender como o apontador laser pode vir a ser utilizado como dispositivo de entrada. Esta etapa implica pesquisar, analisar e comparar as metodologias existentes e agrupar o conjunto de métodos que melhor se enquadra no cenário descrito.

Em seguida é pretendido estudar a área da visão computacional e efetuar um levantamento dos métodos de processamento de imagem frequentemente utilizados. Esta etapa envolve o estudo da biblioteca *OpenCV*, amplamente utilizada na ótica da visão computacional.

Outro objetivo assenta na realização de testes de usabilidade de forma a observar a utilização do sistema e investigar questões que envolvam a interação e entendimento da interface gráfica. Por outro lado, espera-se que o conteúdo do relatório e do projeto possam contribuir para a automação dos sistemas de interação humano-computador, com um sistema de baixo custo, robusto, expansível e que procura diferenciar-se dos métodos tradicionais de interação.

Durante o desenvolvimento foram frequentes as reuniões calendarizadas com os orientadores para verificar o cumprimento ou não cumprimento dos requisitos inicialmente propostos.

1.3. Metodologia

Na Figura 1, encontra-se ilustrado o cronograma que contém as tarefas previstas para a realização do projeto, estabelecendo a ordem de início e do esforço esperado em cada momento.

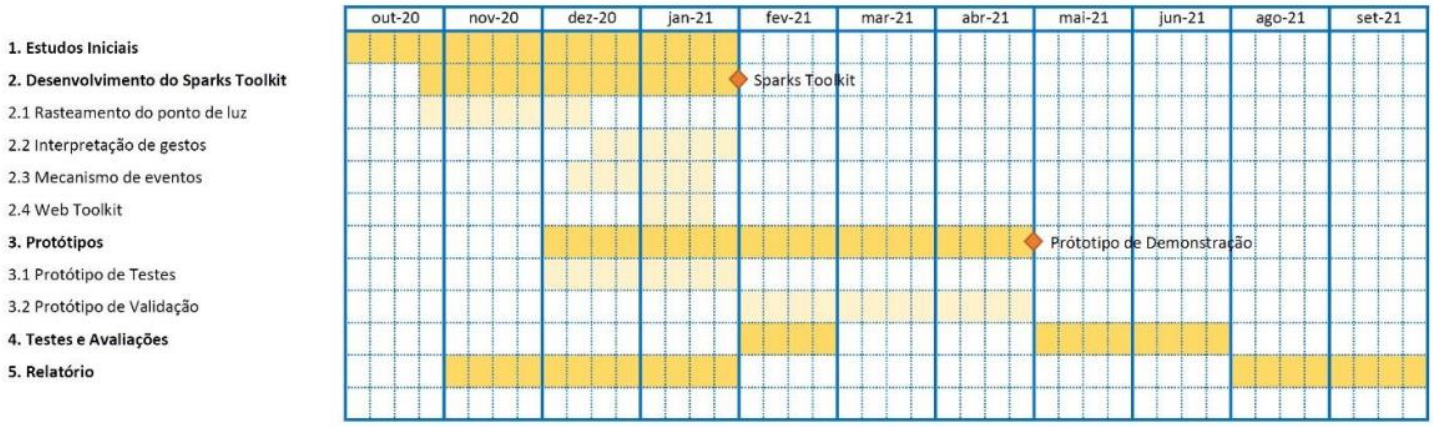

Figura 1 - Cronograma do projeto

A tarefa 1 tem como objetivo efetuar uma avaliação da literatura com o intuito de validar a ideia inicial e identificar as melhores estratégias e possibilidades de melhoria. Adicionalmente, será realizado um levantamento das iniciativas que utilizam ou utilizaram esta forma de interação, para compreender o que existe e os problemas inerentes.

A tarefa 2 está organizada em várias partes. O *rastreio do ponto laser* envolve o estudo dos algoritmos de processamento de imagem. A *interpretação de gestos* envolve estudar formas efetivas que permitam interagir com o sistema. O conjunto mínimo de gestos que o sistema deve suportar é o point & click. O *mecanismo de eventos* indica a principal forma de comunicação (eventualmente, a única) através da troca de eventos. As aplicações deverão subscrever os eventos que desejam subscrever e o Sparks Core anuncia os eventos de acordo com os gestos que o utilizador efetua. O *Web Toolkit* consiste no desenvolvimento de uma biblioteca em JavaScript que permite o conteúdo web beneficiar do conceito de interação proposto.

A tarefa 3 servirá como ferramenta de apoio ao desenvolvimento, com o intuito de desenvolver um protótipo que permita desde cedo testar o conjunto de funções associadas ao sistema.

A tarefa 4 envolve os testes e avaliação do sistema com base no protótipo final, onde serão desenvolvidos testes com o intuito de identificar as limitações e oportunidades de melhoria.

A tarefa 5 consiste na produção de um relatório de Mestrado que satisfaça os requisitos académicos exigidos.

1.4. Organização

O presente relatório é essencialmente composto por 6 capítulos que cobrem da teoria à prática o trabalho desenvolvido. O material encontra-se organizado na tentativa de fornecer um grau de rigor, clareza e os resultados dos testes efetuados.

O primeiro capítulo contextualiza o tema, apresenta os objetivos e indica a metodologia adotada para a realização do projeto.

O segundo capítulo aborda o estado da arte, a visão computacional, os projetos relacionados e os mecanismos que suportam o projeto.

O terceiro capítulo comporta o caso de estudo onde são analisados os requisitos, funcionalidades e a arquitetura que melhor se adequa para o projeto.

O quarto capítulo descreve a implementação do sistema e as técnicas utilizadas durante o desenvolvimento.

O quinto capítulo descreve os testes, os resultados e como foram resolvidos os problemas vividos.

O sexto capítulo contém a síntese dos vários capítulos, descreve os pontos de reflexão e apresenta sugestões para trabalho futuro.

2. Estado da arte

Este capítulo aborda a temática do projeto onde será realizado uma contextualização da visão computacional, as áreas de aplicação e uma perspetiva dos métodos mais comuns de processamento de imagem.

Em seguida, será realizado uma descrição do conjunto de tecnologias e materiais utilizados que permitiram a realização das diversas etapas. Posteriormente, é identificado alguns dos projetos que no passado utilizaram o apontador laser como dispositivo de entrada e os principais problemas associados.

Portanto, é pretendido reunir informação para que o leitor consiga contextualizar o propósito, a interligação e as aplicabilidades práticas de cada um dos tópicos.

2.1. Laser

O laser é uma abreviatura do termo em inglês: "Light Amplification by Stimulated Emission of Radiation". O laser produz um feixe de luz intenso e altamente direcionado, podendo assumir diferentes tonalidades de cores, compreendidas entre o espectro de cores visíveis. A luz do laser é constituída por um comprimento de onda, e todos os componentes do feixe de luz podem ser movidos a elevadas velocidades e com muito mais precisão do que qualquer outro sistema existente [1].

Os lasers foram inventados em 1960 e desde então se tornaram onipresentes, com aplicabilidade em diferentes áreas da sociedade: na ciência, medicina, indústria, entre outros. O primeiro uso dos lasers na vida cotidiana foi através do leitor de código de barras em 1974. Seguidamente, surgiram outras aplicações que utilizam a tecnologia laser, nomeadamente, leitores de disco, impressoras, entre outros.

No âmbito do projeto será utilizado um apontador laser como dispositivo de entrada, contudo, durante a revisão da literatura foi possível constatar que apresenta algumas desvantagens. Como por exemplo, dificuldades de adaptação, alta sensibilidade e pode não ser adequado para ecrãs pequenos. As principais vantagens como dispositivo de entrada é o facto de suportar grandes distâncias independentemente do tamanho da projeção e o facto do ponto laser ser imediatamente visível.

A utilização da tecnologia laser requer alguma preocupação porque o corpo humano é vulnerável a radiações intensas e a exposição prolongada pode provocar danos, por exemplo, no globo ocular. Por este motivo, existem regulamentos que estabelecem o limite máximo em mW (miliwatts) da comercialização de apontadores laser. Na Europa, por sua

vez, foi publicado em fevereiro de 2014, (2014/59/EU), a restrição de comercialização de apontadores laser com potência superior a 1 mW. Contudo, apesar dos riscos é possível encontrar na Internet apontadores laser com potências superiores.

2.2. Fundamento das cores

Isaac Newton foi um cientista, físico e matemático inglês que na década de 1960, através de várias experiências, constatou que a luz solar quando incide um prisma de vidro dá origem a inúmeras cores [2]. O trabalho desenvolvido por Newton, abriu caminho para outros estudos, permitindo perceber que a luz pode-se decompor num conjunto de luzes de várias cores. Assim, a luz visível é apenas uma pequena parte do conjunto de ondas da radiação eletromagnética, designado por espectro eletromagnético.

Na Figura 2, temos a representação do espectro eletromagnético que possui o conjunto de todos os comprimentos de onda, que se estende desde as baixas frequências (ondas de rádio) até as altas frequências (raios gama). No entanto, neste tópico iremos nos focar nas ondas eletromagnéticas que podem ser percebidas pelo olho humano e que dizem respeito ao espectro visível.

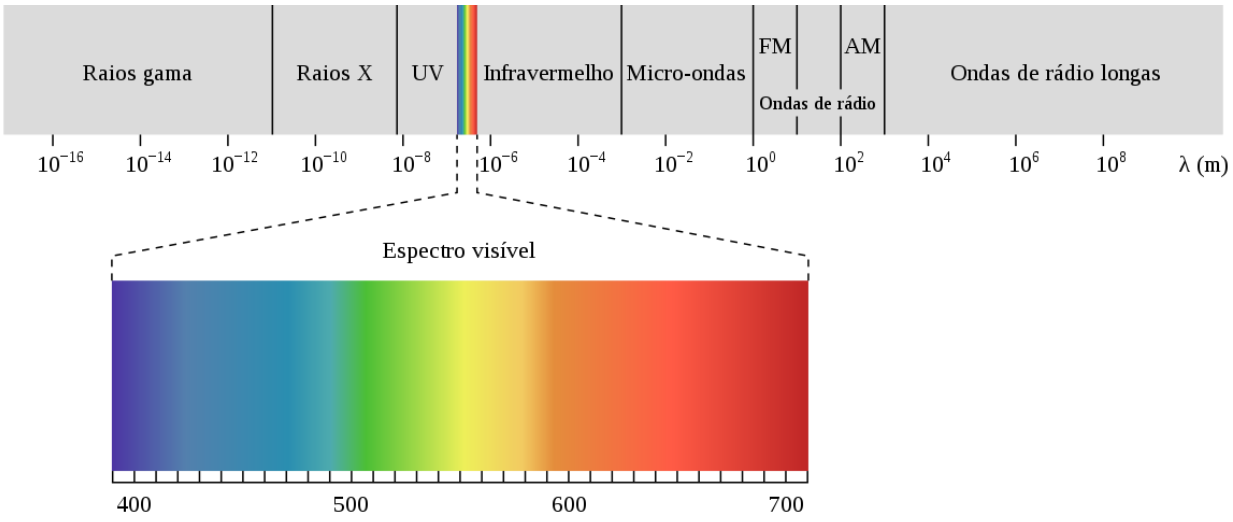

Figura 2 - Espectro eletromagnético – Fonte [3]

O espectro de luz visível restringe-se entre os 400 e 700 nm (nanómetros) [3]. Sendo, as cores do espectro visível: vermelho, laranja, amarelo, verde, ciano, azul e violeta. Por outro lado, enquanto o ser humano está restrito ao espectro de luz visível, as máquinas são capazes de cobrir quase todo o espectro eletromagnético [4].

2.3. Visão computacional

Marengoni & Stringhini (2009), mencionam que a visão computacional procura replicar a visão humana através de software e hardware com o objetivo de compreender o conteúdo das imagens. Da mesma forma, que a visão humana captura as informações que surgem ao longo do caminho (cor, brilho, entre outros), a visão computacional recebe uma imagem como entrada, processa-a e extrai informações para fins de tomada de decisão [5].

A visão computacional está associada à inteligência artificial, no domínio do desenvolvimento de sistemas capazes de compreender o significado de uma imagem e construir descrições a partir dos dados extraídos. Por outro lado, a visão computacional pode operar sobre diferentes frequências do espectro eletromagnético, como por exemplo, raio X, luz infravermelha, entre outros.

Barelli (2018), alude que a visão computacional pode ser um complemento à visão biológica, uma vez que os sistemas podem observar o ambiente através de processos artificiais [6]. O autor defende que a inteligência artificial é uma ferramenta que colabora com o desenvolvimento tecnológico, especialmente na robótica quando são utilizados sistemas de visão computacional para fornecer informações do ambiente.

No quotidiano existem infinitas tarefas que são automatizadas por sistemas de visão computacional, como por exemplo, os robôs industriais que montam automóveis, sistemas que detetam indivíduos através de imagens, entre outras aplicações que serão descritas adiante. Na Figura 3, é possível constatar um exemplo, onde um veículo da Tesla é capaz de realizar um percurso sem a intervenção do condutor, podendo identificar pedestres, sinais de trânsito, veículos, entre outros [7]. O conceito ilustrado faz uso de diversos sensores com o auxílio dos paradigmas de inteligência artificial e visão computacional.

Figura 3 - Automóvel da Tesla utilizando a visão computacional - Fonte [7]

Por outro lado, a visão computacional pode ser utilizada na área da saúde como método complementar de diagnóstico, para detetar precocemente doenças em exames

efetuados por meio de imagens, como por exemplo, a tomografia computadorizada (TAC), ressonância magnética, entre outros. A Figura 4, ilustra um exame de diagnóstico, onde é possível observar um tumor cerebral detetado através de uma ressonância magnética, realizada por meio de um sistema de visão computacional.

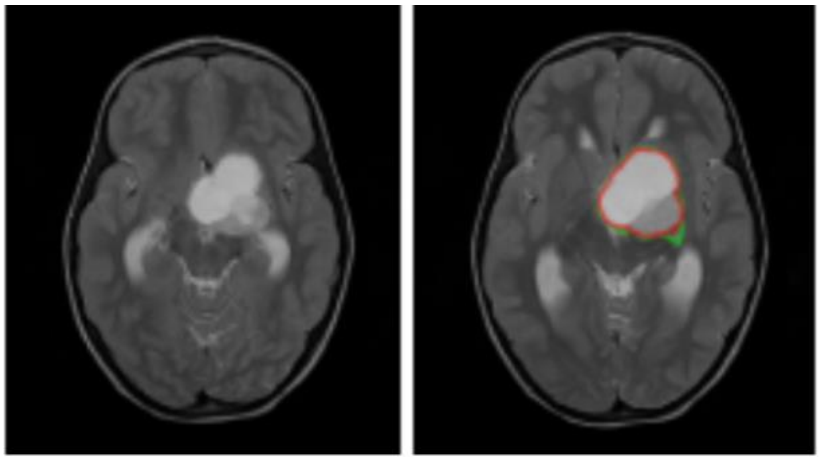

Figura 4 - Identificação de tumor cerebral através da visão computacional - Fonte [8]

Para que os exemplos apresentados sejam possíveis, os sistemas de visão computacional são constituídos por diferentes etapas. A etapa de pré-processamento utiliza técnicas para minimizar o ruído da imagem, de modo a realçar os detalhes. A etapa de segmentação, identifica os objetos ou regiões de interesse para extrair um conjunto de caraterísticas, como por exemplo, a área, o diâmetro, a cor, entre outros parâmetros, com o intuito de identificar cada objeto individualmente. Por fim, a última etapa consiste no reconhecimento do objeto segmentado, defini-lo por meio de uma classe e analisar os resultados.

2.4. Imagem

De acordo com Gonzales & Woods (2008), uma imagem é uma representação visual que pode ser definida através de uma função bidimensional $f(x,y)$, onde x e y são as coordenadas espaciais do plano e f a função de intensidade [4].

Tipicamente, a qualidade de uma imagem depende da quantidade de pixels. Esses elementos contêm informações sobre a cor ou intensidade de cada pixel e traduzem uma pequena porção da imagem [9], como representado na Figura 5.

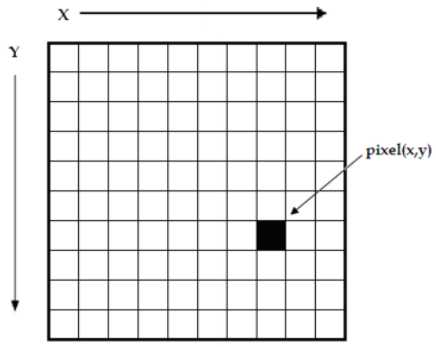

Figura 5 - Representação de um pixel – Fonte [10]

O termo resolução é comumente utilizado para referir o número de pixels que uma imagem contém. A resolução pode ser definida pela largura e altura da imagem, bem como pelo número total de pixels. Por exemplo, uma imagem com 2048 pixels de largura e 1536 pixels de altura (2048x1536) contém 3.145.728 pixels. Deste modo, quando se trata do processamento de imagens é preferível operar sobre imagens com menor resolução para obter maior desempenho, e devemos limitar os níveis de cores a serem atribuídos a cada pixel.

Nos subtópicos que se seguem, serão apresentadas as estratégias utilizadas durante a etapa de segmentação. O objetivo passa por isolar as regiões contidas em uma imagem e para tal, algumas características como a cor, intensidade ou textura são consideradas.

2.4.1. Imagem binária

A imagem binária é a forma mais simples de representar uma imagem, onde cada pixel representa apenas um de dois valores: zero ou um. A Figura 6, ilustra um exemplo da representação dos pixels, onde o valor zero representa a cor preta e o valor um a cor branca.

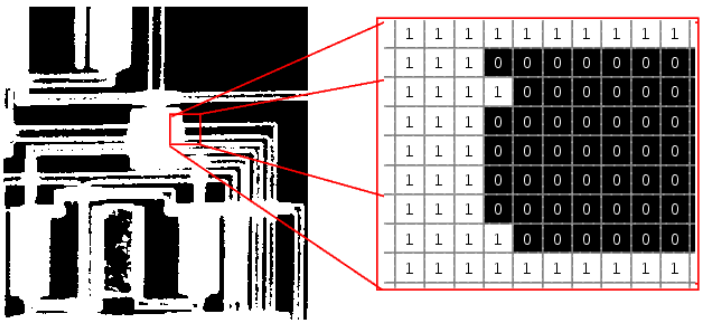

Figura 6 - Exemplo de imagem binária - Fonte [11]

Desta forma, a imagem binária permite obter os dados dos pixels para analisar as formas, tamanhos e orientações dos objetos e, assim, reconhecê-los ou inspecioná-los [4][12].

2.4.2. Imagem em tons de cinza

Uma imagem em tons de cinza é representada apenas pelo valor da intensidade que define a aparência de um pixel. Se o pixel for representado por um bit, ele pode assumir apenas um de dois valores, 0 ou 1, se o pixel for representado por um byte, então pode assumir um valor entre os 256 níveis de cinza, compreendido entre 0 (preto) e 255 (branco) [13].

A Figura 7, ilustra os valores das tonalidades em tons de cinza que um pixel pode assumir, onde podemos verificar que à medida que aumentamos o número de bits os níveis de cinza aumentam.

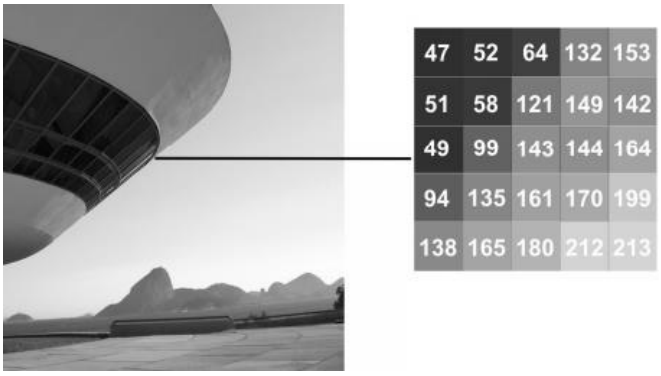

Figura 7 - Representação da escala em tons de cinza - Fonte [12]

2.5. Modelo de cor

A imagem binária ou em tons de cinza são as mais fáceis de representar porque não existe necessidade de tratá-las num espaço de cor. Por outro lado, uma imagem representada em um espaço de cor depende da junção dos valores dos pixels provenientes das diferentes tuplas de canais. Assim sendo, o modelo de cores traduz uma representação matemática que fornece critérios para facilitar a especificação das cores em um padrão e geralmente utiliza um sistema de coordenadas.

Assim, as cores podem ser descritas através de diferentes modelos de cor. A título de exemplo, será abordado, o modelo RGB e o modelo HSV.

2.5.1. RGB

O modelo RGB descreve uma cor através da combinação das cores primárias: vermelho (R), verde (G) e azul (B). Assim, três planos diferentes são utilizados para representar a intensidade de uma cor primária para cada pixel da imagem [15]. Rosebrock (2017), alude que cada cor é definida por um byte, então para cada cor são definidos valores no intervalo entre 0 e 255. Assim, a determinação da cor é realizada através da quantidade de vermelho, azul ou verde que um único pixel contém [16].

Na Figura 8, podemos observar na imagem á esquerda que através da junção da cor vermelha com o verde, obtém-se o amarelo, ao unir o vermelho com o azul obtém-se o rosa, e ao combinar as três cores primárias, obtém-se o branco. Por outro lado, na imagem á direita, podemos ver o modelo RGB representado num sistema de coordenadas cartesianas tridimensionais, onde as cores são representadas por pontos no interior do cubo.

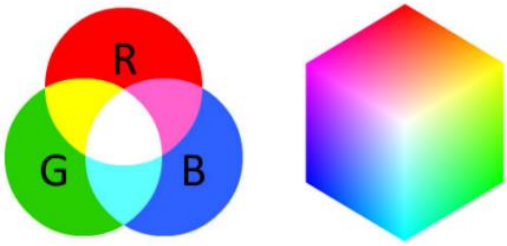

Figura 8 - Representações do modelo de cores RGB – Fonte [16]

Este espaço de cores é projetado para representar cores em sistemas eletromagnéticos, como por exemplo, ecrãs. No entanto, embora seja um modelo projetado para o sentido humano, não é adequado para o processamento de imagens [17].

2.5.2. HSV

O espaço de cores HSV tenta descrever as cores com mais precisão do que o modelo RGB, através das propriedades: Hue (matiz), Saturation (saturação) e Value (brilho) [3]. O modelo HSV é matematicamente cilíndrico, mas pode ser pensado conceitualmente como um cone invertido de cores, cujo eixo central varia do preto na parte inferior ao branco na parte superior.

A Figura 9, ilustra uma representação do modelo HSV, onde na extremidade de cada "fatia", obtém-se a cor no seu tom mais puro. O ângulo em torno do eixo corresponde à matiz, a distância do eixo corresponde à saturação e a distância ao longo do eixo corresponde à luminosidade ou brilho [18].

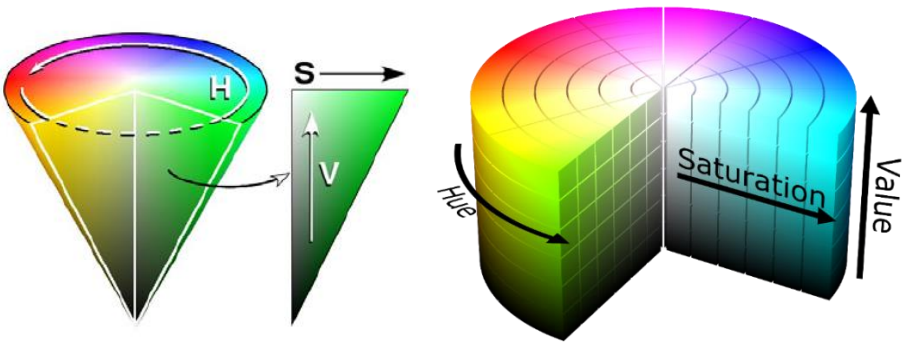

Figura 9 – Representações cônicas e cilíndricas do modelo de cores HSV – Fonte [19]

A matiz determina a cor e é medido em graus num intervalo entre 0 e 359. A Tabela 1, especifica para cada cor, o grau correspondente no modelo de cores HSV.

Cor (Hue)	Grau
Vermelho	0
Amarelo	60
Verde	120
Cião	180
Azul	240
Magenta	300

Tabela 1 - Cores no espaço HSV – Fonte [3]

A saturação corresponde à quantidade de branco que a cor apresenta. Quanto maior for o valor mais pura será a cor, por outro lado, quanto menor o valor mais próximo se aproxima do branco [20]. O brilho determina a intensidade percebida (cor mais clara ou mais escura). Em outras palavras, descreve a quantidade de branco que uma cor contém.

Em termos de aplicabilidade, quando queremos segmentar objetos tendo como referência a cor, a utilização do modelo HSV é benéfica, permitindo distinguir os objetos e acompanhar os movimentos realizados, entre outros.

2.6. Processamento de imagem

Marengoni & Stringhini (2009) mencionam que o processamento de imagens visa a modificação de uma imagem, com o objetivo de sobressair uma coleção de atributos de interesse. O processamento de imagens pode ser favorável em diferentes aplicações, como por exemplo, numa radiografia, onde um conjunto de métodos são aplicados para possibilitar a observação e identificação de uma patologia [5].

No presente tópico, serão abordados os métodos de processamento de imagem mais comuns nas dissemelhantes áreas de utilização.

O **melhoramento de imagem** refere-se á modificação de uma imagem, de forma que o resultado produzido seja melhor que o original. Não existe um método de processamento específico que garanta melhorias, neste processo, cabe ao utilizador decidir se o resultado está de acordo com o seu interesse.

O **restauro de imagem** visa melhorar a aparência de uma imagem, é uma técnica objetiva, uma vez que as técnicas de restauração são fundamentadas por modelos matemáticos e probabilísticos.

A **compressão** corresponde a estratégias para diminuir o tamanho de uma imagem, frequentemente utilizado, durante a transferência de dados.

O **processamento morfológico** é uma ferramenta que extrai informações de uma imagem que podem ser pertinentes para a representação ou descrição de um objeto.

2.7. Suporte

Neste tópico, será abordado os materiais utilizados para a realização do projeto. Numa fase inicial, será discutido o Raspberry Pi, seguido dos sensores de câmara utilizados ao longo do desenvolvimento, de forma a obter diferentes fontes de comparação. Posteriormente, será descrito a linguagem de programação utilizada e um resumo das principais bibliotecas de visão computacional com maior suporte na comunidade Python.

2.7.1. Raspberry Pi

O Raspberry Pi é um microcomputador de baixo custo e de tamanho reduzido, acessível para todas as pessoas que pretendem desenvolver conhecimentos em programação e circuitos eletrónicos [21]. No mercado, existem diferentes modelos a ser comercializados, no entanto, para o desenvolvimento do projeto foi utilizado o modelo Raspberry Pi 3B apresentado na Figura 10.

Figura 10 - Raspberry Pi 3B – Fonte [22]

A Tabela 2, sumariza as especificações técnicas do modelo utilizado e tendo em conta o âmbito do projeto, acreditamos que as tarefas que exigem alta demanda (por exemplo, técnicas de processamento visual) não serão afetadas pelos recursos físicos e consequentemente prejudicar a experiência do utilizador.

Componente	Especificação
Processador	Quad Core 1.2 GHz 64 Bits
Memória	1 GB LPDDR2 SDRAM

Conectividade	2.4 GHz e 5 GHz IEEE 802.11.b/g/n/ac WLAN Bluetooth 4.2 4 × USB 2.0 ports MIPI DSI & CSI
GPIO	40 pinos
Saída de Vídeo e Áudio	1 × HDMI 3.5 mm Jack
Armazenamento	Micro SD
Alimentação	5V/2.5A DC via micro-USB 5V DC via GPIO Power over Ethernet (PoE)

Tabela 2 - Especificações técnicas do Raspberry Pi 3B

2.7.2. Câmaras

No Raspberry Pi 3B existem três interfaces que permitem utilizar uma câmara, nomeadamente, a interface SPI, CSI e USB. No entanto, as câmaras serão conectadas na interface CSI, uma vez que permite uma experiência mais robusta e de alto desempenho [23].

Na Figura 11, podemos visualizar os sensores de câmara que serão utilizados no projeto. O sensor Pi Camera V2 possui um sensor Sony IMX219 de 8 megapixels. Por outro lado, o sensor Pi NoIR Camera V2 possui as mesmas características do sensor anterior, mas não utiliza um filtro infravermelho [24, p. 2].

Ambos os sensores suportam os modos: 1080p30, 720p60 e 480p90, para captura de vídeo ou imagem. Os sensores de câmara são compatíveis com os diferentes modelos do Raspberry Pi e podem ser acedidos através de bibliotecas de terceiros, que no nosso caso particular, será a biblioteca Picamera disponibilizada pelo Python.

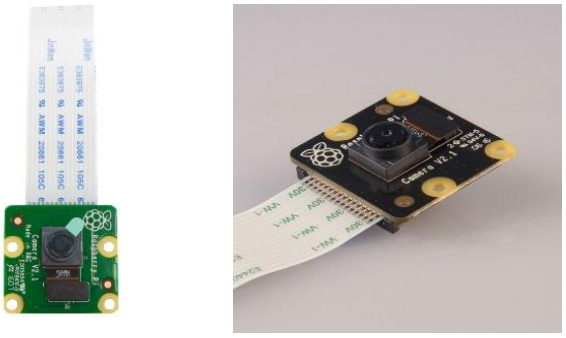

Figura 11 – Pi Camera V2 e Pi NoIR Camera V2 – Fonte [24]

Por outro lado, e para termos uma base de comparação durante os testes, a câmara do telemóvel *Oneplus 5* será utilizada como método de depuração em casos de problemas durante o processamento visual. Este motivo, ocorre pelo facto de a câmara do telemóvel ter características superiores, tal como ilustrado na Tabela 3.

Pi Camera v2 8MP	**Oneplus 5 Camera**
8MP, f/1.2, 25mm, 1/2.5”, 1.12 µm	16MP, f/1.7, 24mm, 1/2.8", 1.12µm 20 MP, f/2.6, 36mm, 1/2.8", 1.0µm
1080p@30fps, 720p@60fps, 640x480@90fps	4K@30fps,1080p@30/60fps, 720p@30/120/480fps

Tabela 3 - Comparação entre a câmara do Raspberry Pi e do telemóvel

Assim, espera-se que as diferentes câmaras possam servir como meio de comparação durante as fases de desenvolvimento e de testes.

2.7.3. Python

O *Python* é uma linguagem de programação de alto nível, dinâmica, desenvolvida por Guido van Rossuml, sendo atualmente uma das linguagens mais populares [12]. O facto de possuir uma sintaxe compacta e um ecossistema de bibliotecas complementares nas diferentes áreas de computação, torna-o uma linguagem de programação poderosa [18].

Todos estes motivos, juntamente com a experiência passada foram fatores eliminatórios para a seleção do Python como linguagem de desenvolvimento.

2.7.4. Scikit-image

O scikit-image é uma biblioteca de código aberto, com uma variedade de algoritmos de processamento de imagem desenvolvidos em Python, e possui uma API bem documentada suportada por uma equipa ativa de colaboradores.

Esta biblioteca permite que os utilizadores principiantes na área de visão computacional possam aprender os algoritmos de maneira prática, ajustando os parâmetros de configuração e modificando o código. Além disso, oferece módulos com os conceitos básicos de processamento de imagem, como por exemplo, conversão do espaço de cores, aplicação de filtros, entre outros [25].

Apesar da simplicidade das operações, o facto de ser uma biblioteca recente (agosto de 2009) e por ainda encontrar-se em desenvolvimento, estes foram alguns dos motivos que fizeram-nos procurar uma solução mais robusta e que será descrita no tópico posterior.

2.7.5. OpenCV

O OpenCV (*Open Source Computer Vision*) é uma biblioteca de código aberto, inicialmente desenvolvida pelos engenheiros da Intel, com o objetivo de tornar a visão computacional mais acessível. O OpenCV disponibiliza uma grande variedade de funções relacionadas com a visão computacional e inteligência artificial, permitindo desde a aplicação de filtros até a execução de operações complexas, como por exemplo, análise de movimentos, reconhecimento de padrões, entre outros [5].

O OpenCV foi desenvolvido nas linguagens de programação C/C++ e mais tarde foi suportada por outras linguagens, como por exemplo, o Python, Java, entre outros. A referência temporal ilustrada na Figura 12, data o lançamento das versões mais importantes, sendo possível constatar a evolução da biblioteca.

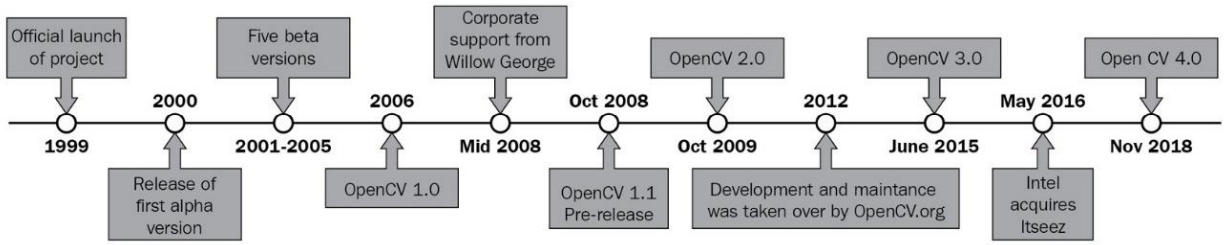

Figura 12 - Evolução da biblioteca OpenCV - Fonte [22, p.]

O OpenCV fornece interfaces de alto nível para capturar, processar e apresentar os dados de uma imagem, permitindo relacionar com bibliotecas científicas, como por exemplo, o NumPy. Como resultado, o OpenCV é frequentemente utilizado para fins acadêmicos, empresariais, entre outros [22].

2.8. Projetos relacionados

Neste tópico serão apresentados os conhecimentos adquiridos e as necessidades verificadas durante a revisão de literatura. Serão abordados alguns dos projetos que utilizaram o apontador laser como dispositivo de entrada, e assim transmitir ao leitor alguns dos avanços já alcançados neste âmbito.

2.8.1. Técnicas de interação utilizando o apontador laser

O estudo proposto por B. Shizuki et al. (2006) [1] investiga uma abordagem que utiliza um apontador laser para interagir e controlar uma apresentação do PowerPoint. O conjunto do material utilizado para a realização do projeto consistiu na utilização de um computador, câmara USB e apontador laser.

Os autores mencionam que existem diferentes soluções no mercado para controlar os diapositivos, geralmente através de um comando com uma interface infravermelha ou Bluetooth, para enviar ações para o computador. Por este motivo, o foco do artigo passa por estudar uma alternativa, tal como ilustrado na Figura 13, que permita o orador controlar um diapositivo através de um apontador laser, evitando a necessidade de estar perto do computador.

Figura 13 - Exemplificação do projeto #1 - Fonte [1]

O estudo apresenta técnicas de interação utilizando o apontador laser sobre uma tela projetada. A Figura 14, ilustra as técnicas utilizadas para a interação com o sistema, através do cruzamento do ponto laser em uma das bordas da tela [1]. Os gestos executados nas bordas foram intitulados por cruzamento In-Out, quando o ponto laser move-se do interior para o exterior da tela, e Out-In, quando o ponto laser move-se do exterior para o interior.

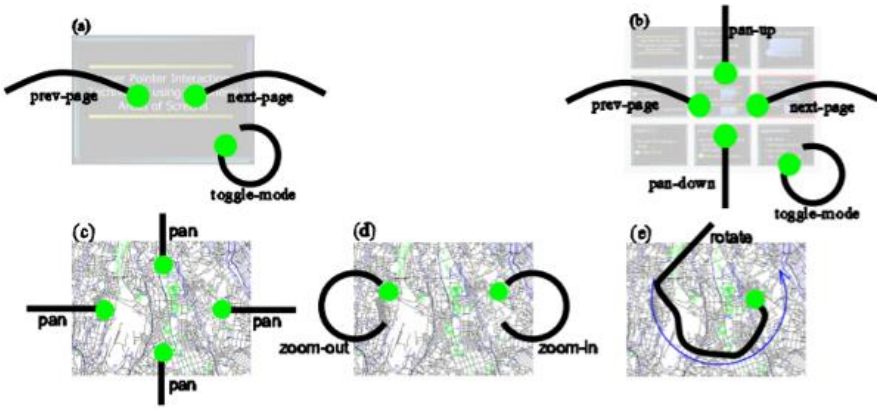

Figura 14 - Técnicas de interação utilizando o apontador laser - Fonte [1]

Do conjunto dos projetos revisados, este foi o que mais se enquadrou no âmbito do projeto. Embora com uma aplicação alvo diferente, o conjunto de gestos adotados pareceu-

nos desde o início a abordagem mais adequada para interação com um navegador web. Assim, os gestos descritos serviram de base para a implementação dos gestos adotados no Sparks, onde serão posteriormente aprofundados no capítulo IV.

2.8.2. Interação humano-computador para telas de projeção

O estudo proposto por M. Wissen (2001), [26] consiste em um sistema baseado no apontador laser para interação com telas de projeção, controlando diretamente o rato do computador.

Na Figura 15, encontra-se ilustrado a arquitetura do sistema, onde os computadores comunicam utilizando a arquitetura TCP/IP. No coração do sistema está um computador com três placas gráficas, cada uma conectada a um dos três projetores, que combinam para criar um único espaço de trabalho contínuo. Cada projetor é atribuído a uma das três câmaras, que por sua vez são conectadas a três computadores. As câmaras verificam em simultâneo a superfície de projeção em procura do ponto laser. Se encontrado, as coordenadas do ponto laser são calculadas e transmitidas através da rede para o computador central, que então posiciona o ponteiro do rato no local apropriado.

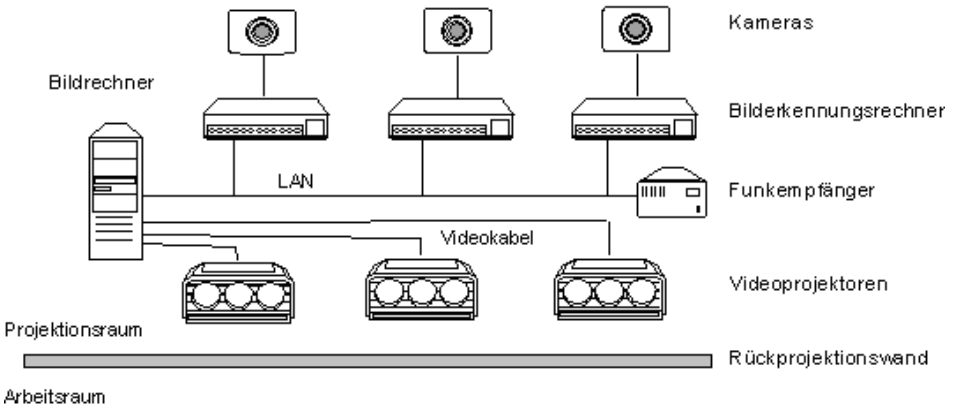

Figura 15 – Exemplificação do projeto #2 – Fonte [26]

Por outro lado, a Figura 16, elucida o apontador laser explicitamente projetado para funcionar no projeto. O botão 1 e 2 simulam a ação do botão esquerdo e direito do rato, respetivamente, e o botão 3 fixa a localização do ponteiro do rato na tela.

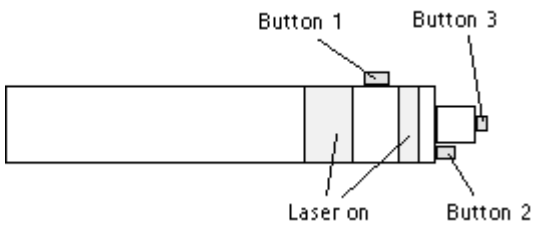

Figura 16 - Apontador laser utilizado no projeto – Fonte [26]

De acordo com o artigo, a utilização do apontador laser apresentou vários problemas, como por exemplo, a instabilidade causada pela mão do utilizador, elementos da interface gráfica pequenos, entre outros. No entanto, o autor salienta que a instabilidade da mão do utilizador pode ser mitigada por meio de software. Relativamente à interface gráfica dos sistemas operativos é difícil de controlar porque as interfaces foram desenvolvidas tendo em mente a interação com o rato.

2.9. Sumário

O ser humano detém como sentido mais apurado a visão e este motivo esclarece a importância das imagens na perceção humana. No entanto, o olho humano encontra-se limitado no que concerne ao espectro eletromagnético uma vez que somente é capaz de visualizar o espectro de luz visível.

Verificamos que a visão computacional abarca uma variedade de utilizações e possui diferentes áreas de aplicação, como por exemplo, a saúde, indústria, entre outros. A visão computacional considera três processos durante a análise de uma imagem. O processo de baixo nível seleciona as operações que permitem diminuir o ruído, melhorar o contraste e nitidez da imagem. O processo de nível médio recolhe os atributos da imagem, como por exemplo, os contornos para permitir uma classificação individual dos objetos. Por fim, o processamento de alto nível compreende o conjunto de objetos que são reconhecidos e executa funções cognitivas que se encontram relacionadas com a visão [5].

Observamos que as imagens binárias e as imagens em tons de cinza são representadas por um único canal. As imagens binárias usam um espaço de cor limitado a 2 bits e as imagens em tons de cinza utilizam um espaço de cor limitado a 8 bits [13]. Vimos que no modelo de cores RGB, cada pixel é representado por uma tupla de três canais: vermelho, verde e azul, podendo o valor do pixel pertencer a um intervalo entre 0 e 255. No entanto, nas etapas de processamento pode não ser a abordagem mais indicada.

Por outro lado, vimos que o OpenCV é a biblioteca mais utilizada no âmbito da visão computacional e utiliza outras bibliotecas científicas (por exemplo, o NumPy) para representar as imagens como matrizes multidimensionais.

Encerramos este capítulo revisando um conjunto de projetos que abrangem o âmbito da visão computacional. A maioria dos projetos revistos indicaram que as falhas na deteção do ponto laser, mesmo por um curto período, podem prejudicar a utilização do sistema. Isto demonstra as dificuldades que nos esperam, e essencialmente, a necessidade do desenvolvimento de automatismos para aperfeiçoar estes mecanismos.

3. Desenho do sistema

Este capítulo trata das principais decisões de desenho resultantes dos requisitos funcionais e não funcionais. Será realizado uma descrição dos módulos que compõem o sistema, a forma como interagem e as propriedades que suportam.

Posteriormente, serão abordados os atributos de qualidade e as táticas arquiteturais utilizadas para garantir que os serviços permanecem em conformidade com a sua especificação em caso de falhas.

Por fim, será discutido a implementação do agente intermediário (broker) responsável por estabelecer a comunicação entre os processos do sistema.

3.1. Requisitos

Neste tópico será descrito os requisitos relevantes para o desenvolvimento do sistema, resultantes das várias pesquisas e reuniões onde foi possível apurar as funcionalidades e as restrições que o software deve enfrentar. Os *requisitos funcionais* (RF) definem as funções que o sistema deve oferecer. Os *requisitos não funcionais* (RNF) definem como o sistema deve ser e auxiliar na implementação das funcionalidades [27]. Deste modo, seguem-se os requisitos auferidos:

RF001: O sistema deve ser autónomo na deteção do ponto laser;

RF002: O sistema deve ter um processo de aprendizagem para testar as funcionalidades;

RF003: O sistema deve monitorizar o estado dos sensores;

RF004: O sistema deve detetar os erros e tentar automaticamente recuperá-los;

RF005: O sistema deve armazenar os dados e assim que não representam informação significativa devem ser eliminados;

RNF001: Os dados devem ser armazenados em memória;

RNF002: O sistema deve ter uma arquitetura modular, sendo a comunicação entre os processos constituída por um broker de mensagens;

RNF003: O tempo de resposta não deve ultrapassar os 3 segundos;

RNF004: A interface gráfica deve ser projetada tendo em mente o apontador laser;

3.2. Cenários de qualidade

Neste tópico, serão apresentados os cenários consolidados para os atributos de qualidade identificados, dos vários objetivos clarificados durante as reuniões. Esta etapa trata-se de um processo importante antes da realização da arquitetura de software.

Dado o âmbito de utilização do Sparks em tempo real, torna-se evidente que o *desempenho* é um atributo prioritário. Deste modo, deve existir um controlo da quantidade de comunicação entre os módulos. Destaca-se também, a *disponibilidade*, uma vez que existem pontos de falha que podem afetar o funcionamento global do sistema. Por último, o atributo de *modificabilidade*, também pareceu prioritário, uma vez que a área de visão computacional está em constante evolução e pode haver necessidade futura de aprimorar os algoritmos.

3.2.1. Modificabilidade

A modificabilidade é determinada pelo modo como uma funcionalidade é dividida e pelas técnicas de codificação utilizadas dentro de um módulo. Este atributo, determina o esforço necessário para realizar alterações de forma eficiente sem a introdução de defeitos [27].

Na Tabela 4, encontra-se apresentado um cenário de modificabilidade, que considera a modificação do algoritmo de deteção do ponto laser.

Descrição	Modificar o algoritmo de deteção do ponto laser sem afetar ou degradar a qualidade do software.
Fonte	Desenvolvedor.
Estímulo	Modificar o módulo de processamento visual.
Ambiente	Durante o desenvolvimento.
Artefacto	Código.
Resposta	A modificação ocorreu com sucesso e as diferentes partes não foram afetadas.
Medida de Resposta	A alteração não deve afetar outras funções ou atributos de qualidade.

Tabela 4 - Cenário de modificabilidade

3.2.2. Disponibilidade

A disponibilidade trata das falhas do sistema e das consequências associadas. Uma falha ocorre, quando o sistema não fornece o serviço de forma consistente com a sua

especificação [27]. As preocupações são diversas relativamente ao que pode causar uma falha, e por esse motivo, temos de saber como as detetar, tratar e evitar.

Na Tabela 5, será apresentado o cenário que faz referência à necessidade de comunicação entre os módulos do sistema.

Descrição	Prevenir falhas durante a comunicação com o agente intermediário (broker).
Fonte	Sparks Core.
Estímulo	O Sparks Core deixa de receber o heartbeat do intermediário.
Ambiente	Modo de operação normal.
Artefacto	Canais de comunicação.
Resposta	O Sparks Core tenta uma nova comunicação (retry), e caso não consiga entra em modo degradado.
Medida de Resposta	Enquanto está no modo degradado, o sistema deverá apresentar uma notificação na tela.

Tabela 5 - Cenário de disponibilidade

3.2.3. Desempenho

O desempenho foi outro atributo considerado prioritário, uma vez que deve existir um limite de resposta para as funcionalidades críticas [27]. O sistema trabalha com vários quadros de imagem por segundo e, portanto, a extração dos dados deve ser rápida. Assim, decidiu-se implementar um sistema de prioridades para distinguir os eventos mais prioritários dos menos prioritários. Eventualmente, todos os eventos serão tratados, contudo, os de baixa prioridade serão processados mais tarde.

Na Tabela 6, encontra-se ilustrado um cenário, onde o utilizador através do apontador laser, realiza um gesto circular para executar a ação de clique.

Descrição	O sistema deve detetar o gesto circular em menos de 3 segundos.
Fonte	Utilizador.
Estímulo	Gesto circular realizado através do apontador laser sobre um botão na página web.
Ambiente	Tempo de execução.
Artefacto	Navegador.
Resposta	Identificar o gesto e reencaminhar as coordenadas da ação de clique para o navegador.
Medida de Resposta	Menos de 3 segundos.

Tabela 6 - Cenário de desempenho

3.3. Táticas arquiteturais

Uma vez consolidados os atributos e cenários de qualidade, passamos à definição das táticas arquiteturais. As táticas são decisões que influenciam o cumprimento de um atributo de qualidade, permitem capturar boas estruturas de desenho e influenciam na resposta do sistema [28].

3.3.1. Modificabilidade

As táticas de modificabilidade referem-se à capacidade do software ser modificado para adição de funcionalidades, melhorias ou adaptação dos requisitos com o menor esforço possível [27]. Através da restrição de dependências e da utilização de um intermediário (broker) consegue-se reduzir o acoplamento e aumentar a coesão. Por outro lado, com o passar do tempo existirão algoritmos mais eficientes, e neste contexto, é espectável que a área de AI possa contribuir para a evolução do projeto, sendo importante deixar em aberto a introdução de novos paradigmas.

As táticas de modificabilidade utilizadas são categorizadas em dois grupos: reduzir o tamanho de um módulo e o acoplamento.

Para a *redução do tamanho de um módulo*, foi utilizada a tática:

- Dividir o módulo – refinar um módulo em módulos menores para reduzir o custo médio das mudanças.

Para *reduzir o acoplamento*, foram utilizadas as táticas:

- Restrição de dependências – limitar a comunicação entre os módulos.
- Utilização de um agente intermediário (broker).

Para a divisão de um módulo, temos o exemplo da classe *Laser* repartida por subclasses, nomeadamente na classe *Dot* e *Gesture*. Para a restrição de dependências, pretende-se o encapsulamento dos dados que garante propriedades importantes, tais como: facilidade de reutilização, deteção de erros e modularidade. Por último, a utilização de um agente intermediário, permitirá coordenar a comunicação e garantir que os serviços comunicantes permaneçam ignorantes da identidade de destino.

3.3.2. Disponibilidade

As táticas de disponibilidade têm como objetivo garantir que em situações de falhas os serviços permanecem em conformidade com a sua especificação [27]. Por exemplo, durante a etapa de calibração é necessário que os processos (Sparks Core e Sparks Web) comuniquem entre si, caso contrário, o sistema será obrigado a seguir um fluxo secundário

dado a impossibilidade de comunicação. As táticas de disponibilidade utilizadas são categorizadas em dois grupos: deteção e recuperação de falhas.

Para a *deteção de falhas* foram utilizadas as táticas:

- Heartbeat – troca periódica de mensagens entre os processos monitorizados.
- Deteção de exceções – detetar erros em tempo de execução.

Para a *recuperação de falhas* foram utilizadas as táticas:

- Tratamento de exceções.
- Retry – tentar novamente a instrução que originou o erro.
- Funcionamento em modo degradado – quando o sistema se encontra com falhas, mantem-se apenas o essencial.

O tratamento de exceções procura evitar que os erros ocorridos em tempo de execução tornem-se falhas, por exemplo, durante a obtenção de dados do sensor de câmara. Por outro lado, a tática hearbeat verifica periodicamente se a comunicação entre os serviços está operacional.

A tática retry pode ser utilizada durante a etapa de calibração, por exemplo, se a tela não for detetada. Nesta situação, o sistema tenta novamente o reconhecimento, se não for possível, o utilizador é informado para definir manualmente os cantos da tela.

Por último, a tática de funcionamento em modo degradado permite que o sistema continue a funcionar, até que uma ação corretiva seja tomada. No entanto, se a falha for crítica é importante notificar o utilizador da indisponibilidade do sistema.

3.3.3. Desempenho

A tática de desempenho caracteriza-se pelo tempo que o sistema leva a responder a um determinado evento. Existem diferentes aspetos arquiteturais que devem ser considerados, nomeadamente a quantidade de comunicação entre componentes e o modo como as funcionalidades são atribuídas [27]. Por outro lado, existem aspetos não arquiteturais que devem ser considerados, como por exemplo, a escolha dos algoritmos a implementar, a forma como são codificados, entre outros.

As táticas de desempenho utilizadas são categorizadas em dois grupos: gestão de recursos e redução do tamanho de um módulo.

Para a *gestão de recursos* foi utilizada a tática:

- Concorrência – as threads podem processar diferentes conjuntos de atividades em simultâneo.

Para a *redução do tamanho de um módulo* foram utilizadas as táticas:

- Priorização de eventos – se os eventos não são todos importantes, deve ser imposto um esquema de prioridade que classifica os eventos de acordo com a importância.
- Eficiência dos recursos – melhorar os algoritmos em áreas críticas.
- Limitar o tempo de resposta – os eventos devem ter uma taxa máxima de resposta.

O sistema de prioridades classifica os eventos de acordo com a importância de os atender. Por exemplo, se a posição da câmara ou da tela, está diferente desde a ultima calibração, o sistema deve forçar a recalibração para garantir a confiabilidade do sistema.

A introdução de concorrência é de elevada importância, uma vez, que a captura dos quadros de imagem deve ocorrer em paralelo com o processamento de informação. A eficiência dos recursos e a limitação do tempo de resposta, contribui para a qualidade do sistema, uma vez que é necessário garantir uma resposta rápida (*feedback*) para o utilizador.

Outro fator benéfico para o desempenho seria o aumento dos recursos computacionais, o que permitiria reduzir a latência e evitar sobrecargas, contudo, não será considerado na implementação do projeto.

3.4. Arquitetura de software

De modo a tornar o sistema versátil é necessário partir de uma base de planificação. Desta forma, é importante definir no início do projeto os requisitos e a estrutura que irá dar suporte ao sistema [29]. Durante o desenvolvimento, o software poderá estar sujeito a alterações, seja para acomodar novos recursos, adaptar-se a novos ambientes ou corrigir problemas. Porém, as mudanças podem ser difíceis quando não existe documentação devidamente elaborada [27]. Assim, a arquitetura de software constitui uma peça fundamental no desenvolvimento, uma vez que incorpora decisões quanto ao modo como o sistema deve ser estruturado.

Na Figura 17, encontra-se representado os dois processos integrantes do Sparks: o Sparks Web e o Sparks Core. A representação alude duas características: a estrutura organizacional dos componentes e dos dados. A estrutura organizacional estabelece as relações entre os objetos e a estrutura de dados determina a organização da informação [27]. Nesta fase, iremos começar por abstrair a estrutura de cada processo, na forma representada. O sistema é constituído por uma arquitetura multicamadas que faz uso de processos distribuídos, o que torna o sistema independente da localização, podendo estar na mesma máquina ou em máquinas separadas. Como resultado, a aplicação pode ser dividida em várias partes, cada uma bem definida, com as características e responsabilidades pré-determinadas.

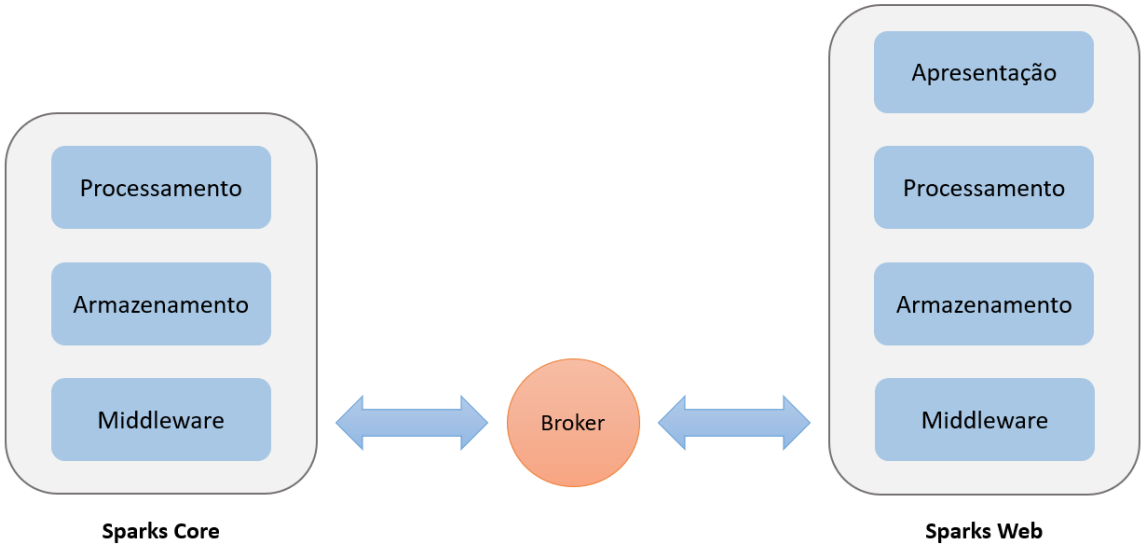

Figura 17 - Processos do sistema

A **camada de apresentação** é a camada diretamente visível para o utilizador através do navegador web.

A **camada de processamento** lida com o tratamento dos dados. No processo Sparks Web, os dados a ser processados estão maioritariamente relacionados com a interface web. No processo Sparks Core, os dados estão relacionados com o processamento visual e configuração do sistema. No entanto, ambos os processos lidam com informação proveniente da comunicação com o intermediário (broker).

A **camada de armazenamento** lida com os fluxos de dados provenientes do exterior ou interior do processo. Os dados provenientes do interior são informações geradas pelos módulos internos. Enquanto os dados provenientes do exterior são eventos recebidos através do intermediário de mensagens (broker).

A **camada de middleware** gere o fluxo de informação do processo. O middleware atua como um tradutor, estabelece as regras de comunicação e atua de forma transparente para o utilizador.

Para a elaboração da arquitetura do sistema foi necessário, numa primeira fase, identificar os cenários de qualidade relevantes. Das várias reuniões, foi possível retirar informações pertinentes e específicas para determinar as funcionalidades que deveriam ser incorporadas no sistema. Desta forma, no Anexo A – Arquitetura de software do Sparks, encontrasse representado a forma como os processos do sistema foram segmentados, sendo abordados com mais detalhe na seção seguinte deste documento.

3.5. Processos do Sparks

A divisão do projeto em processos permite separar responsabilidades, diminuir a complexidade, facilitar a depuração e torna o projeto mais fácil de manter. Assim, os efeitos provocados por alterações ficam limitados, o que reduz a propagação de erros [27]. Assim, é pretendido desenvolver um sistema coeso para garantir qualidade de software.

3.5.1. Sparks Core

O *Sparks Core* coordena as instruções a serem executadas no Sparks Web. É responsável por obter os quadros de imagem do sensor de câmara e analisá-los para fins de tomada de decisão. Portanto, atua como o cérebro do sistema sendo totalmente transparente para o utilizador.

A Figura 18 ilustra a arquitetura do *Sparks Core* com os módulos particionados para garantir a separação de responsabilidades. Este processo contém diversas tarefas distribuídas pelos módulos, entre elas, captura dos quadros de imagem, técnicas de processamento visual, manutenção do estado e histórico do sistema, entre outros.

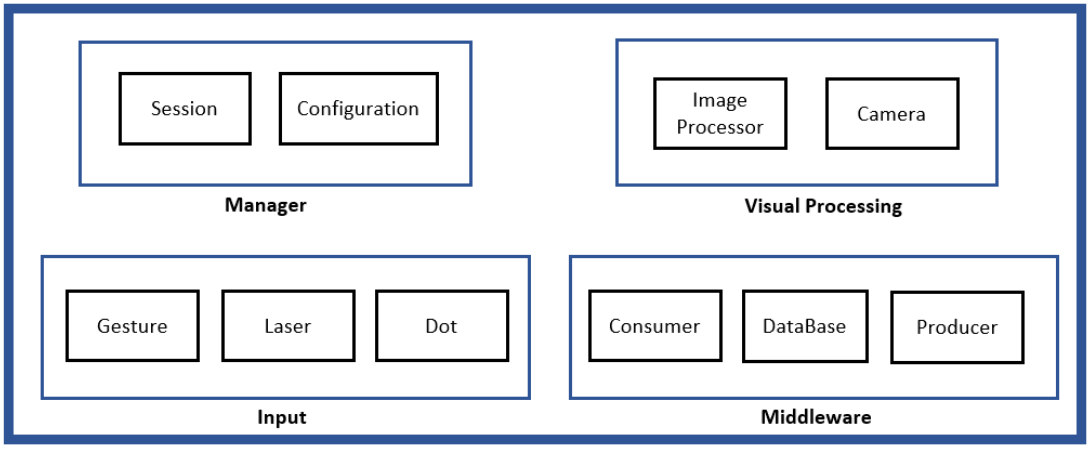

Figura 18 - Abstração do processo Sparks Core

A classe *Configuration* armazena informações do estado do sistema que são encaminhadas para os outros módulos, consoante as necessidades de configuração. As informações podem ser: a resolução da imagem capturada, atualização das coordenadas dos cantos da tela, entre outros parâmetros.

A classe *Session* contém rotinas de manutenção para garantir a confiabilidade, desempenho e disponibilidade do sistema. As rotinas são executadas periodicamente com o objetivo de libertar recursos (por exemplo, memória) e verificar se os parâmetros de configuração correspondem com o estado corrente.

A classe *Image Processor* contém métodos de processamento visual, como por exemplo, transformação de perspetiva, aplicação de filtros, monitorização do ponto laser,

entre outros. Sendo as funcionalidades suportadas através dos métodos da biblioteca OpenCV.

A classe *Camera* contém informações de configuração do sensor de câmara, permitindo o controlo e obtenção de imagens em tempo real.

A classe *Dot* e a classe *Gesture* contêm informações relacionadas com uma instância da classe *Laser*. Um objeto *Dot* corresponde ao ponto laser num determinado instante, e um objeto *Gesture* corresponde a um conjunto de objetos *Dots* que levaram ao reconhecimento de um gesto.

Por outro lado, as classes contidas no Middleware, seguem o paradigma Publish-Subscribe, mas readaptado para ir de encontro com as necessidades de comunicação. O produtor e o subscritor, apenas precisam conhecer o intermediário de mensagens que é responsável por notificar quem estiver inscrito no tópico referenciado. A classe *Consumer* fica à espera de mensagens provenientes do navegador web, através do intermediário de mensagens (broker). Por outro lado, a classe *Producer* é quem prepara a resposta para o intermediário para em seguida ser entregue ao cliente (navegador web).

Na Figura 19, encontra-se representado, as classes da camada de middleware do Sparks Core. Esta representação, ilustra a forma de comunicação, quando chega uma mensagem do exterior. Por exemplo, o processo Sparks Web envia uma mensagem para o Sparks Core, a classe Consumer é notificada através do intermediário (broker), o evento é armazenado na classe Database, e posteriormente o produtor (Producer) é notificado da ocorrência de uma nova mensagem. Assim, tanto o Consumer e o Producer podem aceder à classe Database em termos dos métodos e dos dados.

O Produtor verifica no cabeçalho da mensagem o tipo de evento que ocorreu, através do atributo *State*. Desta forma, o Produtor determina a classe interna do Sparks Core, que irá produzir o resultado a ser reencaminhado para o intermediário.

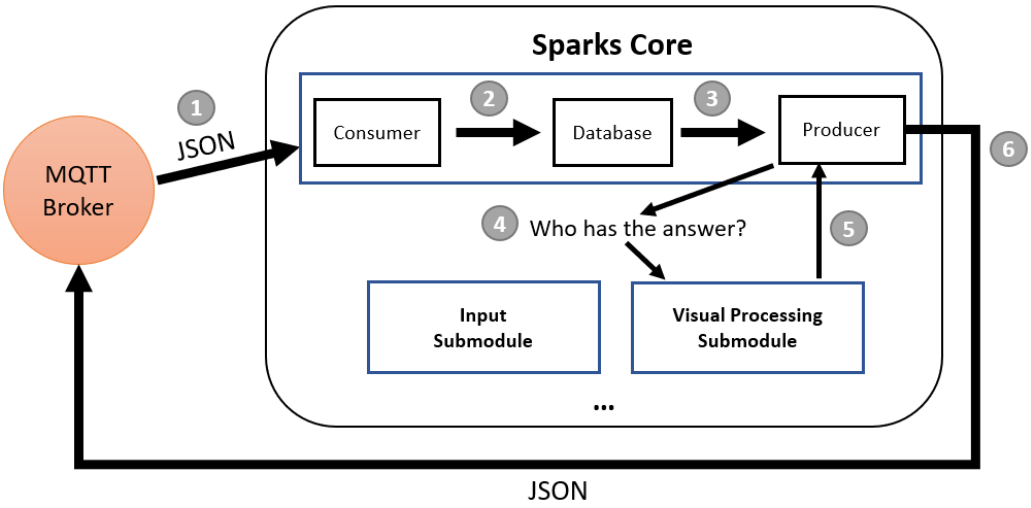

Figura 19 - Funcionamento do middleware do Sparks Core #1

Na Figura 20, apresentamos outra forma que o produtor pode utilizar para desencadear uma mensagem para o intermediário. Esta situação ocorre quando as classes internas do Sparks Core desencadeiam um evento, sem qualquer pedido prévio, e que necessita de ser reencaminhado para o Sparks Web. Por exemplo, quando um gesto é detetado sobre a tela, quando a posição da tela não corresponde com os dados de calibração armazenados, entre outros.

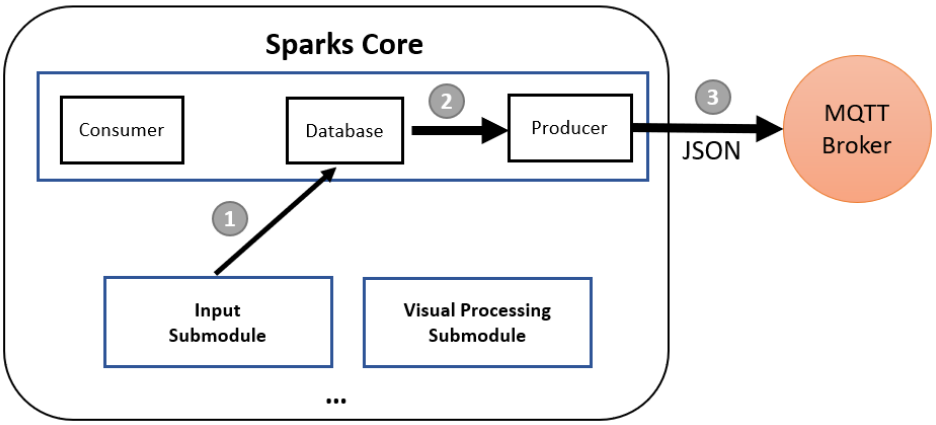

Figura 20 - Funcionamento do middleware do Sparks Core #2

Uma vez que existe uma troca constante de dados com o intermediário, a sincronização será utilizada para garantir a coerência e integridade dos dados. Por outro lado, na Figura 21, encontra-se ilustrado o conceito *Master-Slave*, mas readaptado para o contexto do projeto. O processo *Sparks Core* atua como mestre e o processo *Sparks Web* como escravo. Esta representação permite-nos separar as responsabilidades e definir o *Sparks Core* como o processo principal de tomada de decisões. Os pedidos com origem no *Sparks Core*, são geralmente ordens a ser executadas no processo *Sparks Web*. As ordens podem ser, por

exemplo, restaurar a página inicial do sistema após um período de inatividade, forçar a recalibração do sistema, entre outros.

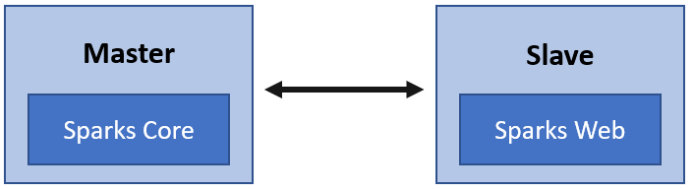

Figura 21 - Paradigma Master-Slave entre a comunicação dos processos

Este paradigma abre portas à *escalabilidade*, permitindo que novos recursos sejam adicionados e controlados por um elemento central. Esta abordagem, também foca a *concorrência* uma vez que a adição de vários escravos favorece o processamento simultâneo. Apesar das vantagens, é de salientar que se o mestre ou o agente intermediário (broker) não responder, o escravo estará limitado quanto ao seu funcionamento.

3.5.2. Sparks Web

Na Figura 22, encontra-se representado a arquitetura do processo Sparks Web constituído pela aplicação web e o middleware.

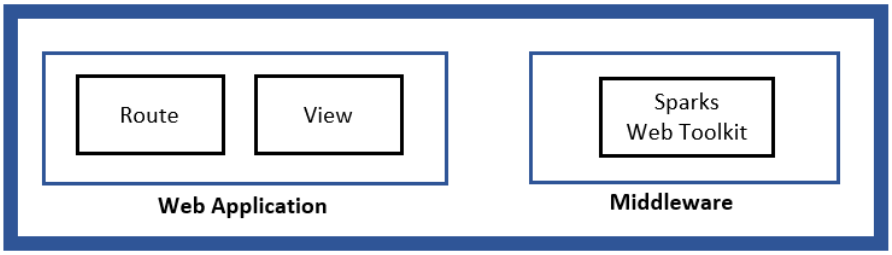

Figura 22 - Abstração do processo Sparks Web

A *aplicação web* é suportada pela framework *Flask* permitindo o desenvolvimento de uma estrutura minimalista e com poucas dependências [30]. Posteriormente, o desenvolvedor pode adicionar novas bibliotecas consoante a necessidade. Esta decisão arquitetural vem de encontro com o âmbito do projeto, uma vez que está a ser utilizado o Raspberry Pi 3B e pretende-se desempenho. Desta forma, a classe *Route* recebe as solicitações HTTP e indica a ação a ser executada para um dado recurso. As solicitações podem variar quanto ao método de requisição: GET, POST, PUT e DELETE. A classe *View* é a resposta que a framework envia para o navegador do cliente, normalmente um documento HTML, mas também pode ser outro tipo de conteúdo.

O *Middleware* efetua a gestão dos dados e estabelece as regras de comunicação. Na Figura 23, encontra-se ilustrado a configuração planeada inicialmente. Nesta representação, podemos ver que o fluxo de informação proveniente do navegador passa pela framework

Flask. Esta abordagem apresenta benefícios em termos de segurança, uma vez que tudo passa por uma entidade controladora antes de chegar ao intermediário de mensagens. No entanto, introduz latência na comunicação, uma vez que os dados passam por duas entidades (Sparks Web e Broker) antes de chegar ao processo Sparks Core.

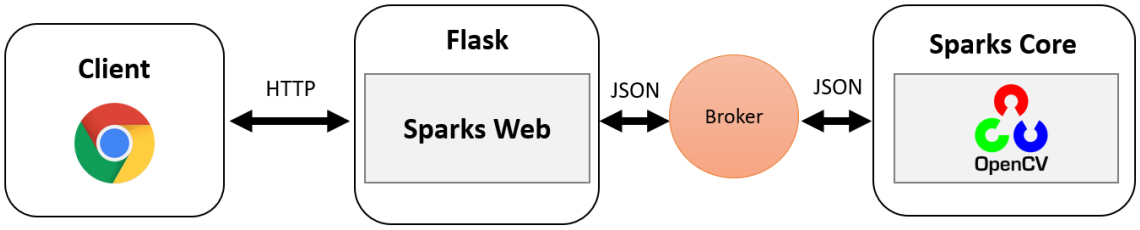

Figura 23 - Middleware (Sparks Web) – Configuração Inicial

Dado a situação descrita, constatou-se que seria necessário simplificar a arquitetura de comunicação. Na Figura 24, encontra-se ilustrado a arquitetura definida para maximizar o desempenho. O middleware que antes residia no Sparks Web, nesta fase, está implementado no lado do cliente (navegador). O que significa que os dados que não estão relacionados com o servidor web, são entregues diretamente ao intermediário de mensagens (broker), que por sua vez encaminha para o processo Sparks Core.

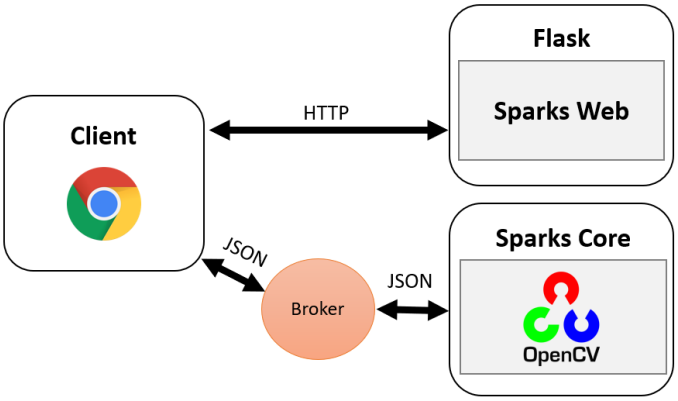

Figura 24 - Gestão dos dados (Sparks Web Toolkit)

Embora a camada de middleware do *Sparks Core* e do *Sparks Web*, estejam arquiteturalmente em localizações distintas, ambas têm o objetivo comum de troca de dados. Assim, para compreender melhor o que tem vindo a ser descrito, no tópico posterior esta matéria será aprofundada.

3.5.2.1. Toolkit

No corrente tópico, será abordado a biblioteca em JavaScript denominada por *Sparks Web Toolkit*, parte integrante do Sparks Web. Na Figura 25, podemos ver uma representação do *Sparks Web Toolkit* integrado no navegador e a lidar com os dados provenientes do intermediário de mensagens (broker) ou do Sparks Web (Flask).

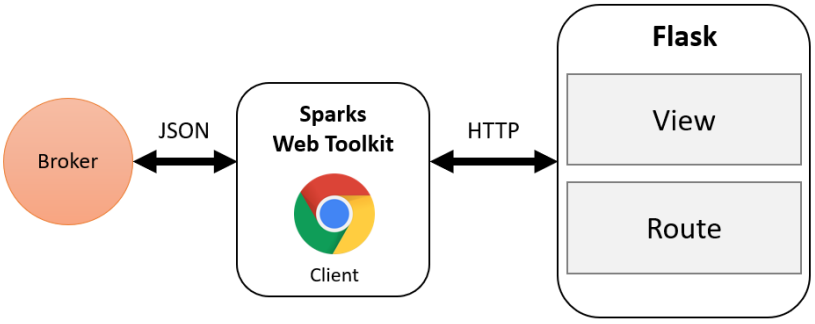

Figura 25 - Funcionamento do middleware do Sparks Web

Os dados no *Sparks Web Toolkit* podem ter como destino: o intermediário de mensagens (broker) ou a framework *Flask*. Quando o *Sparks Web Toolkit* envia os dados para o intermediário, significa que gerou uma resposta ou um pedido para o Sparks Core. Por outro lado, quando o *Sparks Web Toolkit* envia os dados para a framework Flask significa que podem existir informações provenientes do Sparks Core ou do Sparks *Web Toolkit* que devem ser transmitidas ao servidor web ou navegador.

O *Sparks Web Toolkit* pode ser equiparado com a classe *Controller* do padrão MVC (Model-View-Controller), agindo como a classe central que gere as solicitações provenientes do exterior ou interior do processo.

3.6. Broker

Neste tópico, será aprofundado o conceito do intermediário de mensagens, que até aqui tem vindo a ser mencionado em cada processo do Sparks. Para a comunicação entre processos é necessário um componente intermediário que coordene e distribua as mensagens para os subscritores. O broker é o elemento responsável por possibilitar a comunicação entre as partes, de forma a permanecerem ignorantes da identidade, localização e características do serviço de destino [27].

A utilização do broker oferece benefícios, mas constitui um ponto de falha porque se o broker não responder, todo o sistema será afetado. Na Figura 26, encontra-se representado a arquitetura proposta para cada um dos processos discutidos, incluindo o agente intermediário. Em termos de comunicação, o intermediário encontra-se interligado às camadas de middleware do Sparks Core e do Sparks Web.

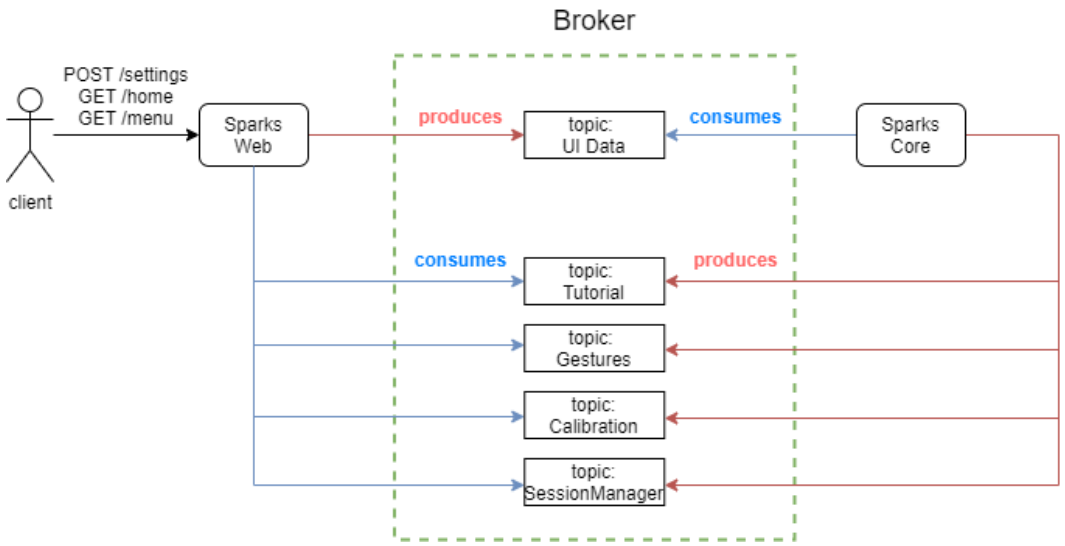

Figura 26 - Representação do intermediário de mensagens

O intermediário contém os canais de eventos, também conhecidos como filas de mensagens. Cada fila corresponde a um canal de comunicação independente, sendo o fluxo de mensagens unidirecional e com a informação organizada de acordo com o seu conteúdo. Esta abordagem permite gerir a informação e depurar mais facilmente a ocorrência de problemas. Sendo assim, existem três elementos distintos: o evento que corresponde a uma mudança de estado; o broker que contém as filas de eventos; e o processador do evento que consiste no processo de destino (*Sparks Web* ou *Sparks Core*) que irá receber e processar a mensagem.

O tópico *Calibration* contém todos os eventos que dizem respeito à etapa de calibração. As mensagens estão na maioria relacionadas com o processamento visual, parte fulcral do processo. Existem duas situações que podem desencadear o início do fluxo de mensagens, nomeadamente quando o sistema é inicializado ou quando existe a necessidade de recalibração.

O tópico *Gestures* contém todos os eventos que dizem respeito à ocorrência de um gesto. Um evento na fila Gestures, pode ocorrer se durante a interação com o sistema, o utilizador desencadear um movimento específico através do apontador laser.

O tópico *Tutorial* contém os eventos que dizem respeito ao processo de aprendizagem do sistema.

O tópico *Session Manager* contém os eventos responsáveis pela configuração do sistema, após a etapa de calibração. Neste canal, os eventos são informações de configuração ou ordens de execução, como por exemplo, forçar a recalibração do sistema.

O tópico *UI Data*, contém os eventos relacionados com o navegador web, como por exemplo, informações da página corrente, cookies, local storage, entre outros. Neste canal,

o fluxo de informação tem origem no *Sparks Web* e como destino o *Sparks Core*. O que significa, que uma mensagem pode ser uma resposta a um pedido do Sparks Core ou uma nova informação gerada pelo Sparks Web.

Como resultado, o MQTT (Message Queuing Telemetry Transport) foi o protocolo utilizado para garantir a comunicação entre os processos. O MQTT funciona sobre a arquitetura TCP/IP e suporta ambientes dinâmicos com enormes volumes de dados. Tem a particularidade de gerar pouca sobrecarga na rede devido à sua simplicidade, baixo consumo de recursos e geralmente utilizado em dispositivos com recursos limitados [31]. Além disso, possui mecanismos de qualidade de serviço (QoS) para garantir a disponibilidade e entrega de mensagens [32].

3.7. Rotinas de manutenção

Neste tópico, será abordado uma das partes integrantes do sistema, totalmente transparente para o utilizador, mas fundamental para garantir a eficiência, desempenho e fiabilidade do sistema. Independentemente dos recursos de hardware disponíveis, existem questões técnicas que implicam uma execução periódica de rotinas. As rotinas mencionadas, ocorrem devido à necessidade de gestão dos recursos, como por exemplo, eliminar eventos passados. No caso do Raspberry Pi, este assunto é ainda mais notório, uma vez que são dispositivos caracterizados pelo seu tamanho reduzido e baixo consumo elétrico [33].

O conjunto de eventos que se enquadram na situação descrita, são: gestos, coordenadas, quadros de imagem, estado do sistema, entre outros. A acumulação dos dados gera sobrecarga, e ao fim de um tempo podem estar a ocupar recursos desnecessariamente. Por este motivo, é importante realizar uma verificação periódica do conjunto de dados que não transmite informação significativa e que pode ser eliminada. Geralmente, são eventos que foram armazenados por motivos de configuração, interpretação ou de reavaliação futura e no estado corrente não são necessários.

Devido aos diferentes fluxos associados ao Sparks, algumas responsabilidades foram segmentadas por threads. O mecanismo de threads permite que vários comportamentos sejam executados aparentemente de forma simultânea, permitindo concorrência dentro do sistema [34]. No *Sparks Core* existem as seguintes threads, cujas funções serão descritas:

- Consumer
- Producer
- Camera

- Session Manager

A thread *Consumer* e *Producer* são objetos ativos com o objetivo de receber e enviar mensagens, respetivamente. O *Consumer* fica à espera de mensagens do intermediário (broker), para posteriormente armazená-las em memória. O armazenamento dos dados, pode ser visto como uma fila, em que as mensagens são processadas conforme a ordem de chegada, seguindo o paradigma *FIFO*. Quando a resposta de um pedido está disponível, a thread *Producer* é notificada e reencaminha a mensagem para o intermediário.

As mensagens entre os processos *Sparks Web* e *Sparks Core* seguem a notação JSON, amplamente utilizada para a troca de dados entre os serviços [35]. A estrutura JSON é constituída por chave/valor, sendo a chave o nome do atributo e o valor, o significado. Assim que a mensagem é recebida no *Consumer* (Sparks Core), é verificado o atributo *State*, a fim de averiguar qual o módulo interno responsável pelo tratamento. Desta forma, é possível filtrar e encaminhar a mensagem para o módulo adequado. Em seguida, o resultado é empacotado em uma mensagem, ficando a cargo da thread *Producer* o reencaminhamento da resposta para o broker.

A thread *Camera* é responsável por controlar e monitorizar o sensor de câmara, estando a cada momento a verificar o estado do sensor e a obter os quadros de imagem. Cada quadro é armazenado e processado no módulo *Image Processor*, com o intuito de extrair informações para fins de tomada de decisão.

A thread *Session Manager* é responsável por garantir a eficiência, fiabilidade e desempenho do sistema. Esta thread executa funções de libertação dos recursos, eliminando os quadros de imagem, coordenadas e gestos antigos, que no estado corrente não representam informação significativa. Para permitir a identificação dos eventos que podem ser eliminados, cada evento contém um atributo data/hora, permitindo comparar posteriormente com os parâmetros temporais. Por exemplo, se um gesto ocorreu há 20 segundos, no corrente instante, pode não representar informação significativa e assim ser eliminado. A thread de gestão também pode restaurar o estado do sistema, por exemplo, se o utilizador se encontra numa página e durante um período específico não ocorreu qualquer interação, então considera-se que o sistema deve retornar ao seu estado inicial.

3.8. Sumário

Neste capítulo foi apresentado o desenho de software elaborado para este sistema. Na fase inicial definiu-se os requisitos funcionais e não funcionais, os cenários de modificabilidade, desempenho e disponibilidade, e as respetivas táticas arquiteturais. Assim,

houve a necessidade de identificar os pontos críticos do sistema e preparar previamente táticas de recuperação dado o âmbito de utilização do sistema em tempo real. Na fase seguinte, procedemos a uma descrição técnica dos processos que compõem o sistema, descrevendo detalhadamente a função específica de cada módulo e a forma como as entidades do sistema se relacionam.

Dado a utilização do sistema em tempo real, é necessário garantir que um evento processado é posteriormente eliminado. Este aspeto é de elevada importância, uma vez que a aglomeração de informação na fila de eventos pode degradar a performance do sistema e gerar sobrecarga. Da mesma forma que um evento é processado, é necessário garantir que a informação associada também é eliminada. Existem informações que são mantidas por algum tempo, como por exemplo, as coordenadas do ponto laser, os quadros de imagem capturados, entre outros. Neste caso específico, quando existe um número elevado de informações, o comportamento passa por eliminar os dados por ordem cronológica (os mais antigos primeiro).

Para garantir a interligação dos processos comunicantes, houve a necessidade de implementar um broker de mensagens, responsável pela coordenação e encaminhamento da informação. Por fim, passou-se ao desenho da arquitetura do sistema, afim, de criar uma estrutura que interligasse todos os componentes necessários para que o sistema pudesse funcionar.

4. Desenvolvimento do sistema

No corrente capítulo serão descritos os desafios envolvidos e as técnicas de processamento de imagens utilizadas para a implementação da camada de processamento visual no processo Sparks Core. Assim, será realizado uma descrição do raciocínio utilizado para a interligação do Sparks Core com o processo Sparks Web, em termos lógicos e computacionais.

Antes de começarmos a abordar as partes integrantes é importante salientar que a revisão de literatura foi crucial para a escolha e definição das estratégias a seguir, permitindo estar um passo adiante dos possíveis problemas que poderiam ocorrer.

4.1. Deteção de bordas

As bordas são recursos relevantes e fornecem informações que permitem reconhecer mudanças dentro de uma imagem [36]. Como resultado, os algoritmos são codificados para encontrar as bordas através das mudanças que ocorrem nos níveis de intensidade dos pixels [37].

A deteção das bordas é uma das operações mais utilizadas na visão computacional e se as bordas forem identificadas com sucesso, os objetos podem ser localizados e identificados em termos das características geométricas e físicas [38]. Desta forma, o algoritmo Canny Edge Detector desenvolvido por John F. Canny [39], foi utilizado no projeto para a deteção das bordas da tela.

O operador de Canny inicialmente aplica um filtro de remoção de ruído através de uma máscara gaussiana. De seguida, calcula os gradientes de intensidade da imagem, cujos gradientes com intensidade mais elevada são os mais prováveis de serem considerados parte de uma borda. Posteriormente, o algoritmo aplica um limiar duplo para identificar 3 tipos de pixels: fortes, fracos e não relevantes. Os pixels fortes têm uma intensidade alta que permitem ter a certeza de que fazem parte de uma borda. Os pixels fracos têm um valor de intensidade baixo, mas não são pequenos o suficiente para serem considerados irrelevantes, e por fim, os pixels classificados como não relevantes são descartados [40].

Na Figura 27, na imagem à esquerda, podemos visualizar o resultado obtido através do operador de Canny. Nesta ilustração podemos observar a capacidade do algoritmo para identificar as diferentes bordas que se encontram no ambiente.

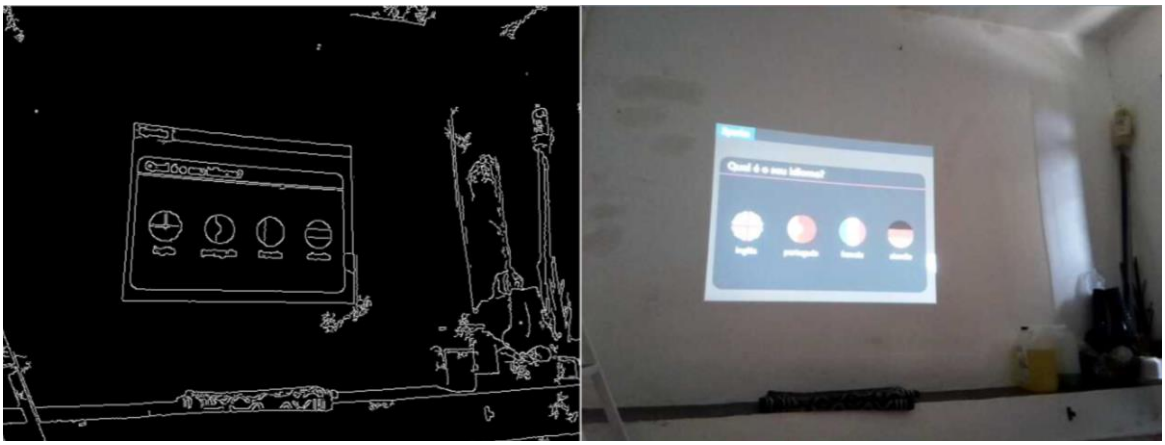

Figura 27 – Resultado do operador de deteção de bordas

No entanto, o que realmente nos interessa é a identificação da tela projetada que será detalhada no tópico processo de calibração. Servindo este tópico apenas como introdução para um dos principais desafios enfrentados durante o desenvolvimento.

4.2. Deteção do ponto laser

Existem diferentes abordagens para a deteção do ponto laser e talvez a mais simples é assumir que é a circunferência com os níveis mais altos de luminosidade na imagem capturada. Teoricamente, esta suposição aparenta ser suficiente, contudo, dado as características da câmara ou do ambiente, a deteção do ponto laser pode ser facilmente condicionada. Por outro lado, a suposição de extrair todos os valores dos pixels na imagem e comparar com um intervalo de cores pré-estabelecido, pode não ser suficiente porque se o ponto laser estiver sobreposto numa cor parecida ou com altos níveis de luminosidade, a deteção do ponto laser pode ficar igualmente condicionada. Devido ao excesso de luz que entra na lente da câmara, o algoritmo de deteção pode não conseguir distinguir o ponto laser das outras partes. O facto do ponto laser se mover rapidamente, os pixels que representam o ponto laser tendem a "espalhar-se" [41] ficando mais distantes, e em detrimento do ruído, o algoritmo pode estar suscetível a falhar.

A identificação do ponto laser implica o reconhecimento de um padrão durante uma sequência de imagens. A procura da sequência sem a utilização de um conhecimento prévio diminui a precisão e eleva o custo computacional. Assim, é importante dotar um conhecimento sobre as caraterísticas do alvo a monitorizar. Embora a biblioteca OpenCV disponibilize diferentes técnicas de identificação de objetos, é necessário ter em atenção que estamos a trabalhar sobre um dispositivo com recursos limitados e futuramente existirão outras tarefas que irão consumir uma quantidade considerada de recursos.

Para a deteção do ponto laser, inicialmente é utilizado um conjunto de filtros com o objetivo de reduzir o ruído da imagem, seguidamente, uma técnica de subtração de fundo é utilizada para reduzir o número de elementos que se encontram na imagem capturada. A técnica de subtração consiste em compor um quadro de imagem que não pode conter objetos estáticos, como resultado, é calculada a diferença pixel a pixel entre o quadro de imagem atual e o quadro de imagem anterior. Caso esta diferença seja menor que um limite definido, esse pixel pertence ao fundo, caso contrário, pertence a um objeto detetado [42].

Na Figura 28, na imagem à esquerda, é possível visualizar o resultado do método de subtração de fundo, que consiste em subtrair os quadros de imagem (possivelmente vários quadros depois) e, em seguida, analisar as mudanças que ocorreram entre as capturas.

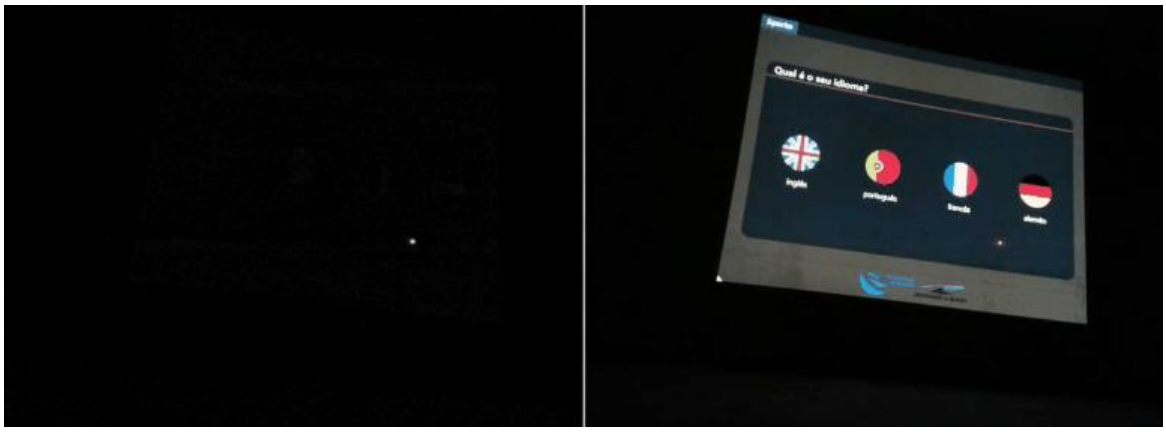

Figura 28 - Técnica de subtração de fundo

Uma vez que o apontador laser está sujeito à instabilidade da mão do utilizador, o ponto laser dificilmente está na mesma posição de quadro para quadro, o que nos permite diminuir o número de elementos a segmentar. Posteriormente, obtém-se um conjunto de segmentos com diferentes propriedades, e nesta fase, é importante procurar pelos segmentos que possuem as menores áreas e cuja representação se assemelhe com uma circunferência. No entanto, devido à instabilidade da mão do utilizador, a assunção que o ponto laser é uma circunferência nem sempre corresponde à realidade. Assim, quando um movimento é realizado rapidamente, a representação do ponto laser pode assumir outra forma, como por exemplo, uma reta e assim dificultar a análise.

4.3. Rastreio do ponto laser

Nesta fase, iremos começar por abordar, a lógica aplicada para a monitorização do ponto laser durante a utilização do sistema. Esta etapa é de elevada importância, uma vez que os gestos podem ocorrer no interior da tela e não existe uma forma direta que os permita diferenciar, tal como acontece nos gestos realizados nas bordas da tela. Assim será descrito

a técnica utilizada para o reconhecimento de gestos no interior da tela, com o auxílio da técnica de transformação de perspetiva que será descrita mais adiante.

Na Figura 29, estamos perante uma captura de imagem com uma resolução de 640x480. As dimensões da resolução foram divididas por um fator de escala (80), podendo ser representada por uma matriz 6x8, com 6 linhas e 8 colunas, respetivamente. O fator de escala deve ser um número cuja divisão pela largura e altura da imagem retorne um valor inteiro. Portanto, do ponto de vista computacional, a imagem capturada pode ser vista como uma matriz, onde cada linha e coluna pode ser representada da forma ilustrada.

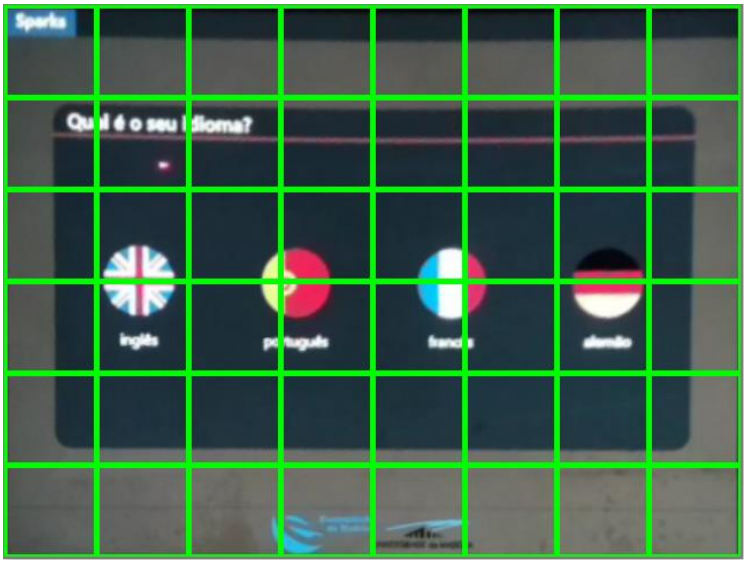

Figura 29 - Matriz desenhada sobre a imagem capturada

O objetivo da representação é permitir que informações como a localização do ponto laser, a célula da matriz sobreposta e outras informações, possam ser armazenadas para análise. Assim, mantendo o histórico de células sobrepostas pelo ponto laser, torna-se possível identificar padrões que quando conjugados podem ter um significado e corresponder a um gesto no interior da tela.

Na Figura 30, é possível visualizar o histórico do apontador laser durante um movimento horizontal. Desta forma, constata-se que ocorreu um movimento com origem na célula (2,3) até a célula (2,6), sendo o presente caso, uma representação da lógica aplicada. Contudo, no tópico posterior as identificações das células da matriz serão clarificadas para transmitir ao leitor o raciocínio aplicado.

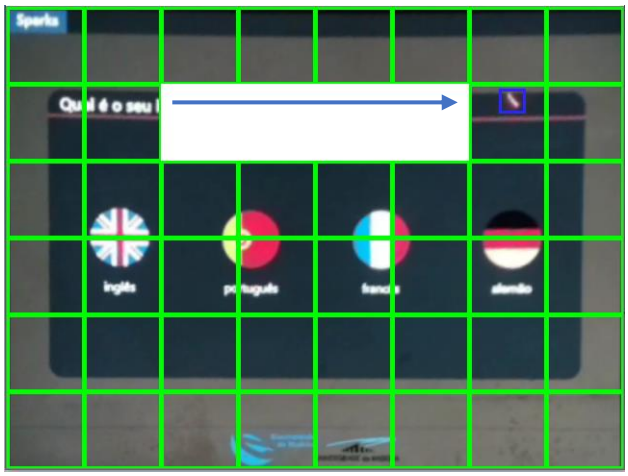

Figura 30 - Histórico do ponto laser sobre a matriz (Fator de escala - 80)

Na Figura 31, é possível observar outro exemplo do percurso realizado pelo ponto laser, mas neste caso, foi utilizado o fator de escala (*40)*, o que significa que ao dividir o valor da resolução pelo fator de escala, obteve-se uma matriz 12x16, composta por 12 linhas e 16 colunas.

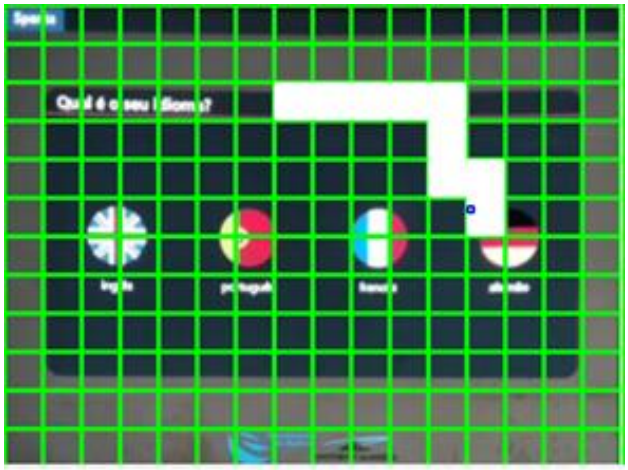

Figura 31 - Histórico do ponto laser sobre a matriz (Fator de escala - 40)

A principal diferença entre os fatores de escala, é que quanto menor for o valor, maior será o número de linhas e colunas da matriz, e consequentemente teremos mais células sobrepostas consoante o percurso do ponto laser. Assim, esta abordagem permite-nos maior sensibilidade durante a monitorização do ponto laser no interior da tela. Por outro lado, quanto maior for o valor do fator de escala, menor será o número de linhas e colunas da matriz.

4.4. Gestos circulares

De acordo com a literatura existem diferentes abordagens para simular o clique do rato através do apontador laser, como por exemplo, permanecer o ponto laser sobre o alvo,

desligar e ligar o apontador laser, entre outros. No entanto, estas estratégias forçam o utilizador a esperar, e por outro lado, para manter o ponto laser no mesmo local pode ser difícil devido à instabilidade da mão do utilizador [43].

A técnica utilizada no Sparks para simular o clique do rato tem por base o conceito definido por Kelvin e Kevin (2006), que consiste na realização de um gesto circular sobre o botão. De acordo com os autores, esta abordagem apresentou benefícios em termos de interação, não sendo um constrangimento para o utilizador.

Na Tabela 7, encontra-se representado a região da tela sobre a forma de uma matriz permitindo identificar cada célula e a ocorrência de um padrão. Neste caso específico, pretende-se detetar o gesto circular, através do padrão ilustrado nas células destacadas a laranja {(2,4), (2,5), (3,4), (3,5)}.

(1,1)	**(1,2)**	**(1,3)**	**(1,4)**	**(1,5)**	**(1,6)**	**(1,7)**	**(1,8)**
(2,1)	**(2,2)**	**(2,3)**	**(2,4)**	**(2,5)**	**(2,6)**	**(2,7)**	**(2,8)**
(3,1)	**(3,2)**	**(3,3)**	**(3,4)**	**(3,5)**	**(3,6)**	**(3,7)**	**(3,8)**
(4,1)	**(4,2)**	**(4,3)**	**(4,4)**	**(4,5)**	**(4,6)**	**(4,7)**	**(4,8)**
(5,1)	**(5,2)**	**(5,3)**	**(5,4)**	**(5,5)**	**(5,6)**	**(5,7)**	**(5,8)**
(6,1)	**(6,2)**	**(6,3)**	**(6,4)**	**(6,5)**	**(6,6)**	**(6,7)**	**(6,8)**

Tabela 7 - Identificação do gesto circular

Desta forma, espera-se que pelo menos quatro células estejam armazenadas no histórico do sistema e que o conjunto de células sobrepostas corresponda com a relação descrita na Ilustração 1. No entanto, mesmo que existam mais células armazenadas, o que realmente interessa, é que a condição ilustrada seja estabelecida. Assim, o reconhecimento do padrão é realizado pelo processo Sparks Core que em seguida comunica ao processo Sparks Web da ocorrência de um gesto circular.

$$\{ (x , y) , (x , (y + 1)) , \ ((x + 1), y) , ((x + 1), (y + 1)) \}$$

Ilustração 1 –Identificação de um gesto circular

Na Figura 32, encontra-se representado a ocorrência de um gesto circular, mas nesta fase é pretendido obter o ponto médio. Assim, sendo $\{(x_1, y_1), \dots, (x_4, y_4)\}$, as células da

matriz onde ocorreu o padrão, nesta fase é importante determinar o ponto médio para especificar ao navegador a localização do clique.

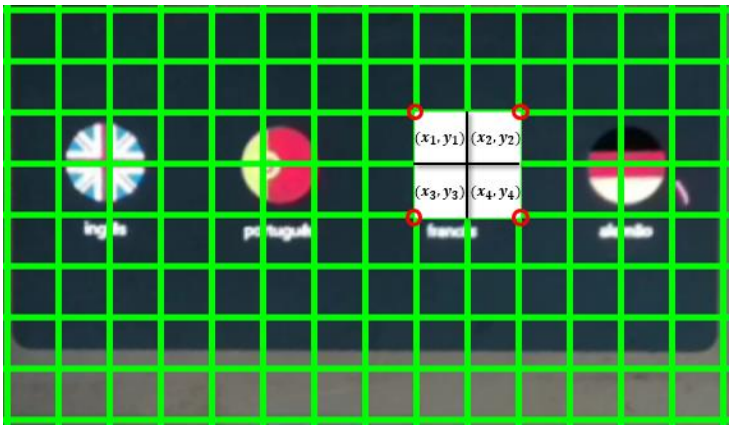

Figura 32 - Identificação do ponto médio do gesto circular

Assim, a Equação 1 ilustra a função *f* que representa o ponto médio (x, y) da região onde foi detetado o gesto circular.

$$f(x) = \frac{\sum_{n=1}^{4}(x_n)}{4} * Fator\ de\ escala \qquad f(y) = \frac{\sum_{n=1}^{4}(y_n)}{4} * Fator\ de\ escala$$

Equação 1 – Identificação do ponto médio do gesto circular

O resultado da equação permite obter as coordenadas do ponto médio, nomeadamente em termos da resolução da imagem onde deve ser efetuado o clique no navegador, tal como ilustrado na Figura 33.

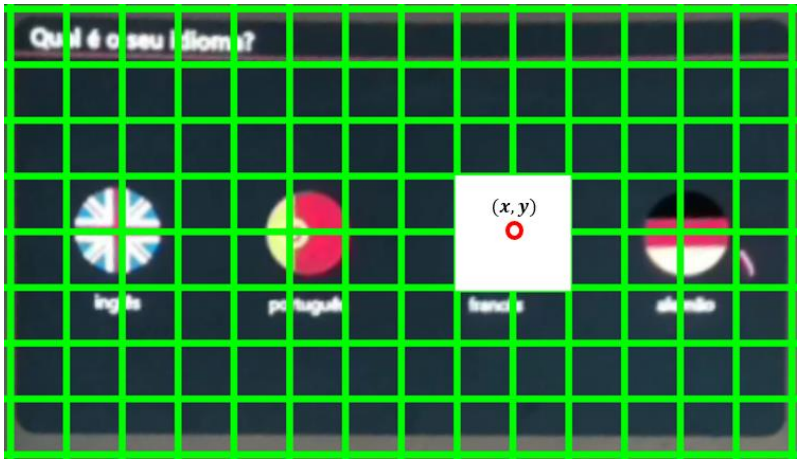

Figura 33 - Ponto médio do gesto circular

Nesta fase, as coordenadas calculadas correspondem à localização do elemento na página web. Posteriormente, a Figura 34 ilustra a informação recolhida durante a análise do gesto circular para ser entregue ao processo Sparks Web, através do intermediário de mensagens.

```
{'State': 'Gestures', 'TypeOfGesture': 'Circular', 'Coordinates': [320, 400]}
```

Figura 34 - Dados do gesto circular

4.5. Gestos nas bordas

Os gestos nas bordas têm por base o conceito definido por Shizuki et al. (2006) [1], o que permite utilizar as bordas da tela como fronteira para a identificação dos gestos, minimizando a execução de ações não pretendidas [1]. Na Tabela 8, encontra-se representado os gestos InOut e OutIn que podem ocorrer quando o ponto laser cruza uma das bordas da tela. Sendo *InOut* um movimento com início no interior da tela e fim no exterior, e *OutIn* um movimento com início no exterior da tela e fim no interior.

Borda da tela	**Gesto**	**Evento**
Direita	InOut	InOut-Right
	OutIn	OutIn-Right
Esquerda	InOut	InOut-Left
	OutIn	OutIn-Left
Superior	InOut	InOut-Top
	OutIn	OutIn-Top
Inferior	InOut	InOut-Bottom
	OutIn	InOut-Bottom

Tabela 8 - Eventos gerados no cruzamento de uma borda da tela

Esta abordagem possibilita a combinação de múltiplos eventos como ilustrado na Tabela 9. Por exemplo, para executar um gesto no canto superior esquerdo, primeiro teria de ser detetado um evento InOut-Left e seguidamente um evento OutIn-Top. Claramente, que para um evento ser considerado parte do anterior, teria de ocorrer dentro de um intervalo de tempo, como por exemplo, 2 segundos.

Eventos		
Evento 1	**Evento 2**	**Designação**
InOut-Right	OutIn-Top	CornerTop-Right
	OutIn-Bottom	CornerBottom-Right
InOut-Left	OutIn-Top	CornerTop-Left
	OutIn-Bottom	CornerBottom-Left
InOut-Top	OutIn-Right	CornerTop-Right
	OutIn-Left	CornerTop-Left

InOut-Bottom	OutIn-Right	CornerBottom-Right
	OutIn-Left	CornerBottom-Left

Tabela 9 - Exemplo de combinações possíveis utilizando as bordas

Dado o elevado número de combinações de eventos, decidiu-se restringir o número de gestos. Embora a abordagem permita a implementação de outras combinações, como por exemplo, três ou mais eventos seguidos, este número foi restrito para não ser um constrangimento durante a utilização do sistema.

Para compreender a identificação dos gestos, podemos imaginar a área da tela através de um sistema de coordenadas cartesiano, tal como ilustrado na Figura 35.

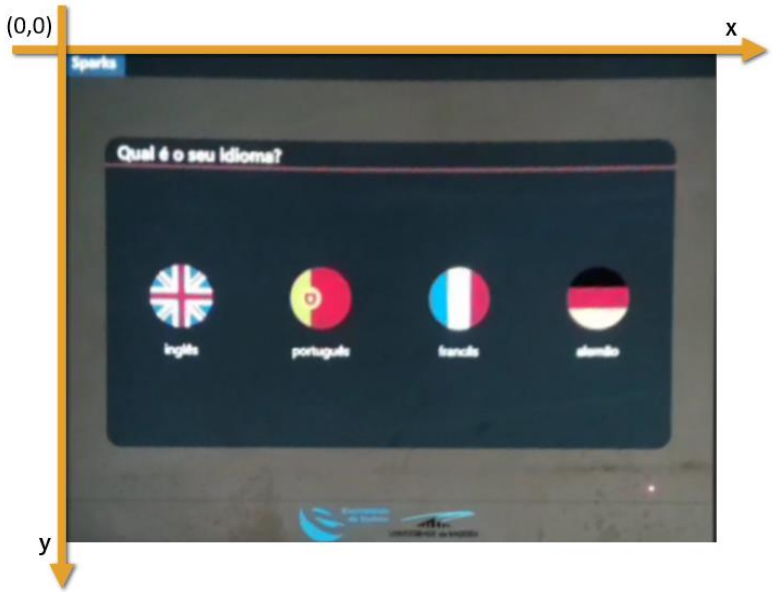

Figura 35 - Representação da imagem capturada num plano cartesiano

Durante a monitorização do ponto laser, verifica-se qual o eixo de coordenadas $(\boldsymbol{x}, \boldsymbol{y})$ que apresenta maior variação. Por exemplo, se num dado momento o ponto laser encontra-se a deslocar no sentido horizontal, o eixo do x apresentará maior variação, o que significa, que existe maior probabilidade de ocorrer um gesto na borda esquerda ou direita. A mesma lógica aplica-se para o eixo do y, o que significa que se a variação for maior no sentido vertical, existe maior probabilidade de ocorrer um gesto na borda superior ou inferior. Para verificar o eixo de coordenadas com maior variação compara-se as coordenadas do ponto laser a cada iteração e posteriormente verifica-se o tipo de gesto que ocorreu.

Na Tabela 10, temos uma representação do raciocínio descrito, onde as células a laranja identificam a ocorrência de um gesto InOut-Right, e as células a verde, a ocorrência

de um gesto OutIn-Right. Desta forma, quando o ponto laser está a deslocar-se na horizontal ocorre uma comparação entre as coordenadas do eixo do x do ponto laser e da borda direita.

Borda Direita (x, y)	**Ponto Laser** (x, y)
(503,400)	(411,200)
	(442,210)
	(463,206)
	(482,208)
	(549,212)
	(570,215)
	(596,206)
	(581,204)
	(508,200)
	(463,207)
	(429,209)
	(388,210)

Tabela 10 - Deteção de gestos na borda direita da tela

No entanto, existem situações que são necessárias prever e por isso garantir que as coordenadas dos eventos passados são eliminadas. Por exemplo, se um utilizador realiza um gesto InOut, desliga o apontador laser, e em seguida volta a ligar, mas com o ponto laser no interior da tela (sem cruzar nenhuma borda) poderia desencadear um gesto "falso positivo" (OutIn). Por este motivo, e como referido anteriormente é importante associar a cada gesto um evento temporal para prevenir a ocorrência de gestos involuntários.

4.6. Processo de Calibração

A calibração é o procedimento utilizado para determinar o conjunto de parâmetros necessários para garantir a confiabilidade e qualidade exigida durante a utilização do sistema. Durante o processo são conduzidas operações para qualificar se as condições do sistema e do ambiente estão de acordo com as especificações.

O processo de calibração utiliza o protocolo de mensagens MQTT para a comunicação entre os processos. O processo Sparks Web fornece informações visuais sobre o estado corrente do sistema e o processo Sparks Core é quem dá início à etapa de calibração e indica se é possível avançar para a etapa seguinte.

Seguidamente, será detalhado cada etapa do processo de calibração e os parâmetros necessários para garantir o correto funcionamento.

4.6.1 Conectividade

O teste de conetividade é a primeira etapa de calibração. Nesta fase, ocorre uma troca de mensagens entre os processos para determinar se existe conetividade entre as partes comunicantes. Esta etapa é de elevada importância, uma vez que durante todo o ciclo de vida do sistema são trocadas mensagens.

A cada iteração da etapa de calibração, o processo Sparks Core é responsável por analisar e confirmar se é possível avançar para a próxima etapa, sendo apenas a função do Sparks Web apresentar o resultado para o utilizador. Na Figura 36, é possível visualizar a primeira etapa de calibração, onde é efetuado um teste de conetividade entre os processos Sparks Core e Sparks Web.

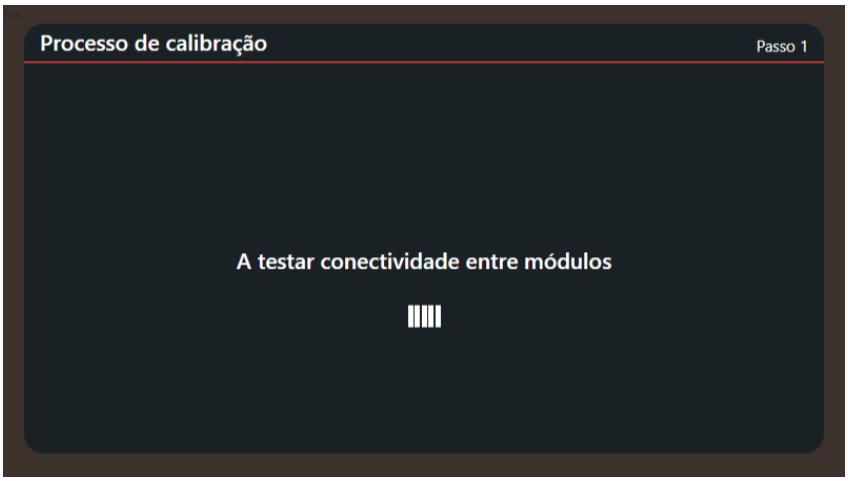

Figura 36 - Etapa de calibração (Teste de conetividade)

Se existir comunicação entre as partes (processos, módulos e sensores), o *Sparks Core* permite transitar para a próxima etapa de calibração.

4.6.2. Reconhecimento da tela

Após a verificação da conetividade entre os processos, a próxima etapa consiste em detetar a tela projetada. A deteção da tela constitui um processo fundamental porque a imagem capturada geralmente obtém tudo o que se encontra ao redor do ambiente. Desta forma, o algoritmo *Cany Edge Detector* é utilizado para identificar todos os segmentos da imagem e assim categorizá-la em dois planos. O primeiro plano consiste na região de interesse (a tela projetada) e o segundo plano tudo o que se encontra ao redor.

Na corrente etapa, espera-se que o Sparks realize o reconhecimento da tela de forma automática, contudo, o sucesso desta etapa depende das técnicas de segmentação. Nesta fase,

consoante os segmentos obtidos, procura-se pelo segmento com maior área e cuja forma se assemelhe com um retângulo.

Inicialmente a identificação da tela foi realizada sem qualquer suporte da interface gráfica, contudo, os resultados não foram os pretendidos. Na Figura 37, na imagem à direita, é possível verificar que as bordas da interface gráfica são segmentadas como se fossem os cantos da tela.

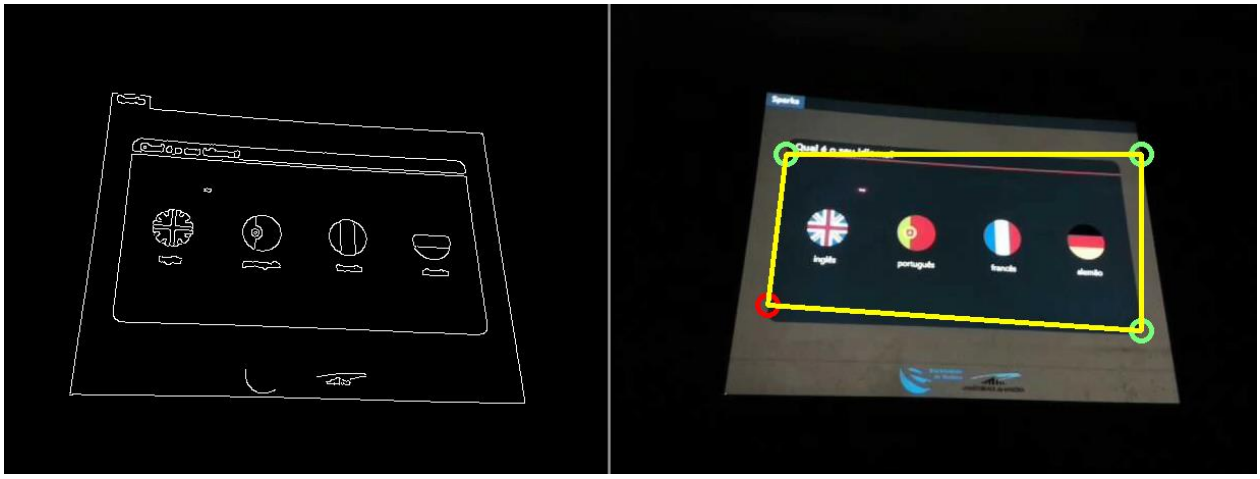

Figura 37 - Tentativa de identificação da tela sem sucesso

Após o problema mencionado, verificou-se que seria necessário utilizar a interface gráfica como suporte do processo de identificação. Na Figura 38, encontra-se ilustrado a primeira estratégia utilizada, que consistia em utilizar um fundo branco para criar um efeito de alta luminosidade, de forma a salientar a região de interesse. Embora a abordagem permitisse o resultado pretendido, a alta luminosidade gerada introduzia ruído nos quadros de imagem subsequentes, e em termos de interação com o utilizador transmitia a ideia de que o sistema estava a ter algum problema.

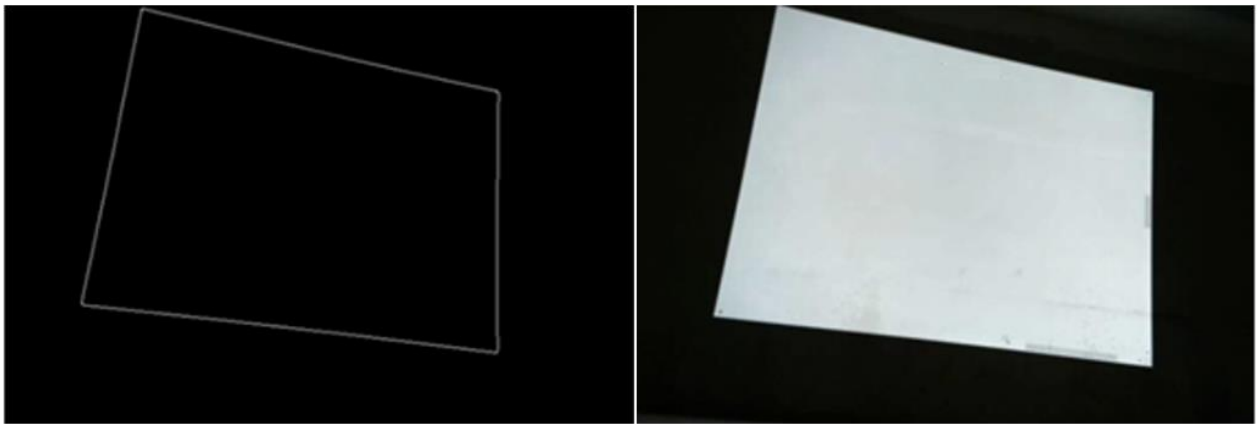

Figura 38 - Deteção da tela projetada (primeira estratégia)

Uma vez que a abordagem descrita introduziu incertezas e não apresentava *feedback* para o utilizador, decidiu-se descartar a estratégia mencionada. Deste modo,

procedemos à identificação da tela com o suporte de uma borda branca ao redor da página. Na Figura 39, na imagem à direita, é possível constatar uma borda branca ao redor da página. A borda era colocada durante dois segundos com o objetivo de realçar as bordas da tela durante a etapa de segmentação. Esta abordagem permite o utilizador estar informado acerca do estado corrente, e na imagem à esquerda, podemos verificar o resultado produzido pelo algoritmo Canny Edge Detector, onde se verifica um efeito de duplicação das bordas, o que fornece um padrão para a identificação da tela.

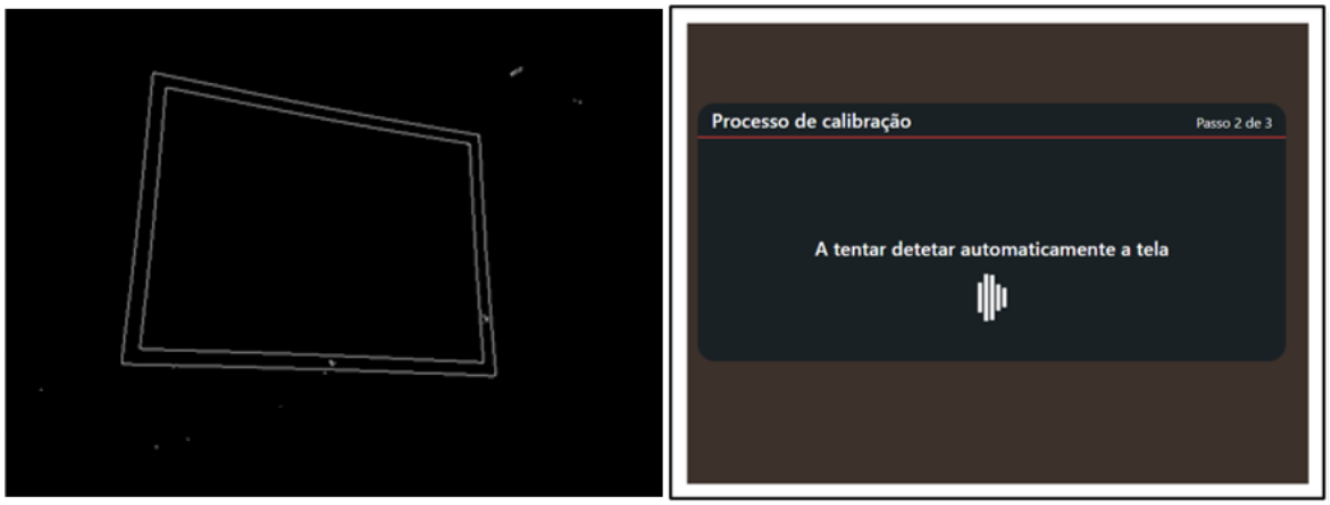

Figura 39 - Deteção da tela projetada através da utilização de uma borda

Nesta fase, o algoritmo Canny Edge Detector é utilizado para a segmentação da tela. Posteriormente, cada segmento é analisado individualmente com o objetivo de verificar se a área é composta por quatro lados. Contudo, somente esta suposição não é suficiente, uma vez que a localização da tela, geralmente sobreposta sobre uma parede ou outro plano, pode induzir o algoritmo a erro. Desta forma, a utilização da borda apresentou benefícios, uma vez que o efeito de borda dupla reduz a probabilidade, de por exemplo, uma parede vir a ser segmentada como a tela projetada. Esta abordagem, além de apresentar melhorias na deteção, é menos intrusiva para o utilizador, dado que permite continuar a ter perceção do estado corrente do sistema.

Nesta fase, espera-se que o Sparks seja autónomo, contudo, devemos estar cientes que existem fatores externos que podem dificultar o reconhecimento. Neste sentido, as condições ambientais são determinantes, uma vez que a quantidade de luz que afeta as lentes da câmara podem prejudicar a deteção. Assim, no próximo tópico será descrito o processo de calibração manual, caso a corrente etapa não seja bem-sucedida.

4.6.3. Calibração manual da tela

A situação ideal é que a tela projetada seja identificada automaticamente durante o processo de calibração. Contudo, o sistema está dependente de fatores externos que podem prejudicar o reconhecimento, e por este motivo, pode ser necessário a definição manual dos cantos da tela.

Conforme apresentado na Figura 40, o utilizador é indicado para posicionar o ponto laser sobre o canto superior esquerdo. Em seguida, deve realizar a mesma ação para cada um dos cantos, conforme indicado pelo sistema.

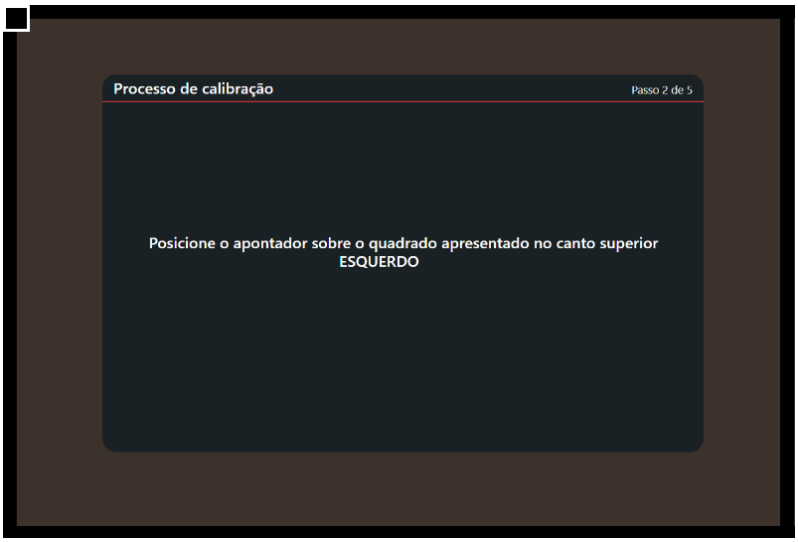

Figura 40 - Calibração manual da tela projetada

Embora esta abordagem possa resolver o problema de reconhecimento automático da tela, é sempre necessário garantir que o ambiente apresenta condições favoráveis porque se estivermos perante elevados níveis de luminosidade, as lentes da câmara serão afetadas e tornará a imagem capturada impercetível. Por outro lado, na impossibilidade de deteção da tela, o sistema será obrigado a terminar o processo de calibração, uma vez que as condições não foram reunidas para garantir a confiabilidade do sistema.

4.6.4. Deteção de colisões

A deteção de colisões é utilizada para assegurar que a tela projetada se encontra devidamente enquadrada com a posição da câmara. Esta etapa é de elevada importância, uma vez que se a câmara não estiver centrada com a projeção pode prejudicar a utilização do sistema.

Na Figura 41, é possível observar um exemplo, onde os cantos da tela estão no interior da imagem capturada, no entanto, se fosse permitido concluir o processo de

calibração na forma apresentada, no futuro, poderiam existir problemas na deteção dos gestos realizados na borda superior.

Figura 41 - Área que deve ser ocupada pela tela projetada

Assim, a tela projetada deve estar posicionada no interior da região azul para que se consiga assegurar a confiabilidade do sistema durante a execução dos gestos executados nas bordas. Por este motivo, a área a azul tem como objetivo auxiliar o utilizador a entender os limites que são utilizados para calibrar a posição da tela projetada que deve estar devidamente enquadrada com a câmara.

4.6.5. Transformação de perspetiva

A transformação de perspetiva é um mapeamento entre dois espaços geométricos permitindo transformar as imagens horizontalmente e verticalmente, rodar, ajustar a escala, entre outros [44]. Na Figura 42, podemos verificar na imagem à esquerda, o quadro de imagem capturado através da câmara. Se a partir desta situação, obtivéssemos as coordenadas do ponto laser e enviássemos para o navegador como uma ação de clique iriamos ter uma falha de precisão. Este motivo ocorre pelo facto do canto superior esquerdo da tela projetada (na imagem capturada) não estar alinhado com o canto superior esquerdo da imagem. Nos tópicos anteriores, vimos que uma imagem pode ser vista como um sistema de coordenadas cartesiano com origem no canto superior esquerdo. Desta forma, temos de garantir que a origem da tela coincide com a origem da imagem.

Assim, foi utilizado a função *cvPerspectiveTransform* da biblioteca OpenCV, onde foi indicado os pontos de origem a serem transformados para a matriz de destino, permitindo obter a transformação ilustrada na imagem à direita.

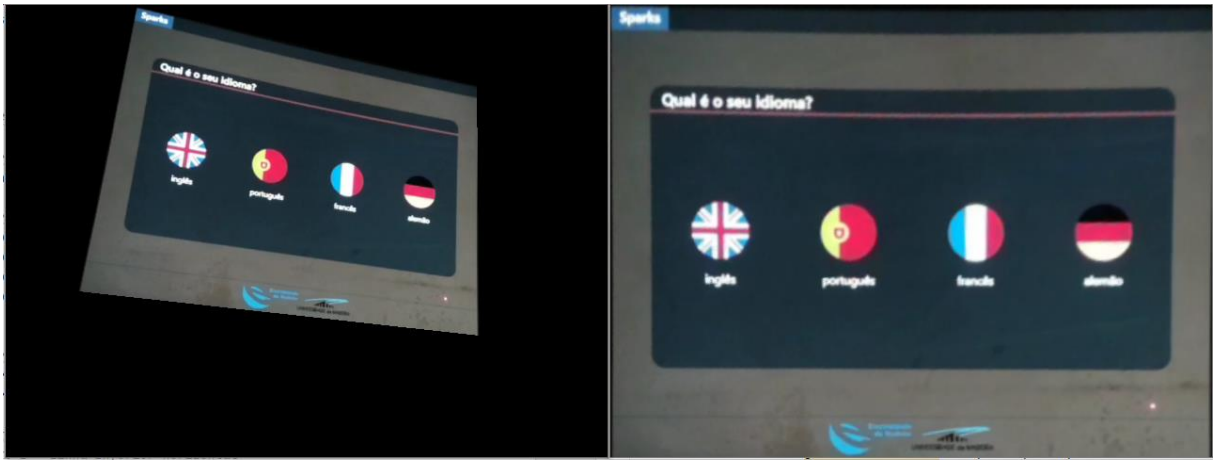

Figura 42 - Transformação de perspetiva

No contexto descrito, estamos a mapear os pontos de um plano bidimensional embutido em um espaço tridimensional, para outro conjunto de pontos de um plano bidimensional diferente.

4.7. Local Storage

Durante o processo de calibração é importante armazenar os dados obtidos. A Figura 43, ilustra os dados do processo de calibração armazenados no local storage do navegador. Deste modo, sempre que o sistema é inicializado, se existir informação no armazenamento, os dados são comparados com os dados correntes obtidos através da câmara. Nesta fase, verificamos se as coordenadas da tela permaneceram inalteradas desde a última calibração. Por outro lado, se não existir dados armazenados ou as coordenadas da tela não coincidem com as atuais, o processo de calibração necessita de ser realizado.

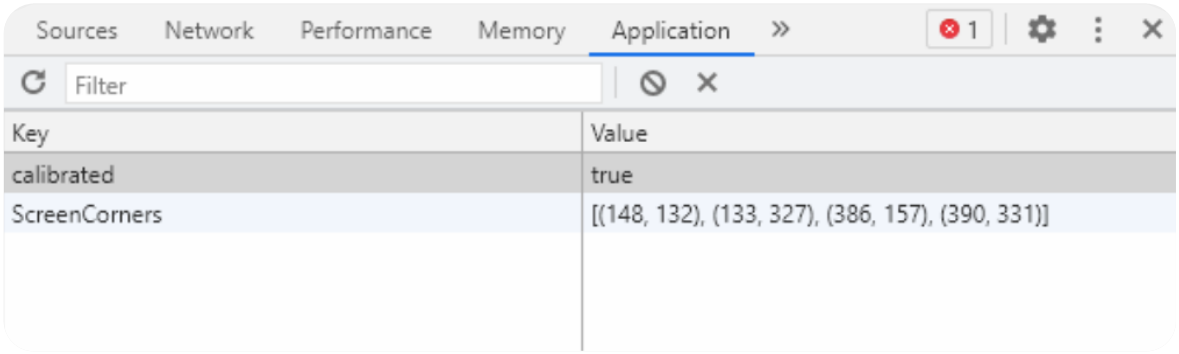

Figura 43 - Armazenamento dos dados de calibração

De salientar, que este processo acelerou o desenvolvimento do sistema porque toda a vez que o projeto era reinicializado, os parâmetros de configuração da última calibração eram na maioria das vezes reutilizados.

4.8. Interface Gráfica

Dado o âmbito do projeto, além da eficiência dos algoritmos de processamento visual e das técnicas de interação, é importante garantir que a interface gráfica é simples permitindo o utilizador saber como utilizá-la. Portanto, a interface deve ser projetada tendo em mente o apontador laser como dispositivo de entrada, e por este motivo, os elementos devem estar devidamente identificados e posicionados.

De acordo, com Shneiderman e Plaisant (2004), existem conceitos que devem ser preservados durante o desenvolvimento de uma interface gráfica, como por exemplo, o modelo mental, o modelo conceitual e a acessibilidade virtual [45]. O modelo mental é a representação que o utilizador desenvolve através da experiência e ajuda a decidir o que fazer em situações futuras. O modelo conceitual é o que o desenvolvedor tem em mente quando projeta o sistema, e tem como objetivo fornecer modelos mentais ao utilizador [46, p.]. A acessibilidade virtual refere que a interface gráfica deve fornecer sugestões ao utilizador [45].

Posto isto, e dado a existência de problemas de luminosidade relacionadas com a interface gráfica, desde cedo adotou-se a utilização de temas escuros nas páginas web. No entanto, este aspeto será descrito em detalhe no capítulo posterior. Contudo, a boa notícia é que os fundos escuros contrastam com praticamente qualquer tipo de cor, e com isso somos capazes de criar soluções elegantes e que cativam a atenção do utilizador.

4.9. Sumário

Neste capítulo foram apresentadas as técnicas de processamento de imagem utilizadas para segmentar as regiões de interesse, detetar e monitorizar o ponto laser, entre outros. Vimos que a deteção do ponto laser e a etapa de calibração necessitam de ser eficazes para garantir a confiabilidade do sistema, contudo, mais pesquisas poderão ser necessárias para determinar o melhor curso de ação.

Além das técnicas de processamento, foi descrito o raciocínio utilizado para a implementação dos gestos. Existem diferentes abordagens, contudo na maioria dos projetos revistos na literatura, os gestos eram utilizados sobre apresentações. Assim, a seleção dos gestos a implementar constituíram um desafio, uma vez que são o único meio de interação e por esse motivo devem ser facilmente manuseáveis pelo utilizador.

Por outro lado, e em complemento à situação descrita, a interface gráfica deve ser projetada tendo em mente o dispositivo de entrada. Por este motivo, os elementos da interface devem ser grandes o suficiente para facilitar a ação de clique. Desta forma, durante as reuniões constatou-se que as bordas da tela seriam o melhor recurso para a execução dos gestos, e para executar a ação de clique, o melhor movimento seria através do gesto circular.

A estrutura proposta tem vindo a ser testada exaustivamente para identificar o funcionamento e comportamento do sistema. Entretanto, é espectável que algum módulo venha a ser modificado no decorrer dos testes que serão descritos no próximo capítulo.

5. Experiências e resultados

Neste capítulo serão descritas as experiências realizadas para a validação do projeto e os resultados obtidos. Inicialmente serão descritas as dificuldades verificadas durante o desenvolvimento do sistema, e posteriormente serão mencionadas as experiências com os utilizadores de modo a identificar as limitações e oportunidades de melhoria.

Desta forma, as experiências serão agrupadas em duas categorias: os testes executados durante o desenvolvimento e os testes de usabilidade.

5.1. Testes durante o desenvolvimento

No início do desenvolvimento do projeto, o único material disponível era o apontador laser, então para poupar algum tempo decidi preparar um cenário que permitisse desde cedo avançar com o desenvolvimento dos algoritmos.

Na Figura 44, temos um exemplo de uma imagem capturada através do telemóvel, onde o ponto laser foi filmado a deslocar-se sobre a superfície de projeção. Em seguida, o vídeo foi interpretado com o objetivo de identificar e monitorizar o ponto laser e sempre que o mesmo cruzava alguma das bordas azuis, procedia-se ao reconhecimento de um gesto.

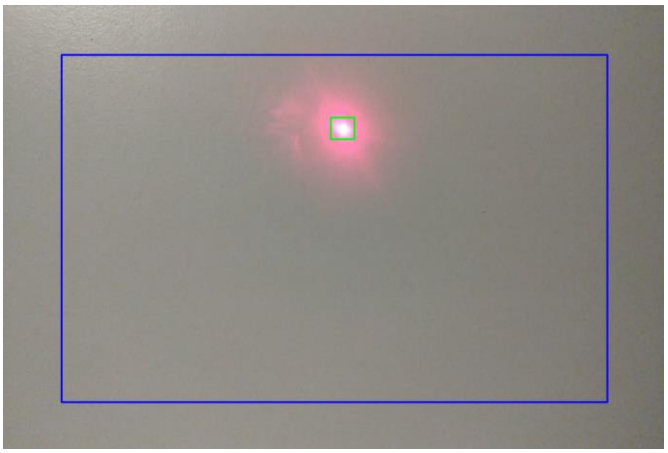

Figura 44 - Cenário de testes inicial

Assim, até antes da chegada do projetor, Raspberry Pi e sensor de câmara, alguns progressos já haviam sido realizados. Contudo, assim que o material em falta foi integrado, verificou-se que alguns dos algoritmos previamente programados não funcionavam como esperado. Esta situação ocorreu porque até à data estávamos a testar num ambiente simulado com fundo estático e cuja câmara tinha melhores características. No entanto, esta estratégia permitiu desde cedo testar diferentes cenários e algoritmos de processamento visual, que mais tarde viriam a ser aprimorados consoante os testes em tempo real que serão descritos adiante.

5.1.1. Ambiente de testes

De forma a criar as condições necessárias às experiências foram utilizados dois quartos com uma área útil de aproximadamente $10m^2$. Durante os testes, o projetor foi colocado a uma distância de 2,5 m da superfície de projeção e as janelas foram cobertas para impedir a entrada de luz. A particularidade dos ambientes de teste difere nos materiais que revestem as paredes. Na Figura 45, podemos verificar na imagem à esquerda que a parede se encontra na cor natural do reboco, enquanto na imagem à direita, a parede está pintada de branco.

Figura 45 - Ambientes de teste utilizados durante o desenvolvimento

As diferentes superfícies de projeção oferecidas pelos ambientes de testes foram úteis para o desenrolar deste capítulo, uma vez que permitiram compreender os problemas de luminosidade mencionados na literatura.

5.1.2. Efeitos da luminosidade

Neste tópico serão apresentadas as experiências realizadas com diferentes condições de luminosidade. Na Figura 46, podemos visualizar os quadros de imagem obtidos num ambiente noturno a partir dos diferentes ambientes de testes. Na imagem à esquerda, temos a captura de imagem a partir da parede não pintada, e na imagem à direita, a captura de imagem na parede branca. À primeira vista é possível constatar que o resultado obtido através da superfície não pintada permitiu obter baixos níveis de luminosidade, enquanto na parede pintada, temos uma imagem com altos níveis de luminosidade provenientes do reflexo da parede.

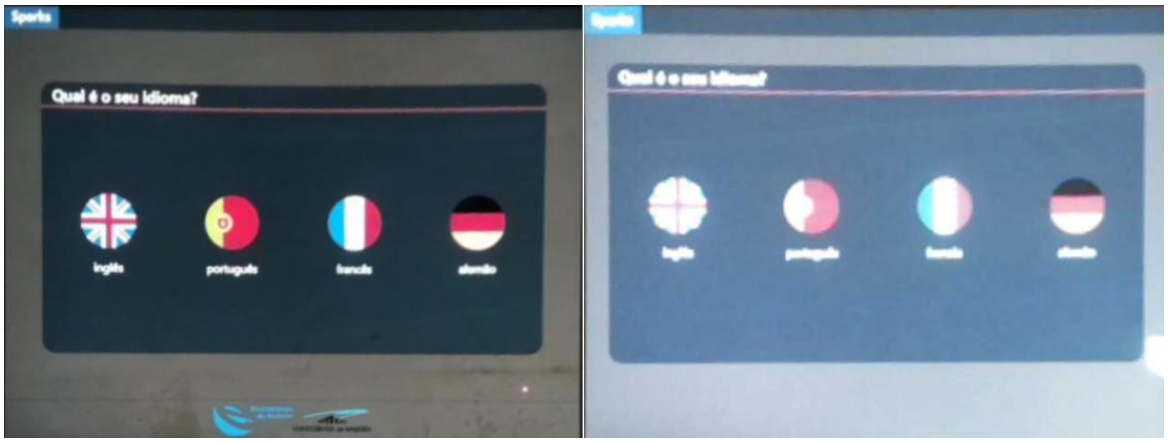

Figura 46 - Imagens capturadas sobre diferentes superfícies de projeção

Na Figura 47, temos um exemplo a ilustrar o problema da utilização de cores de alta luminosidade. Na imagem à esquerda, temos a página projetada e na imagem à direita, a captura de imagem através do sensor de câmara do Raspberry Pi. O ambiente encontrava-se totalmente escuro, contudo, as cores utilizadas geraram um efeito de alta luminosidade, introduzindo ruído e prejudicando a perceção do conteúdo da tela.

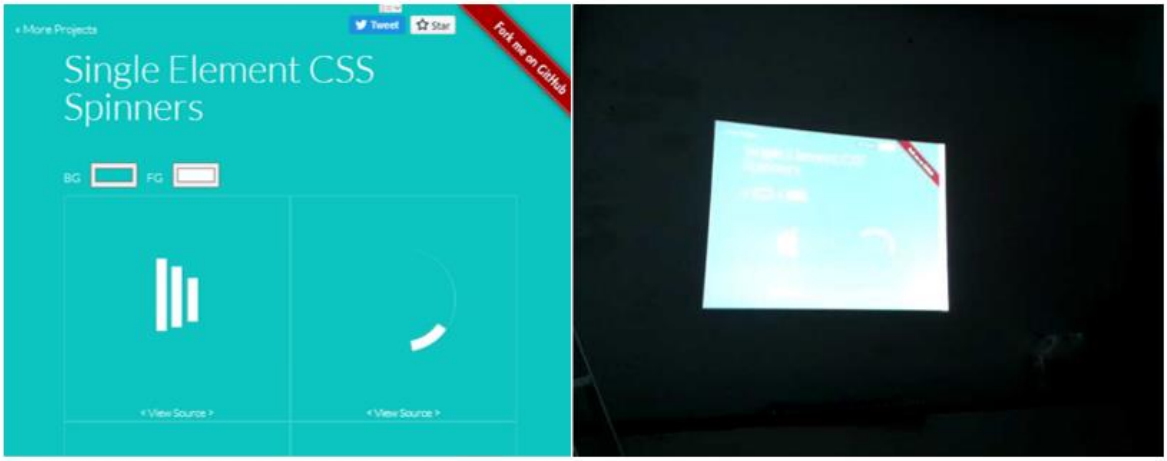

Figura 47 - Testes com cores de alta luminosidade

Deste modo, durante o desenvolvimento existiram situações que o algoritmo de deteção do ponto laser apresentou dificuldades no reconhecimento. Inicialmente, pensou-se que o problema estava relacionado com o algoritmo de deteção do ponto laser, contudo, após alguns testes verificou-se na imagem capturada que era impercetível a visualização do ponto laser quando sobreposto em cores de alta luminosidade.

Para entender o fenómeno descrito, irei ilustrar alguns exemplos. Na Figura 48, na imagem à esquerda, temos uma captura de ecrã, e na imagem à direita, a imagem capturada através do Raspberry Pi. Desta forma, é possível constatar que a cor de fundo da imagem capturada através do Raspberry Pi, pouco se compara com a cor inicialmente definida para projeção, transmitindo altos níveis de luminosidade.

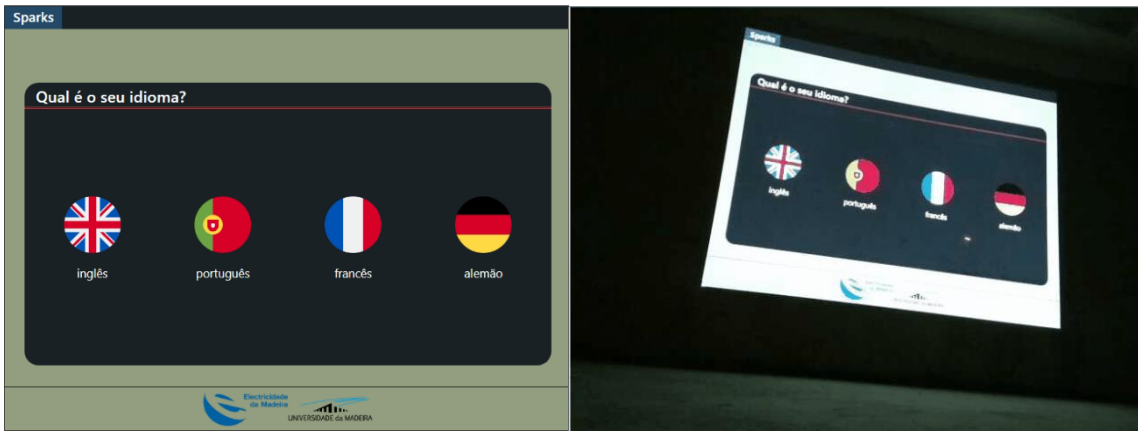

Figura 48 - Página de teste com cor de fundo verde

Por outro lado, na Figura 49, temos a mesma página, mas com um plano de fundo castanho. Igualmente se verifica na imagem capturada que a cor de fundo projetada não corresponde com a cor inicialmente definida, mas desta vez não se verifica um excessivo efeito de alta luminosidade.

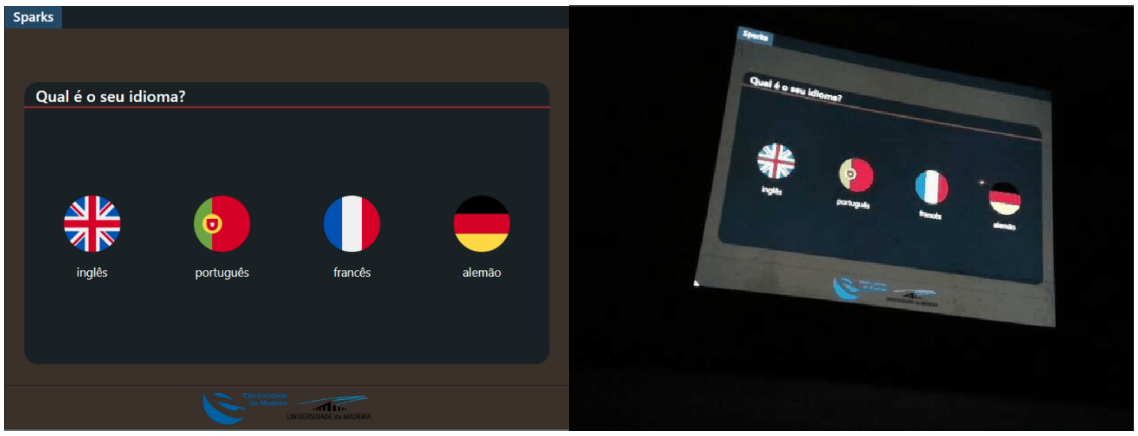

Figura 49 - Página de teste com cor de fundo castanho

Nas imagens apresentadas até agora, o ponto laser encontra-se sobreposto no formulário azul, posicionado estrategicamente para exemplificar o problema que será descrito adiante. Na fase que se segue, será ilustrado a mesma situação, mas desta vez com o ponto laser posicionado sobre as cores de fundo apresentadas.

Na Figura 50, na imagem à esquerda, podemos visualizar o algoritmo a filtrar somente o ponto laser, e na imagem à direita, um retângulo branco a delinear a localização do ponto laser sobre a tela.

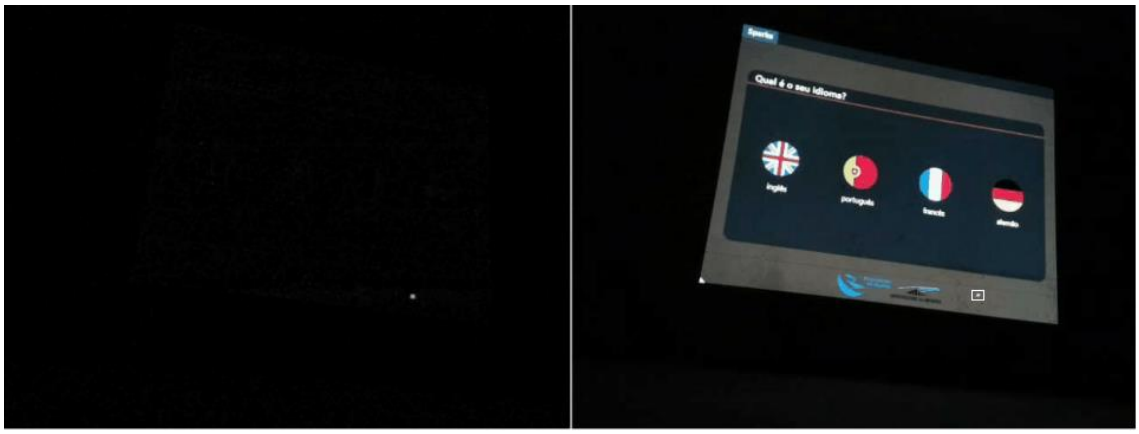

Figura 50 - Resultado da deteção do ponto laser sobre fundo castanho

Na Figura 51, a imagem à esquerda demonstra as dificuldades do algoritmo de deteção do ponto laser sobre o plano de fundo com elevados níveis de luminosidade. Por outro lado, na imagem à direita, a localização do ponto laser encontra-se destacada através de um retângulo vermelho, definido manualmente pelo autor, para elucidar a localização do ponto laser.

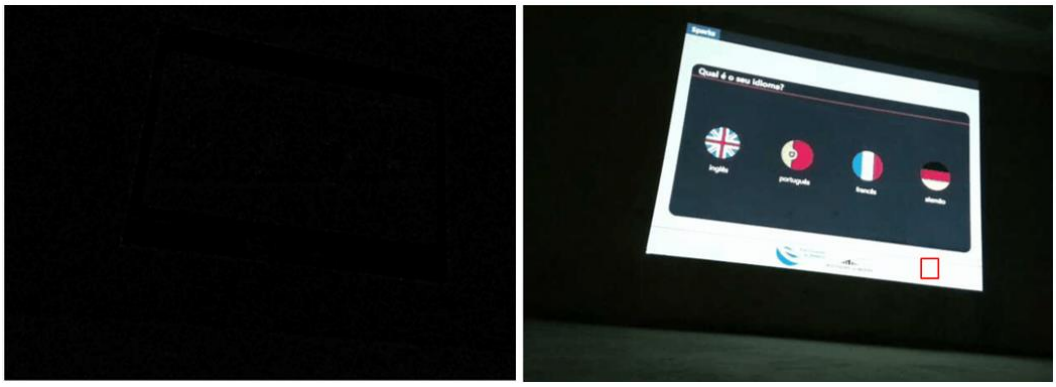

Figura 51 - Resultado da deteção do ponto laser sobre fundo verde

Embora os benefícios da parede não pintada em ambientes de baixa luminosidade, a Figura 52 ilustra um dos problemas quando o ambiente está exposto a luminosidade proveniente do exterior. Neste exemplo, a tela projetada encontra-se pouco percetível na imagem capturada e a utilização de temas escuros não apresenta benefícios. Demonstrando assim, que o sistema é facilmente condicionado pela alta luminosidade, pelo que no decorrer do documento iremos dar continuidade aos testes, mas utilizando ambientes escuros.

Figura 52 - Teste em ambiente diurno

5.1.3. Testes de Luminosidade

Neste tópico será comparado a visibilidade do ponto laser sobre os diferentes níveis de luminosidade, através da imagem capturada a partir dos diferentes sensores de câmara.

Dado a incapacidade de deteção do ponto laser em condições de alta luminosidade, será discutido no corrente tópico o conjunto de testes para estabelecer os valores de luminosidade que influenciam a confiabilidade do sistema. Como resultado, planeou-se as Tabelas 11 e 12 com vários níveis de luminosidade para diferentes cores.

Luminosidade	**Vermelho**			**Verde**			**Azul**		
	R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
90	255	204	204	204	255	204	204	204	255
80	255	153	153	153	255	153	153	153	255
70	255	102	102	102	255	102	102	102	255
60	255	51	51	51	255	51	51	51	255
50	255	0	0	0	255	0	0	0	255
40	204	0	0	0	204	0	0	0	204
30	153	0	0	0	153	0	0	0	153
20	102	0	0	0	102	0	0	0	102
10	51	0	0	0	51	0	0	0	51

Tabela 11 - Valores de luminosidade #1

Luminosidade	**Amarelo**			**Roxo**			**Azul**		
	R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
90	255	255	204	255	204	255	204	255	255
80	255	255	153	255	153	255	153	255	255
70	255	255	102	255	102	255	102	255	255
60	255	255	51	255	51	255	51	255	255
50	255	255	0	255	0	255	0	255	255
40	204	204	0	204	0	204	0	204	204
30	153	153	0	153	0	153	0	153	153
20	102	102	0	102	0	102	0	102	102
10	51	51	0	51	0	51	0	51	51

Tabela 12 - Valores de luminosidade #2

As duas tabelas anteriores permitiram obter a matriz de cores ilustrada na Figura 53. Assim, obteve-se uma matriz 6x9 constituída por 6 linhas e 9 colunas, onde cada linha representa uma cor e cada coluna um nível de luminosidade. A célula mais à esquerda constitui a célula com maior valor de luminosidade (90), e a célula mais à direita, a célula com o menor valor de luminosidade (10).

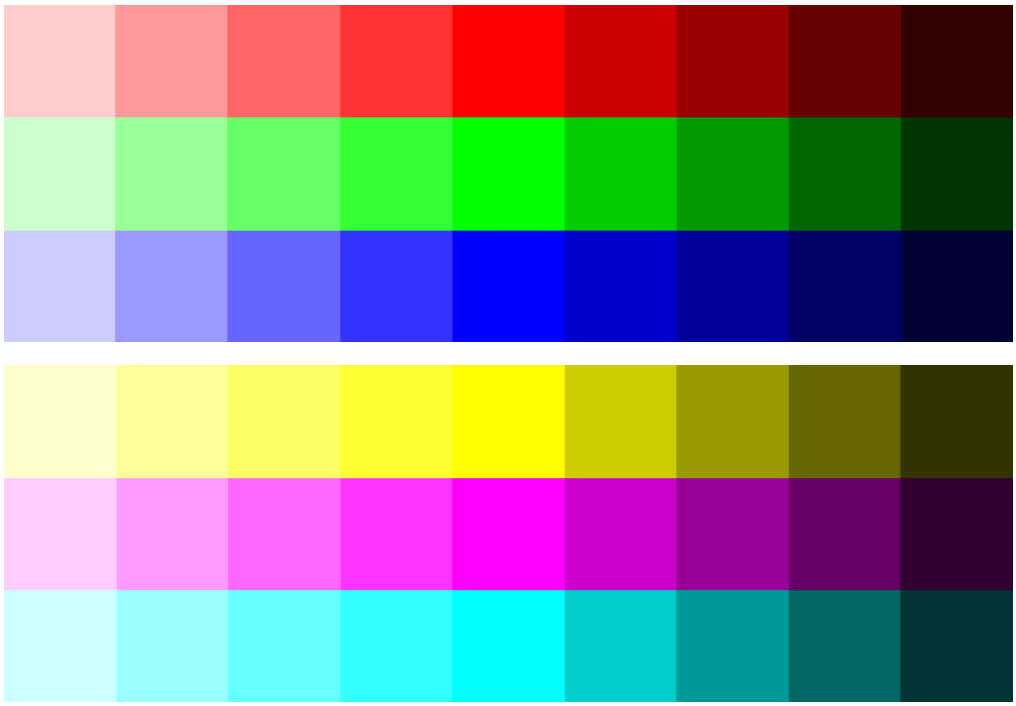

Figura 53 - Matriz de cores

Os testes em cada faixa de cor tiveram início com o ponto laser sobreposto na célula com menor valor de luminosidade, e de seguida posicionado nas células adjacentes, até se perder a visibilidade do ponto laser. Assim, os testes de seguida apresentados foram executados no ambiente sem a parede pintada, uma vez que se verificou que a parede branca expôs elevados níveis de reflexão.

5.1.3.1. Testes sem tela de projeção

A cada etapa deste tópico será comparado a visibilidade do ponto laser sobre os diferentes níveis de luminosidade através da imagem capturada a partir dos sensores do Raspberry Pi. Assim, no lado esquerdo teremos a imagem capturada através do sensor Pi NoIR Camera V2, e no lado direito, a captura através do sensor Pi Camera V2.

Na Figura 54, a imagem do sensor Pi NoIR Camera transmite altos níveis de luminosidade, contudo, o ponto laser encontra-se mais nítido quando comparado com a imagem gerada pelo sensor Pi Camera V2. No entanto, em ambas situações o ponto laser encontra-se visível e foi detetado pelo algoritmo.

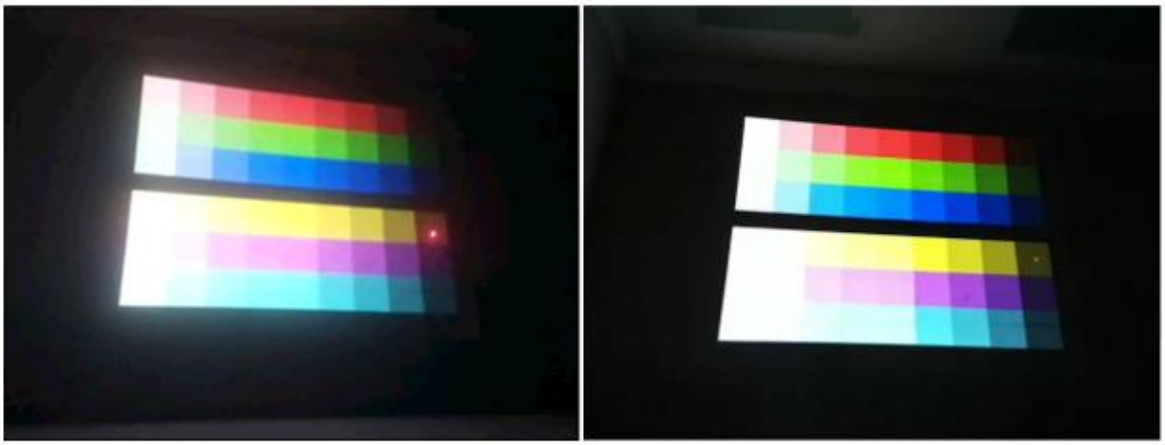

Figura 54 - Comparação entre os sensores Pi Camera #1

Na Figura 55, o ponto laser encontra-se sobreposto na célula amarela com valor de luminosidade (30), onde é possível verificar que a alta exposição gerada pelo sensor Pi NoIR começou a deteriorar a visibilidade do ponto laser. No entanto, em ambas situações é possível verificar a localização do ponto laser, sendo mais nítido, no sensor Pi NoIR Camera.

Figura 55 - Comparação entre os sensores Pi Camera #2

Na Figura 56, o ponto laser encontra-se sobreposto na célula amarela com valor de luminosidade (50). No entanto, pode-se constatar nas capturas que não é percetível a visualização do ponto laser nas células da faixa amarela com valor de luminosidade superior ou igual a 50.

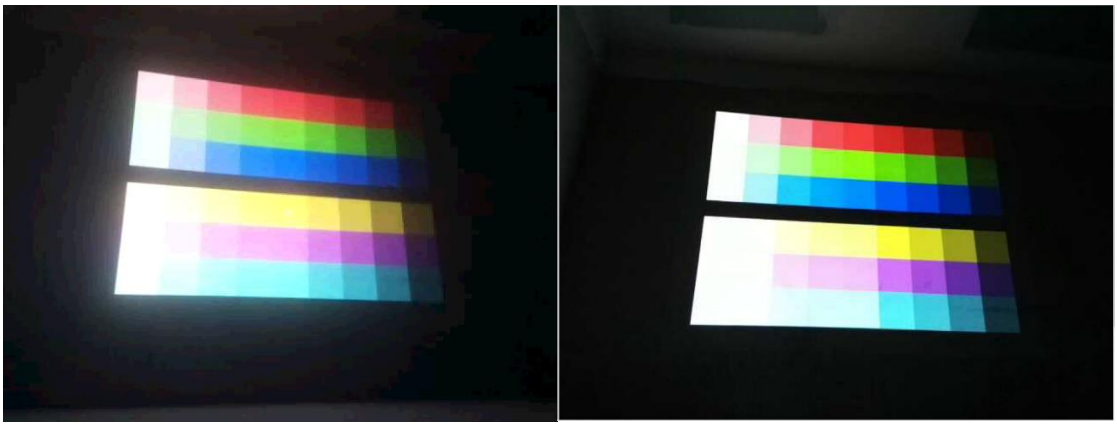

Figura 56 - Comparação entre os sensores Pi Camera #3

Desta forma, é possível concluir que a comparação entre os sensores de câmara do Raspberry Pi poucos benefícios apresentaram na deteção do ponto laser na faixa amarela.

Assim, as Tabelas 13 e 14, sumarizam a visibilidade do ponto laser nas restantes faixas de cor. O símbolo 'X' foi utilizado para descrever a impossibilidade de visualização do ponto laser, o símbolo 'O' para descrever a pouca visibilidade do ponto laser e as células vazias correspondem a uma perfeita visualização do ponto laser.

Cor da célula	**Valor de luminosidade**								
	90	**80**	**70**	**60**	**50**	**40**	**30**	**20**	**10**
Vermelho	X	O							
Verde	X	X	O	O					
Azul	X	O							
Amarelo	X	X	X	X	X	X	O		
Roxo	X	X	X	O	O				
Ciano	X	X	X	X	X	O			

Tabela 13 - Visibilidade do ponto laser sobre a matriz de cores (Pi Camera V2)

Cor da célula	**Valor de luminosidade**								
	90	**80**	**70**	**60**	**50**	**40**	**30**	**20**	**10**
Vermelho	X	O							
Verde	X	O	O	O					
Azul	X	O							
Amarelo	X	X	X	X	X	O			
Roxo	X	X	X	O	O				
Ciano	X	X	X	X	X	O			

Tabela 14 - Visibilidade do ponto laser sobre a matriz de cores (Pi NoIR Camera V2)

De acordo, com os dados recolhidos é possível verificar que o sensor Pi NoIR Camera V2, apresentou benefícios em termos de visibilidade do ponto laser sobre as células com valor de luminosidade reduzido. Contudo, durante a utilização do sistema com o fundo em constante mudança, apresentou saturação excessiva e alta luminosidade, tornando impossível a visibilidade do ponto laser nas capturas.

Por outro lado, embora nos quadros capturados através do sensor Pi Camera V2, o ponto laser não apresentasse a sua cor real, o que realmente importa é a visibilidade do ponto laser e a estabilidade da luminosidade. Assim, verificou-se que este sensor apresenta menos ruído, independentemente das variações do plano de fundo, tornando-se o sensor mais adequado para utilização em tempo real.

5.1.3.2. Testes com tela de projeção

No corrente tópico, o ambiente de teste e a lógica do tópico anterior mantiveram-se, sendo a única diferença a utilização de uma tela de projeção sobre a parede. Até aqui, a parede estava a ser utilizada como superfície de projeção, entretanto, para esclarecer quaisquer dúvidas, foi montado uma tela para verificar os níveis de luminosidade na imagem capturada.

Na Figura 57, é possível constatar que o ponto laser é facilmente observável, contudo, a captura através do sensor Pi NoIR Camera apresenta o ponto laser com maior nitidez, tal como tinha acontecido anteriormente no teste sem tela.

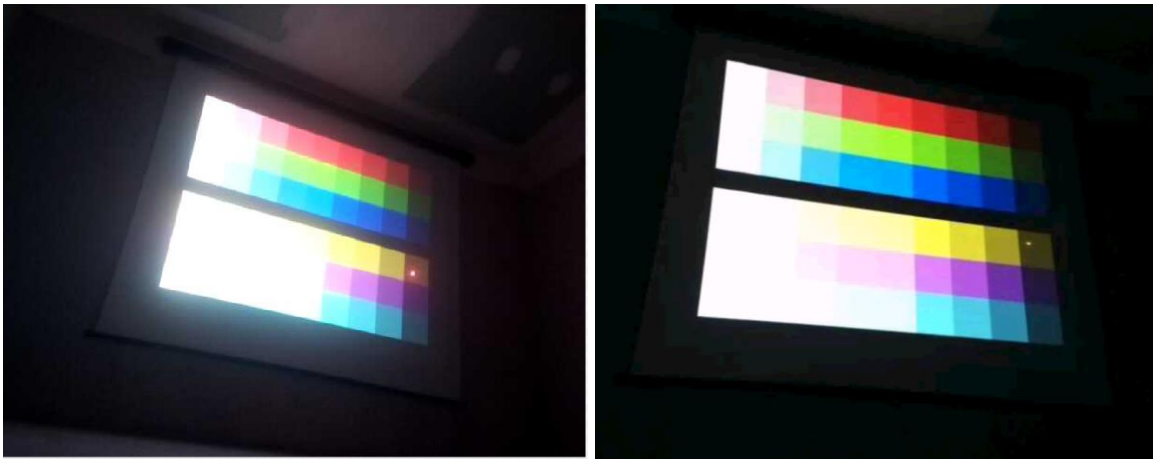

Figura 57 - Comparação dos níveis de luminosidade (com tela) #1

Na Figura 58, o ponto laser encontra-se sobreposto na célula amarela com valor de luminosidade (30). Contudo, já se começa a notar dificuldades na visualização do ponto laser em ambas as capturas, principalmente na imagem capturada através do sensor Pi Camera. Por outro lado, na imagem capturada através do sensor Pi NoIR Camera, o ponto laser é praticamente impossível de ser visualizado.

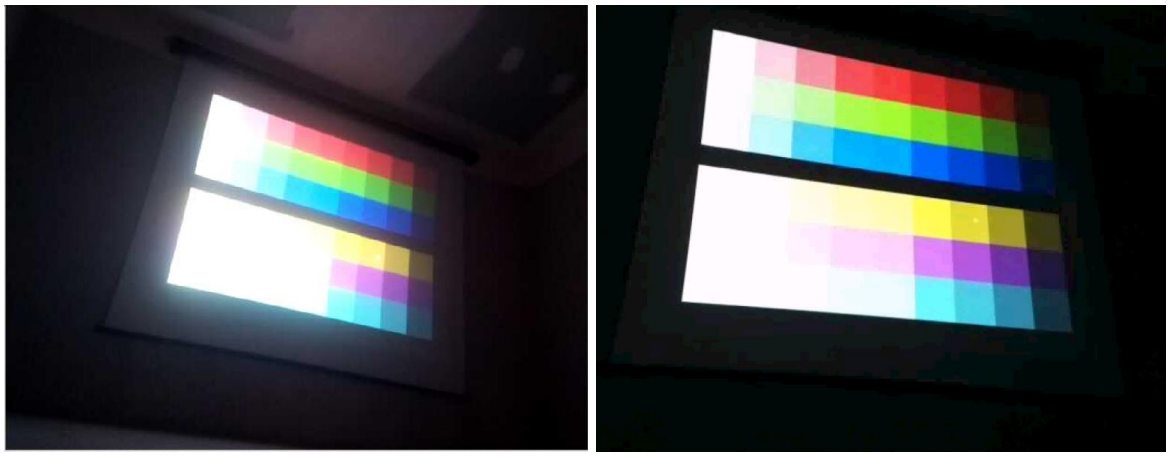

Figura 58 - Comparação dos níveis de luminosidade (com tela) #2

Para terminar o conjunto de testes com tela, o ponto laser foi posicionado na célula amarela com valor de luminosidade 50, tal como tinha acontecido no teste anterior. Nessa fase, foi possível constatar a impossibilidade de visualização do ponto laser nas capturas a partir de ambos os sensores.

Por este motivo, verificou-se que a utilização da tela fez com que os níveis de luminosidade ficassem concentrados na área da tela, impossibilitando a visualização do conteúdo das células com altos valores de luminosidade. Por outro lado, nos testes sem tela, a luminosidade tendia a espalhar-se por toda a imagem capturada. Assim, a utilização da tela pouco beneficiou a visibilidade do ponto laser, contudo, pode ser útil quando estamos a utilizar o sensor Pi Camera e uma superfície de projeção com altos níveis de reflexão.

5.1.3.3. Testes com o telemóvel

No corrente tópico, manteve-se o ambiente de teste e reutilizou-se a lógica dos testes anteriores, mas nesta fase utilizamos a câmara do telemóvel. Assim, a cada etapa iremos comparar a visibilidade do ponto laser nas diferentes superfícies de projeção (com e sem tela).

Uma vez que já se percebeu que as células com baixo valor de luminosidade não representam um problema para a deteção do ponto laser, neste conjunto de testes, iremos começar a partir da célula da faixa amarela com valor de luminosidade (50). O motivo para esta escolha deve-se por estarmos a utilizar uma câmara com características superiores e pretendemos saber se o ponto laser é visível sobre as células com alto valor de luminosidade. Desta forma, nas imagens que se seguem, teremos no lado esquerdo a captura sem a tela de projeção, e no lado direito, a captura com a tela de projeção.

Na Figura 59, o ponto laser encontra-se sobreposto na célula amarela com valor de luminosidade (50). Em ambas as capturas, o ponto laser encontra-se visível, no entanto, a captura sem tela apresentou maior nitidez e facilidade de deteção do ponto laser.

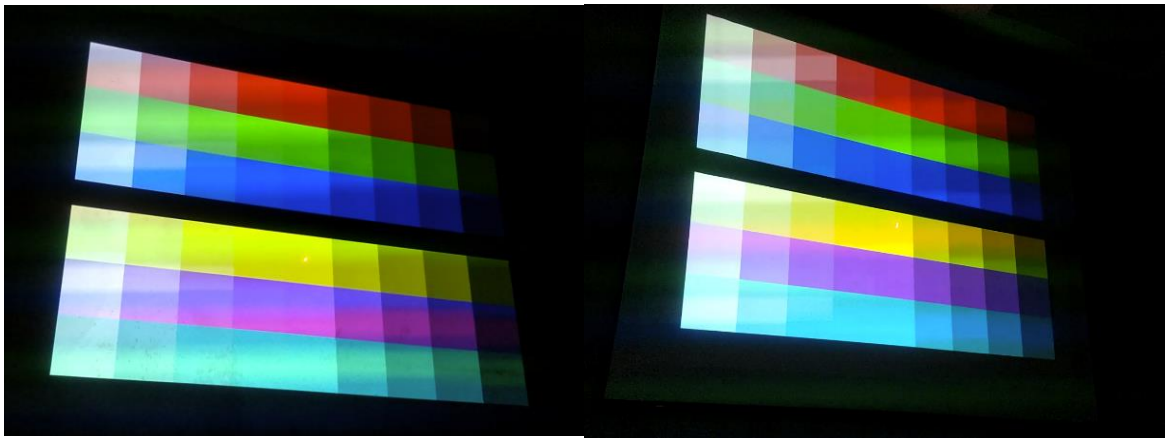

Figura 59 - Comparação dos níveis de luminosidade #1

Na Figura 60, o ponto laser encontra-se sobreposto na célula amarela com valor de luminosidade (70). Nesta situação, o ponto laser encontra-se facilmente observável, no entanto, tal como aconteceu anteriormente, a captura sem tela apresenta maior nitidez e representação do ponto laser.

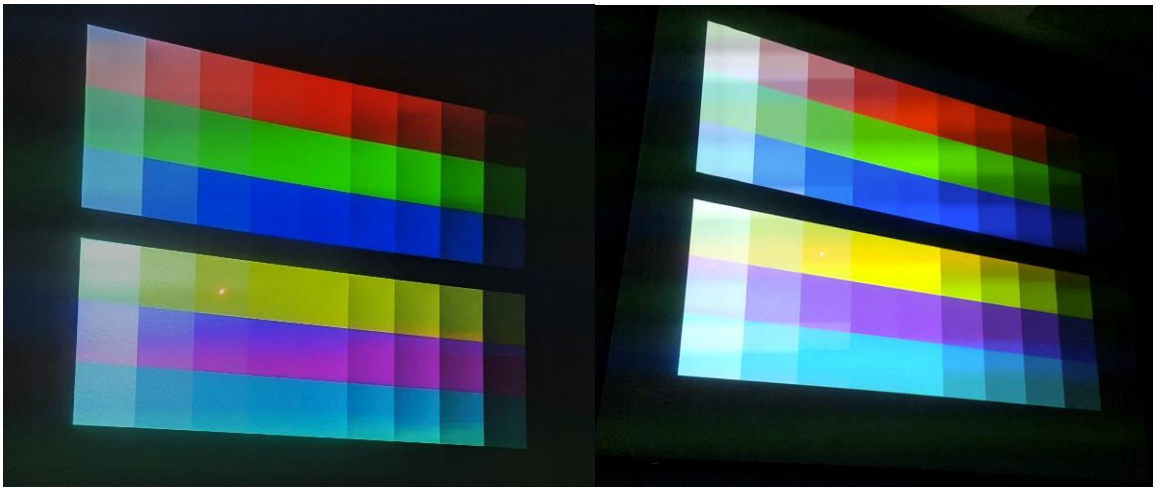

Figura 60 - Comparação dos níveis de luminosidade #2

Por fim, na Figura 61, o ponto laser encontra-se sobreposto na célula amarela com valor de luminosidade (90). Nesta situação, podemos verificar que a captura sem tela permitiu continuar a ter perceção do ponto laser, enquanto na captura com tela, o ponto laser encontra-se ocultado pela luminosidade.

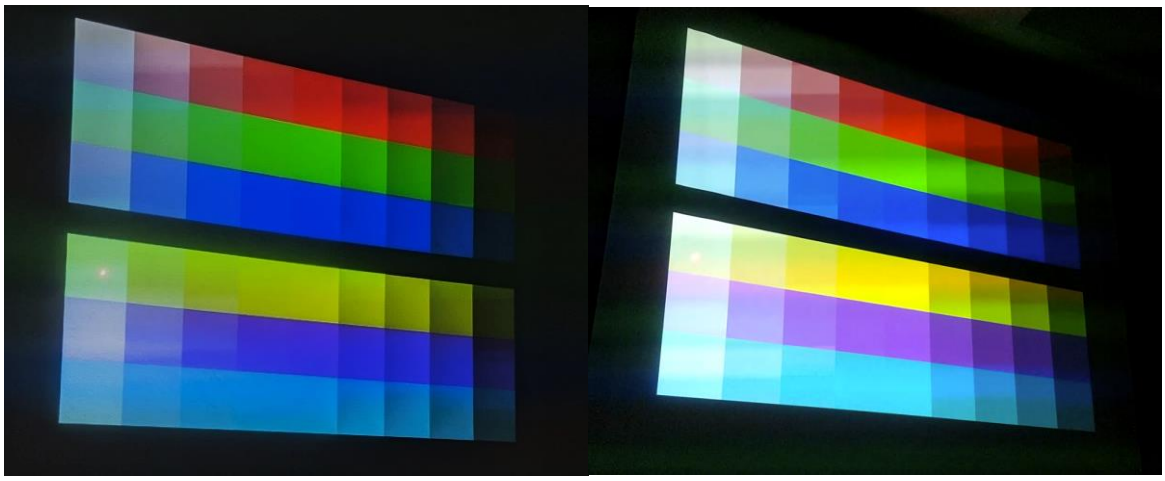

Figura 61 - Comparação dos níveis de luminosidade #3

Assim sendo, conclui-se que a câmara do telemóvel conseguiu detetar o ponto laser em toda a faixa amarela independentemente dos níveis de luminosidade. No entanto, é de salientar que nesta situação, a introdução da tela projetada pouco beneficiou no reconhecimento do ponto laser. Como resultado, torna-se cada vez mais evidente que a adoção de temas escuros é preferível para evitar os problemas que temos vindo a deparar.

5.1.4. Medidor de luminosidade

Dado o conjunto de testes e as dificuldades sentidas, decidiu-se utilizar um luxímetro para medir a intensidade luminosa (lux) sobre a superfície de projeção. O luxímetro é um equipamento utilizado para determinar os níveis de luminosidade em um ambiente [47], sendo constituído por uma célula fotoelétrica e um miliamperímetro. A célula fotoelétrica é um material semicondutor sensível à luz, onde a luz incidente forma corrente no semicondutor que depois é medida no amperímetro para determinar o nível de iluminância [48].

A Figura 62, ilustra o luxímetro LX-101, o modelo utilizado para o conjunto de testes que se segue. Este modelo tem uma precisão típica de 5% e permite capturar os dados nas escalas (x1, x10 e x100), sendo a intensidade luminosa para cada intervalo compreendida entre 0-1.999 lux, 2.000-19.990 lux e 20.000-50.000 lux, respetivamente [49].

Figura 62 - Medidor de luminosidade - Lutron Lx-101

Na Figura 63, é possível visualizar a sonda fotoelétrica do luxímetro encostada na superfície de projeção e direcionada para o projetor. Esta ação repetiu-se a cada célula da matriz nas diferentes faixas de cor, durante três segundos, para permitir a estabilização do sensor aquando da transição.

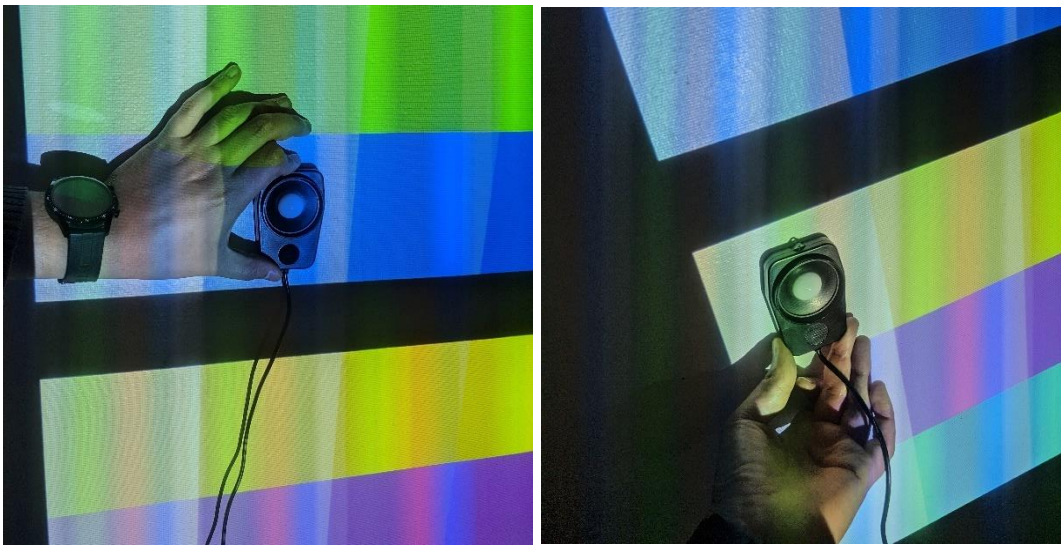

Figura 63 – Medição através do luxímetro

Na Tabela 15, encontra-se representado a média dos dados de intensidade luminosa provenientes de três medições através do luxímetro. Desta forma, pretende-se compreender a tendência central dos valores de intensidade luminosa em cada célula da matriz de cores. Os dados foram obtidos em condições ambientais de baixa luminosidade e o projetor encontrava-se aproximadamente a dois metros de distância.

Cor	**Valor de luminosidade**								
	90	**80**	**70**	**60**	**50**	**40**	**30**	**20**	**10**
Vermelho	478	256	137	84	73	48	28	15	7
Verde	669	462	411	394	405	255	141	61	17

Azul	553	281	145	87	74	48	29	15	7
Amarelo	956	845	790	779	783	484	266	115	29
Roxo	692	396	247	179	165	102	58	29	11
Ciano	828	638	593	571	575	358	199	90	25

Tabela 15 - Média dos valores de luminosidade #1

No conjunto de dados apresentados na Tabela 16, utilizou-se a mesma lógica descrita anteriormente, mas com a diferença do projetor encontrar-se aproximadamente a três metros de distância da superfície de projeção.

Cor	**Valor de luminosidade**								
	90	**80**	**70**	**60**	**50**	**40**	**30**	**20**	**10**
Vermelho	291	164	83	50	44	30	19	12	9
Verde	399	290	253	242	241	153	86	39	13
Azul	333	177	91	56	46	28	17	11	7
Amarelo	591	505	473	460	459	273	151	67	19
Roxo	426	252	153	110	100	61	35	20	9
Ciano	496	386	346	330	328	204	113	52	16

Tabela 16 - Média dos valores de luminosidade #2

Comparando os resultados obtidos, é possível constatar que o afastamento do projetor da superfície de projeção permitiu diminuir os níveis de luminosidade na imagem capturada. Contudo, na Tabela 15, é possível verificar que a transição do ponto laser entre as células com valor de luminosidade 60 e 50, nas faixas de cor: verde, amarelo e ciano, apresentaram um aumento dos valores de intensidade luminosa. No entanto, quando o projetor foi posicionado a maior distância da tela, verificou-se uma redução nos níveis de luminosidade que afetam as lentes da câmara.

5.1.5. Realização de gestos

Apesar do sucesso na identificação dos gestos realizados nas bordas da tela, existiram por vezes problemas na deteção do gesto circular. Durante um gesto circular, o movimento irregular da mão do utilizador pode fazer com que as células sobrepostas pelo ponto laser não correspondam com o padrão definido (Capítulo IV – Gestos Circulares). Por outro lado, os problemas de luminosidade podem fazer com que as células sobrepostas não sejam registadas, prejudicando o reconhecimento de um gesto.

Para simplificar a situação descrita, a Figura 64, ilustra um exemplo, onde o conjunto de células registadas durante a execução do gesto circular não correspondem com o padrão pré-definido.

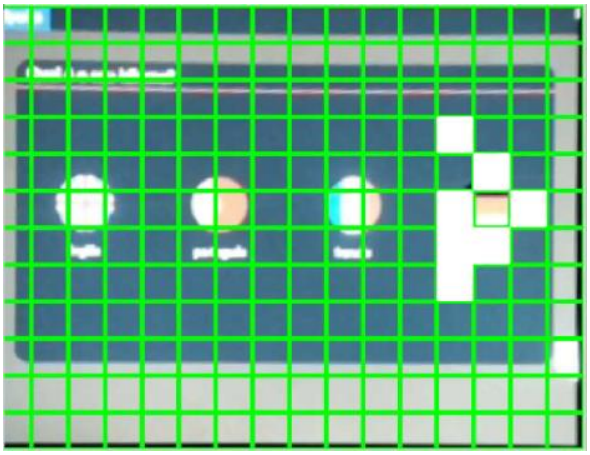

Figura 64 - Problemas na deteção do gesto circular

Este problema pode ser atenuado, aumentando o número de padrões que significam a ocorrência de um gesto circular. Na Tabela 17, é possível verificar a abordagem que sugere o aumento do número de padrões que significam a ocorrência de um gesto circular.

(0,0)	**(0,1)**	**(0,2)**
(1,0)	**(1,1)**	**(1,2)**
(2,0)	**(2,1)**	**(2,2)**

(0,0)	**(0,1)**	**(0,2)**
(1,0)	**(1,1)**	**(1,2)**
(2,0)	**(2,1)**	**(2,2)**

Tabela 17 - Padrões de deteção de um gesto circular

Embora o aumento de padrões que significam a ocorrência de um gesto circular seja uma solução para o problema descrito. Posteriormente, verificou-se que a implementação da matriz computacional sobre a área da tela não foi a solução mais adequada porque introduz alguma latência (aproximadamente 3 segundos) na deteção do padrão circular.

5.2 Avaliações Heurísticas

Nas últimas décadas, tem sido dada cada vez maior importância à interface gráfica, dado que envolvem todas as funções da aplicação [50]. Nielsen (1994), alude que para caracterizar a usabilidade podem ser conduzidos métodos de avaliação. Dentre os métodos mais difundidos destaca-se a Avaliação Heurística (AH), um método empírico que julga a interface consoante um conjunto de heurísticas que procuram potencializar a usabilidade da interface e da interação [51]. A avaliação heurística é um método informal de análise de usabilidade em que vários avaliadores são apresentados a uma interface e solicitados a

comentá-lo, tentando chegar a uma opinião sobre os aspetos positivos e negativos. O ideal é que as avaliações sejam realizadas de acordo com as diretrizes de design, no entanto, no presente documento, os participantes realizaram as avaliações heurísticas com base na sua própria intuição e bom senso.

Para testar a aplicabilidade da avaliação heurística, foram conduzidas três experiências de usabilidade, onde os participantes foram solicitados a descrever os problemas de usabilidade na interface da forma mais precisa possível. As heurísticas utilizadas para medir a experiência do utilizador, foram examinadas tendo em conta: a atratividade, eficiência e estimulação [52]. Assim, é pretendido aumentar a produtividade durante a realização das tarefas e não exigir muito tempo para conhecer o sistema. Desta forma, no tópico posterior serão descritos os testes realizados, os problemas verificados e assim determinar como as regras e diretrizes de Shneiderman e Nielsen podem ser utilizadas para melhorar e avaliar a interface [53].

5.3. Testes de usabilidade

Os testes de usabilidade permitem os utilizadores avaliar e descrever a qualidade de interação com o sistema. O processo pode exigir a execução de tarefas, entrevistas, entre outros métodos, com o objetivo de recolher dados sobre a experiência de utilização do sistema [54]. O estudo de usabilidade teve por base três fases distintas. A primeira fase centrou-se num estudo, onde foram definidas as tarefas que seriam executadas pelos partitcipantes. Na segunda fase efetuou-se o conjunto dos testes de usabilidade. Na terceira e última fase fez-se uma análise dos resultados obtidos com foco nos problemas detetados e com base em algumas observações. O ambiente selecionado para a realização dos testes de usabilidade foi numa sala do departamento de sistemas de informação da Câmara Municipal do Funchal, tal como ilustrado na Figura 65.

Figura 65 - Ambiente onde foram realizados os testes de usabilidade

De modo a avaliar a usabilidade foram selecionados 6 participantes, na tentativa de identificar diferentes padrões de utilização. Desta forma, a Tabela 18 sumariza as informações.

Participante	Idade	Faixa Etária
1	16	[10,25]
2	20	
3	28	[26,41]
4	31	
5	43	[42,57]
6	57	

Tabela 18 - Informações dos participantes dos testes de usabilidade

Inicialmente, os participantes foram instruídos que o sistema tinha por base a interação através de um apontador laser e que cada borda da tela correspondia a uma ação para o sistema. Relativamente ao procedimento, cada teste foi realizado isoladamente e cada participante era acompanhado. O meu papel enquanto avaliador consistiu em analisar tudo o que ocorria durante a sessão de teste para garantir que os objetivos eram alcançados e proceder ao registo dos tempos de execução das tarefas, erros e observações.

Os testes de usabilidade foram executados na sequência a seguir descrita:

1. Realizar um tutorial.
2. Monitorizar a estabilidade do apontador laser.
3. Realizar o gesto circular sobre botões aleatórios.

5.3.1. Tutorial

Os gestos são a única forma de interação com o sistema, e por este motivo, é importante instruir o utilizador do conjunto de gestos disponíveis. O tutorial representa o primeiro teste de usabilidade e tem como objetivo permitir o utilizador compreender e executar os gestos que podem ser executados. Na Figura 66, o utilizador é indicado para realizar um gesto InOut-Right. Se o gesto for executado corretamente, o utilizador é alertado do sucesso da operação, caso contrário, possui até 3 tentativas. Por outro lado, se nenhum gesto for detetado, o utilizador é notificado que um gesto deve ser executado, caso contrário, o tutorial termina sem sucesso.

Figura 66 – Teste de usabilidade #1

Como resultado, quando o utilizador desencadeia um gesto, o evento é analisado pelo processo Sparks Core e comunicado ao processo Sparks Web. Consoante a resposta obtida, cabe ao Sparks Web determinar se o gesto corresponde com a etapa corrente do tutorial.

5.3.2. Estabilidade do ponto laser

Conforme relatado na literatura, a instabilidade da mão do utilizador pode ser um fator desencorajante para a utilização do sistema. A Figura 67, ilustra o segundo teste de usabilidade que consiste em monitorizar a estabilidade do ponto laser e verificar o comportamento dos utilizadores perante um espaço restrito. Desta forma, pretendeu-se observar a estabilidade do ponto laser nas diferentes faixas etárias, sendo o objetivo evitar sobrepor as paredes do labirinto.

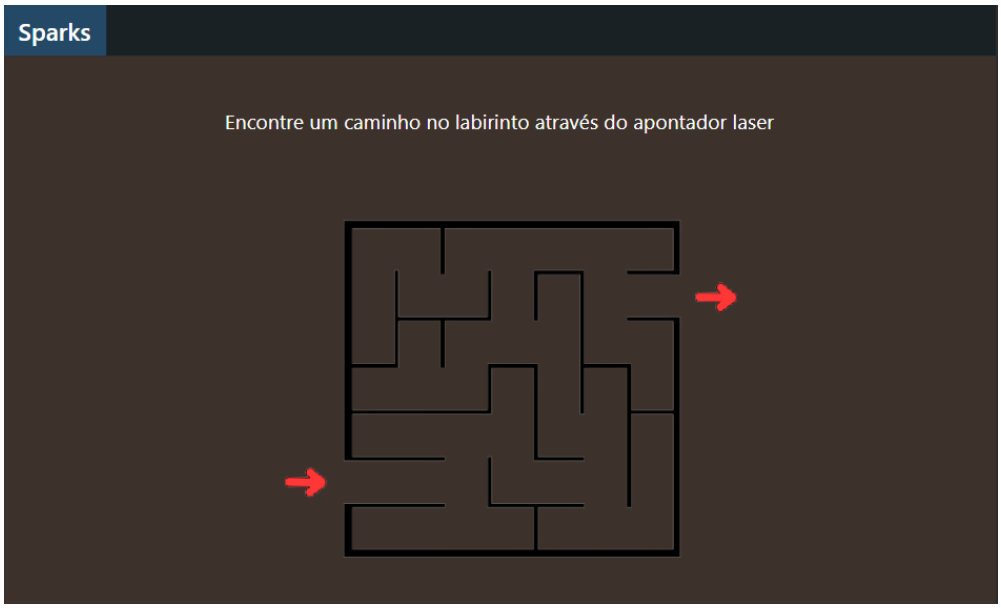

Figura 67 - Teste de usabilidade #2

5.3.3. Gesto circular sobre botões aleatórios

Na Figura 68, encontra-se ilustrado o último cenário de testes, onde o utilizador era instruído para clicar sobre um botão através de um gesto circular. Durante 3 iterações, o utilizador clicava sobre um botão, cujos números eram gerados e atribuídos aleatoriamente. Os principais objetivos foram testar a precisão do apontador laser na ação de clique, o tempo que os utilizadores levavam a executar a operação e o tempo que o sistema demorava a responder.

Figura 68 - Teste de usabilidade #3

Por outro lado, espera-se que esta abordagem permita compreender se o gesto circular corresponde com o movimento mais adequado para a realização da ação de clique.

5.4. Resultados

Neste tópico será debatido os resultados, seguido de algumas notas referentes ao comportamento dos participantes durante a execução dos testes de usabilidade. Nos primeiros momentos de utilização, os utilizadores revelaram alguma confusão na execução dos gestos e tinham a tendência de cruzar as bordas da tela com o apontador, não tendo perceção de que a ação desencadeava um gesto. Assim, a realização do tutorial como primeira etapa dos testes de usabilidade foi crucial para instruir os utilizadores do conjunto de gestos disponíveis. Como mencionado, cada etapa do tutorial era constituída por 3 tentativas pelo que houve situações que os utilizadores tiveram de repetir o gesto.

O teste de monitorização da estabilidade do ponto laser foi considerado o mais fácil. Os participantes percorreram o labirinto durante 3 iterações e a cada iteração verificou-se melhorias na realização do percurso. Por outro lado, foi evidente que a distância do utilizador para a tela, afeta consideravelmente a estabilidade do ponto laser. Contudo, embora o tremor causado pela mão do utilizador verificou-se no geral que a maioria dos participantes conseguiram realizar o percurso com sucesso sem sobrepor as paredes do labirinto.

Por outro lado, a Figura 69, ilustra um dos participantes a utilizar uma mão para apoiar a outra mão enquanto segurava o apontador laser. Esta ação se verificou em todos os testes de usabilidade com o mesmo utilizador, e no final quando questionado, o mesmo relatou que de natureza tem alguma instabilidade nas mãos, pelo que preferiu optar por esta estratégia para reduzir o tremor visível.

Figura 69 - Execução do teste de usabilidade #2

Relativamente ao teste do gesto circular, é de salientar que algumas das vezes o gesto não era bem-sucedido, pelo que o utilizador era obrigado a repetir o movimento. O principal motivo das dificuldades não ocorreu necessariamente devido ao movimento circular, mas sim pelos botões não terem o tamanho suficiente para que o ponto médio do gesto circular coincidisse com a área do botão. Por este motivo, o tamanho dos botões são de extrema importância, de acordo, com a estratégia utilizada para a identificação do gesto circular. Neste teste de usabilidade, a cada ação de clique com sucesso, os botões eram novamente gerados com um novo número e localização diferente.

A Figura 70 ilustra um dos participantes a realizar o gesto circular para concretizar a ação de clique.

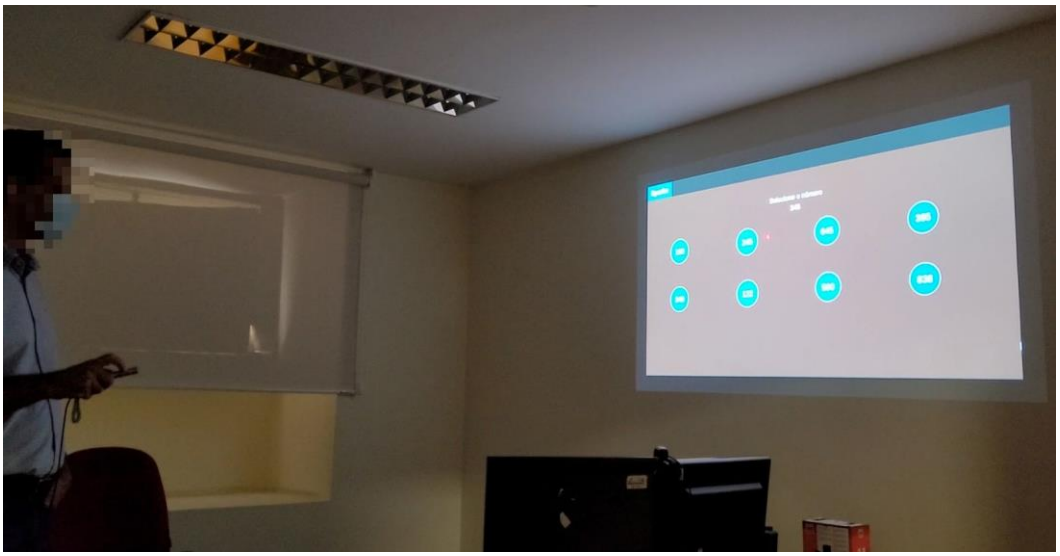

Figura 70 - Execução do teste de usabilidade #3

Durante toda a sessão, as experiências foram registadas permitindo desta forma obter os tempos de execução de cada tarefa, tal como ilustrado na Tabela 19. Desta forma, é possível constatar que o tutorial (Teste 1) consistiu na experiência de maior duração, de certa forma compreensível, uma vez que tem como principal objetivo familiarizar o utilizador do conjunto de ações disponíveis. O teste de estabilidade (Teste 2) demonstrou ser o mais rápido uma vez que teve como objetivo verificar os níveis de sensibilidade entre utilizadores de diferentes faixas etárias. Por fim, a realização do gesto circular (Teste 3) acabou por revelar que dependendo do utilizador o tempo para a execução da ação de clique pode variar. Esta situação ocorreu porque os utilizadores tinham a tendência de realizar um gesto circular de diâmetro pequeno, o qual acabava por não coincidir com o padrão definido. Desta forma, os utilizadores eram obrigados a repetir o movimento para que o diâmetro do gesto circular coincidisse com as quatro células da matriz sobre a área de projeção.

Participante	Teste 1	Teste 2	Teste 3	Total
1	2m 02s	32s	45s	3m 19s
2	1m 50s	35s	39s	3m 04s
3	1m 46s	30s	41s	2m 57s
4	2m 05s	28s	49s	3m 22s
5	2m 11s	37s	54s	3m 42s
6	2m 30s	41s	50s	4m 01s

Tabela 19 - Tempos de execução dos testes de usabilidade

Depois de completadas as tarefas, os participantes foram reunidos para relatar as experiências e responder a uma breve entrevista. Assim, os testes de usabilidade duraram em média, cerca de dez minutos para cada participante e todas seguiram o mesmo protocolo. As questões colocadas durante as entrevistas, seguiram a ordem especificada:

1. Considera o sistema fácil de utilizar?
2. Os testes de usabilidade permitiram acelerar o processo de aprendizagem?
3. Considera pertinente um teclado virtual no sistema?
4. Quais os pontos fortes e fracos desta forma de interação?

No geral, os resultados foram positivos apesar de nos momentos iniciais existir a tendência de cruzar involuntariamente as bordas da tela com o apontador. Embora um movimento involuntário, assim que os utilizadores realizaram o tutorial, começaram a perceber que o cruzamento da borda da tela corresponde a uma ação para o sistema, e por este motivo, passaram a ter mais cuidado com a movimentação do ponto laser. Relativamente à interface gráfica, ficou saliente que deve ser cuidadosamente projetada para a interação com o ponto laser. Os botões não devem estar muito próximos (como acontece regularmente) e devem ter um tamanho razoavelmente grande para que o ponto médio do gesto circular coincida com a localização do botão.

Os utilizadores consideraram as cores da interface gráfica simples e apelativas, no entanto, surgiram sugestões no sentido que o sistema deveria apoiar mais o utilizador. Por exemplo, quando um gesto é realizado numa borda, deveria existir um *feedback* visual ou auditivo que transmitisse o sucesso do gesto. No mesmo âmbito, mencionaram que a interface do tutorial poderia ser complementada com imagens para reforçar o sentido do texto e facilitar a aprendizagem do sistema. Por último, também surgiu a sugestão que

quando o ponto laser estivesse sobre um botão ocorresse um efeito visual. Desta forma, estas críticas construtivas, vêm de encontro com as regras de Shneiderman, no sentido, que o utilizador deve saber onde está, o que está a acontecer e para cada ação deve existir um feedback apropriado e legível [55]. Assim ficou claro, que a interface gráfica do tutorial, deve ser reestruturada com o intuito de aumentar a perceção do utilizador relativamente ao gesto que deve ser executado em cada momento. Por outro lado, durante o processo de calibração, uma tela apresenta constantemente o estado do sistema, indo de encontro com as regras de Nielsen, no sentido, que o sistema deve sempre manter os utilizadores informados, permitindo-os reconhecer e diagnosticar os erros.

Relativamente à execução do gesto circular, os testes de usabilidade provaram que o movimento circular não acarreta dificuldades para executar a ação de clique. No entanto, verificou-se a tendência de executar mais do que um gesto circular em redor do botão, este fator, ocorreu porque os utilizadores esperavam uma resposta rápida e dado a falta de *feedback*, tinham a tendência de repetir o movimento circular. Assim, a estratégia utilizada para a deteção do gesto circular demonstrou que necessita de melhorias para que no futuro o sistema possa responder mais rápido à ação de clique.

Em suma, os utilizadores mencionaram que o sistema tem potencial, contudo, a interface gráfica deve ser projetada com cuidados para acomodar o dispositivo de entrada. Por outro lado, quando questionado a possibilidade de implementação de um teclado virtual para interação através do ponto laser, os utilizadores não demonstraram uma reação positiva, salientando que o sistema pode estar limitado quanto a esse aspeto. Assim sendo, os testes de usabilidade permitiram identificar os problemas, observar o comportamento dos utilizadores e perceber o motivo pelo qual escolheram seguir determinados caminhos.

5.5. Sumário

Conforme os testes realizados verificou-se que a deteção do ponto laser apresenta vários desafios. A luminosidade do ambiente, o reflexo da superfície de projeção e a utilização de cores de alta luminosidade, podem introduzir ruído na imagem capturada, causando saturação excessiva. De acordo, com Huang e Putnam (2006), para minimizar os efeitos prejudiciais descritos, a utilização de um filtro de densidade neutra em frente da lente da câmara, pode ajudar a reduzir a luminosidade formando uma distinção mais clara do ponto laser.

Em termos do reconhecimento do ponto laser, conseguiu-se melhor desempenho quando a página era constituída por cores escuras. O que significa que a adoção de temas

escuros pode ser favorável perante câmaras com características reduzidas. Por outro lado, a câmara do telemóvel apresentou melhor desempenho independentemente da intensidade luminosa das cores projetadas. No entanto, quando o ambiente era exposto à luz externa (por exemplo, abertura de uma janela), as dificuldades voltaram a se verificar, o que é perfeitamente compreensível. Por outro lado, o luxímetro permitiu-nos compreender os níveis de intensidade luminosa que o ambiente estava exposto, e a partir de que luminosidade começámos a ter dificuldades em visualizar o ponto laser sobre a captura de imagem.

Os testes de usabilidade permitiram comprovar que pode ser difícil para o utilizador controlar a mão, e por esse motivo, um gesto nem sempre é bem-sucedido. Por outro lado, quanto mais distante o utilizador encontra-se da tela, mais notável é a vibração da mão. Os testes de usabilidade foram importantes para perceber como os utilizadores reagiam aos procedimentos e às técnicas utilizadas para interação com o sistema. Deste modo, foi possível constatar que os desenvolvimentos efetuados foram importantes e permitiram estabelecer uma base que no futuro pode ser aprimorada. Contudo, para o sucesso do sistema é necessário que a interface gráfica acompanhe as evoluções introduzidas, para assim o sistema ser compreendido mais facilmente pelos utilizadores.

6. Conclusão

Neste último capítulo, serão apresentadas as conclusões com menções aos capítulos anteriores, reflexões pessoais e opiniões sobre o trabalho futuro a realizar.

Este projeto de mestrado contribuiu para estudar, compreender e aplicar conceitos substanciais da área da visão computacional em contexto real. A complexidade do sistema torna clara a necessidade e o aparecimento de uma arquitetura. Portanto, o desenho e implementação do sistema foi sem dúvida o elevar da fasquia nos desafios propostos. Esta etapa exigiu investigação, recolha de requisitos e o desenvolvimento de aptidões sobre as metodologias de visão computacional e das técnicas relacionadas com a biblioteca OpenCV.

A revisão de literatura permitiu desde cedo estar ciente das possíveis dificuldades inerentes ao sistema. Contudo, é espectável que se tivéssemos utilizado equipamentos com melhores características, o projeto poderia ter alcançado resultados ainda mais positivos. No entanto, tendo em conta os recursos disponíveis e comparando os resultados com o problema inicialmente gerado, é possível comprovar que a implementação do sistema em Python, permitiu obter os resultados dentro do prazo estabelecido no cronograma. Por outro lado, o Python acabou por refletir alguns problemas de desempenho em tarefas de alta demanda, como por exemplo, na monitorização do ponto laser. Desta forma, a adoção de outra linguagem de programação, como por exemplo, o C++, pudesse ser preferível.

Terminantemente, a realização deste projeto permitiu colocar em prática as técnicas de visão computacional, compreender o seu funcionamento e aprender uma área que está em constante evolução. Por outro lado, a oportunidade de ver outras pessoas a testar a framework desenvolvida por mim, e verificar que grande parte do raciocínio introduzido foi percebido pelos participantes, permitiu-me obter satisfação e vivenciar uma experiência incrível, o que permite pensar mais além.

6.1. Trabalho Futuro

São várias as ideias e as possibilidades de crescimento oferecidas pela área de visão computacional e de inteligência artificial. A utilização de redes neuronais poderia aumentar a confiabilidade do sistema, contudo, seria necessário aumentar os recursos computacionais, e entender as funcionalidades que podem vir a ser treinadas. Sendo a monitorização do ponto laser um dos aspetos mais vulneráveis, a utilização de diferentes fontes de dados contendo informações sobre o ponto laser, poderia expandir a informação para o treino das redes, permitindo identificar o ponto laser com maior precisão. No entanto, apesar dos benefícios, seria necessário mais tempo para obter uma rede estável e com resultados satisfatórios.

O processamento dos quadros de imagem é um aspeto que requer desempenho e, portanto, o módulo de processamento visual poderia ser reescrito em C++. Por outro lado, devido á grande quantidade de dados provenientes do sensor de câmara, seria preferível ter um equipamento dedicado apenas para a recolha e processamento dos dados, de modo, a possibilitar a distribuição da carga. Desta forma, a utilização de mais de um equipamento poderia ser benéfica, permitindo uma arquitetura distribuída com as tarefas segmentadas pelos diferentes equipamentos.

Consoante os problemas causados pelos efeitos de alta luminosidade, algum algoritmo poderia vir a ser desenvolvido para desencadear uma forma mais inteligente de previsão da localização do ponto laser, considerando o histórico de coordenadas.

Em suma, e pensando um pouco mais além, o sistema poderia ser aprimorado para disponibilizar uma API que permitisse os desenvolvedores criar aplicações para integrar no Sparks, tendo em conta a interação baseada no apontador laser.

Referências

[1] B. Shizuki, T. Hisamatsu, S. Takahashi, e J. Tanaka, «Laser pointer interaction techniques using peripheral areas of screens», em *Proceedings of the working conference on Advanced visual interfaces - AVI '06*, Venezia, Italy, 2006, p. 95. doi: 10.1145/1133265.1133284.

[2] C. Martins, «A "Nova Teoria sobre Luz e Cores" de Isaac Newton: uma Tradução Comentada». Mai. 16, 1996. [Em linha]. Disponível em: http://www.sbfisica.org.br/rbef/pdf/v18a33.pdf

[3] J. M. Brisson Lopes, «Cor e Luz», Maio de 2003, [Em linha]. Disponível em: http://disciplinas.ist.utl.pt/leic-cg/textos/livro/Cor.pdf

[4] R. C. Gonzalez, R. E. Woods, e B. R. Masters, «Digital Image Processing, Third Edition», *J. Biomed. Opt.*, vol. 14, n. 2, p. 029901, 2009, doi: 10.1117/1.3115362.

[5] M. Marengoni e S. Stringhini, «Tutorial: Introdução à Visão Computacional usando OpenCV», *Revista de Informática Teórica e Aplicada*, vol. 16, n. 1, Art. n. 1, 2009, doi: 10.22456/2175-2745.11477.

[6] E. P. F. Neto, «Visão computacional para identificação de cores em tempo real com OpenCV e Python», p. 17. [Em linha]. Disponível em: https://repositorio.uniceub.br/jspui/bitstream/prefix/15103/1/Eur%C3%ADpedes%20Vis%C3%A3o%20Computacional%20VF%20PB.pdf

[7] J. Cohen, «Computer Vision at Tesla», *Medium*, Out. 15, 2020. https://heartbeat.fritz.ai/computer-vision-at-tesla-cd5e88074376 (acedido Mar. 30, 2021).

[8] S. Roy, S. Nag, I. K. Maitra, e S. K. Bandyopadhyay, «A Review on Automated Brain Tumor Detection and Segmentation from MRI of Brain», 2013, p. 41. [Em linha]. Disponível em: https://www.researchgate.net/publication/259262689_A_Review_on_Automated_Brain_Tumor_Detection_and_Segmentation_from_MRI_of_Brain

[9] N. Ransom, «Visual Resource Centre School of Humanities», 2008, p. 8. [Em linha]. Disponível em: https://www.reading.ac.uk/web/files/using-images/Understanding1.pdf

[10] B. Girod, «Digital Image Processing», Jul. 2006, Acedido: Mai. 09, 2021. [Em linha]. Disponível em: https://web.stanford.edu/class/ee368/Handouts/Lectures/2014_Winter/1-Introduction_16x9.pdf

[11] «Binary Images :: Introduction (Image Processing Toolbox User's Guide)». http://matlab.izmiran.ru/help/toolbox/images/intro4.html (acedido Mar. 30, 2021).

[12] U. Rajashekar, A. C. Bovik, D. Sage, M. Unser, L. J. Karam, e R. L. Lagendijk, «2.4 - Image Processing Education», em *Handbook of Image and Video Processing (Second Edition)*, A. Bovik, Ed. Burlington: Academic Press, 2005, pp. 73–95. doi: 10.1016/B978-012119792-6/50069-3.

[13] W. K. Pratt, *Introduction to Digital Image Processing*. CRC Press, 2013. [Em linha]. Disponível em: http://nana.lecturer.pens.ac.id/index_files/referensi/image_processing/Digital%20Image%20Processing.pdf

[15] A. Backes, «Modelos de Cor». Fev. 10, 2014. [Em linha]. Disponível em: http://www.facom.ufu.br/~backes/gsi058/Aula11-ModelosCor.pdf

[16] H. Singh, *Practical Machine Learning and Image Processing For Facial Recognition, Object Detection, and Pattern Recognition Using Python*. Berkeley, CA: Apress, 2019.

Acedido: Out. 05, 2020. [Em linha]. Disponível em: https://link.springer.com/book/10.1007/978-1-4842-4149-3
[17] M. Stone, *A Field Guide to Digital Color*. New York: A K Peters/CRC Press, 2019. doi: 10.1201/b12887.
[18] A. L. Trifan, «The importance of color spaces in robotic vision», p. 2. [Em linha]. Disponível em: http://sweet.ua.pt/an/publications/2012e.pdf
[19] «HSV color space as a conical object», *ResearchGate*. https://www.researchgate.net/figure/HSV-color-space-as-a-conical-object_fig2_242293325 (acedido Jun. 26, 2021).
[20] J. Ramos, «Modelo de cor HSV». Fundação Universidade Federal do Vale do São Francisco. [Em linha]. Disponível em: http://www.univasf.edu.br/~jorge.cavalcanti/comput_graf06_Cores.pdf
[21] Cook e S. McManus, «Raspberry Pi for Dummies». [Em linha]. Disponível em: http://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Raspberry-Pi-For-Dummies.pdf
[22] *Understanding computer vision - Raspberry Pi Computer Vision Programming - Second Edition*. 2020. Acedido: Mar. 29, 2021. [Em linha]. Disponível em: https://subscription.packtpub.com/book/data/9781800207219/1/ch01lvl1sec02/understanding-computer-vision
[23] L. Jackson, «Use Almost Any MIPI Camera Module on Raspberry Pi (up to 18MP)», *Arducam*, Mar. 15, 2019. https://www.arducam.com/use-almost-any-mipi-camera-module-on-raspberry-pi/ (acedido Dez. 29, 2020).
[24] T. R. P. Foundation, «Pi NoIR Camera V2», *Raspberry Pi*. https://www.raspberrypi.org/products/pi-noir-camera-v2/ (acedido Jun. 22, 2021).
[25] S. van der Walt *et al.*, «Scikit-image: image processing in Python», *PeerJ*, vol. 2, p. e453, Jun. 2014, doi: 10.7717/peerj.453.
[26] M. Wissen, «Implementation of a Laser-based Interaction Technique for Projection Screens», Jul. 2001. https://www.ercim.eu/publication/Ercim_News/enw46/wissen.html (acedido Set. 27, 2020).
[27] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. Palgrave Macmillan, 2005.
[28] L. Garcés, B. Oliveira, e C. Arenas, «Arquiteturas de Software para o Domínio de Saúde», p. 47. [Em linha]. Disponível em: https://repositorio.usp.br/directbitstream/d387b8e9-0739-4b03-9858-a93d55ad9b32/3012103.pdf
[29] I. O. de Nunes e V. Zapalowski, «Recuperação de arquitetura de software com a ferramenta ArRTool», 2016, Acedido: Mar. 09, 2021. [Em linha]. Disponível em: http://hdl.handle.net/10183/151024
[30] «Welcome to Flask — Flask Documentation (1.1.x)». https://flask.palletsprojects.com/en/1.1.x/ (acedido Mar. 11, 2021).
[31] «Light - 2017 - Mosquitto server and client implementation of the.pdf». Acedido: Fev. 17, 2021. [Em linha]. Disponível em: https://joss.theoj.org/papers/10.21105/joss.00265.pdf
[32] «Wireless Heart Rate Monitoring System Using MQTT - ScienceDirect». https://www.sciencedirect.com/science/article/pii/S1877050916303817 (acedido Fev. 18, 2021).
[33] U. Madhuri Ganesh e R. A. Khan, *Raspberry Pi Home Automation Based on Internet of Things (IoT)*. 2015. Acedido: Fev. 27, 2021. [Em linha]. Disponível em: https://www.ijarcce.com/upload/2015/december-15/IJARCCE%2069.pdf

[34] R. Scheffer, «Uma visão geral sobre Threads», 2007, p. 6. [Em linha]. Disponível em: https://www.researchgate.net/publication/279504040_UMA_VISAO_GERAL_SOBRE_THREADS
[35] N. Nurseitov, M. Paulson, R. Reynolds, e C. Izurieta, *Comparison of JSON and XML Data Interchange Formats: A Case Study*. 2009. [Em linha]. Disponível em: https://www.researchgate.net/publication/220922905_Comparison_of_JSON_and_XML_data_interchange_formats_A_case_study
[36] «Visão Computacional Na Detecção de Fadiga | Scada | Visão Computacional», *Scribd*. https://pt.scribd.com/document/374078021/Visao-Computacional-Na-Deteccao-de-Fadiga (acedido Mai. 27, 2021).
[37] R. Lukac e K. Plataniotis, Eds., em *Color Image Processing: Methods and Applications*, vol. 20065568, CRC Press, 2006. doi: 10.1201/9781420009781.
[38] E. R. Davies, *Computer vision: theory, algorithms, practicalities*, Fifth edition. London, United Kingdom ; Cambridge, MA, United States: Elsevier/Academic Press, 2018. [Em linha]. Disponível em: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Computer%20and%20Machine%20Vision_%20Theory%2C%20Algorithms%2C%20Practicalities%20%284th%20ed.%29%20%5BDavies%202012-03-19%5D.pdf
[39] A. Pajankar, *Raspberry Pi computer vision programming: design and implement computer vision applications with Raspberry Pi, OpenCV, and Python 3.x*. 2020. Acedido: Dez. 03, 2020. [Em linha]. Disponível em: https://go.oreilly.com/university-of-alberta/library/view/-/9781800207219/?ar
[40] C. Shetty, «What is a Canny Edge Detection Algorithm». https://towardsai.net/p/computer-vision/what-is-a-canny-edge-detection-algorithm (acedido Jun. 30, 2021).
[41] M. Marengoni e D. Stringhini, «Introdução a visão do OpenCV». https://pt.scribd.com/document/37502870/introducao-a-visao-do-opencv (acedido Jan. 04, 2021).
[42] L. Rauta e A. Fernandes, «Algoritmos de Subtração Básica de Fundo». 2012. [Em linha]. Disponível em: https://www.aedb.br/seget/arquivos/artigos12/61916773.pdf
[43] K. Cheng e K. Pulo, «Direct Interaction with Large-Scale Display Systems using Infrared Laser Tracking Devices», p. 9. [Em linha]. Disponível em: https://www.researchgate.net/publication/221536145_Direct_Interaction_with_Large-Scale_Display_Systems_using_Infrared_Laser_tracking_Devices
[44] A. A. Block, «Affine transformation», 2020. Acedido: Dez. 15, 2020. [Em linha]. Disponível em: https://www.maa.org/sites/default/files/pdf/pubs/books/meg/meg_ch12.pdf
[45] B. Shneiderman e C. Plaisant, *Designing the user interface: strategies for effective human-computer interaction*, 4th ed. Boston: Pearson/Addison Wesley, 2004.
[46] A. Brajdic, «Understanding mental and conceptual models in product design», *Medium*, Mai. 12, 2020. https://uxdesign.cc/understanding-mental-and-conceptual-models-in-product-design-7d69de3cae26 (acedido Mar. 24, 2021).
[47] C. Jennifer e H. Sacht, «Avaliação de desempenho lumínico através de medições com luxímetro». [Em linha]. Disponível em: https://dspace.unila.edu.br/bitstream/handle/123456789/1398/EICTI%202016_485-488.pdf?sequence=1
[48] L. S. Pedroso, J. A. de Macêdo, M. S. T. de Araújo, e M. R. Voelzke, «Construção de um luxímetro de baixo custo», *Rev. Bras. Ensino Fís.*, vol. 38, n. 2, Jun. 2016, doi: 10.1590/S1806-11173812136.

[49] «Lux Meter Model: LX-101». SGS. [Em linha]. Disponível em: http://www.sunwe.com.tw/lutron/LX-101eop.pdf
[50] C. Souza, J. Leite, R. Prates, e S. Barbosa, «Projeto de Interfaces de Utilizador», em *1999*, p. 46. [Em linha]. Disponível em: http://www-di.inf.puc-rio.br/~clarisse/docs/JAI_Apostila1999.pdf
[51] J. Nielsen e R. Molich, «Heuristic evaluation of user interfaces», em *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*, Seattle, Washington, United States, 1990, pp. 249–256. doi: 10.1145/97243.97281.
[52] T. Zuk, L. Schlesier, P. Neumann, M. S. Hancock, e S. Carpendale, «Heuristics for information visualization evaluation», em *Proceedings of the 2006 AVI workshop on BEyond time and errors novel evaluation methods for information visualization - BELIV '06*, Venice, Italy, 2006, p. 1. doi: 10.1145/1168149.1168162.
[53] F. K. Mazumder e U. K. Das, «Usability Guidelines For Usable User Interface», p. 5. [Em linha]. Disponível em: https://ijret.org/volumes/2014v03/i09/IJRET20140309011.pdf
[54] K. G. Ferreira, «Teste de Usabilidade», Ago. 2002. Acedido: Jan. 15, 2021. [Em linha]. Disponível em: https://homepages.dcc.ufmg.br/~clarindo/arquivos/disciplinas/eu/material/referencias/monografia-avaliacao-usabilidade.pdf
[55] J. Adekunle Akinjobi, *Introduction to Human Computer Interaction*. 20213. Acedido: Dez. 28, 2021. [Em linha]. Disponível em: https://nou.edu.ng/sites/default/files/2017-05/CIT%20353%20%20INTRODUCTION%20TO%20HUMAN%20COMPUTER%20INTERACTION.pdf

Anexos

Anexo A – Arquitetura de software do Sparks

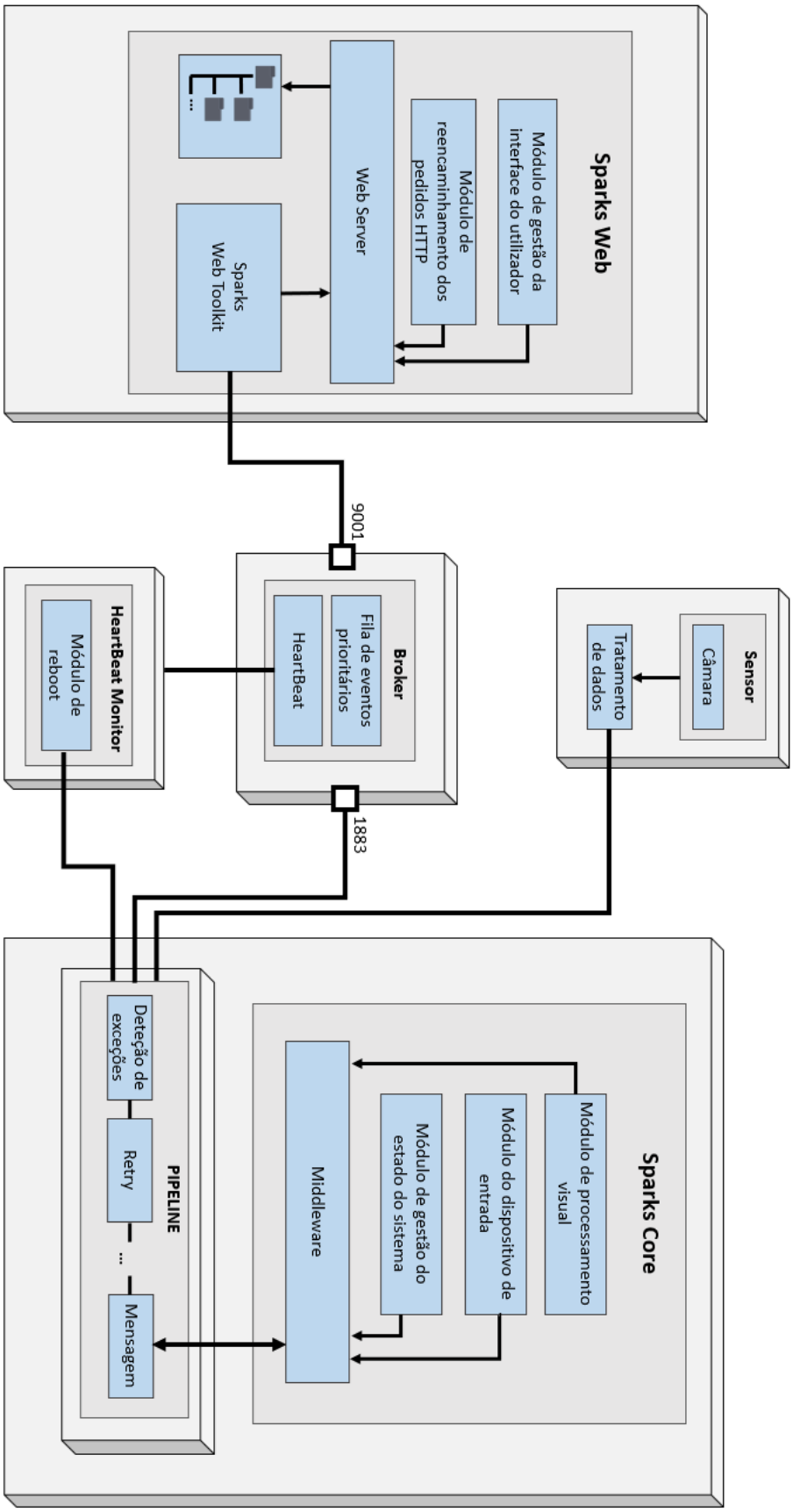

Figura 71 - Arquitetura de software

Anexo B – Processamento visual

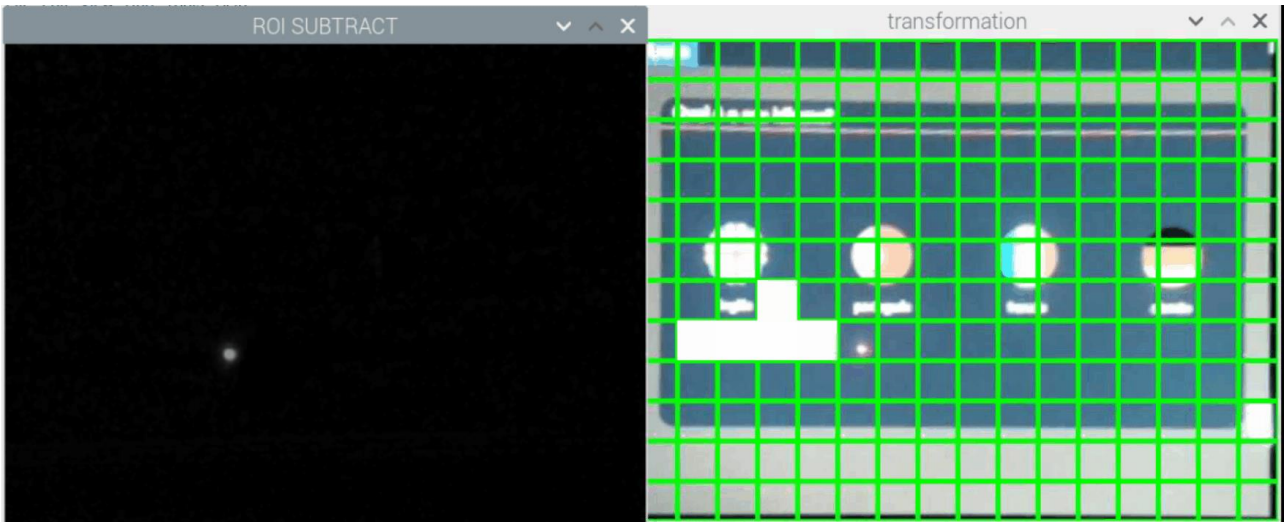

Figura 72 - Monitorização do ponto laser (técnica de subtração de fundo)

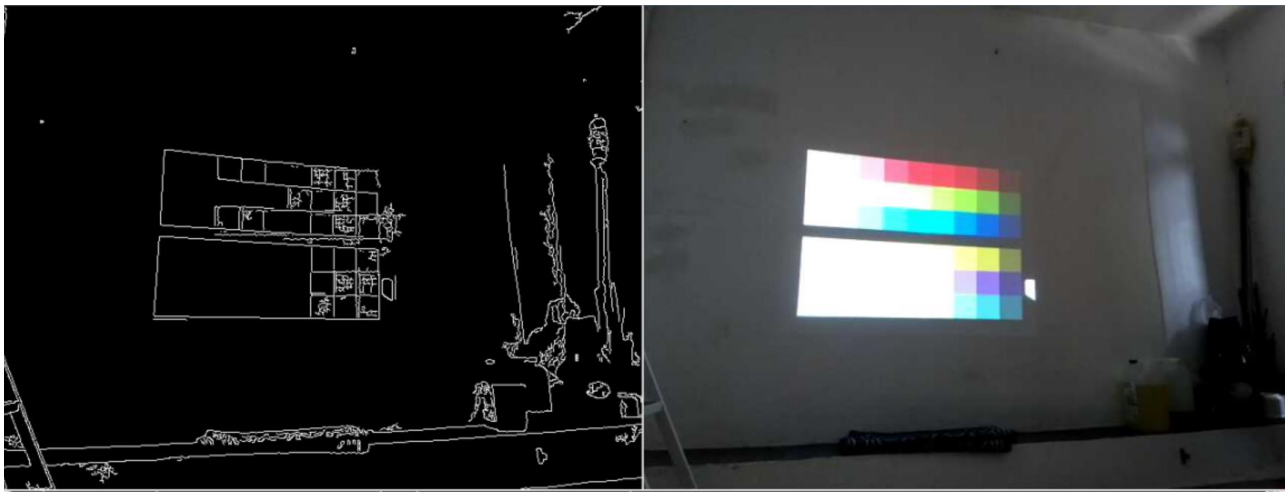

Figura 73 - Segmentação da tela projetada

Anexo C – Protótipos de alta fidelidade

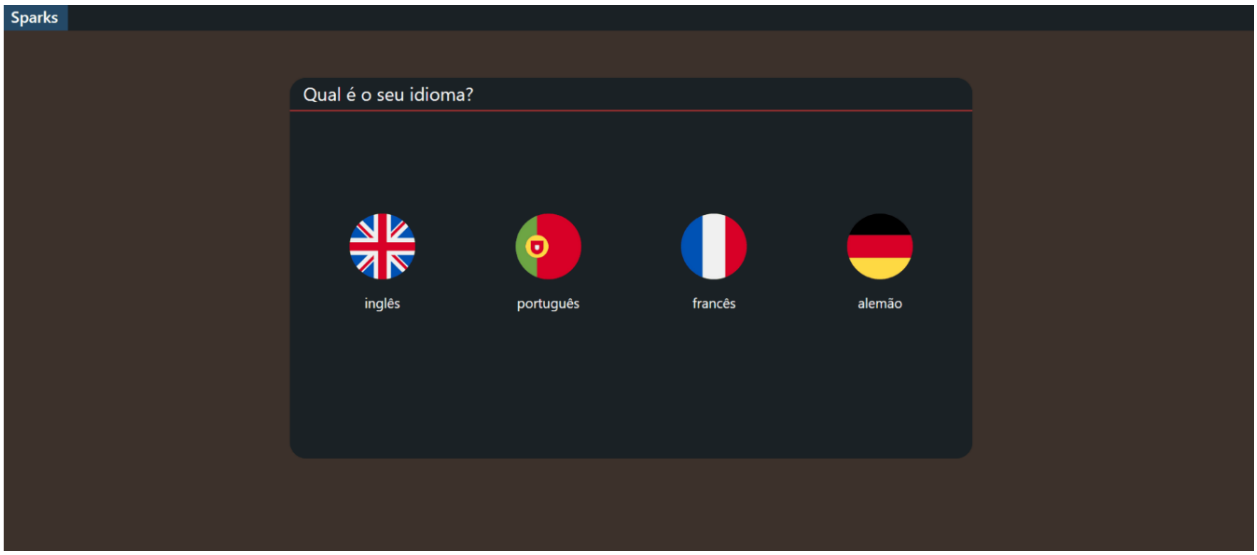

Figura 74 - Página inicial do sistema

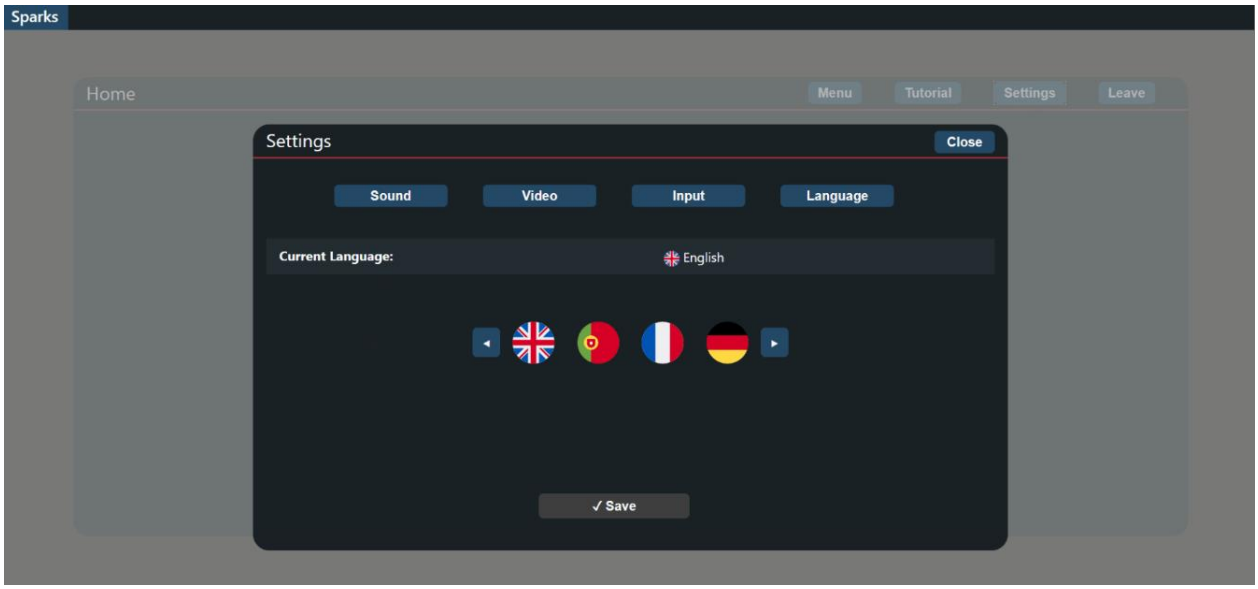

Figura 75 - Página de configuração #1

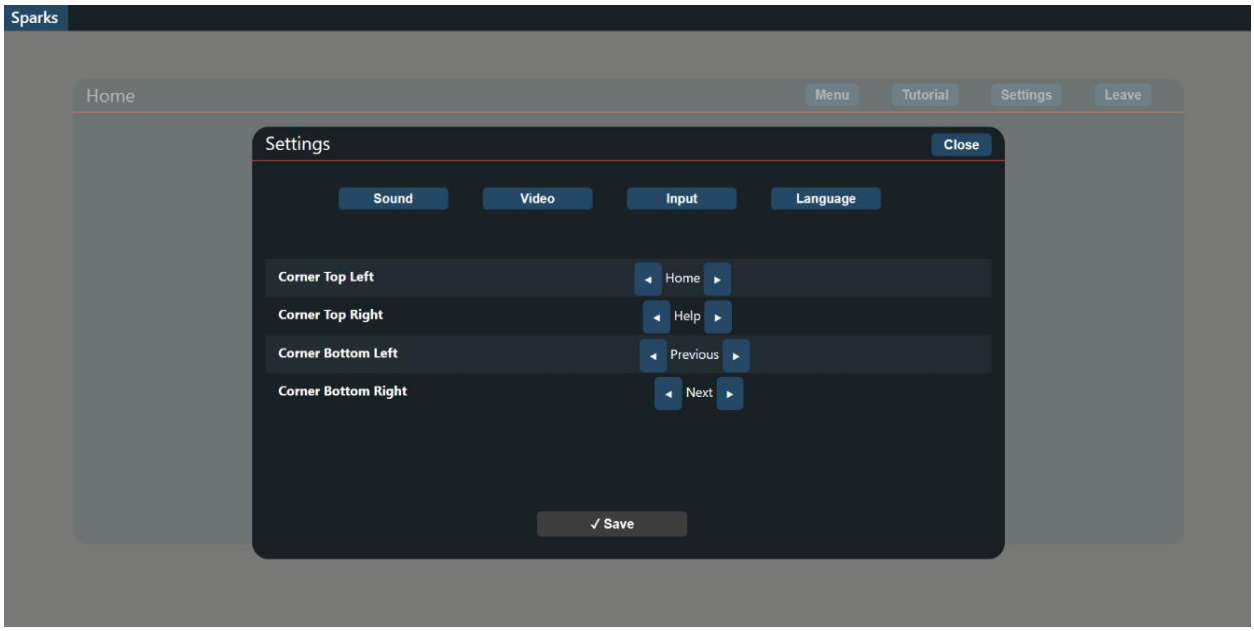

Figura 76 - Página de configuração #2

Anexo E – Depuração do gesto circular

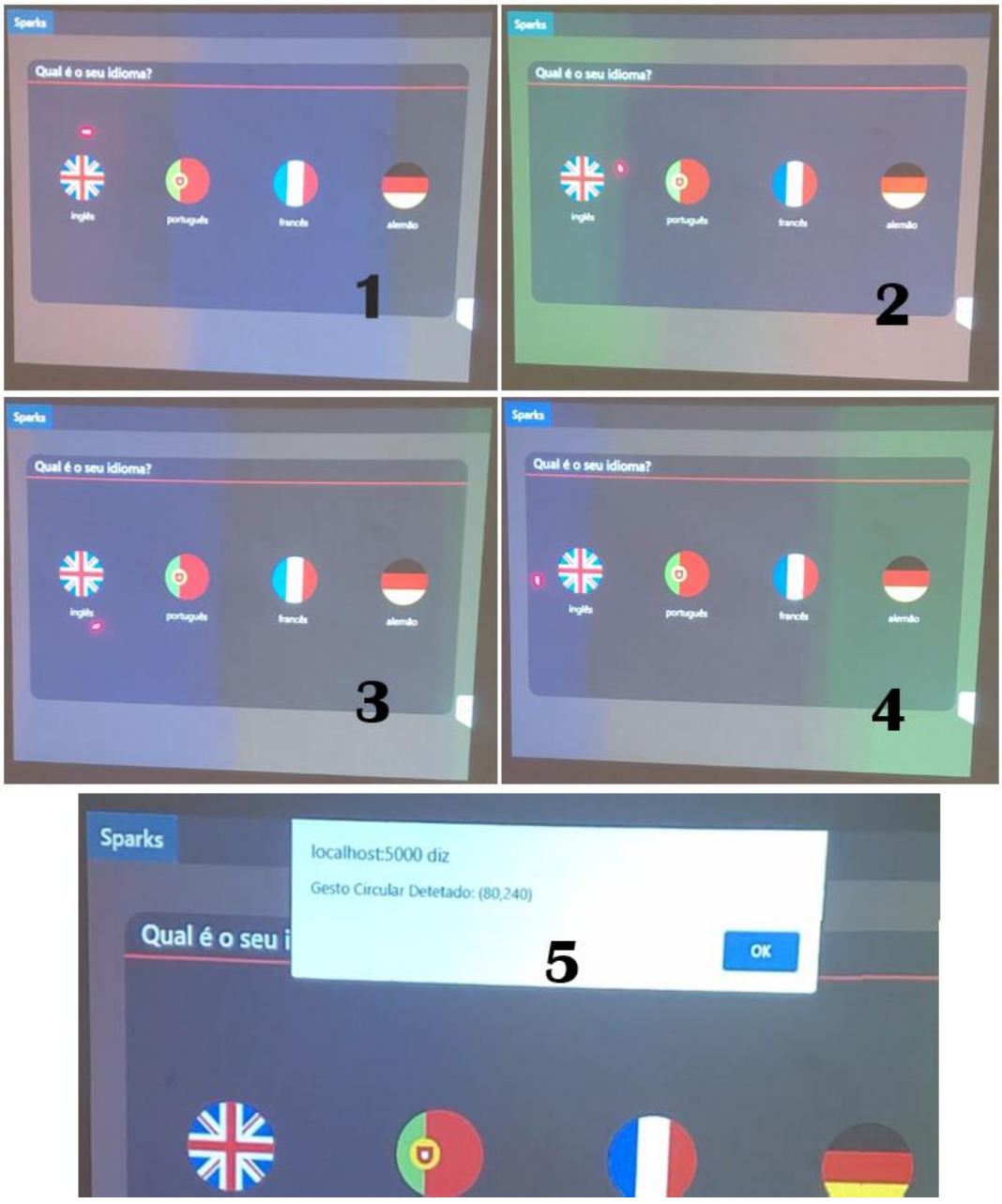

Figura 77 - Deteção de um gesto circular

Anexo D – Testes de usabilidade

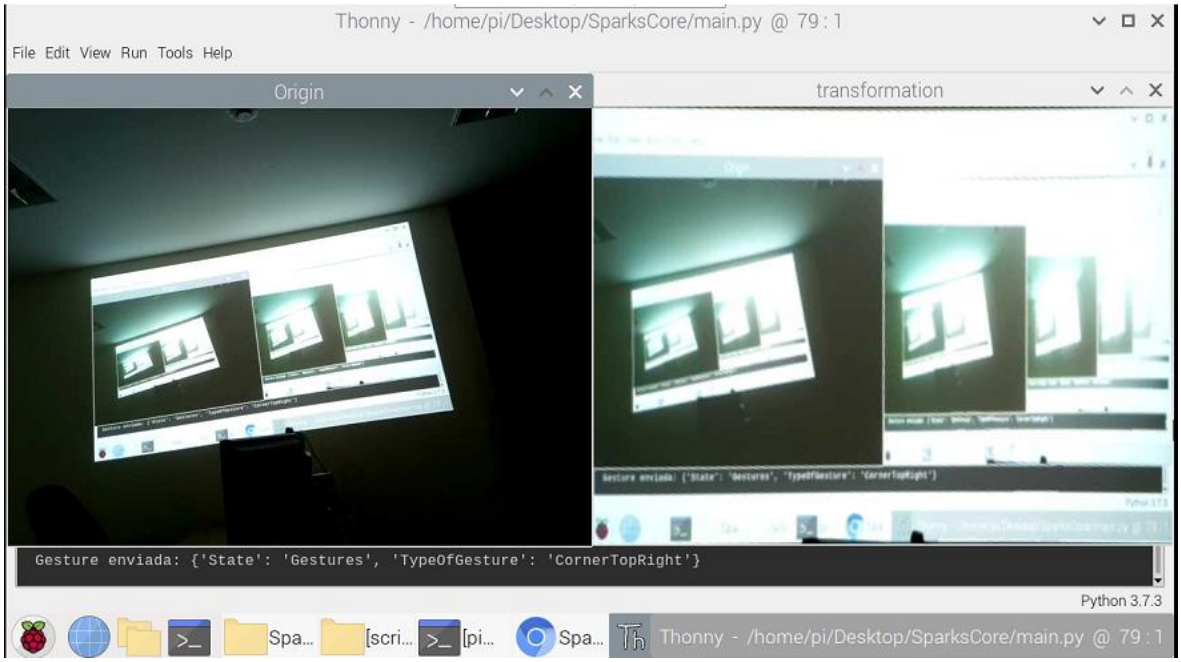

Figura 78 - Efeitos de alta luminosidade

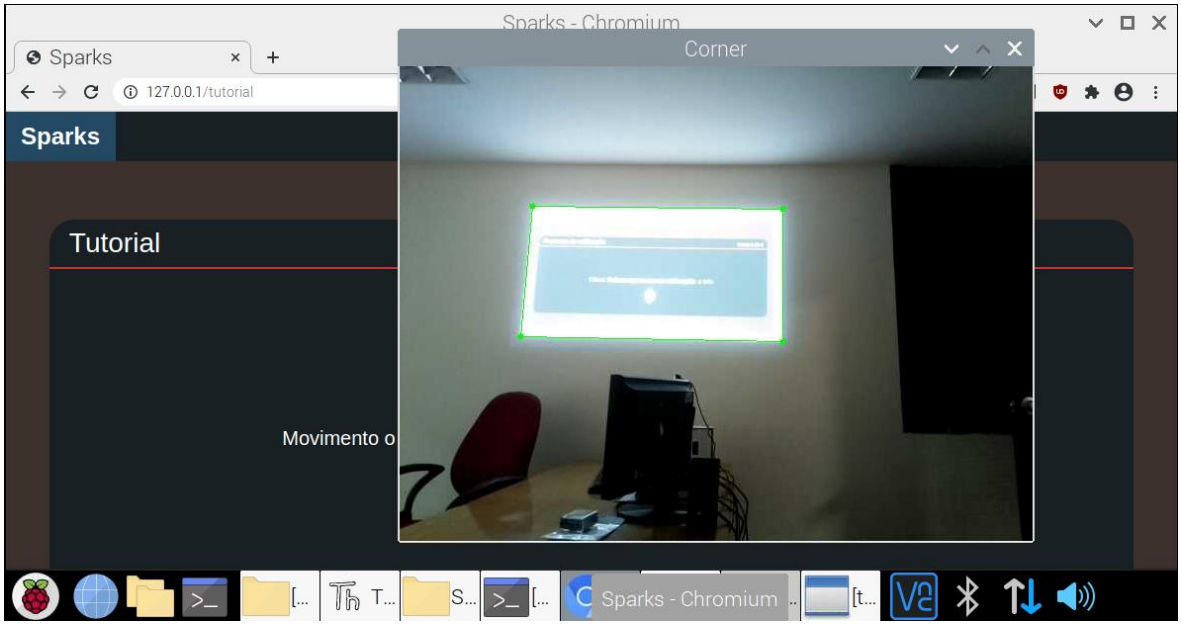

Figura 79 - Deteção da tela projetada antes dos testes de usabilidade #1

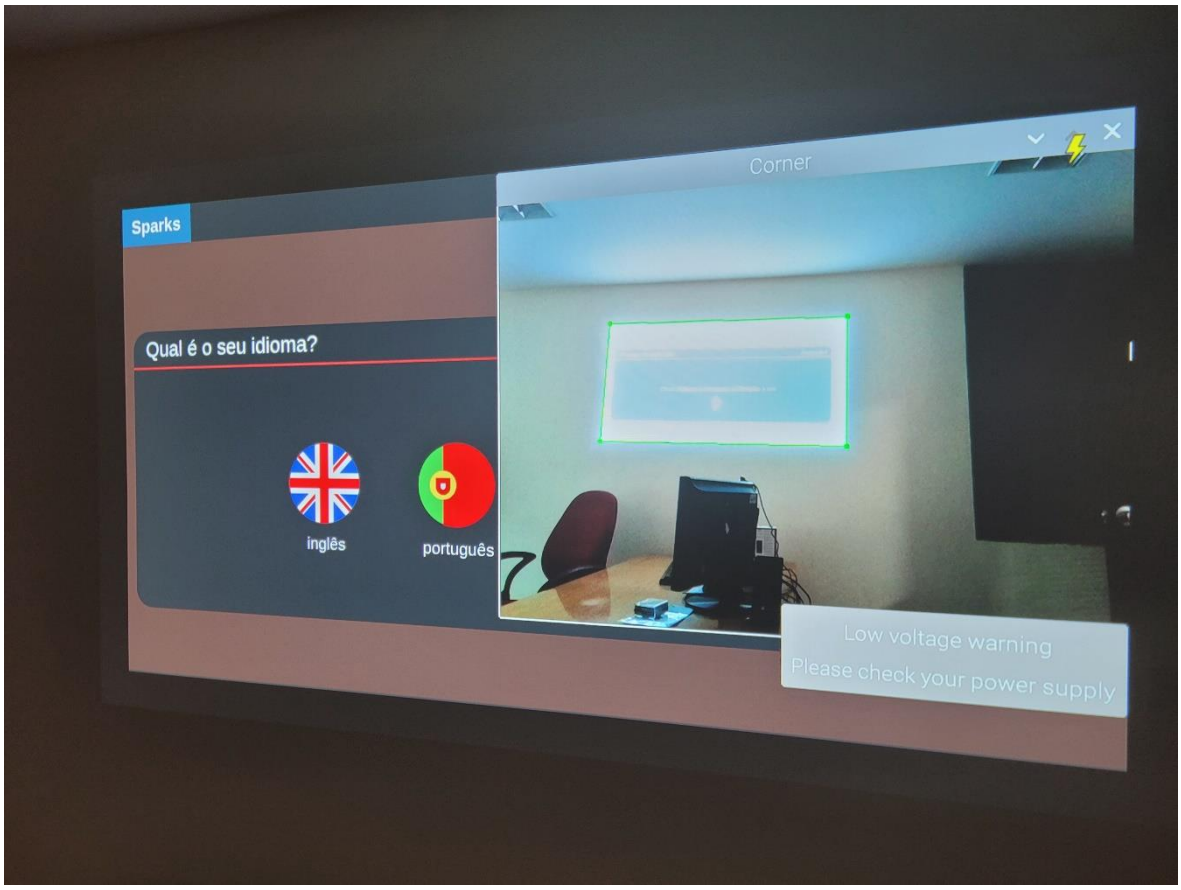

Figura 80 - Deteção da tela projetada nos testes de usabilidade #2

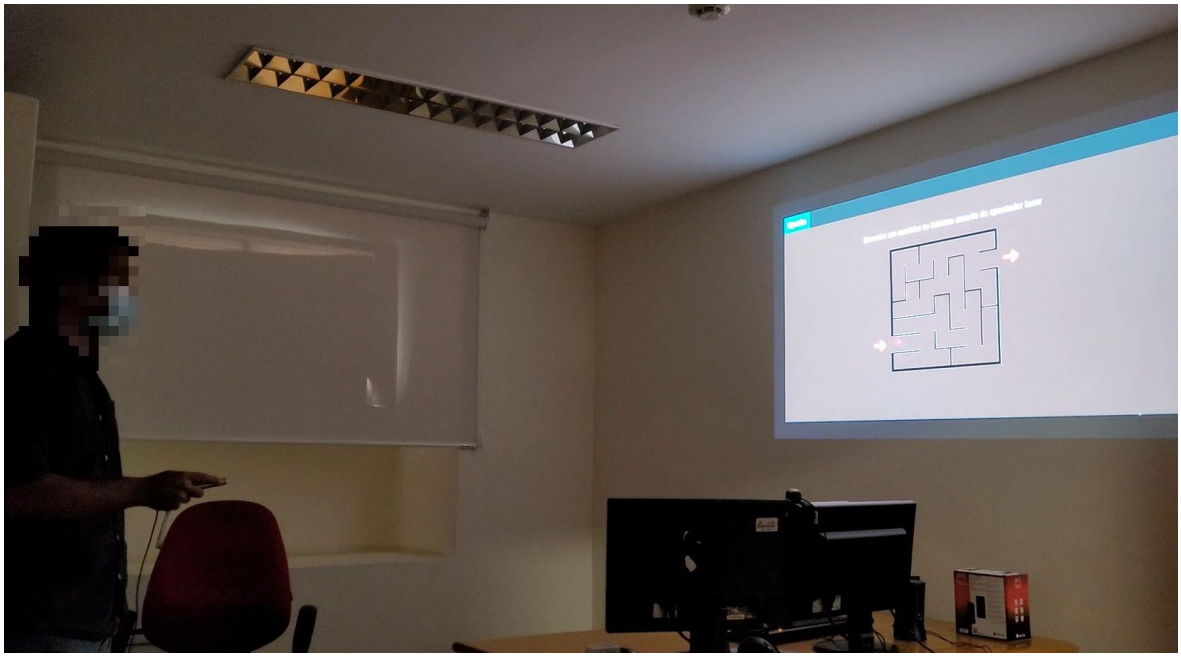

Figura 81 - Testes de usabilidade com os participantes #1

Figura 82 - Testes de usabilidade com os participantes #2